WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science

Department of Computer Science and Engineering

Thesis Examination Committee:
William D. Smart, Co-Chair
Robert Pless, Co-Chair
Chris Gill
Caitlin Kelleher
Bilge Mutlu
Annamaria Pileggi

Contextualized Robot Navigation

by

David V. Lu

A dissertation presented to the
Graduate School of Arts & Sciences
of Washington University in
partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2014
Saint Louis, Missouri

Contents

List of Figures

List of Tables

Acknowledgments

First, I thank the many fine faculty who have helped me throughout my graduate career: Bill Smart, Robert Pless, Caitlin Kelleher, Chris Gill, Bilge Mutlu, Michael Brent and Jeremy Buhler.

Many thanks to the people with whom I have shared the WashU robotics lab, including Erik Karulf who taught me the art of getting ancient robot technologies to work, Chris Wilson who introduced me to the fascinating intersection of robotics and theatre while we rehearsed a play about vampires, and particularly Dan Lazewatsky, who has debugged my ROS code, been a sounding board, and helped me complete this dissertation in more ways than I can count.

I am also grateful to the actors and others from the performing arts department who let me introduce crazy technologies into their life and work. This includes Justin Rincker and Anne Marie Mohr, for their participation in the Forgiveness piece, Justin Rincker (again) and Amanda Spector for allowing me to capture their motion, and most of all, Anna Pileggi, the fearless leader of all things theatrical. Additional thanks to Greg Bligard and Kavita Joshi, for letting me steal you away from med school to act opposite a robot. I would also like to thank Bobby Bodeheimer and Julie Adams for the use of their motion capture studio.

From the ROS community, I would like to thank Thibault Kruse, Eitan Marden-Eppstein, Dave Hershberger, Leila Takayama, Doug Dooley, Caroline Pantofaru, Steve Cousins, William Woodall, Eric Christensen and Austin Hendrix.

I would also like to thank my numerous collaborators from around the country, including Zeke Maier, Brian Haynes, Dan Allan, Ross Mead, Akhil Madhani, Agata Kargol, Karen Eng and Aaron Graubert. Special thanks to Mark Bober for his limitless technical support.

Special thanks to Lewis T. Robot and Harris T. Robot, who have shared in my love of theatre, and have willingly driven into many different obstacles in the name of science.

My most sincere thanks to my family. First and foremost to my parents, Vincent and Beverly Lu, my brother Rob Lu, my sister Karen Eng, and the rest of the Engs (Steve, Lydia and Carolyn). Without your constant love, support and encouragement I would never have been able to complete this long endeavor. Thanks for putting up with me writing papers during every family vacation. Finally, the largest thanks go to my collaborator-for-life, my beautiful wife Elise. Without her scientific insight, constant support and enthusiasm, I never would have even had the courage to start this crazy thing, and for that I am forever grateful.

David V. Lu

Washington University in Saint Louis
December 2014

ABSTRACT OF THE DISSERTATION

Contextualized Robot Navigation

by

David V. Lu

Doctor of Philosophy in Computer Science

Washington University in St. Louis, 2014

Professor William D. Smart, Chair

In order to improve the interaction between humans and robots, robots need to be able to move about in a way that is appropriate to the complex environments around them. One way to investigate how the robots should move is through the lens of theatre, which provides us with ways to analyze the robot's movements and the motivations for moving in particular ways. In particular, this has proven useful for improving robot navigation. By altering the costmaps used for path planning, robots can navigate around their environment in ways that incorporate additional contexts. Experimental results with user studies have shown altered costmaps to have a significant effect on the interaction, although the costmaps must be carefully tuned to get the desired effect. The new layered costmap algorithm builds on the established open-source navigation platform, creating a robust system that can be extended to handle a wide range of contextual situations.

Chapter 1

Problem Statement

1.1 The Difficulty of Human-Robot Interaction

Robots are cold, unfeeling machines. However, for better or for worse, people treat computers and machines in social ways and want to treat the machines like they treat other people. People have been shown to be polite to computers [73], form effective long-term social relations with robots [42] and generally want to use traditional, familiar social cues when interacting with robots [10]. When humans see robots as mutual social actors, they extrapolate from themselves and expect that behaviors that are intuitive to them will be easily integrated on the robot. They expect some level of intelligence and social awareness, and a user's a priori expectation of how the robot will behave will impact their perception of the robot [75].

We contend that this expectation stems from the lack of appropriate *theory of mind* for robots. A theory of mind is defined as the belief that an entity has "beliefs, goals, and percepts" [90] and attributing a theory of mind to another being allows one to predict what it will do, reason about its intentions and understand its motivations. Robots fit betwixt people and machines. On the one hand, there is a very simple theory of mind for machines, such as a vending machine. A vending machine will react to buttons being pressed and other sensory input, do some processing on the input (are there enough coins? is that item in stock?) and move some motors as a result. This is similar to how robots work: sense, think, act. However, people generally do not think of vending machines as intelligent agents, since they follow our commands, and (usually) produce the appropriate delicious output. Ergo, a very simple theory of mind.

On the other hand, humans have a very complex theory of mind for other humans. We cannot know what other people are thinking but, for practical purposes, we often make the assumption that their cognitive process is similar to our own, which gives them agency. We believe them to be rational agents just like ourselves, capable of making complex decisions and acting on the world around them. This allows us to plan our behavior accordingly, drawing on our past experiences and interactions with other people. We are comfortable in our interactions with other people when everyone is following the same social rules, i.e. obeying the social contract.

Robots however fall into the middle ground, fitting between these two extreme models. Due to their embodiment in the physical world (and possibly their portrayal in science fiction), people tend to attribute robots with theories of mind closer to those of humans. They seem to be rational and have free will, and as a result, robots are attributed a level of agency that is near-human-like. As a result, people may initially try to treat robots as humans.

However, the robots fall quite short of the mark. Even when they succeed in some goals, there will be other components that do not match human agency levels. For instance, in robot navigation, if the robot successfully navigates from one location to another in an efficient manner without colliding with obstacles, humans may perceive it as unintelligent because it blocked someone's path, or failed to acknowledge someone waving to it. In social interactions, there will always be layers upon layers of additional contexts that are taken into account by people that robots just have not been programmed to integrate. The result is un-human-like behavior.

The Uncanny Valley theory[69] suggests that in general people become more comfortable with something the more human-like it is, but at a certain point, when the entity is almost but not quite human, there is a deep drop-off (valley) in people's comfort level, as seen in Figure 1.1. The theory pertains to general appearance, and the effects are heightened when the entity is in motion. Robots that suggest some intelligence but fail to mirror all facets of human behavior often fall into the valley.

The reason for this discomfort is that most people do not have a theory of mind for something that seems intelligent and yet is not alive. When people see the mismatch between what they expect and what they are interacting with, it breaks their expectation, leading to a seeming break in the implicit social contract, in which people follow the set social customs, and other people will do the same. This forces people to move away from their familiar theory of mind

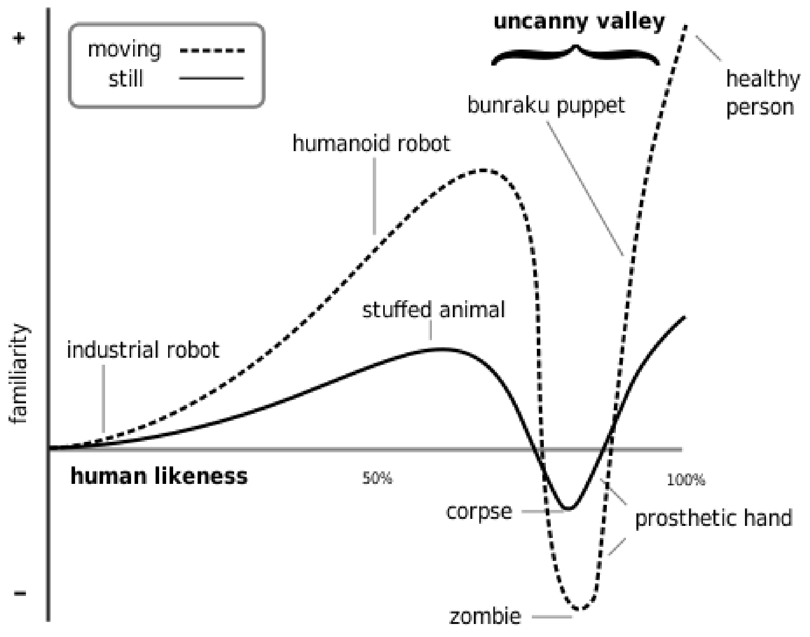


Figure 1.1: The Uncanny Valley - Do robots that move “robotically” fall into the valley, making them particularly unsuitable for interacting with people? Image used under Creative Commons License.

for how the robot should behave, and leads to them not knowing how to behave around the robot. Being unable to predict what the robot will do leads to uncertainty in action, and thus less efficient task behavior. They are stuck in an unfamiliar middle ground with no useful frame of reference. The traditional social cues used by humans will no longer apply.

The real-world ramifications of the breach in social contract have two primary effects. First, people will perceive the robot differently, perhaps for the worse. The person's image and perception of the robot dictates what they expect it to do and be able to do. Breaking the contract can lead to a perception of the robot as less intelligent or less reliable, which may affect the person's willingness to work with the robot. Experiments by Paepcke and Takayama showed that robots that failed to meet expectations were rated more disappointing and less competent [75]. This leads to the second, more important problem, a decrease in task performance. If the people interacting with the robot are concentrating on figuring out what it is going to do and what the rules of the new social contract are, then they will not be focusing on the task at hand. Their uncertainty and the robot's unpredictability will cost time.

These problems could possibly be fixed by forging a new social contract between people and robots with explicit communicative cues. For example, the robot could simply announce it's intentions or have a large monitor explaining what exactly it was about to do. The DON-8R donation-seeking robot had a flag raised behind it explicitly stating how to interact with it [82]. However, this mediates the interaction, which is undesirable since it does not leverage the existing social conventions that humans are able to respond to intuitively.

Instead, we focus on a mode of communication that all robots have some access to: physical actions through space. While the robot's physical embodiment does impose a number of high expectations onto the robot, the expectations exist alongside a large number of implicit rules about how bodies move through space. We contend that the closer we move toward meeting some of the expectations, the less disruptive the gap between human behavior and robot behavior will be. Ergo, the goal of this dissertation is to program robots to better fit people's theory of mind for them, solely by changing the robot's physical actions. [1]

[1] This consciously does not include the use of language from our research. While there is a great body of work detailing the integration of movement and language, we chose to focus solely on the former.

1.2 “Robotic Motion”

When people think about how robots move and behave, the characteristics that often come to mind are not always flattering. Their moves are jerky and have sharp accelerations, as though they were dancing “The Robot.” They are unresponsive or slow to respond to the environment around them. They act with very narrow scope, and anything outside of that scope “does not compute.” They do the same thing over and over with little variation in how they do it. This collection of attributes has come to be represented by the term “robotic” even in non-robotic scenarios. These qualities can be reinforced by the actual behavior of certain types of robots. In our language, the term robotic has become short-hand for an almost inhuman theory of mind. There is something distinctly “other” about a robot’s jerky motions.

This behavior stems from optimizations for overall efficiency. Robots are programmed to perform specific tasks in efficient ways, regardless of how such an action may appear to a person observing the robot. Jerky movements are often the quickest way to get from one pose to another, but the movements are very sudden, which can be off-putting to an observer. The narrow scope of things that the robot optimizes for is a good starting point, and often is sufficient for certain tasks. However, in human-robot interaction tasks, a narrow scope can result in a much simpler theory of mind than people expect, resulting in the aforementioned mismatch and a problematic interaction.

To improve this, the goal here is not to change *what* robots do. It is not our intent to invent new tasks for the robot to perform or create different goals for the robot to achieve. Instead, we want to change *how* the robot performs the tasks and actions. We define the *what* as the separate functional actions that are needed to accomplish a task. In the navigation domain, you could consider moving from point A to point B. Without performing these actions, the robot would not have completed the task. The *how* portion of the task entails all of the states in between the functional states required by the task. There are an infinite number of paths between two states. A small subset of them will have a minimum length, i.e. be the most efficient paths. One might go to the left around an obstacle, the other might go to the right. How the robot accomplishes its task depends on which paths it takes, as well as any additional non-functional behavior the robot employs.[2]

[2]It has been shown previously[40] that the trajectory humans take is communicative and can be used by robots to predict behavior. We believe the inverse is also true.

The choice of path among the set of paths is determined by which contexts are integrated into the planning algorithm. In the past, the only contexts were factors like avoiding collisions and traveling in minimal time, but those may not be optimal in the scope of a larger interaction. We must determine what additional contexts influence the quality of the interaction and optimize with the additional contexts integrated into the planning.

In this research, we want to improve how robots move, particularly during navigation tasks. We modify the robot's navigation behavior to include a multitude of contexts and contextual factors as additional metrics into the robot's planning algorithms.

1.2.1 Prior Work

A body of existing work suggests that the way in which a robot moves affects how people think about and interact with it. Cassell [19] shows that body pose, hand gestures, and eye-gaze direction signal changes in a conversation, initiating or terminating interactions, or facilitating turn-taking. Further, it seems that the addition of these body-language cues reinforce the use of human conversational protocols (greetings, turn-taking, etc), and make them more effective [18]. This view is strengthened by Sidner et al. [92] who found that users interacting with a robot penguin were more engaged, and used more head gestures, when the robot itself nodded appropriately. Further evidence is provided by Bruce et al. [15], who note that robot head movements are much more likely to engage passers-by in conversation than if they were not used. However, they also report that breaking social norms in this situation is "unpleasant and unnerving" for the humans involved once these physical actions are in place.

Much of the work in using motions to convey information and fit human-like theories of mind has been done in the realm of conversational agents. In these systems, an animated on-screen character holds a conversation with a human. Often the purpose of this conversation is to impart specific information, such as directions, and much of the work has focused on gestures by the character that help convey this information. Norman Badler's lab developed an approach that uses Laban Movement Analysis, a dance/theatre based way of notating and parametrizing the qualities of motions performed on stage, which creates 'procedurally generated movements' to integrate qualitative styles into the motion of on-screen characters [20]. However, since as others[4, 43] point out, robots are similar to animations, but far more engaging because of their physical presence.

The Laban approach has also been used for robots as a way to mentally structure movements, learn the measurable qualities of those motions and then apply them to social robots [85]. However, the application was primarily to have the robot recognize human actions with the system. Our work uses a different theatrical approach and is geared towards motion generation.

There has been some work to investigate the perceptions of the generated motions by looking at some very specific motion styles such as exaggeratedness [32]. However, there are many other qualities to these motions that have yet to be explored.

1.3 Creating a Theory of Mind with Acting

The field of human robot interaction has tried numerous interdisciplinary approaches to improving robots' ability to conform with social expectations. In addition to technical fields like computer science and the various engineering disciplines, researchers have brought in knowledge from fields such as psychology, design and sociology. A core principal of our work has been that ideas from theatre and acting will help inform the robot's actions.

Theatre is a very specific subfield of human interaction in which the participants (actors) interact with one another within a specific situation or set of circumstances. The given circumstances are usually presented in a script, which provides the setting and general action of the interactions. The challenge for actors is to perform their actions on stage in a way that is truthful to their specific character, the given circumstances, and the actions of the other actors. Theatre functions as a way to examine human interaction, one specific context at a time. It is a particularly useful way of thoroughly exploring the interactions, because theatre provides controlled, repeatable interactions. A "normal" interaction (i.e. non-staged) happens only once, but with theatre, the interaction can be repeated and refined. This is one of the benefits of theatre to everyone, in that it focuses people's attention on how interactions take place, which can then inform the audience about the actions in their own lives.

Theatre and HRI both aim to replicate some element of humanity in their interactions. For our interdisciplinary approach, we mainly focus on the processes that actors use on the stage. Acting is a form of embodied communication. Actors are specifically trained to convey the inner state of their character. The audience can understand what the character is thinking and what they are trying to do without them explicitly stating these things. Much of this information can be conveyed strictly through physical means. A good actor can perform an action in a way that takes into account a multitude of different contexts, like their relation to each of the other characters (among many others). Such an action will be influenced by all of these factors, even if it is not the primary focus of said action. When an actor performs an action, it is done in a way that is designed to make the audience believe that the actor has specific attributes relating to the character within the circumstances of the play. With a good performance, the audience ceases to think of the actor as such and instead substitutes a theory of mind aligning with the character the actor is portraying.

The core analogy is as follows. Both robots and actors want those observing them to believe certain information about them. Actors want their audience to believe that they are a character. We want robots to behave in such a way that observers of the robot believe it conforms to the same social rules as people do. The process that actors undergo to achieve their goal requires them to go through a given acting method. We believe that applying the same rules to robots will help them to achieve their goal as well, and that the same transfer of the theory of mind will occur. The end result will be either an audience that believes that the person on stage is the character, or observers that believe that the robot will match their social human state of mind.

During good performances, audiences suspend their disbelief, which is made easier as the actors/characters on stage fit with the audience's given models of human behavior. In making robots "less robotic", we aim to make it that much easier for humans to "suspend their disbelief" and treat the robots as social, even though they are actually not.

Furthermore, HRI and theatre are both inherently interactive. Actors must make constant adjustments to their actions based on what is happening around them. For actors, especially those in live theatre, there must be constant listening and reacting. Otherwise, if they just "go through the motions" and do not change their behaviors, the performance becomes unnatural looking, leading to the illusion of the character breaking down. Such behavior can lead to the actor being reviewed as "robotic", fulfilling the same qualities mentioned earlier.

This work exists in parallel to Brenda Laurel's work viewing human-computer interfaces through a theatrical lens [51]. The primary difference between our work and Laurel's rests in the division between the human's world, and the computer's or robot's world. Human-computer interfaces generally rely on physical input and output devices, such as mice, keyboards, and graphics on a screen. In human-robot interfaces, the interaction takes place more on the human's terms, using devices like speech synthesizers and actuated movement in the "real" world. In both cases, the computer must define itself in terms easily accessible to the human, but in human-robot interactions, it is less about defining the interface, but defining the robot's role within the context of a normal social interaction.

In addition to theatre, another art-based approach to HRI includes using principles from traditional animation to help guide how robots should move [65, 104]. There is also the previously mentioned work using the Laban Movement Notation, which was originally developed for dancers.

Given the similarity between the processes and the effectiveness of the metaphor, we endeavor to use this multidisciplinary approach to program robots to convey the same sort of agency.

1.4 Robot Platforms

1.4.1 The Robot Operating System - ROS

The work in this dissertation builds upon the Robot Operating System, or ROS. ROS is a completely open-source middleware and library suite[83], initially developed by Willow Garage, and now maintained by a large community of individuals and companies, most notably the Open Source Robotics Foundation. One of the key features of ROS is a messaging system that can be used as the transport layer which has bindings for many programming languages. This enables robot systems to function as a collection of independently running executables, called nodes, which use a publish/subscribe model for communicating with other nodes on a named topic. Each "topic" publishes streams of structured messages, providing standardized interfaces for frequently used data structures. For example, all laser range-finders produce the same kind of message (called $LaserScan$), regardless of the manufacturer or properties of the device.

This structure has several key benefits. Firstly, nodes may be developed independently, providing that they adhere to a fixed message format. Each node can be written in an appropriate programming language, since the messaging system works across languages and operating systems. The separation of functionality into domain-specific nodes also enables flexibility in the configuration of robot system pipelines. Consider an image processing node that subscribes to an image topic, and publishes a position estimate based on its analysis of the image. First, it should be noted that such a node could easily be swapped out for another node with the same functionality, so that we could replace an older node written in C++ with a newer one written in Python. Secondly, the node will function regardless of what happens upstream or downstream of the node. The image could be published directly by any of a number of camera drivers, or it could have already gone through some pre-processing. This means that the core algorithms for robots can be developed independently of specific robot capabilities and drivers.

The standardization creates an environment where code-reuse across multiple platforms. One notable example is a demonstration done by Brown University in the early days of ROS[3], where the goal was to sense an AR tag with a camera, and then drive towards it. This was demonstrated on three robots with diverse sensing capabilities and a variety of actuation configurations. However, due to the abstraction layer provided by the common message types, the implementation completely separated the high-level logic and task from the low-level implementations of the camera and motor drivers.

This means that high-level modules can be implemented on numerous different robot platforms with less effort than if the code was implemented specifically for each platform. There is a ease with which a developer can get a sophisticated algorithm working on a new robotic platform, by creating low level drivers that connect with previously developed packages.[4]

We chose to work with ROS for a variety of reasons. Beyond the structure described above, it also has gained wide-scale adoption, more so than any previous open-source robotics platform. There are thousands of users engaged in discussions and editing the community documentation from around the globe[27]. Not only does this pool of people working on the same platform mean that there are more resources, but more importantly, it means that there are numerous labs around the planet who may want to actively reproduce your results.

[3]`https://www.youtube.com/watch?v=mKmqgVUbQQM`

[4]All ROS code is organized into 'packages', and groups of packages are grouped into a 'stack.'

In that vein, all of the source code for this thesis is released into the ROS community. We have embraced this philosophy throughout the work on this project. Early on, I contributed code, documentation and tutorials for enabling ROS users to create more of their robots in a ROS-standard specification called the Unified Robot Description Format, or URDF. These artifacts have become central to many projects, and although they are not specifically used within the scope of the work described in this thesis, they proved useful for my internship with Walt Disney Imagineering Research and Development and engendered positive karma from the ROS community which led to, among other things, my internship at Willow Garage.

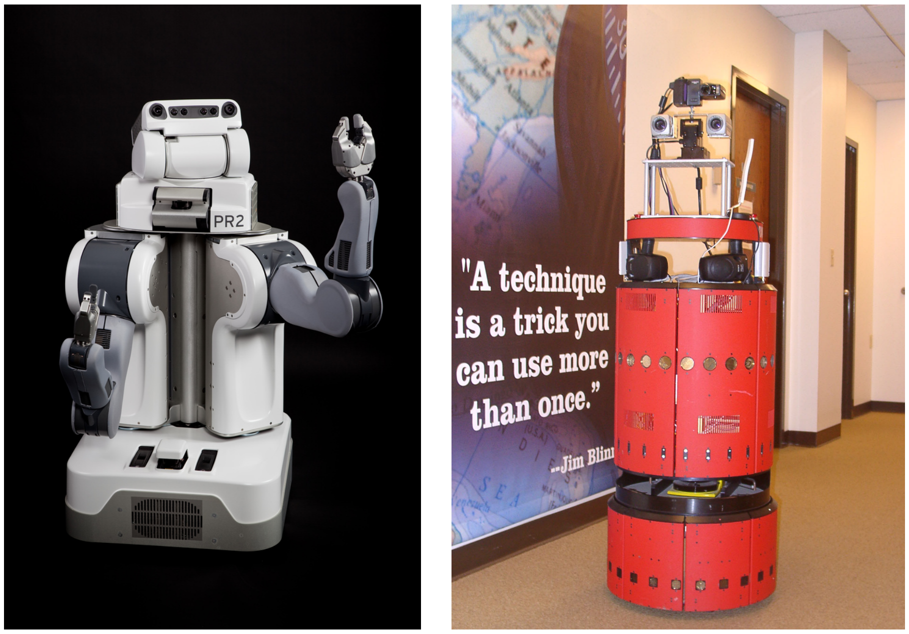
Releasing our code and contributing back to the ROS community follows the doctrine articulated by Buckheit and Donoho:

> An article about computational science in a scientific publication is **not** the scholarship itself, it is merely **advertising** of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures. (Buckheit and Donoho [16])

1.4.2 Robots

Due to the nature of ROS, much of the work described herein can be explored with any number of robots. The work has primarily used the PR2, a robot developed and manufactured by Willow Garage. as seen in Figure 1.2a. The PR2 has a near-holonomic base (meaning that it is capable of moving in any direction almost instantaneously), an articulated head capable of panning and tilting and two seven-degree-of-freedom arms. The PR2 has a full slate of sensors, including two laser range finders, five head-mounted cameras, and a camera in each of the forearms. The PR2 was developed alongside ROS, and therefore has ROS drivers for each of its sensors and actuators.

The other robot that was used for much of the pre-PR2 work was the iRobot/Real World Interfaces B21r, as seen in Figure 1.2b. The B21r, known as "Lewis", has two degrees of freedom it the base as well as a pan-tilt unit for the "head." The sensors include a Hokuyo laser range finder, sonar sensors mounted around body, infrared scanners and two video cameras mounted on the pan-tilt unit. The B21r stands approximately 1.2m tall "at the shoulder", making it human-sized and useful for interactions with humans.



(a) (b)

Figure 1.2: (a) The PR2 platform by Willow Garage (b) The iRobot RWI B21r.

1.4.3 Navigation

One of the most widely used high level behavior ROS module is the navigation stack. Creating this module was central to ROS's early development, and figured prominently into Willow Garage's aims as a young company[5]. The navigation stack runs on dozens of robot platforms, each using the same core code to navigate and avoid obstacles.

One of the defining features of this dissertation is that it is not intended as a one-off demonstration of how different navigation behavior could theoretically be achieved. The code for this research has been integrated into the main ROS open-source repository. It has been released as part of the Hydro release of ROS, enabling people using ROS Navigation to benefit from the improvements made to the code without needing to consciously try to do so. It is hard to estimate how many people are actively using the code developed as part of this research. ROS has an estimated 100,000 users[31]. All of those users running the most recently released version of ROS (code named Hydro) to move real or simulated robots through the world are using the code developed as part of this work. This work also enables

[5]`http://www.willowgarage.com/blog/2009/06/03/watch-milestone-2`

future researchers to use our code as a starting place, giving them the ability to stand on our shoulders research-wise.

Chapter 2

Robots in Theatre

In this chapter, we take a look at the intersection of the fields of robotics and theatre and how it motivates our work. First, we make a brief survey of the prior work in this field. Then we discuss the primary motivation for using the two fields together, first from an acting perspective, and then from a philosophical perspective. These arguments convey the fact that in terms of the interaction, all that matters is performing actions truthfully, such that it does not matter what the underlying process is. Outside of the interaction, it is important to discuss the process, in order to understand the technical achievements of a given work. To this end, we present a structure for discussing such works in the form of an ontology of robot theatre.

To conclude, we present our views on the strongest areas where the two fields will prove beneficial. In the following chapter, we will show how these methods have been put to use in our research.

2.1 Robot Theatre Work

Throughout their histories, the fields of robotics and theatre have been occasionally intertwined. The word "robot" itself first appeared in the context of a stage play [103], in 1923. However, mechanical devices have been integrated into theatre starting long before Čapek's play. The term "deus ex machina", referring to a mysterious outside force that enters the play and cleanly wraps up the plots, stems from the Ancient Greek practice of lowering a mechanical god onto stage as a plot device. However, it has only been in recent years with cheaper and more prevalent technology that the intersection of the two disparate fields has really flourished.

Initially, robot theatre was dominated by robot-only performances. The first live work that we are aware of is by Ullanta Performance Robotics, a group of Brandeis (and later USC) graduate students who staged a small number of plays using an all-robot cast [105]. Wurst [107] created three small robots to perform a highly stylized improvisational piece called "The Lazzo of the Statue", in which one robot pretends to be a statue that moves when the other robot actors' backs are turned. Bruce et al. [14] used two autonomous robots to perform comedic improvised pieces.

More recently, robots have been integrated into performances with human actors as well. Hoffman et al. [37] described a set of performance pieces involving a robotic desk lamp and a single human actor. Ishiguro and his collaborators produced a play, recently performed at HRI2010, with two human actors and two Wakamaru robots, exploring the relationship between humans and robots [5]. In more traditional theatrical settings, Les Freres Corbusier theatre company mounted a well-reviewed production of *Heddatron* [9], in which five robots kidnap a Michigan housewife and force her to take part in their production of the Ibsen play *Hedda Gabler*. This piece is typical of "real" theatrical appearances of robots, in that the robots were essentially self-propelled props, used for comedic effect [23]. *A Midsummer Night's Dream* has provided inspiration for a number of robot-theatre collaborations (possibly due to its themes of different worlds colliding). Duncan et al. [25] employed robot helicopters to act as flying fairies in a mostly human production of the show. Also, the UPenn GRASP Lab put on a series of pieces with humans and robots inspired by the same play [81]. Furthermore, the new opera *Death and the Powers* explores the relationship between people and technology, while human singers share the stage with custom-built robots controlled by human actors [6].

Less traditional performance pieces with robotic elements, such as Breazeal's "Public Anemone" [12] are also appearing more and more frequently at conference venues such as SIGGRAPH. One of the main venues for people to see robots in a theatrical setting has been theme parks [12]; however, in the last few years, the animatronic characters have evolved to interact with guests on a personal level [109].

The first work at Washington University using a robot in a theatrical setting was the B21r Lewis performing as an actor in a student-run play festival along side four human actors.

The plot of the play, titled “Dr. Oddlust”, involves a woman introducing her new boyfriend to two of her friends, with the twist that the boyfriend is actually a robot. Our work in this domain continues a line of research that started with Chris Wilson’s Masters Thesis [106].

2.2 Physical Action

In Chapter 1 we discussed the similarities between theatre and HRI as both attempting to create interactions in a given context, and that our approach to improving HRI is to use the techniques that actors use on robots. However, this requires employing an acting philosophy that is fundamentally compatible with the robot’s task. We need the ability to discuss theatre as embodied communication.

The approach we chose draws from the idea of physical action, one of the main ideas that we draw from theatre, based on the acting methodology espoused by Jerzy Grotowski [34]. Under this philosophy, there is a line drawn between an activity and a physical action, where an activity is something a character does, and a physical action is something the character does for a specific purpose. This is not coincidentally very similar to the distinction we have between ‘what’ and ‘how.’ A simple activity is walking toward another character. A physical action is walking toward another character in order to confront them as a means to achieve the character’s objective. As Thomas Richards describes Grotowski’s teachings, “It was the how and the why that made them, not activities, but physical actions” [86]. The physical action brings a clarity of purpose and a depth of reasoning to the action. By putting a reason behind the activity, it makes the action have meaning. Richards describes it being “ essential that in this moment I remember my intention: for whom I was walking...I had to concentrate on how I did it, on the precise way of walking, and for whom” [86]. In essence, it conveys information about why the action is being performed.

Whether robots can fit into a theatrical production depends on how one approaches theatre. For instance, if one subscribes to “The Method” school of acting, in which actors must recall and relive a powerful emotional memory from their life in the moment of acting in order to properly convey that emotion on the stage, robots will not fit due to their inability to have authentic emotions. However, there are other approaches which fit a great deal better.

The approach that Grotowski espouses is much more in line with robotics. Robots can be programmed to have an objective and perform actions in ways to achieve those objectives. Programmers can imbue every movement with sufficient purpose to have them cross the line into physical action. This school of acting also does not require emotion as a precursor to action. Instead, the emotion functions as the result of the action, tied to whether it succeeds or fails in accomplishing the objective. The great director Constantin Stanislavski was quoted as saying "Do not speak to me about feeling. We cannot set feeling; we can only set physical action" [86]. Instead of formulating all of their motivations around pure emotion, actors must compose their character's physical actions in terms of an objective and goals they are trying to attain.

The focus on objective gives each movement performed a grounded specific motivation, and thus expands upon the communicative potential of such an action. Furthermore, without requiring emotion for performing the physical actions, we can program the robot to perform such physical actions without needing to simulate emotions.

2.3 The Chinese Room and The Stage

The similarity between robot interaction and theatre can be viewed through a comparison to Searle's Chinese Room argument [91]. The essence of Searle's argument is that no system that simply manipulates symbols without truly understanding them can be said to be intelligent. The thought experiment starts with a person locked in a room. He receives messages in Chinese on pieces of paper pushed through a hole in the wall. He looks up these symbols in a book, copies the "answer" symbols that the book shows onto a new piece of paper, and then shoves this paper through the hole in the wall. The translator does not speak Chinese, but is simply manipulating symbols based on the book he has. However, an external observer who *does* speak Chinese can write a message on a piece of paper, push it through the hole, and get a perfectly valid response back, as if the person in the room *did* understand the original note.[6] This, Searle claims, shows that a system that purely manipulates symbols (the person in the room) cannot be said to be intelligent (understand Chinese), even though the outputs

[6]As with many arguments in philosophy, this example is meant to prove a point, rather than suggesting that it would be practical to construct such a system. As a result, it glosses over many details, such as how to write the translation book in the first place.

that it generates in response to inputs are exactly the same as those that a person (assumed to be an intelligent system) would.

While we will not argue for or against Searle's claim here, we note that he draws a distinction between two types of systems, those that are "intelligent" (or understand Chinese) and those that are merely manipulating symbols without understanding, even though these two systems generate the same input-output mappings (respond in exactly the same way). One system has *true* intelligence, and the other only *appears* to be intelligent.

We can extend Searle's line of thinking to robots in social situations. Instead of manipulating Chinese symbols on pieces of paper, the system takes in social cues as input, and generates social cues as output. People do this naturally, and can probably be said to be "social," in the sense that they understand these cues and what they mean. Robots, on the other hand, are like the person in the Chinese room; they are presented with a set of social cues, and they give a response calculated to be appropriate. If we accept Searle's argument, then we must also concede that truly social robots are an impossibility; a robot can only give the appearance of social behavior while actually only manipulating symbols.

Introducing the idea of acting to the Chinese Room scenario reveals some useful parallelism. Instead of engaging in conversation in Chinese, the actor communicates with their physical actions on the stage, and the "input" is the context of the play around them. Consider the titular character in the Shakespearean tragedy, *Hamlet.* Throughout the play, the character of Hamlet is presented with many dramatic situations, such as his dead father appearing to him as a ghost.

While the text provided to the actor gives the language the actor can use as a means to action, directions on physical movement are mostly limited to entrances and exits, leaving many possibilities for the physical score of any given scene. Given all of the factors in the situation, such as Hamlet's likely fear of ghosts and his grief over his father's death, the set of valid responses/actions is constrained. In most cases, it would be wrong for Hamlet to rush forward and hug his father; that is a wrong output for the given input. Instead, a hesitant step backwards before questioning the apparition would be much more appropriate in the context presented to him.[7]

[7]Once again, as we did in Chapter 1, we will limit our discussion to physical actions while excluding language as a means of accomplishing a character's objective.

The "correct output" for an actor consists of actions that are consistent with the context of the play and the character's objective. These actions are what will be perceived by the audience. The audience cannot know (and does not care to know) what process the actor is going through to create such a performance. All that matters to them is whether he *appears* to be the Prince of Denmark. The actor will never actually *be* Hamlet; all they can do is act in a way consistent with how Hamlet would act, which the audience will judge in terms of what the actor's physical actions are. [8]

This reflects the essential performative nature of theatre, and why theatre is a good model for improving HRI. The ultimate goal is the audience perception, getting them to both understand and believe what the character is doing. The audience is not judging based on what is going on inside the actor's mind, just as the Chinese speaker does not care what happens inside of the room. This is the essential nature of performing. Since we need the robot to appear to be social, it is valuable to think of the robot as an actor so that it can employ the same performing techniques to generate the proper physical actions.

The focus on the audience's perception fits with Grotowski's view of emotion. To get the "correct output" is it enough that the actor playing Hamlet *appear* to be fearful, or does he actually need to *be* fearful? The famed school of Method acting generally suggests the latter, where the actor must internalize real emotion in order to correctly portray it [94]. Meanwhile, our chosen process based on Grotowski does not require emotion. All that matters is the physical actions borne out of objective; the emotion will follow.

The similarity of the relations between actor and play and between robot and interaction presents a strong case for using theatre as a model for HRI. Both exist within a contextualized interaction in which each must fill a specific role. The goal is to give the appearance consistent with the unobtainable ideal (normal social player/actual character) by performing correct actions.

[8] To put it into the terms of a theory of the mind, the actor wants the audience to adopt a theory of mind consistent with Hamlet, and not the actor himself.

2.4 The Ontology of Robot Theatre

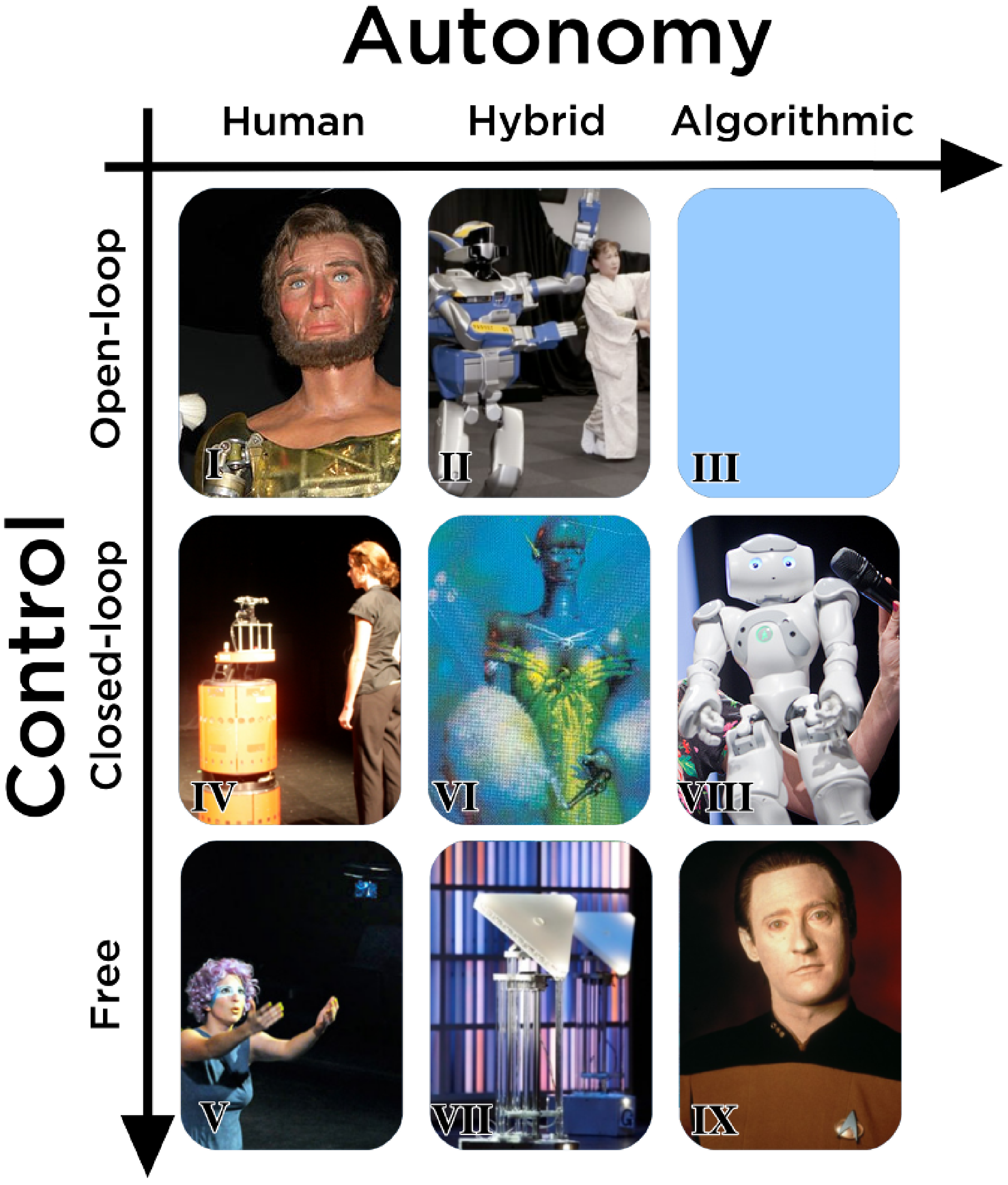
Any time that a piece of interesting technology is placed on stage, a fundamental question arises as to what role the technology plays. In many cases, it is often just a prop, much like a skull, spoon or other object that an actor might interact with on stage. However, once the technology starts moving on the stage, apparently under its own volition, it starts to become something else. It moves closer to the role that the humans on stage have: actor.

It is tempting to say that any such robots are acting much like their human counterparts. This is especially true when the robots are humanoid or recite lines of dialogue. However, making the claim that they are truly acting is not that easy. There is a large gray area in between things that are known not to be acting and people who are undeniably acting. On one side, there could be an immobile robot that occasionally beeps placed on stage, playing the part of "inert robot" in a scientist's lab. On the other side of the spectrum, one could envision a future robot which can read the script for "Hamlet," make decisions about how it wanted to perform its assigned role, and then perform on stage with other actors, altering its own performance to react in real time to the others' performances. Between these two extremes exists a continuum of robots that could be considered "acting robots" but are clearly not doing the same things.

Where should the line be drawn between acting and not acting? Who gets artistic credit for robot performance? In order to answer these questions, it would be useful to have a common framework with which to compare the different types of performance. In this section, I propose an ontology for classifying the different categories of robot performances and explore where some recent robots fit into the proposed classification system.

The categories in this ontology are differentiated along two different axes. First, there is the automation level, which describes how much of the robot's actions are prescribed by a human, and how much of it is derived by some algorithm running on the robot. The second variable is the control of the system, exploring how reactive the robot is to the immediate situation on stage, ranging from simple playback algorithms which never change their behavior, to fully reactive systems which adjust their performance to any number of factors.

As you can see in Figure 2.1a, we divide the area created by these two axes into nine separate regions, which correspond to to nine different classes of robot actor. The regions



(a)

Figure 2.1: Examples in The Ontology. (I) Audio-Animatronic Lincoln at Disneyland (II) Dancing HRP-4 from AIST (III) None (IV) B-21r performing at Washington University (V) Quadcopter doing Shakespeare at Texas A&M (VI) Cover for the short story "The Darfstellar" (VII) Operabot from "Death and the Powers" (VIII) Data (Nao) performing comedy at TED (IX) Lt. Data from Star Trek. All images copyright their respective owners.

are numbered in one possible ordering corresponding to increasing difficulty of the task. Class I includes robot systems that are open-loop with human-produced actions, like the Audio-Animatronics at Walt Disney World, which perform the same hand-animated actions show after show. Class II is also open-loop but introduces a level of computer control, in which actions are largely dictated by humans, but some portion is derived by the computer, like the dancer created at AIST [72] which bases its movement on motion capture of a human dancer, but then modifies the motions in order to maintain stability and stay upright. Class III is the last open-loop system, but its motions are algorithmically generated, possibly precomputed. Such a system is now known to exist at this time.

The robots in Classes IV and V are controlled by people in real-time, giving additional levels of reactivity. Class IV robots can react to things on stage in real time, but only in very limited ways, such as the human operator choosing between one of a few discrete choices. In Section 3.2 we describe a robot falling into this class in our own theatrical presentation. Fully teleoperated robots fall into Class V, since the operator can command any action. This class includes the robots from the production of Heddatron in New York [9] and the production of A Midsummer Night's Dream performed at Texas A&M with quadcopter faries [25].

Class VI is defined for all closed-loop hybrid systems, where the performance combines human generated elements with algorithmically generated elements, although only a finite number of ways, like the PR2 in the play "Roboscopie" created at LAAS [53]. Class VII robots also have hybrid control, but with fewer constraints on how the robot can react, as in the performances in MIT's Robot Opera "Death and the Powers" [39] in which the technology on stage is the result of teleoperation combined with algorithmically generated behavior.

Classes VIII and IX feature robots that are primarily algorithmically controlled, with varying levels of reactivity. Class VIII can autonomously react to stimuli in a limited number of ways. One of the closest examples of Class VIII is Data, the robot stand-up comedian [46, 47], autonomously selects which (pre-written) jokes to tell based on audience feedback. The final class, Class IX has free control of its algorithmic performance. To the author's knowledge, no autonomous systems of this type exist currently. However, in the realm of fiction, the examples are plentiful, most notably another Data, Lieutenant Data from "Star Trek: The Next Generation"[52, 66, 67]

For a more thorough explanation of the classes, see our 2012 paper on the subject[56].

2.5 Practical Crossover

The techniques for using actors and theatre as a model for robot interactions fall into three categories. The first uses actors and their movements as an explicit model for robots, integrating the actors' actual movements onto the robot. This includes strategies such as motion-capturing actors and transferring those behaviors into the robot's movements. The second category relies on using the general approach that actors use to approach interactions in order to give purpose to robot action and give it clear structure, using theatre as an implicit model for robot behavior. The third category is using theatre as a venue for testing the quality of social robots.

Ultimately, we chose to focus on the second approach, in which the problem of interaction is approached as though it were a piece of theatre. As such, we will examine each movement the robot makes to see whether it is in service of a Grotowski-esque objective. This will require looking at each physical action as an actor would, to figure out what the motivation behind it is.

In order to properly collaborate with people in another field, there needs to be a common language. The work in this chapter describes several explorations into the commonalities between the fields where we can find middle ground as well as the philosophical issues arising from the field. The key difference is that there are numerous separate subareas in the field of robots and theatre, and that we must recognize improvements in each in turn.

What we gain from the analogy to Searle by way of the Chinese Room argument is that, in both theatre and social robotics, all that matters is the "performance." On the stage and in an interaction, if the correct actions are performed for the given inputs, then it is a success This is not necessarily a universally held belief. There are some, particularly in the theatre community, who believe that human actors have an undeniable human element that a robot can never have. This mirrors Searle's belief about the human/room system not truly understanding Chinese. However, Searle assumed a system that produced the correct output, whereas it is possible to believe that a robot cannot produce the proper output because it lacks an essential human element. This remains a fundamental divide, one that seems to demarcate a separation between scientists/engineers (who believe in the capacity to model all complex systems) and artists/humanists (who believe that some things cannot be modeled, and those things are distinctly human traits). At this point in the development of technology, it is not necessary to choose a side. There is plenty of improvement to be made at modeling the system and increasing the quality of the interactions before it is necessary to worry about whether robots can do everything as well as humans.

One thing that Searle's argument highlights is that it is important to know what is going on behind the scenes in order to fully judge the intelligence of the system. This is a separate judgment from the quality of the Chinese conversation/interactions/performance. This is why developing the ontology was important. One could imagine a performance by the Lincoln Animatronic from Disneyland using a Class I system conceivably giving the same exact performance as a Class IX robot. In one sense, if the performance is effective and affects the audience in the same intended way, then the process is irrelevant. However, for academics and developers working in the field, there is a world of difference between Class I and Class IX. Knowledge of the mechanisms behind the performance allow for better contextualization of the robot's capabilities.

The best analogy to understand the magnitude in class differences with demo videos. Over the past few decades of robotics, in addition to the traditional means of sharing one's work with others, i.e. academic publishing, demos are increasingly common as ways to prove the worth of the research. Whether in the form of live demonstrations or videos, these demos can often conceal both the complexities and the shortcomings of a particular technology. A working demo can be indicative of an incredibly complicated recognition task working with the latest adaptive articulation platforms, or it could be the result of carefully controlled conditions. Nowadays, most researchers attempt to make it clear where their deceptions are, by acknowledging them clearly. This means labeling videos when they are the result of teleoperation, or when they are running open loop, or when they are sped up considerably. Despite the impressive performance in the video, the true value can only be judged if the audience knows what is going on behind the scenes.

Consider the difference between the systems behind Willow Garage's PR2 in the study by Takayama et al [99], and Georgia Tech's SIMON programmed by Gielniak et al [32]. Both systems perform animated expressive gestures. However, the PR2 is a Class I system with the motions developed by a Pixar animator, and SIMON is a Class VI system that uses an algorithm to autonomously exaggerate pre-programmed motions and maintain eye contact with the study participants. The different class rankings do not necessarily mean one system is superior to the other. However, in the former, the credit for an effective performance should go to the animator, whereas in the latter, the credit should go to the algorithm and its creator.

Chapter 3

Theatre as an Implicit Model

One method for accomplishing the communication is by considering the character's goals and objectives. The clichéd question heard from actors is "What's my motivation?" The reason they ask is because it is clear what they should be doing, but not why they are doing it. Much of the motivation in certain schools of acting is objective-based. Actors must constantly ask what it is that their characters want, and use that motivation to drive what they are doing in each scene in order to obtain that objective.

This is beneficial to robots seeking to emulate humans, since it is easy to formulate problems in goal-based terms. At its core it reduces down to a planning or objective-evaluation problem. A robot must decide what its objective is, and then come up with a plan on how to achieve it, figuring out along the way what the most effective way to achieve the objective is. It is easy to see the actor/robot parallelism here, since both actors and robots naturally formulate their plans in this way.

We use theatrical techniques in three ways. First, we use the movement exercise known as Viewpoints to explore the space of possible movements without regard to the context of those movements. Next, we add context into the movements by figuring how what movements are appropriate using the objective formulation described above. Finally, we use the ideas from these two exercises to formulate a larger context for the navigation problem specifically to help figure out proper objectives while navigating.

3.1 Non-contextual Movement

Teachers of acting initially train students to become more aware of the verbal and physical effects they have on one another. This heightened awareness is a necessary first step in effectively "acting" on another person to obtain some goal or objective [7]. This physical awareness is often taught through a series of exercises, known as *Viewpoints*, where students first learn to act without intent or context, simply responding to the situation around them. This is difficult for beginning students, since it is hard to discard *all* intent from one's movements. The Viewpoints exercises aim to have the students ignore the context of their motions, and instead just explore different types of "pure movement." The exploration is guided by the titular Viewpoints, each of which highlights a particular component of movement, such as gesture or tempo.

Despite its difficulty for people, the task of ignoring context and generating pure movement is easier for a robot. Robots have no intent other than to execute the program given to them. Any actions the robot performs can only going to have intent in the eyes of someone observing the robot. Furthermore, any contextual factors must be explicitly factored into the robot's program, so it is easy to leave them out.

These qualities make a robot an ideal acting partner in the Viewpoints exercises. In this section, we give an overview of Viewpoints and describe our experiences with inserting robots into a movement class in the Performing Arts Department at Washington University in St. Louis. We describe the specific exercises performed in the class, and what the robot did in those exercises to help teach the students. In addition to the benefits to the class, we also show how the work helped us gain insight to the different types of motions possible with the robot.

3.1.1 Viewpoints: A Definition

Viewpoints is an improvisational acting exercise originally developed by the dance choreographer Mary Overlie, and later adapted for stage actors by Bogart and Landau [7]. Viewpoints is "a philosophy translated into a technique for (1) training performers; (2) building ensemble; and (3) creating movement for the stage" [7, p. 7]. The exercise is designed to train students' awareness of their fellow actors and of the physical space that they are in, so that they can act responsively to one another in their environment. Students move about a large open space, performing unscripted movements, reacting to the actions of their fellow actors, and to the physical characteristics of the space. Actors with a good awareness of their environment respond more impulsively to situations on stage and, consequently, give better, more believable performances.

In acting terms, participants in the exercise explore space and time through movement. The impulses to move and explore are derived from different categories of environmental stimulus, called Viewpoints. The nine physical viewpoints can be divided into two categories, Viewpoints of Time, and Viewpoints of Space.

Viewpoints of Time

1. Tempo - The speed of a movement
2. Duration - How long a particular movement continues
3. Kinesthetic Response - When a movement is a reaction to someone else's movement
4. Repetition - How a movement is repeated

Viewpoints of Space

1. Shape - The positioning of the body (in relation to itself)
2. Gesture - A sequence of movements of a subset of body parts
3. Spatial Relationship - The distances between entities
4. Topography - The pattern of movement relative the floor
5. Architecture - How movements exist in relation to elements in the surrounding environment

Each of the Viewpoints presents a way of examining movements. Particular Viewpoints can be emphasized or diminished, but all of them are present in some capacity in every movement. Within the scope of the exercises, the Viewpoints give structure to movements when context is not present. Instead of moving with a particular contextual intent, the actor's moment-to-moment movements are created by exploring new components of any particular Viewpoint or set of Viewpoints.

It is easy to construe this example simply as a set of random movements in the space, but once they are deconstructed into a set of responses to the Viewpoints in the environment, they emerge as a clear set of actions and reactions. Actors move in response to the characteristics of the space and the movements of the other actors in time. Developing the skill of reactively responding to the (often unpredictable) actions of the other actors is one of the central aspects of the exercise.

In this early phase of training, an emphasis is often placed on "pure" movement. The goal is to perform these actions without an overt agenda, context or emotion. (i.e. not trying to play a pre-conceived role). Actors in the exercise are encouraged to respond to their surroundings confidently, without hesitation, and without consciously thinking about the response. Two of the common pitfalls that novice actors fall into are coming to the exercise with a pre-planned agenda ("I'm going to be energetic today!"), and reading an agenda into another actor's actions ("She looks like she's escaping from something, so I'll follow her."). This causes the actor to choose actions that are not completely based on his surroundings but, rather, on some notion of what the "right" action for the role is. Failing to truly react to the environment on stage leads to a forced, stilted performance.

The pure movement is movement for its own sake; a physical language of exploration. This agenda-less exploration accomplishes several important goals. Most importantly, it fosters a wider range of physical expression than would be possible if the actor were performing in some pre-specified context. It also gives a specific language with which to discuss different types of movement. In more scientific parlance, each viewpoint is an axis along which specific movements can be placed. As Bogart and Landau said, "Viewpoints leads to greater *awareness*, which leads to greater *choice*, which leads to greater *freedom*. Once you are aware of a full spectrum... Range increases. You can being to paint with greater variety and mastery." [7, p. 19]

3.1.2 The Robot Acting Partner

Since the essence of Viewpoints is to explore motion without an agenda, a mobile robot is an excellent teaching tool and acting partner. If the robot behaves according to a set of reactive rules, it has no intent, other than that implicit in the rules. The lack of body language and expression makes it hard for the actors to infer intent, whether it exists or not, in the way that they can do with other humans. In acting terms, this represents "pure movement", the ultimate goal of the Viewpoints exercise.

Chris Wilson was the first to integrate the robot into the Viewpoints exercise, prior to the work described in this thesis. Students were briefly introduced to the robot as "an expert in Viewpoints" before it started moving, and were told to interact with it in the same way that they interacted with their fellow actors. No technical details of the robot were given. In the first demonstration, the robot was teleoperated by Wilson, which provided useful preliminary results, but since the human's intent was still present and the robot lacked autonomy, the results were ultimately dissatisfying.

In work within the scope of this thesis, we used an Erratic robot to perform the exercise with a smaller group of four students, each experienced with Viewpoints. In this demonstration, the robot was completely autonomous and operated without human intervention. The robot relied on its laser scan data and odometry to interact with the other actors and its environment. Based on the model of the space gathered from this information, the robot performed a number of simple reactive behaviors that corresponded to the Viewpoints concepts of *architecture*, *shape*, *floor pattern*, *spatial relationship* and *tempo*, and determine where the robot moved and how it moved there. Behaviors were sequenced probabilistically, based on the proximity of humans, embodying *duration* and *kinesthetic response.* Some of the behaviors were closed-loop and did not use the sensors (such as standing still or spinning in place, for example), while others were open-loop and responded to changes in the environment (stopping when close to an actor or following them, for example). The robot participated in three separate exercises with the human actors, each lasting approximately five minutes.

3.1.3 Robots and Viewpoints

This project served two distinct purposes. The first purpose benefited the theatre side of the collaboration by using the robot as a pedagogical aid for instructing beginning actors. By having the robot perform its autonomous movements, it forced the actors to truly react to its movements in the moment, since it moved different than they were used to.

The second purpose benefiting the robotics side of the collaboration helped to explore the full range of movements available to the robot and give us a concrete language with which to discuss them.

Much as the goal for the students in the class is to explore different types of motions, the Viewpoints exercises gave us a chance to explore movements for the robot. Wilson's previous Viewpoints robot was limited by having no variability in speed, other than on and off. This highlighted the robot's *tempo* limitations. While this was fixed in later versions, we still discovered additional timing-based limitations that we had not previously considered. In the initial autonomous tests, the robot switched which activity it was performing rather frequently, based on a probabilistic model. This resulted in many short movements, and as a result, the robot generally stayed confined to one small area. This highlighted a *duration* limitation.

The experience with Viewpoints led to our development of expanded definitions of each of the Viewpoints in ways that simultaneously apply to robots in non-theatrical settings and can be calculated empirically.

Tempo

The concept of speed of execution is the most familiar to roboticists, and does not require much elaboration here. Tempo covers the velocity, acceleration and jerk profiles of the robot, pertaining to all degrees of freedom (i.e. both translational movement as well as rotational).

Duration

Duration is related to acceleration ultimately, since changing from one movement to another will result in an acceleration spike. In more concrete robotic terms, it can also refer to how often replanning (and the subsequent change from one plan to another) occurs. The separation of sequential actions can be framed by the differences between a robot with traditionally "robotic" motions (i.e. doing the robot), with many short discrete motions, and a industrial robot arm, which moves fluidly from one pose to the next.

Kinesthetic Response

We consider the kinesthetic response to encapsulate how quickly and to what extent robots react to stimuli. For contrast, consider a situation where the robot's path is suddenly obstructed by an obstacle. On one hand, the robot could stop suddenly as soon as its plan is disrupted regardless of whether a collision was imminent. Alternatively, the robot could continue along, altering its path slightly to avoid the obstacle.

Repetition

Repetition can fall into two categories. If the robot repeats/mimics the movements of the people in its environment, the robot can learn socially acceptable behavior. If the robot repeats its own motions, it creates a predictable model of behavior for the robot.

Shape

Shape translates to the robot's joint pose. For a robot with arms, this marks the difference between a robot moving with a neutral pose, to one with its arms extended in a Heismann-esque pose, or perhaps a robot that carefully tries to use its arms to needle through a crowd.

Gesture

Gesture is generally well defined in robotics, and can pertain to where the robot directs its gaze and how the robot moves its arms throughout a movement.

Spatial Relationship

Many roboticists have already begun to consider spatial relationship, drawing on studies in *proxemics*.

Topography

This feature is most closely related with path planning, and the path the robot takes. It can include things like curvature.

Architecture

The traditional view of navigating views the world in terms of obstacles and free space. The architecture element opens the possibility for giving special consideration to architectural features, such as doors, hallways and other such features.

3.1.4 Transition

After enumerating these possibilities, it becomes clear that our robots are very limited in what they do. Using Viewpoints has helped expand the physical vocabulary of the robot, allowing us to be aware of places where the robot is only using one variant of a Viewpoint, and begin to explore other options.

However, in order to decide which of the motions within the Viewpoints is best, we must first reintroduce context into the planning process.

3.2 Contextualized Movement

Once an actor learns all the different ways that they are able to move, they must develop the skill of choosing among the different options. In the school of acting that we discussed earlier, that choice is based on accomplishing a given objective. The character's objective informs *how* actions should be performed, and is derived from both the character and the context in which they are placed. The objective forces the actors to need something specific and to have a goal. These elements give the actor a new form of structure to guide physical interaction: movement becomes more than pure stimulus/impulse/response in time and open space.

In this section, we explore the use of robots in theatrical settings where they must use purely physical motions to express information and their objective. Some of our previous work has looked at generating communicative robot behavior for a variety of media [55, 57], the most noteworthy and relevant is the work we did directly in collaboration with our performing arts collaborators.

3.2.1 Composition of Forgiveness

The primary work in our contextualized movement research focused on a presentation in September 2010 at a joint colloquium between the performing arts and computer science departments at Washington University in St. Louis. The centerpiece was a short movement piece, where a robot and a human interact on stage with the goal of conveying a meaningful story just through their physical actions. In the particular scenario we focused on, entitled "Forgiveness," the context is that Actor A has recently betrayed Actor B. The physical actions in the piece are motivated by this scenario. Actor A's objective is to seek forgiveness, while Actor B wants an apology. An initial version of this work was done in conjunction with Wilson's thesis[106].

The piece was first composed and rehearsed using two human actors, with one of the actor's motions constrained to movements which our robot, a B21r, could also perform. Thus, the human actor was limited to moving around the room and directing his gaze by moving his head, and could not do such things as gesture with his arms or change his facial expression.

In pieces such as these, three things dictate an actor's movements: the given circumstances/context, their objective and the movement of their partner. For instance, if an actor is seeking forgiveness, he might move towards his partner in an effort to appease them. As a response to this movement toward, the partner may look away from him in an effort to reject his attempt at appeasement. His response then may be to move away from his partner and sit down in order to give the partner some time and space. This physical "talking and listening" comprised the "score" of the movement piece. Each actor must try to achieve his objective. The piece can end in one of two ways: (1) Actor A is forgiven, (2) Actor A is not forgiven. Through the rehearsal process, the motion score was tweaked and refined, ultimately arriving at the set of movements and objectives shown in Table 3.1.

To incorporate the robot, the actors' performances were video recorded, and the timings of the actions were coded. Thus far, the process mirrors that done by Wilson. However, the performance the robot gave in Wilson's work was inflexible, since the robot used the exact same timing in every performance, regardless of what the other actor did. This forced the other actor to constrain their performance to fit with the robot's preprogrammed timing, thus taking away from the essential interactive components of the performance. To quote Grotowski,

> Theatre is an encounter. The actor's score consists of the elements of human contact: "give and take." Take other people, confront them with oneself, one's own experiences and thoughts, and give a reply. In these somewhat intimate human encounters there is always this element of "give and take." The process is repeated, but always *hic et nunc*: that is to say it is never quite the same. [34, p. 212]

Even the correct actions are useless if their timing is wrong. It is this timing that breathes life into a performance, and makes the interactions in it seem organic. In this vein, we introduced an element of variable timing into the robot's performance, where a human offstage would observe when certain actions were taken by the human actor and send a signal to the robot marking the event. This takes the place of a more robust set of sensors that could signal these actions autonomously, but ultimately was not the focus of our research. This allowed for the human actor to react freely (within the score of the piece) to the robot's actions. The use of this minimal Wizard-of-Oz controller enabled the robot to transition from being a Class I actor to a Class IV actor, according to the ontology described in Section 2.4.

Physical Action	**Objective**
A starts to move forward.	to make B feel invited
A stops.	to make B feel powerful
A starts toward B again.	to make B feel forgiving
B sits up.	to make A feel warned
A moves to B's (left) side.	to make B feel justified
B gets up and crosses to stage left, facing away.	to make A feel rejected
A turns to face B.	to make B feel urgently needed
A moves to be behind B.	to make B feel challenged
B turns around to face A.	to make A feel confronted
A "steps" forward.	to make B feel acknowledged
B crosses back to the chair and sits, facing stage right.	to make A feel rejected
A looks down.	to make B feel sympathetic
B looks down.	to make A feel considered
B sits up, turns to sit forward	to make A feel hopeful
A turns toward B.	to make B feel important
B leans forward.	to make A feel unsure
A moves to her left side again, looking forward.	to make B feel supported
B sits up.	to make A feel encouraged
B places hand on A's back.	to make A feel loved
A looks at B.	to make B feel forgiven
A looks straight forward again.	to make B feel resolved

Table 3.1: Forgiveness Movement Score - The part of A was played by both a human actor and our robot, while the part of B was only played by an actress. At the start of the scene, A is up stage right, with B sitting in a chair center stage, leaning forward.

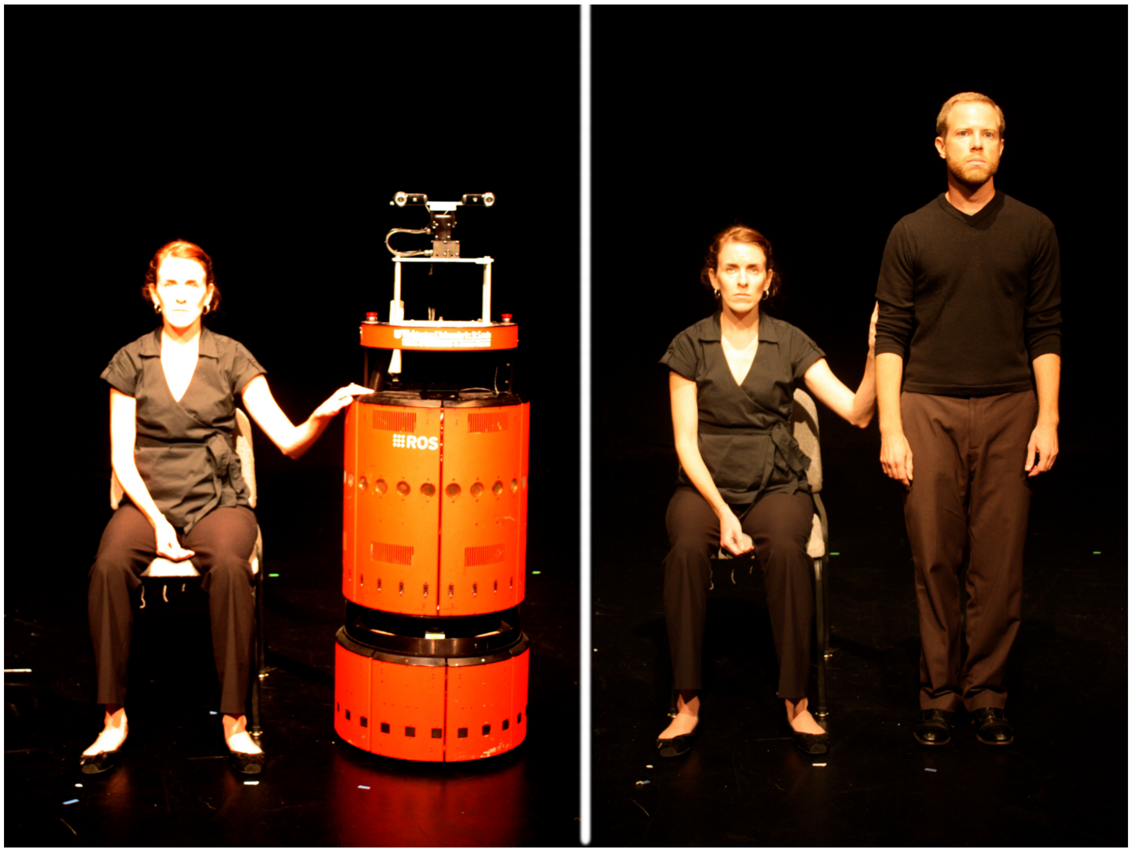


Figure 3.1: The same scene from two performances, one with one robot and one human actor (left) and one with one two human actors (right). The full human/robot performance can be seen at `https://www.youtube.com/watch?v=-FyK5tVuaKk`

3.2.2 Performance

For the actual performance at the colloquium, the piece was then performed twice: once with one human and one robot, and then again with the two human actors. The audience was not informed to the specific context of the movement piece prior to watching it for the first time. Subsequent to watching the human/robot performance, there was a question/answer session with the audience, led by the director of the piece. Through this discussion and informal questioning, we discovered that the audience was able to correctly infer the characters' intents. The robot was also able to generate a degree of sympathy from the audience, with many audience members feeling bad when the robot dropped its 'head' in disappointment during the piece, as indicated by the discussion and audible cues during the performance.

When the audience then watched the version of the piece with two human actors, they were still able to see the context of the piece, however, many in the audience liked this version *less* than the human/robot version according to the discussion after the second performance. One of the key stated reasons for this difference was that the human actor was using a smaller percentage of his full faculties as an actor while doing the same actions as the robot. Many found the human actor not using his arms or facial expression to be off-putting, making something seem missing from the performance, whereas the robot was using all of the degrees of freedom available to it, thereby making the same motions seem like a fuller performance.

3.2.3 Lessons

There are several points that we take away from this work that are applicable to our non-theatrical robot research.

First, this work underscored for us the need to have *every* action backed by a specific objective. As discussed in the previous section, there are a wide range of possible actions, even for robots as constrained as the B21r. Choosing among those actions requires strong consideration of the context and the objective. Even small differences in the sequencing of actions can change the perceived intention. Turning and then driving forward is a more deliberate action, communicating the robot's intended direction of travel before moving, whereas driving forward while turning prioritizes moving toward the goal as the objective. Too often these little changes are overlooked in robotics because they are minor, or because they do not contribute to the overall efficiency of the robot, but once placed into a social context, they can carry a great deal of meaning.

While the physical score specifies much about how the robot was to move about the stage, performing those actions in any fashion would not have been enough to replicate the expertise the actors brought to crafting the piece. The other key component was the precise timing and spatial relationships that the actors enacted with each performance. The physical actions listed contribute *what* the actor is doing, but not *how*. The precise manner in which the actions are performed is instead drawn from the objective. The variance in possible paths for the robot takes the shape of varying the different Viewpoints of each action. For instance, moving toward the other actor can be penitent or aggressive, depending on the *tempo* and the precise *spatial relationship*. Having an actor direct their gaze at the other can make the other person feel threatened or curious depending on the *duration* and quality of the *gesture*. It is the actor's job to take the context of the scene into consideration to chose precisely *how* to effectively convey their objective.

We also noted the effect of limiting an actor from their full potential. The audience thought less of the human's performance of the robot part because it did not use all of his physical capabilities as an actor. The argument could be made that the audience had a less-than-complete theory of mind for the actor in this performance, because they only utilized their legs and neck. This leads to questions of the form, "I understand what he is doing, but why is he not doing X." The absence of some types of behavior move the entity further from understood theories of mind, and thus diminishing the observer's capability to interact with the entity. The implication for robot behavior is clear, in that there is the potential for robots with unused actuators to be viewed as unintelligent, or at least to fall short in the potential interactions.

3.3 Theatre as a Structure

One clear key feature of theatre that we can take advantage of is dramatic structure. Theatre should engage the audience, draw them in and elicit a lasting response from them at the conclusion. This makes it a complete experience, due in part to its inherent structure. Traditionally, the dramatic arc of theatre involves four distinct parts: the initial exposition, rising action, the climax, and the resolution. In a manner similar to Laurel's exploration of "dramatic potential", we can see how human robotic tasks can be structured into these distinct parts as well. In both cases, the goal is to create a cohesive interaction, where one action follows logically after the previous ones. If this experience is broken, then it creates a jarring context switch that takes the audience/user out of the interaction, lessening the overall quality of the experience.

In a play, the initial exposition is when the setting and characters are introduced. In Laurel's terms, this is when "the potential for action in that particular universe is effectively laid out" [51, p. 64]. This is analogous to the initial introduction of the robot and human in HRI settings. The two must meet and become acquainted with each other. In particular, the robot, which the human is likely to be unfamiliar with, must be very explicit in introducing its "character" and what its "potential for action" is, i.e. what it is capable of doing and what it is likely to do. In this phase, it is crucial to explain the context and set the expectations for the rest of the interaction.

Then, as the play progresses, the characters' objectives and how they plan to accomplish their objectives is revealed, hopefully continuing in the same vein as the potential laid out in the exposition. This sets up the primary conflict, and results in an increase in tension, called the rising action, leading to the other characters (and the audience) becoming more engaged. In HRI, the goal is not to have rising tension, but to increase the engagement by making each action causally follow the previous one, with the actions building on one another. If this causality is broken, the direction of the interaction becomes unclear, and becomes less effective as a result.

Ultimately, the dramatic arc reaches its climax. At this point, the characters' objectives are met, and the only thing that remains is a short resolution to close the piece. A similar moment happens when a human and robot achieve their goals, and then part ways after the interaction.

Consider the robot receptionist installed at Carnegie Mellon University [33]. Engaging interactions with such a character need to have the four parts of the dramatic arc. The robot's actions at the beginning need to have clear clues as to what capabilities it has. If it initially looks at people when they arrive, it necessitates that the rest of the interaction should include eye contact. The exposition for this interaction specifies what modes of communication it has and is willing to accept. The robot receptionist does this through locating people, greeting them and prompting them to use the keyboard to type to it. As the interaction progresses and the interacting person specifies their goal (to obtain directions, get the weather forecast, etc.) the robot attempts to keep them engaged by providing the requested information, telling stories and avoiding the default response. These all help the user maintain the belief that they are interacting with a social actor. Then, once the goals have been achieved and the climax reached, the robot and human must conclude their interaction, usually with the human walking away and the robot bidding them farewell. This allows the interaction to gracefully be resolved, and finish appropriately.

3.3.1 Three Act Structure for Navigation

One particular context that is not taken into account in navigation is the robot's progress on its task. Consider the navigation task of moving from one location to another. In order for a robot to navigate in dynamic environments with uncertain elements, navigation algorithms are written to allow the robot to continuously re-plan and change its high-level behavior (the global path) almost instantly. This creates relatively robust behavior, but does not take into account how people viewing the robot's behavior may interpret it. This behavior implements an implicit Markov assumption, in that the robot only takes into account its current state, rather than also including its progress through the action.

One way to explore this context is to confer every physical action the robot performs with a narrative arc, specifically that of the archetypal three act structure. The first act is the exposition, in which the characters and setting are established. Second is the rising action, the main course of action in which the protagonist faces multiple obstacles as they move toward their goal. Then finally, the protagonist will arrive at a point where there is only one possible scenario: the climax in which we see the protagonist either achieve or fail to achieve their goal.

To properly use this structure with a navigation task, we must first define what is meant by "goal". For navigation, it may refer to simply the goal pose of the robot, or the goal pose with some constraints on how to get there (no collisions, minimal path length). In a theatrical context, the term for goal is objective, the motivation behind every action the character does. In a play, if a character crosses the stage, it might be to move away from someone to make someone feel isolated. These objectives are always posed in relation to others and not in isolation. Hence, the robot's objective cannot just be to move from place to place, but to move from place to place in relation to others around it. (Note, in this discussion we will use "objective" to refer to the motivation, and "goal" to refer to the desired pose.)

In the exposition, we need to establish the robot's "character," i.e. what it is capable of and likely to do. In most scenarios, it is impossible and impracticable to endow the robot with as much character as traditional dramatic characters like Austen's Mr. Darcy. Instead, the aim is merely to introduce static qualities of the robot that will be present throughout the action as a way of providing information to help people predict what the robot will do. This could mean exploring the different modalities of the robot (i.e. the different ways the robot can move/act). Establishing this is important, even if the modalities are not functionally necessary, so that if/when the robot employs these behaviors later, they do not come as a surprise to the audience. Furthermore, establishing the type of movements the robot will perform can also be beneficial. Consider the difference between a robot that starts moving in a straight line to its goal, and one that moves more erratically. An observer may think the latter may need more attention or that the former is more deserving of trust. Not only does introducing these qualities early on have the benefit of helping predict future behavior, but also molds an observer's vital first impression.

The first act is also where the robot will begin to move toward its goal, which may require some preparation. The start of the action must be done in a way that is consistent with the robot's objective. For many robot navigators, the objective is simply to move towards the goal. However, in social navigation, the objective includes moving toward the goal in a way that does not disturb the people around it or cause them to be uncomfortable. For some large robots, the simple act of them starting to move their bulk toward the goal can be unquieting. One way around that is to use the additional modalities of the robot besides its mobile base to indicate that the robot is about to move. This could entail moving the head around to ensure the area is clear or a slight raise in the torso to indicate imminent action. This sort of anticipatory gesture is also suggested by Van Breemen [104] and Mead and Matarić [65].

During the middle act, i.e. the bulk of the movement toward the goal, the robot's objective must be to move to the goal, deal with unforeseen obstacles it encounters along its planned path, and to make people aware of those activities in a way that makes them continue to be comfortable. Importantly, the robot should react to the obstacles during this middle act in a way appropriate to the context of the action as a whole. The robot should not stop completely and act as though it were planning a brand new motion from the beginning again. It should react in a way that indicates that it is still pursuing the same goal while taking into account new information about the obstacles. Similarly, the scale of the reaction to unforeseen obstacles needs to be adjusted based on when it happens. One would not expect a robot to react the same way to a change in plans at the beginning of an action than at the end when it is almost at its goal.

The relationship between the robot and the people in the environment is centered around the idea of legibility, i.e. making the robot's actions clear and readable[24, 93]. Legibility is particularly important in this middle act for ensuring a smooth transition between the robot's initial goal and the ultimate outcome of that goal, since illegible behavior could be read as not acting toward that goal. Certain people in the environment require extra consideration since the robot is actively moving in the same space as them. As a result, the robot must take into account contexts related to them, their objectives, and the robot's relationship to those objectives. A person should be treated differently than other mere obstacles, in that they are often mobile and have personal space which it is better not to enter. The person may be particularly sociable and want explicit interaction with the robot. On the other hand, if the person's sole objective is to get to their destination as quickly as possible, then the robot may need to adjust its behaviors for that. If the robot's only objective is its own goal, then it can ignore the other person's objective and move in a way that my detrimental to their goal. However, if the robot's objective does include other people's objectives, then it should move in a way that enables both to complete their objectives.

In the final act, it becomes clear whether the robot has achieved its goal or not. If a robot just stops, it is difficult to determine whether the robot is actually at its goal position or whether there is a problem, especially if the goal is unknown to the people observing the robot.[9] Thus, adding cues like looking at the goal or changing the robot pose in some manner will help indicate the final outcome, making its behavior more legible. One particularly useful approach that has not seen common-place usage in navigation tasks is including success and failure animations to explicitly mark the outcome of the action[99].

3.4 Conclusions from Theatre

Why do theatre? Not, why is there theatre in this thesis, but why do theatre at all outside of the computer science department? Theatre is about examining assumptions in the so-called real world by examining the lives of characters explicitly created to raise those questions. Theatre is the allegorical "mirror-to-society" that allows people to focus in on interactions and learn from them, perhaps even empathize with them. As a result, we begin to learn about our own place in society.

[9]This is especially true while testing new iterations of navigation algorithms. Spoken from experience.

Our goal with injecting theatre into robotics mirrors this purpose. Regardless of the technical contributions they can help with, the discussions of the theoretical implications of theatre and robots allows us to come to a better understanding of the very idea of human robot interaction. Grappling with the idea of whether robots can do what humans can do on the stage has challenged our fundamental motivations for programming robots.

As roboticists our mindset is often focused on algorithmic deconstructions of problems working through the process of how we can make a particular thing happen. Many times these are things that humans can instinctually do. Theatre artists can as a result focus on the end product more clearly and have an entire language and structure with which to talk about it. It is not a question of roboticists not being able to focus on these things, but it is not the training that most go through.

In the rest of this work, we will integrate two of the fundamental building blocks that we have gained. First, the explorations with Viewpoints has given us the language and structure to talk about the possibilities of movement. Secondly, we will use the idea of context to motivate which of the myriad movements to select.

3.4.1 Collaboration

Above, I have discussed many of the benefits that we, as roboticists, have gained from the collaboration with artists of the theatre. However, thus far I have not mentioned any of the benefits this work has had on the theatrical side of things.

One fundamental benefit has resulted from the many discussions about robots and theatre, which is forcing our theatrical collaborators to challenge their own assumptions about why they do the work that they do. Many times the discussion has not changed their beliefs but made them clarify precisely what motivates them to create the art that they do.

Another benefit has been on the pedagogical side. Based on informal discussions with them afterward, interaction with the robot proved to be a valuable experience to the students of movement that interacted with the robots in the Viewpoints exercises. The exercise with the Forgiveness piece has also inspired an assignment in Pileggi's classes where the goal is to perform movement pieces in the same way as the robot to entice students to put clear motivation into every one of their physical actions.

Chapter 4

The State of Contextualized Navigation

Navigation is one of the fundamental tasks in mobile robotics. For robots with reasonable dynamics and operating speeds in indoor environments, efficient collision-free navigation is considered a solved problem from a practical standpoint. Navigation algorithms can process large amounts of sensor data to keep track of the locations of obstacles and free space with great accuracy. Such an algorithm is fine for many use cases, or for navigation in the abstract, if all that matters is getting from point A to point B. However, many navigation algorithms suffer from the same problem: the algorithms optimize based on the single constraint of finding efficient collision-free paths.

Such single constraint algorithms are insufficient when humans are introduced into the environment. People must be treated differently than static obstacles in order to respect their social norms. For example, driving a few centimeters away from a table is perfectly fine in most cases. However, driving that closely to a person is socially undesirable. There are many additional scenarios, beyond respecting people's personal space, where choosing the shortest collision-free path may not be optimal. Given information about where people often are, a longer path that avoids probable obstacles may be preferable. The robot must also consider the utility of entering potentially hazardous areas, such as kitchens, which are valid paths, but they come with a cost. Even simple factors like driving on the right side of a hallway will need to be considered. Which path the robot takes will depend on having additional information about the larger contexts.

However, each additional factor adds a layer of complexity to the algorithms, and there has been no general algorithm for introducing these constraints in a structured way.

In this chapter, we outline previous work in contextualized robot navigation, and discuss the platform with which we will be working and its limitations.

4.1 ROS Navigation

4.1.1 Why ROS Navigation

In this research, we focus on the ROS [83] Navigation Stack for a variety of reasons. First and foremost is its ubiquity. It runs on dozens of robots,[10] despite the variety of locomotion methods. The Navigation Stack runs on numerous wheeled platforms, as well as several walking and flying robots. Due to the generality of the interfaces, regardless of the physical actuation. This flexibility is shown on the opposite side as well, being able to support numerous types of sensor input as well, including planar sensors like laser range finders, and three-dimensional scanners like the Microsoft Kinect. The ROS Navigation Stack also has the ability to plug in different planners that conform the given interface. Leveraging this system gives us the opportunity to build upon a robust navigation platform, already equipped with reliable collision avoidance. Furthermore, due its ubiquity, working with this platform gives the advances we make in this area the potential to have a larger impact than if we created a new competing system.

4.1.2 Algorithmic Design

The ROS navigation stack implements the dual costmap navigation algorithm, the general structure for which is shown in Figure 4.1. All of the work in this dissertation works within the design of this algorithm, and is backed by one particular implementation of the algorithm based on ROS.

The algorithm receives a goal pose (x, y, θ) from an external source. That goal is input into the *global planner*, which plans a path from the robot's current position to the goal. The resulting *global path* is a series of points between the current position and the goal. However, the global path is an entirely geometric solution of how to get to the goal, when what the

[10]`http://wiki.ros.org/navigation/RobotsUsingNavStack`

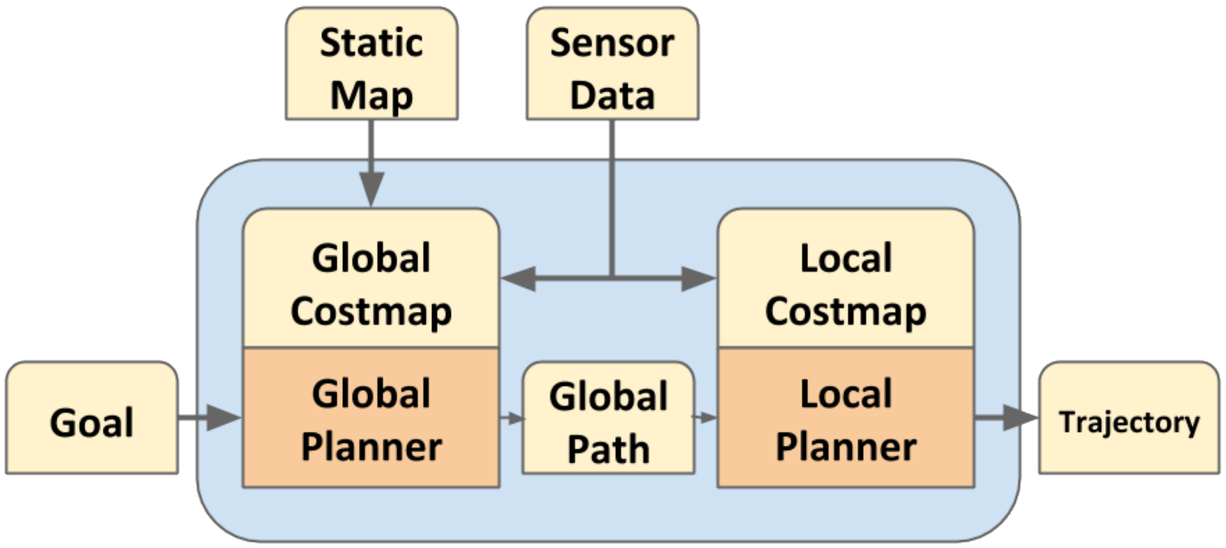


Figure 4.1: Structure of Dual Costmap Navigation Algorithm

robot ultimately needs is a solution with respect to space and time. The job of the *local planner* is to translate the global path from geometric points to a command trajectory to send to the robot's motor drivers, a process that is constantly done to create a stream of commands that will be executed until the robot reaches its goal.

The implementation of the two planners can be done in a myriad of ways, however, one commonality of most of them is the use of a *costmap*. The costmap is a two-dimensional grid representing the robot's environment, where each cell has a cost for traveling through that area of the environment. In its simplest form, the costmap only has two values in it: a high value that represents cells which the robot cannot be in, and a low value for cells which the robot can be in. Since the high values will result in a collision between the robot and an obstacle, such values are often called *lethal* in the costmap. Throughout this dissertation, we will discuss how the values in the costmap are calculated and additional schemes for what the values can represent.

The global planner generally maintains a costmap that represents the robot's entire world, i.e. all the places it can travel, such as the entire floor of a building. Then, by using the costmap as a locally connected graph, the global planner can use Dijkstra's algorithm or some similar shortest-path algorithm to create the global path from the robot's position to the goal.

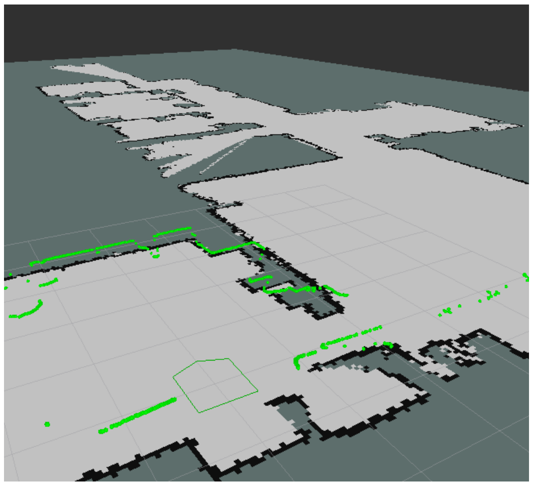
The local planner maintains a costmap as well, which covers only a small area around the robot. While the specifics of how the local planner is implemented can vary, the costmap will generally be used to evaluate potential robot trajectories to determine if the robot will drive into any of the obstacles recorded in the costmap if it takes a given trajectory. The generated trajectory must be compatible with the robot's capabilities. For holonomic robots, i.e. robots that can move in any direction (forward/backward, left/right, clockwise/counter-clockwise) the trajectory will be of the form (x, y, θ). Non-holonomic robots will not use all three of those components. For example, robots that cannot move laterally will always have the y component set to 0.

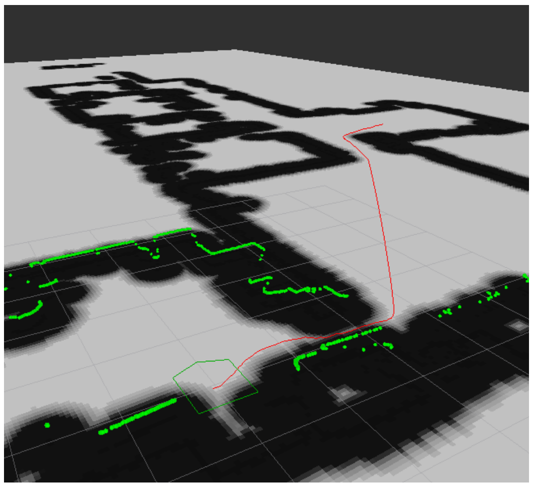
4.2 Navigation in Practice

To better understand the existing elements of the dual costmap navigation algorithm, let us walk through a short example of how the system works when implemented in ROS. First, Figure 4.2a shows the two data sources for our costmaps, the live sensor data and the static map. The sensor data here comes from a planar laser range finder, giving the robot a sense of how far obstacles are away from it. The static map can be built from this same data, but is compiled prior to running the navigation algorithm, using a mapping algorithm such as AMCL. Note that this static map contains three values, for lethal, clear and unknown cells.

Both of these data sources are integrated into the global costmap. The values from the static map are copied directly into the global costmap. The sensor data provides two types of information. First, it demarcates where obstacles are (e.g. the endpoint of a laser beam), allowing us to place lethal values into the costmap. It also demarcates where obstacles are not (the space between the sensor and the endpoint of the laser beam), allowing us to set the value of cells between the obstacle and the sensor to zero.

Integrating the data sources into the costmap as in the previous paragraph is compatible with a wide variety of planning algorithms. However, the next step uses a specific semantic convention for ROS planning algorithms. The global planning algorithm calculates a set of points where the center of the robot will be. One method for determining whether a particular location is a safe place for the robot to be is to look at all cells in the costmap that overlap with the footprint of the robot. This is computationally intensive, which is why ROS planning algorithms use the convention that given all locations such that if the


(a) Costmap Data Sources


(b) Global Costmap and Plan

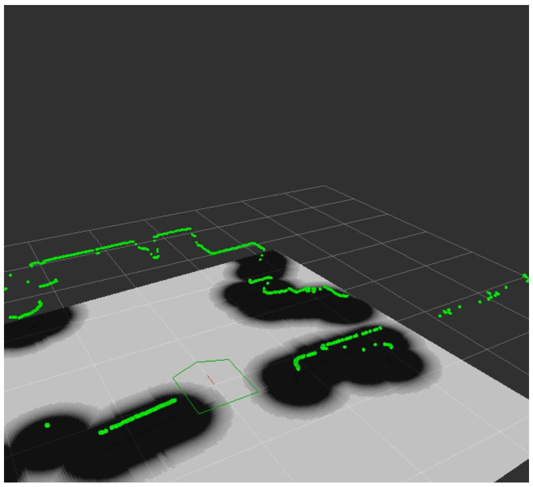

(c) Local Costmap and Plan

Figure 4.2: Dual Costmap Algorithm in Practice - Figure (a) shows the two basic data sources for the costmap, the static map and the live sensor data (shown in light green). The robot's outline is in dark green. Figure (b) shows the global costmap generated from the two sources as well as the global plan to reach the goal (red line). Figure (c) shows the local costmap and its initial local plan (small red line).

center of the robot were there then somewhere on the robot would definitely be in collision, the value in the costmap at those locations is also a lethal value. This process is known as *inflation*. Using this convention, the global planning algorithm can just check the cells in the costmap for where the center of the robot will be. The ROS implementation also adds some non-zero non-lethal costs around the obstacle to discourage driving very close to obstacles. Those costs are shown in Figure 4.3.

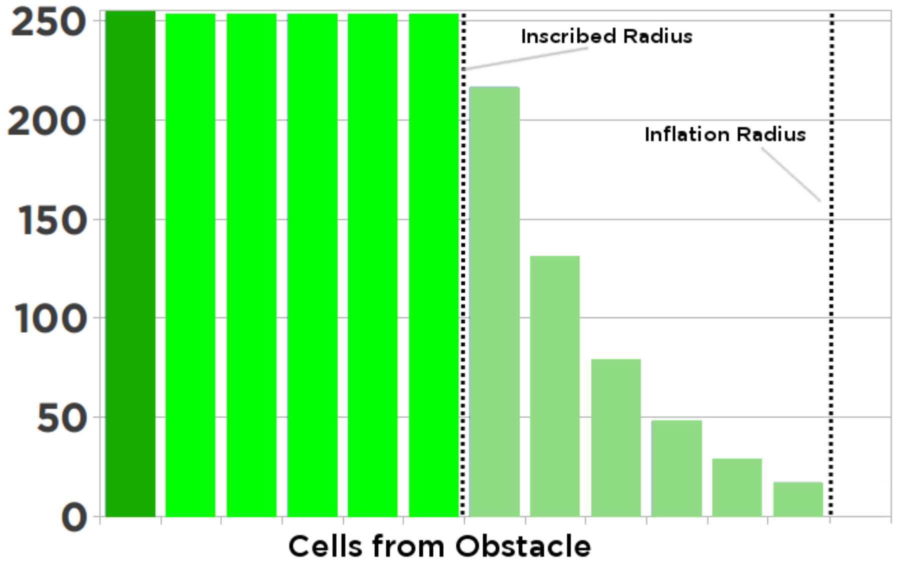


Figure 4.3: Inflation Cost Function - When an obstacle is detected, the associated cost in the costmap is set to the highest lethal value (254). The adjacent cells where the robot would be in collision (i.e. cells that are less than the inscribed radius of the robot away from an obstacle) are assigned a slightly lower lethal value (253). Cells slightly further away are assigned a non-lethal cost that exponentially decreases as the distance to the obstacle increases. The distance that the non-lethal costs extend out is a configurable parameter known as the inflation radius.

The result of combining the sensor data and static map, and then running the inflation step is shown in Figure 4.2b. Note that the use of the static map creates a plan that avoids obstacles that are currently out of the robot's sensor range.

The local costmap, shown in 4.2c is built in much the same way, except that it does not use the static map as a data source. It instead only relies on the sensor data. Since the local planner is only going to check the results of a very short trajectory, the local planner only needs to maintain a costmap for the area immediately surrounding the robot.

The local planning process used commonly in ROS is based on the Dynamic Window Approach[28] (frequently abbreviated DWA). It generates a collection of possible trajectories using the robot's position and velocity, based on its dynamics, and then chooses a trajectory based on the optimization of some cost function. In this example, the cost function is a combination of how close the trajectory moves towards the goal pose, how far it is from the global path, and how well it is aligned such that it drives forward.

4.3 Related Work

The problem of navigating with respect to additional contexts in the past has primarily focused on navigating [1, 74, 108] or localizing [100] robots in the presence of people.

However, most systems effectively ignore the problem, treating all obstacles, human or not, equally (see, for example [62]). This is a relic of using Occupancy Grids[64, 68] instead of costmaps, where each grid cell is an estimate of whether that location is free or occupied, and not a continuous cost. Konolige [48] and Thrun [102] improved the probability model of Occupancy Grids to better localize obstacles, but it was not until sensors got more accurate (i.e. not sonar-based) that costmaps came more into use.

Recently, there has been an increase in the number of people using more complex costmaps to model human obstacles in a more socially-aware way, mostly for modeling personal space. Using the full range of values in the costmap is vital for representing many social constraints for navigation algorithms. They represent general preferences or guidelines rather than hard and fast rules. As Kirby et al. [44] observed, "Human social conventions are tendencies, rather than strict rules," This approach discourages a subset of paths without disallowing them outright.

Dautenhahn et al. [22] constructed recommendations for planning motions where humans would be comfortable based on live HRI trials, taking proximity, visibility and hidden zones into consideration. These were formulated into a costmap system by Sisbot et al. [93], creating a Gaussian based "human aware motion planner." Kirby [44, 45] used an algorithm that in addition to minimizing path distance and avoiding obstacles, modeled proxemics and behaviors like passing on the right into the costmap, also using Gaussians.

Work by Svenstrup et al. [96] created even more complicated models of personal space, integrating a mixture of four different constraints modeled as Gaussians. [97] expanded this work to maneuver among a field of multiple people while moving toward a goal. Soft constraints are occasionally used for other fields like autonomous vehicles. Ferguson and Likhachev [26] used large constant valued areas to favor driving on the right side of the road and to avoid curbs. Among the latest work in this field, Mainprice et al. [60] have expanded their original model to a three dimensional costmap in order to control positioning during hand off tasks, taking safety, visibility and the human's arm comfort into consideration. Scandolo and Fraichard [89] have also created a complex model that included proxemic, visibility and motion models as Gaussians, and "interaction areas" as constants. Lu and Smart [58] also have modified robot behavior using intermediate costmap values to improve the efficiency of human task completion.

Algorithms that *do* adjust their navigation in people-specific ways take a number of forms. The patrol robot studied by Hayashi et. al [36] is an excellent example of a robot that uses different paths and gaze behaviors around people in order to affect the interaction. However, that effect was only studied using subjective survey-based measures, and did not look at the resulting efficiency of the interaction.

There is also work that views pedestrians as dynamic obstacles, which the robot then uses to create a spatio-temporally optimal plan [80]. However, this work assumes fixed human trajectories, and thus does not take into account how the humans might react to the robot's behaviors as it approaches them.

In addition to manipulating the path, many robots also use gaze to communicate information about the task [29]. Gaze communicates “joint attention” and helps establish mutual understanding of the scope of the interactions[95]. Such gazes can also help to resolve ambiguities that are not otherwise communicated [87], and can help the human decide on an appropriate action to take. Gaze has also been shown to affect a person’s proxemic preferences around a robot [98], resulting in people varying their approach distance to a robot based on the robot’s gaze.

Most of the obstacles added to the costmaps follow Gaussian distributions or constant values (with the minor exception of the representation of the intimate personal space in the work by Scandolo and Fraichard [89]). Little to no discussion is given about how the authors of the previous works found the parameters that worked best with their system.

For a more exhaustive examination of all human-aware navigation algorithms, please see the survey paper by Kruse et al. [49].

Chapter 5

Towards More Efficient Navigation for Robots and Humans

In this chapter, we examine an experiment which aimed at showing how we could improve navigation interactions between a human and a robot by making subtle changes to the robot's behavior. While we were successful in changing the robot's behavior, it did not have the significant effect on the interaction that we had expected. We describe our motivation behind the study, how it was implemented, and the experimental design and results.

5.1 Improving Navigation Interactions

We must first define what we mean by *improve*. For the purposes of the work reported here, there are three ways to improve an interaction: (1) to increase the efficiency with which the robot performs the task in which the interaction happens; (2) causing the human's subjective perceptions of the robot to become more positive; and (3) to increase the efficiency with which the *human* performs whatever task they are engaged in. This third aspect is one that is often neglected, but is one of the main reasons that we want people to interact with robots; to make the humans more efficient at what they do.

In this work, we consider a robot navigating in a hallway with a person as seen in figure 5.1. As a robot and a person approach each other in a hallway, it is unclear what the robot should do in order to most efficiently pass the person. When two people pass each other, they have a shared body of implicit knowledge about social situations, and swap a multitude of subtle social cues in order to manage the interaction. Both of these are typically missing in the human-robot setting.

If the robot used its standard navigation algorithm, the results are socially suboptimal. The robot is programmed to take the most efficient path, which often results in a path that drives down the center of the hallway until a collision with the person seems imminent. This behavior treats the person as it would any other obstacle, ignoring the context of the obstacle as a moving, decision-making entity, that will not only move in a particular way, but also react to the robot's movements. In essence, the robot has no effective theory of mind for the people it encounters. This is problematic from the perspective of the person because they will have no way to predict which side of the hallway the robot will pass on, leading to higher uncertainty and less effective task behavior.

The use of navigation behavior that is not contextually aware has been shown to be quite problematic. In a long-term ethnographic study by Mutlu and Forlizzi [70], an autonomous delivery robot in certain hospital environments was found to not only be ineffective in its tasks (bumping into people and obstacles) but also made the people around it feel "disrespected" and anxious. The study found that these problems persisted after up to three years of interaction with the robot, indicating that the disconnect with how they expect it to act is a lasting problem.

The path that the robot takes is only one portion of the problem. Even if the robot did move to one side of the hallway or another, the person has no way to know why the robot performed such an action. It could have been because the person was there, or it could have been something else. Without implicit social cues, it is hard for the person to tell the difference.

Our hypothesis is that if the robot modifies its behavior to use a predictive model for the person, in which the person's social behavior is taken into account, the person will recognize the social behavior and be able to create a theory-of-mind for the robot that is more similar to people. We can examine how long it takes for the robot and the human to reach their goals, as well as how the person's opinion of the robot differs as a result of the interaction. As a result or our additional social navigation, the person will be able to complete their task more efficiently, since they are then able to use all of their (often implicit) prior knowledge of social interactions to predict how the robot will behave.

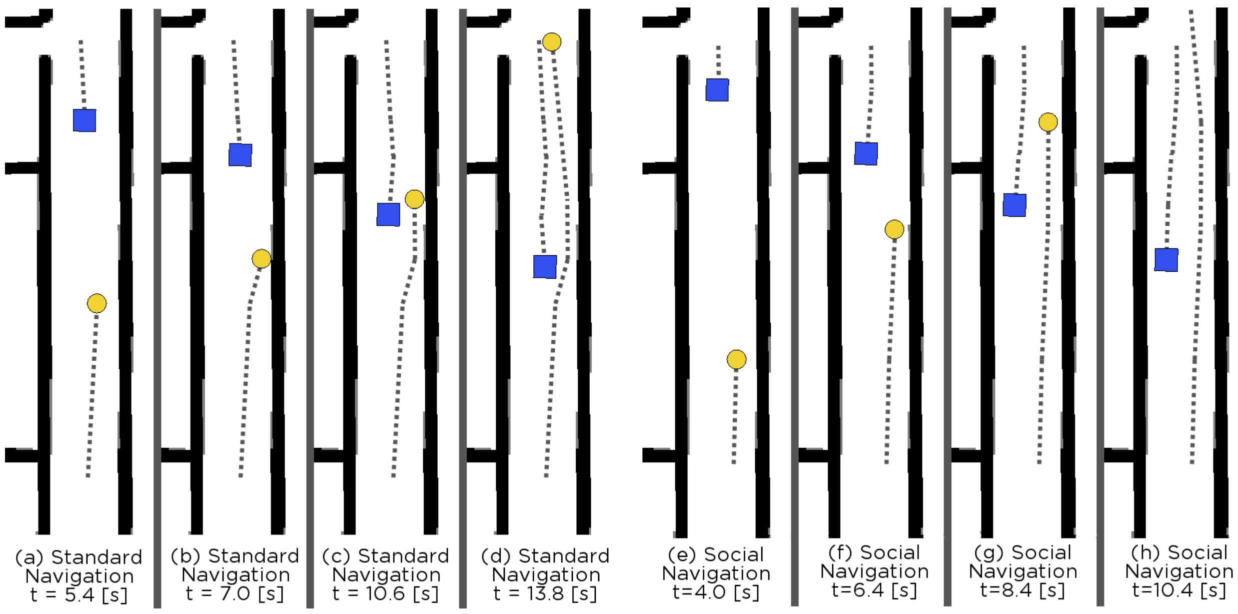

(a) Standard Navigation t = 5.4 [s]
(b) Standard Navigation t = 7.0 [s]
(c) Standard Navigation t = 10.6 [s]
(d) Standard Navigation t = 13.8 [s]
(e) Social Navigation t=4.0 [s]
(f) Social Navigation t=6.4 [s]
(g) Social Navigation t=8.4 [s]
(h) Social Navigation t=10.4 [s]

Figure 5.1: *An Example of Standard vs. Social Navigation* - These diagrams show results from an experiment as a robot (blue square) and a human participant (yellow circle) pass each other in a hallway. In the top example, using the standard navigation, the person is forced to slow down drastically while the robot passes (c). With the social navigation in the bottom example, the person is able to pass the robot passes the robot with much greater ease and ends up completing the task more quickly as a result.

In this work, we address these two problems, inappropriate paths and poor signaling of intent, with two techniques. Similarly to the work discussed in section 4.3, we modify the robot's costmaps to reflect the social behaviors we want to show in the planned paths. To better communicate the robot's intent to the human, we use a gaze behavior that directs the robot's head at either the human or at the hallway ahead. Our goal in implementing these behaviors is to make the interaction more natural, and hence efficient, for the human. In a pilot study with two human actors in a motion-capture environment [59], we explicitly studied the scenario of two people passing each other in a hallway. We observed that both gaze and the relative position of each person played a role in the passing behavior.

5.2 System Implementation

5.2.1 Lasers for People Detection

Before causing the robot to act appropriately in the presence of people, we must first determine where the people are. In the work reported here, we use the laser range-finders to detect people, primarily because of their accuracy in our experimental environment. We note, however, in a more general setting, another more powerful people-detector might be more appropriate.

We reduce the problem of detection people to that of detecting legs. Our leg detection technique is based on the algorithm of Arras et al [3] and extends an implementation developed at Willow Garage by Caroline Pantofaru.[11] Like Arras and Pantofaru, we use a group of low-level classifiers to determine the probability that a sequence of laser readings is a leg or not. These leg probabilities are then passed to an algorithm that pairs the individual legs, based on distance constraints, and tracks the resulting leg-pairs, which correspond to a person under our assumptions. While there may be more advanced information about the person beyond their location through time that may help adjust the robot's behavior, this approach uses less processing than more advanced state recognition and provides an adequate advance over the previous implementation. To improve recognition accuracy, we also employ a filter on the laser readings to remove obstacles that are part of the static map, ensuring that components of the architecture are not mistakenly identified as legs.

5.2.2 Gaze Behavior

Gaze is an important element of social interaction [41] and has been identified as a key behavior for human robot interaction [11], particularly in conversations [71] and cooperative tasks [8]. We implemented a gaze behavior in order to help the human and robot with the collaborative task of jointly navigating a hall.

We use the detected locations of humans to control the gaze behavior. The default behavior is to have the robot looking straight ahead. When a person is detected, the position of

[11]`http://ros.org/wiki/leg_detector`

their head is estimated approximately, using the center of their detected legs and an average human height. The robot's pan/tilt head is then pointed at this position, simulating gaze. Initial prototyping revealed that having the robot look at the person constantly gave an impression of being "creepy", which is consistent with two-person psychological studies [2]. Furthermore, this persistent gaze lowered the person's confidence that the robot knew where it was going, since it was constantly not looking in the direction it was traveling. Hence, for the actual experiment, we chose the robot's gaze to be on a cycle: 5 seconds of looking at the person, followed by 5 seconds of looking ahead. The intent was to give the person acknowledgment that the robot saw them, while avoiding the downsides of a constant leer. We acknowledge that more sophisticated models are possible, but chose instead to focus on improving the robot's path planning, which turned out to be more problematic.

5.2.3 Socially Aware Costmaps

In order to affect the change we wanted in the paths the robot took, we needed to change fundamental parts of the navigation system. We chose to implement our changes by modifying the existing ROS navigation and costmap architecture, for the reasons enumerated in section 4.1. In Chapter 7, we discuss the architectural changes we made for this experiment, as well as the generalized solution that those changes grew into. In this section we will instead discuss the changes in the costmap values.

The default implementation of the navigation system often results in the robot often driving within a few centimeters from contact with obstacles, which can be quite unsettling when that obstacle is a person. People are generally not comfortable with a robot entering what Hall defines as the person's intimate proxemic space [35], despite the fact that the robot does not collide with them.

The problem is that the current implementation only deals with hard constraints on the costmap space. If we were to try to increase the space around each person with these constraints, many paths that are valid (non-colliding) would be excluded. If the environment is cluttered, or more relevantly, a narrow hallway, then there may be no possible path with hard constraints applied. A more appropriate approach would use softer constraints on the planned paths, where we can specify that we would rather a robot did not enter certain spaces, but would allow it if needed.

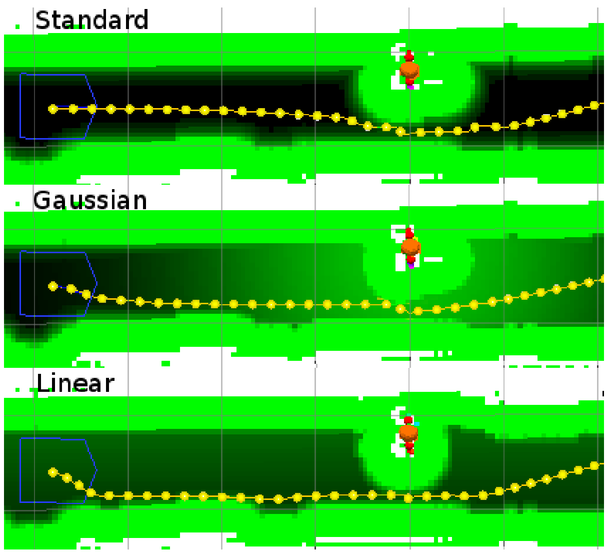


Figure 5.2: Three different paths with three different costmaps. The robot is marked in blue, the detected person in red and the path in yellow. The white areas show the locations of obstacles (either the walls or the person's legs). The tints of green show the other costs, ranging from bright green lethal points to darker tints for lower values.

We experimented with the use of Gaussian costmap adjustments, as suggested by Kirby [45] and others, as seen in the middle portion of figure 5.2. In this approach, values are added to the costmap around the human's detected location, proportional to a two-dimensional Gaussian distribution, possibly taking into account the person's direction of travel. This approach works well in the general case, as it causes the robot to take smooth paths that are farther away from people. It also works as an analogy to the personal space/proxemic concerns discussed earlier. However, it requires fine tuning of the parameters to get the desired behavior, and we found that sometimes there are no (readily-found) values of the parameters that realize the desired behavior. For instance, in our corridor, we wanted the robot to move to the side of the hallway it was going to pass on as soon as possible. In theory, increasing the amplitude or variance on the Gaussian function should result in the paths moving further away from the center of the Gaussian. However, the path planning algorithms need to find balance between the lowest cost paths and the shortest paths. Once the costs become sufficiently high (either by increasing the amplitude or variance), the decrease in penalty for moving further from the obstacle is outweighed by the cost of taking a longer path. This results in the robot reverting to its earlier behavior of driving straight toward the person along the shortest path. We discuss this problem further in Chapter 6.

Instead of the Gaussian approach, we implemented a cost function that was more specific to the environmental context. This “linear” cost function varied depending on what side of the hallway the person was detected on. If the person was detected on the left, the costs would decline linearly from the left side to the right side (and similarly if they were on the right), with values ranging from 140 on the undesirable side of the hall to 20 on the other. Similar to the Gaussian function, this would cause the lowest cost next to the person to be all the way against the wall. However, this would persist through the entire hallway, resulting in the robot moving to the side of the hallway earlier than in the Gaussian, as can be seen in figure 5.2. This sends a social signal to the human sooner and more clearly, giving them time to interpret and react to it. Furthermore, this costmap modification requires little tuning to get the designed behavior.

While this solution is less general than the Gaussian function, we argue that the environment informs human-human social behavior. We instinctively follow the architectural cues in buildings. This context can be, we claim, captured in special-purpose costmap modifications that depend on the current location of the robot.

5.3 Experimental Design

5.3.1 Procedure

The goal of our experimental design was to test how the two social navigation behaviors we implemented would affect interactions between people and the robot. We devised a fetch and carry task, in which participants were told that they would be working with the robot to deliver boxes to different rooms on a hallway. This scenario was selected for three reasons. First, it is typical of the kind of interaction a person unfamiliar with robots is likely to encounter in the near future. Robots are being used in uncontrolled environments, such as hospitals, where people may encounter a robot with no prior explanation as to what is is likely to do. Secondly, delivering packages or mail is a task that robots are already capable of doing. Finally, giving the person a specific task gives them reason to complete the task in an efficient manner. The boxes were shoe-box sized and empty, so that any changes in speed due to the weight or impedance of carrying a box is negligible. The robot is 67cm wide and the hallway is approximately 150cm wide, meaning that people will be able to pass the robot in most cases, regardless of whether the robot is all the way to one side or not.

Participants were recruited from the Washington University campus community. All participants were 18 years or older, with English language skills, and no prior experience working with robots. After the consent process, the scenario was explained to them. They were instructed to fetch or deliver a box three times, in the environment shown in Figure 5.3. This necessitated them walking from room A (at one end of the hallway) to room C (on the far end) and back. On the first trip, the robot would be driving toward the participant as they went to room C and stationary at the door to room B (midway down the hall) on the way back. On the second and third trips, the participant would encounter a stationary robot on the way out, and the robot driving toward them (and room C) on the way back.

To be able to generalize our results to a wide range of human-robot encounters, the experiment involved two tasks: one where the robot was moving, and one where the robot was stationary. To compensate for participant-specific variation in speed, we wanted to measure the participants' general walking speed. We felt that this would be best measured by having the participant walk down the hallway around the stationary robot obstacle.

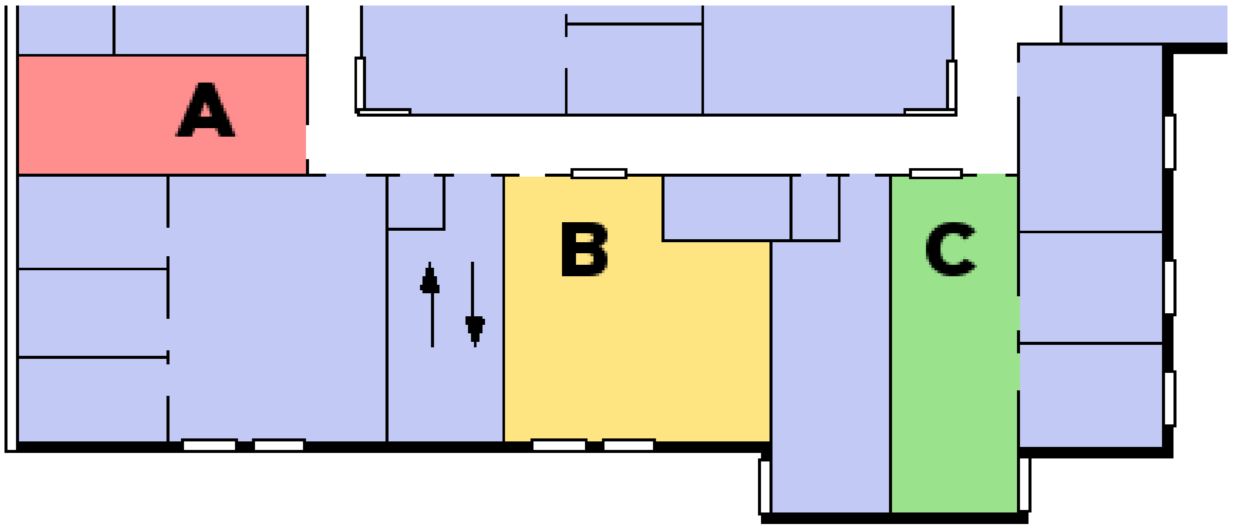


Figure 5.3: Map of office where experiment was held. Experimenter sat in Room A, and participants traveled from Room A to Room C. The robot moved along the hall, occasionally stopping at Room B. The hallway is 17 meters long and approximately 1.5 meters wide.

We use our two navigation behaviors to form a 2x2 between-subjects study design. The robot's interaction in the hallway was dictated by one of these four sets of behaviors, randomly assigned.

C1 Standard navigation, no gaze behavior (N=8)

C2 Standard navigation, gaze behavior (N=7)

C3 Social navigation, no gaze behavior (N=7)

C4 Social navigation, gaze behavior (N=8)

At the conclusion of the three deliveries, the participant was given a questionnaire to fill out. They provided basic demographic information, as well as a 7 point Likert scale rating of the following statements, from strongly disagree(1) to agree (7).

- The robot is intelligent.
- The robot was aware of where I was going.
- The robot is confident.
- I felt safe when I passed the robot.
- I knew where the robot was going.
- The robot knew what it was doing.
- The robot navigated around me.
- The robot seemed to know what I was doing.
- I was concerned the robot might run into me.
- The robot let me know where it was going.

The questions were designed to measure the participants' perception of the robot's competency, awareness of the participant and ability to communicate intent.

Thirty people participated in the study (N=30), with ages ranging from 18 to 70 (mean=27.7, SD=12.4, median=23). 60% (N=18) of the participants were female, and roughly half (53%, N=16) were native to the United States. Each participant was compensated for their time. The data for each participant are divided up into six trials, defined as going from one end of the hallway to the other. Out of a total of 180 possible trials down the hallway, 171 are eligible for analysis, with the remaining nine removed due to sensor malfunctions or external interference.

For measuring the location of both the robot and the participant, in addition to the robot's laser scanner, a Microsoft Kinect device was placed at each end of the hallway. These devices were stationary, and the only elements moving within the field of view were the robot and participant, allowing for very accurate measurements of the the locations of the targets through time. Signaling distance information was derived from these measurements as well.

5.3.2 Hypotheses

We chose to study the following hypotheses with our study.

H1: People encountering the robot in the Gaze condition will take less time to walk past the robot than those encountering the No Gaze condition.

H2: People encountering the robot navigating with the social algorithm will take less time to walk past the it than when it uses the standard navigation.

These hypotheses were based on our assumption that the biggest problem with the interactions was that people were unsure whether the robot knew where they were.

Thus, a robot that looked at faces or used social navigation would communicate that the robot was aware of their position, and introduce greater certainty about the robot's beliefs. Therefore, we hypothesized that people would be more confident about where the robot would go, and thus be able to complete the task in less time.

5.4 Results

The primary metric we will examine is the average speed of the human participant to walk from one end of the hallway to the other. For each participant, we average the speed for all of the hallway traversals in which the robot was moving, which we annotate HS. [12]

An illustrative example of the differences in navigation styles can be seen in Figure 5.1. Consider the top example from a trial under condition C1. Both robot and human start by moving straight down the middle of the hallway (1a). As they get nearer, the human moves to the side of the hallway (1b); however, the robot continues on the same path as before. This causes the person to slow down almost to a stop as they wait for the robot to pass by (1c). Although the robot does in fact yield slightly, it only does so once the person has already reacted to its path by stopping. Both parties continue on their way after they pass (1d). Contrast this to the bottom example under condition C4. This trial starts off similarly with both moving down the center of the hallway (1e). However, they both move to the sides of the hallway early on (1f) allowing for a nice quick pass (1g). This allows both the human and robot to reach their destinations sooner than the previous example (1h). In these exemplars, the HS was $0.88m/s$ and $1.18m/s$ respectively.

The aggregate results are shown in Figure 5.4. We will discuss the lack of statistical significance in the results in the following subsection. We must be cautious to not over extrapolate from this data, given its limitations, but there are some weak trends worth mentioning. First, in minor support of H2, the social navigation algorithm generates faster human speeds than the standard navigation algorithm. Contrary to H1, the gaze condition lead to slower speeds than the no gaze condition. Neither of these results is significant, but does mirror trends we saw in some of the users.

There is one additional interesting result we were able to gather from our data. As expected (though not explicitly hypothesized about), over all conditions, the average human speed in the passes where the robot was stationary was significantly higher than the passes where the robot was moving ($p < 0.003$, $1.179ms^{-1}$ vs. $1.251ms^{-1}$ based on a one-tailed T-test, $df = 58$). This comes as no surprise since it is easier to predict where a stationary object will be.

[12]In Lu and Smart [58], other metrics were also considered.

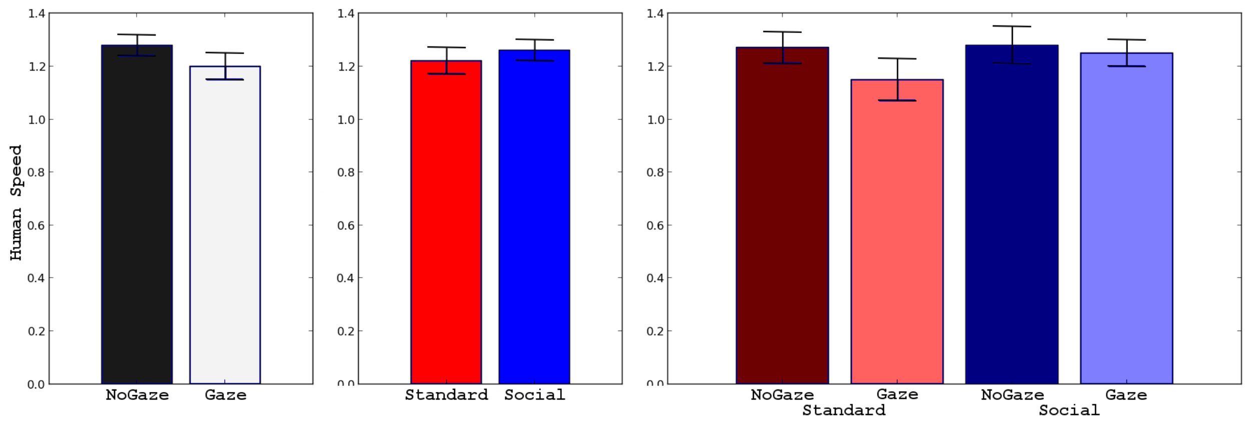


Figure 5.4: Average Human Speed in the User Study

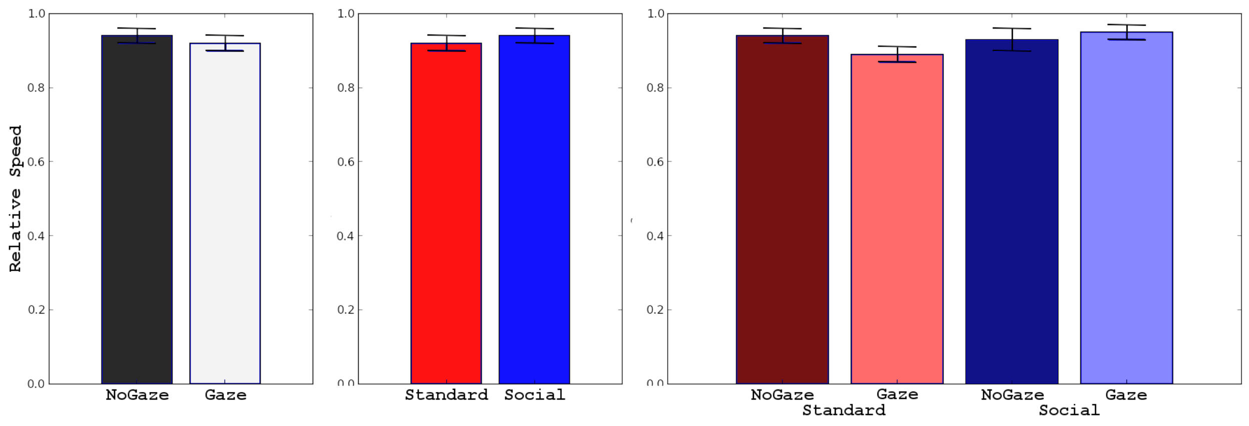


Figure 5.5: Relative Speeds in the User Study

Table 5.1: ANOVA Analysis of Average Speed

	df	Sum of Squares	Mean Square	F	Pr(>F)
C(gaze)	1	0.04	0.04	1.26	0.27
C(nav)	1	0.02	0.02	0.64	0.43
C(gaze):C(nav)	1	0.01	0.01	0.42	0.52
Residual	26	0.84	0.03		

Table 5.2: ANOVA Analysis of Relative Speed

	df	Sum of Squares	Mean Square	F	Pr(>F)
C(gaze)	1	0.0015	0.0015	0.38	0.54
C(nav)	1	0.0039	0.0039	0.97	0.33
C(gaze):C(nav)	1	0.0068	0.0068	1.71	0.20
Residual	26	0.1038	0.0040		

We also calculated the relative speed of the person, with the person's average speed when the robot was moving divided by the average speed when the robot was stationary. Since the participants generally moved faster during the stationary passes, the ratios are usually less than 1.0. Ratios closest to 1.0 indicate that the robot had less of an impact on the participants' path. These ratios are shown in Figure 5.5. They generally mirror the results shown for average overall speed, and do not add any statistical significance.

The survey data did not produce any statistically significant results. We discuss this further in section 5.5.

5.4.1 Statistical Analysis

We calculate two-way ANOVA scores, using the navigation condition (*nav*) and gaze condition (*gaze*) as our independent variables, and human speed as our dependent variable. As indicated by the right-most column in Table 5.1, there is no statistical significance to our results. The same is true when using the relative speed, as seen in Table 5.2.

The questionnaire provided no statistically significant data. Since the statements were designed by us, there is the likely possibility that they were not as reliable as we had hoped.

5.5 Discussion

As mentioned in the introduction, this user study failed to produce the desired effect on the interaction. This was in part due to the large variance in individual behavior when encountering the robot, as reflected by the relatively large standard deviation found in our results. We performed a post-hoc power analysis on our study. With k-means power analysis, $\alpha = 0.05$ and a standard deviation of 0.178, as well as the means of the average speed, we found that our sample size of $N = 30$ resulted in a power of 0.205. With an ANOVA power analysis, we get a similar result, with a power of 0.1924. In retrospect, given the number of trials we completed, our study had low power. It may be that additional trials could have helped clarify or uncover trends. However, running additional participants the study are impractical due to logistical constraints.

There are three separate ways to examine the implications of this study and its lack of significant results. We'll first look at the takeaways for designing human-robot behavior in general, followed by looks at the specifics for gaze and navigation behavior.

Our first conclusion is that there are different strategies for robots to navigate around people depending on the desired priorities. If we place sole priority on getting *people* to where they need to go, the clear solution is to park the robot and not interact at all. Keep the robot out of the way, and people will get to their destinations promptly. If the priority is solely on the *robot's* navigation effectiveness, the best strategy might be the (untested) strategy of ignoring the obstacles associated with detected people, on the assumption that they will move out of the way as needed. However, this strategy requires that the human endow the robot with higher status and thus give way to its priorities, which may not be the case. Nevertheless, most situations call for a mixing of priorities. The system cannot be optimized for just one party or another. The best solution for the combined robot/human system likely involves elements of social navigation and additional secondary actions that help the robot convey its intent to the humans with whom it interacts.

We could also speculate on the implications of the lack of statistical significance in the questionnaire. We found no evidence of relationships between people's perceptions of the robot, the robot's behavior or the person's behavior. This might suggest that people are not consciously aware of many of these subtle social cues, regardless of the effect they have on the interaction. This fact lends support to part of Reeves' and Nass' findings [84] that social interaction with machines is often subconscious and inherently social. However, it is most likely to be reflective of an improperly designed survey that left too much ambiguity in what the person's actual opinion of the robot was.

The gaze behavior did not affect the interaction as we predicted. We had hypothesized that the gaze behavior would let the person know that the robot saw them and, hence, the person would walk more confidently, and more quickly, down the corridor. Instead, we discovered no significant relationship between the gaze behavior and task completion speed.

The lack of differentiation likely stems from the large number of factors that are involved in designing gaze behavior. Our stated belief that the gaze behavior would increase the person's confidence of the robot's awareness is just one way the gaze might change human behavior. Moving the robot's gaze toward people means that the robot would often travel in a different direction than the where it was looking. Kirby defines traveling in the direction one is looking to be a desirable quality. After all, humans have a tendency to look at where they are about to go. When this quality is broken, it may lead to confusion about where the robot is about to go. Another factor is that glancing at the other entity in the hallway could be construed as looking to start an interaction. For instance, Miles Patterson [78] lists gaze as a key "nonverbal involvement behavior" that can lead to more intimate interactions. If a participant encountered such behavior, it would make sense if they slowed down to possibly begin a more involved encounter. There are myriad other ways that gaze could be interpreted to affect the interaction, whether it be with a sharp glance intended to intimidate the other party to yield, or with a glance away to signal deference to the other party.

We started with the simple belief that adding gaze behavior would be better than no gaze behavior, i.e. something is better than nothing. This led to the simplistic gaze behavior that we implemented in Section 5.2.2 of five seconds of looking at the user, and five seconds of straight ahead. This behavior covers some of the motivations discussed above, but not all, and based on our results, most likely, not enough. The simple implemented behavior can be reduced to tracking roughly three variables: the location of the person, whether the robot is currently looking at the person or straight forward, and how long the robot has been looking at the current thing. This gives the robot a five second working memory. Such a model makes no differentiation between the beginning of an interaction and the end, nor any additional contextual information about what has come before. This mirrors what we discussed in Section 3.3. One possible model for gaze could include an initial long glance when the robot first detects the person, subsequent glances as the person changes position significantly and perhaps a final social glance before the person passes out of the range of view. While there has been a fair bit of research on creating models for gaze behavior during explicit interactions further exploration and experimentation is needed to actually build an effective gaze behavior for this scenario.

Similarly, the social navigation uses a simple model that is minimizes the state kept by the robot. The social navigation tracks which side of the hallway the person is currently on, ignoring where they have been and where they are likely to go. If the person has primarily been walking on their right side of the hallway, and move momentarily to the left (to pass an obstacle, say) the robot will briefly start moving to the person's right. Such a model may be useful for certain scenarios, but will not be sufficiently complex to capture all the variance in human behavior around navigating robots.

These simple models exist in one of the most difficult tensions in human-robot interaction research. The instinct in science is to try to make the simplest models possible, not only to have an elegant solution but also to make performing comparative experiments easier. In order to make a clean A/B experiment, it is most convenient to have a clear division between behaviors with the fewest moving parts. However, simplicity stands in direct contrast to human behavior. It is difficult to fit human social behavior into a neat model, which is why real people are needed for user studies. However, with more complex models, greater numbers of people are needed to gain statistical significance for small changes in behavior. However, obtaining greater numbers require significant resources, which in this experiment, turned out to be infeasible. Additionally, there is further tension in designing interactions for experiments which are simultaneously free enough that the user will feel they can act "naturally" but constrained enough to actually test the behavior desired.

While this experiment started with the sole focus on the robot's gaze, our initial experimentation led to the conclusion that the standard navigation behavior was distractingly problematic. We hypothesized that it would not matter how useful our gaze behavior was, it would not make the interaction comfortable enough when the robot came into the user's personal space so obtrusively. This led to the inclusion of altered navigation behavior into our model and experiment. Afterwards, we judged the navigation behavior research to be more promising. This led us to two new research directions:

1. The use of the linear costmap adjustment was ultimately beneficial, but the inability to replicate an effective use of the Gaussian distribution (like that used by Kirby [44]) was disappointing. While the previous work was able to get the robot to effectively navigate around people using a Gaussian distribution, we found that the parameters were not very robust, generating unpredictable behavior in many scenarios. In Chapter 6 we explore the mathematical properties of costmaps that explain why we ran into trouble, and how to avoid it.

   The work in Chapter 6 relies on simulation rather than user studies. We know from the study of proxemics [35] that the distance between people and other entities does make a difference. While we would need user studies to figure out the ideal distance between a robot and a person in specific scenarios and cultures, we instead use the simulated interactions in the next chapter to explore what parameters are needed to achieve a particular distance once that ideal has been specified.

2. The second research direction stemmed from our difficulty in creating custom behavior using the ROS navigation platform, and lead to the work in Chapter 7.

   The greatest portion of the time prior to the experiment was not spent designing precisely the type of robot behavior that would be most effective, but implementing the changes required to get something approximating our desired behavior. To make the changes to the costmap model more flexible and easier to manipulate in the future, we systematized our algorithms and released them as open source.

   With the infrastructure and implementations devised in the rest of this thesis, the intent is to make it easier to implement complex behaviors that encompass the variation of human behavior. Our contribution is to expedite the process for future researchers of implementing social robot behavior so that they may instead focus on developing effective models.

Chapter 6

Tuning Cost Functions for Social Navigation

Raising the value of the cells in the costmap makes it less likely for the robot to travel into those cells. But how much less likely? Under what conditions? The answer is, it depends.

In practice, creating the desired behavior around people is surprisingly difficult. While previous researchers have tuned their parameters to create working configurations, there exists no general guide for how to do this to effect a specific change in robot behavior. Furthermore, as we show below, the resulting behavior is not always intuitive from a consideration of the individual costmap elements.

6.1 Problem Statement

Each planned path depends on two separate components: the costmap and the planning algorithm. The costmap is represented by a two-dimensional grid, where each grid cell has a value $f(x, y)$. Values above some predefined threshold are designated as "lethal" values, representing states that must be avoided, like those resulting in collisions. Standard algorithms such as Dijkstra's and A* are typically used as the path planner. However, if the total cost of a path is defined as the cost of the cells the path traverses alone, then the resulting path will be a very long path that avoids any cost. To avoid this scenario, wavefront planners are often used, which add a constant value, P, to each cell traversed to create a gradient from start to finish[21]. This is equivalent to adding the value P to each cell and running Dijkstra's algorithm on the new costmap, but for purposes of discussion here, we keep it as a separate constant to differentiate the values in the costmap and the parameters of the path planning algorithm.

Formally, we define $C(p)$ as the total cost of the path p (including both the costmap costs and the path planning costs). Finding the best path involves minimizing the cost over all possible paths.

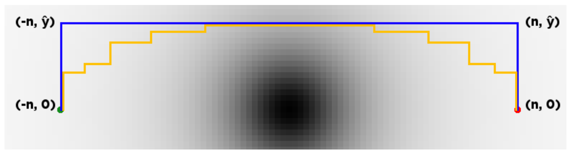
$$\min_{\forall \text{path} p} C(p) = \min_{\forall \text{path} p} \sum_{(x,y) \in p} \Big[f(x,y) + P \Big] \tag{6.1}$$

This definition for path cost assumes that each step in the path moves to another grid cell exactly 1 unit away, implying each cell is connected to its four immediate neighbors. We use this assumption throughout this paper, although it is possible to generalize it, for example, to also include diagonal moves.

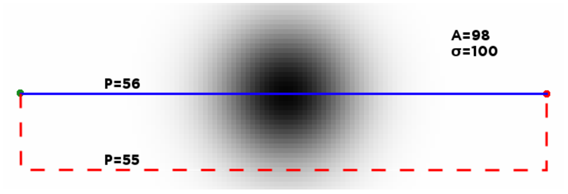
For purposes of this paper, let us further refine the problem to reduce the number of cases we must consider. First, without loss of generality, let us consider paths that go from $(-n, 0)$ to $(n, 0)$. We further assume that there are no lethal cells in our costmap, since path planning algorithms already do a fine job of avoiding these.

In addition to the actual planning problem, there is also the parameter tuning problem. We would like to be able to design robot behavior from a high level and not need to fiddle with the parameters endlessly to find the perfect balance for the desired behavior. One goal of this is to find functions in which the parameter space is either intuitive or limited to only the best possible values.

The interplay between the values in the costmap and the path planning constant turns out to be crucial for determining the course of the planned path. The duality of optimizing for path length or for path cost results in a continuum of different paths that could be considered optimal depending on the weighting of the two sides.



(a) The Bracket Shape



(b) Gaussian Cost Function

Figure 6.1: Types of Paths - In 6.1a, an optimal bracket-shaped path in blue and a suboptimal path in yellow. In 6.1b, the discontinuity of optimal paths between $P = 55$ and $P = 56$.

6.2 Mathematical Properties of Gaussian Obstacles

Our analysis focuses on the frequently-employed two-dimensional Gaussian distribution, defined as

$$f(x, y) = A \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{6.2}$$

The cost in each cell under the Gaussian depends on both the amplitude, A, and the variance, σ. One key feature of the Gaussian function (and all other monotonically decreasing functions) is that the optimal path is always bracket-shaped: from $(-n, 0)$ to $(-n, \hat{y})$ to $(n, \hat{y})$ to $(n, 0)$ for some $\hat{y}$. The proof of this is elided for space, but centers on the fact that all paths that reach $y = \hat{y}$ will have the same length (as seen in Figure 6.1a), and the bracket-shaped paths are the farthest paths of that length from the obstacle. Note that a direct path qualifies as a bracket with $\hat{y} = 0$.

The only variation in the paths is how far away from the obstacle they are, i.e. their $\hat{y}$ values. In this section we seek a relationship between the model parameters P, A and σ and the resulting distance of closest approach, $\hat{y}$.

In order to cause the optimal path have to a smaller value of $\hat{y}$, the obvious strategies are to decrease the costs (decrease A or σ) or to increase the path constant P. This is generally true, but there exist certain conditions in which incrementally changing the parameters in this way will result in a drastically different path.

For instance, one would expect that increasing P incrementally would result in a gradual decline in $\hat{y}$. However, the resulting change sometimes is a discontinuous jump. Consider the paths in Figure 6.1b. With $A = 98$ and $\sigma = 100$, when P is increased from 55 to 56, one would expect $\hat{y}$ to undergo a small decrease. Instead, the path jumps suddenly through the origin ($\hat{y} = 0$). There is, in fact, no value of P which causes the optimal path to occur between these two paths.

6.2.1 Theory

Our goal is to show why there are certain parameters that result in one of three behaviors: **(1)** Minimum path cost is at $\hat{y} = 0$ **(2)** Minimum path cost is at finite $\hat{y} > 0$ **(3)** Minimum path cost is infinitely far away. The cost of a bracket path in terms of the model parameters and some choice of $\hat{y}$ is

$$\begin{aligned} C(p_{\hat{y}}) &= C(\text{segment 1}) + C(\text{segment 2}) + C(\text{segment 3}) \\ &= \left[\hat{y}P + \sum_{i=0}^{\hat{y}-1} f(-n,i)\right] + \left[2nP + \sum_{x=-n}^{n} f(x,\hat{y})\right] + \left[\hat{y}P + \sum_{i=0}^{\hat{y}-1} f(n,i)\right]. \end{aligned}$$

Now let us assume that we begin far from the obstacle, so $n \gg \sigma$. In this limit, the cost due to the obstacle on segments 1 and 3 is very small compared to the baseline cost and the cost along segment 2.

$$C(p_{\hat{y}}) \approx (2n + 2\hat{y})P + \sum_{x=-n}^{n} f(x,\hat{y}) \tag{6.3}$$

We are only concerned with the cost of each path in relation to other paths, so we will express the cost of some $p_{\hat{y}}$ relative to the cost of the direct path.

$$\begin{aligned} \Delta C(\hat{y}) &= C(\hat{y}) - C(0) \\ &= \left[(2n + 2\hat{y})P + \sum_{x=-n}^{n} f(x,\hat{y})\right] - \left[(2n + 2(0))P + \sum_{x=-n}^{n} f(x,0)\right] \\ &= 2P\hat{y} + \sum_{x=-n}^{n} \left[f(x,\hat{y}) - f(x,0)\right] \end{aligned}$$

When $\Delta C(\hat{y}) < 0$ for some $\hat{y}$, the direct path is not optimal.

With a Gaussian obstacle as f,

$$\Delta C(\hat{y}) = 2P\hat{y} + \sum_{x=-n}^{n} \left[A \exp\left(-\frac{x^2 + \hat{y}^2}{2\sigma^2}\right) - A \exp\left(-\frac{x^2}{2\sigma^2}\right) \right]$$

We have already assumed $n \gg \sigma$, the tails of the Gaussian will contribute negligibly, so we can approximate the sum over $x \in [-n, n]$ as the sum over all x, which has a simple solution. We use the following approximation.

$$\begin{aligned} \sum_{x=-n}^{n} A \exp\left(-\frac{x^2}{2\sigma^2}\right) \exp\left(-\frac{y^2}{2\sigma^2}\right) &\approx \sum_{x=-\infty}^{\infty} A \exp\left(-\frac{x^2}{2\sigma^2}\right) \exp\left(-\frac{y^2}{2\sigma^2}\right) \\ &= A\sigma\sqrt{2\pi} \exp\left(-\frac{y^2}{2\sigma^2}\right) \end{aligned}$$

Finally, we have a closed-form expression for the cost of bracket path $p_{\hat{y}}$ compared to the direct path.

$$\Delta C(\hat{y}) = 2P\hat{y} + A\sigma\sqrt{2\pi} \left[\exp\left(-\frac{\hat{y}^2}{2\sigma^2}\right) - 1 \right] \tag{6.4}$$

Now, we locate the $\hat{y}$ that minimizes $\Delta C(\hat{y})$.

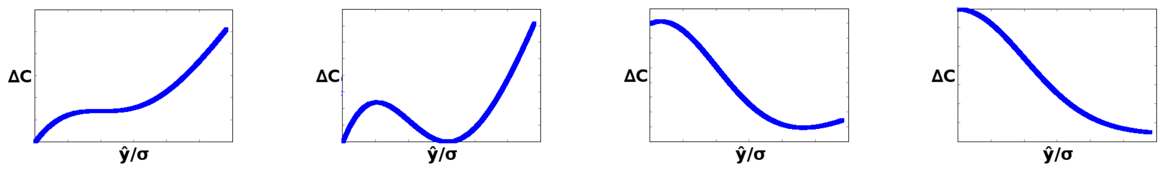
$$\frac{d\Delta C}{d\hat{y}} = 2P - \frac{\hat{y}}{\sigma} A\sqrt{2\pi} \exp\left(-\frac{\hat{y}^2}{2\sigma^2}\right) = 0 \tag{6.5}$$

$$\frac{P}{A} = \sqrt{\frac{\pi}{2}} \frac{\hat{y}}{\sigma} \exp\left(-\frac{\hat{y}^2}{2\sigma^2}\right) \tag{6.6}$$

The solution for $\hat{y}$ is related to the Lambert W-function [13], which cannot be written in closed form. Depending on the cost ratio P/A, it admits 0, 1 or 2 solutions. We can make several observations.

1. Eq. 6.6 peaks when $\hat{y}/\sigma = 1$, attaining $P/A = \sqrt{\pi/2e}$. If we set P and A such that $P/A > \sqrt{\pi/2e} \approx 0.760$, there is no solution to Eq. 6.6, and the only minimum of ΔC occurs on the boundary at $\hat{y} = 0$ like in Figure 6.2a. The direct path is optimal.

[13] See `http://en.wikipedia.org/wiki/Lambert_W_function`



(a) $P/A = \sqrt{\pi/2e}$: The direct path $\hat{y} = 0$ minimizes ΔC.

(b) $P/A = 0.57$: A new global minimum is established at $\hat{y} > 0$.

(c) $P/A < 0.57$. $\hat{y}$ moves farther from the origin, and the minimum grows deeper.

(d) $P = 0$, and the global minimum is at $\hat{y} \to \infty$.

Figure 6.2: Cost ΔC relative to the cost of the direct path, as a function of ratio of the baseline cost-per-step P and the amplitude A of a Gaussian obstacle. The minimum of this function determines where the optimal path is.

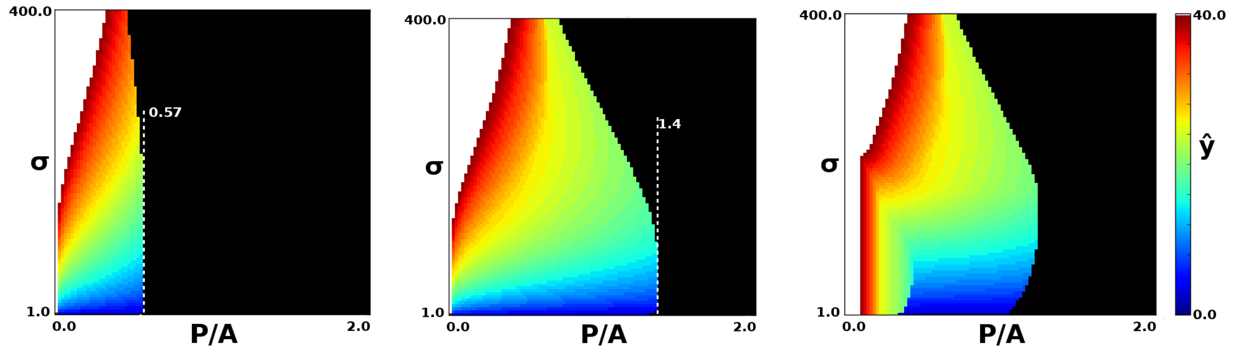
2. For values of P/A less than .760, the inflection point at $\hat{y}/\sigma = 1$ decreases as well, creating a local minimum. This minimum remains a local minimum until $P/A \approx 0.57$ (Figure 6.2b), a point we determined numerically where the minimum becomes the global minimum. For values even less than 0.57 (Figure 6.2c), the direct path is no longer optimal and the center of the obstacle is avoided.

3. When $P = 0$, $\Delta C(\hat{y})$ has a minimum at infinity (Figure 6.2d). This results in the path being as far away from the center of the obstacle as possible.

Thus, we have shown, with some simplifications, the mathematical underpinning for the relationship between the different parameters and why certain configurations lead to the three behaviors/values of $\hat{y}$ discussed at the beginning of this section.

6.3 Results from Simulation

To further explore the relationships between the parameters, we ran path planning algorithms over simulated costmaps as described in the Section 6.1. Instead of showing all of the paths for each configuration, we represent the resultant $\hat{y}$ values in heatmaps as seen in Figure 6.3. As is evident from Equation 6.6, P and A are inversely related, a fact we could have surmised through dimensional analysis. As long as the ratio P:A remains constant, the value of $\hat{y}$ also remains constant (plots not shown). This means we can explore the entire parameter space



(a) Four Connected (b) Eight Connected (c) Two Gaussians

Figure 6.3: Heatmaps - The value of $\hat{y}$ as a function of P/A and σ. Black values represent paths where $\hat{y} = 0$; white values represent paths where $\hat{y}$ is infinitely far away. Redder values indicate higher values of $\hat{y}$ and bluer values represent lower values.

as a two dimensional heat map relating that ratio to σ, which we did by varying the value of P.

The three distinct behaviors are seen in the different colorations. Configurations represented in black result in paths that move straight through the obstacle ($\hat{y} = 0$). On the opposite side of the spectrum, white configurations represent paths that are as far away as possible, i.e. the optimal $\hat{y}$ is infinitely large. The intermediate hues represent the finite values of $\hat{y} > 0$, with red values being the farthest away and blue values being the closest (smallest $\hat{y}$).

Let us begin the discussion with Figures 6.3a and 6.3b. Figure 6.3a was created using the von Neumann neighborhood (i.e. four connected) and corresponds with the math from the previous section. Figure 6.3b utilized the Moore neighborhood (i.e. eight connected), and thus the properties from the previous section need no necessarily apply. However, as evident in the two figures, both the von Neumann and Moore neighborhoods operate similarly, although with slightly different scaling. This lends substantive support to our hypothesis that the properties operate similarly regardless of the precise path planning implementation used.

The next thing to note is the relationship between σ and $\hat{y}$ for a given P/A. For any given finite positive value of $\hat{y}$ we can decrease the variance to get a smaller $\hat{y}$. This leads us to our first general observation about the parameter space. **Decreasing the variance on a Gaussian will always lead to paths closer to the obstacle.** Similarly, we can assert that **Lowering the ratio $P : A$ will always increase $\hat{y}$ from a positive value to a greater value.**

However the inverse of these statements is not true. Increasing the variance will sometimes increase the distance from the obstacle, but can also lead to decreasing the distance down to $\hat{y} = 0$. **Increasing the variance on a Gaussian will only sometimes result in paths further from the obstacle.** Increasing $P : A$ will always lead to to decreased $\hat{y}$ values, however, at some point, those values jump to $\hat{y} = 0$. This discontinuity means that **For a given σ, some values of $\hat{y}$ cannot be expressed.** If this were not the case, the right-hand edge of the colored areas in the heatmaps would be blue for the lowest $\hat{y}$. Similarly, some values of $\hat{y}$ cannot be expressed for a given ratio $P : A$. These two limitations lead to the most important observation about the parameter space: **Finding paths with the full range of $\hat{y}$ values cannot be achieved by tuning just one parameter.**

This point is reiterated by the fact that there are some values of $P : A$ that do not have any values of σ resulting in $\hat{y} > 0$. Our simulated experiments also validated that these dead zones occur when $P : A > 0.57$ when the grid is four-connected, which can be seen in Figure 6.3a and since these values result in no solution to Equation 6.6. Values of $P/A < 0.57$ generally result in non-zero $\hat{y}$, until the variance gets sufficiently large, which is where our initial assumption that $n >> \sigma$ breaks down, resulting in unpredictable behavior. Based on the results shown in Figure 6.3b, we have further estimated that for eight connected neighborhoods, the dead zone starts with $P/A > 1.4$.

6.4 Results using ROS Navigation

Much of this work is motivated by our experiences with the ROS navigation stack [61]. In the experiments from Chapter 5, we added the ability to express such non-lethal obstacles. In the costmaps used in that experiment, the amplitudes can range from $[0, 254]$ with no limits on the variance. Initially we were unaware that the default path planning constant was set so that $P = 50$, and we did not know why we were getting paths that had such small passing distances. However, through the work in this chapter, we came to realize that setting $P = 50$ meant that our options for A were limited, and which explained our difficulty in tuning the parameters.

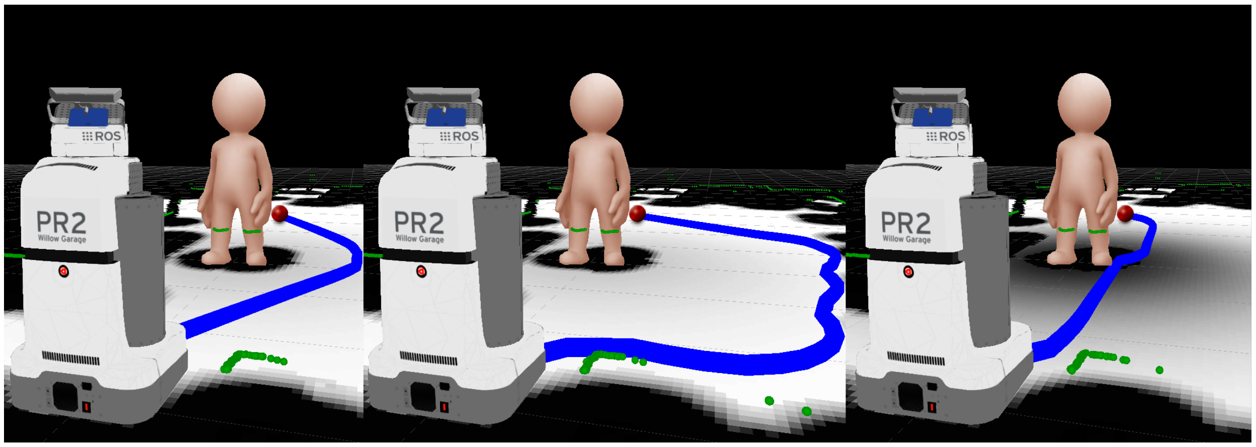


Figure 6.4: Results using ROS Navigation, $\sigma = 1.75$, and $P/A = \{3/3, 3/60, 3/235\}$

To further validate the principles in this chapter, we used ROS navigation stack with real sensor data from a PR2. Paths were planned around a live human using a modified wavefront planner that performs interpolated gradient descent to determine a smooth path, i.e. not constrained to either the von Neumann or Moore neighborhoods. The results are shown in Figure 6.4 with three different values of A. As A increases, the value of $\hat{y}$ increases too, but then when A is very large, it contracts back down to $\hat{y} \approx 0$, as close to the person as possible. The full heatmaps are not included for space.

We found that even using the smooth-path planner, the same principles apply, although the constants have different values. Since costmaps and the discretized paths they produce are all approximations of the same continuous space, it is logical that the different grid connectivities would have similar results; however, it is nice to have the technical confirmation to back up the ideas. Not only is the solution more general, but it has the added benefit that the resulting smoother paths are more legible to people observing the robot.

6.5 Tuning in Two Dimensions

Thus far, the analysis of the robot's path relative to the center of the Gaussian has been restricted to one dimension. In most cases, the path can be reduced to the single data-point, $\hat{y}$. This simplifies the analysis and allows us to gain the insights about the Gaussian that we got from earlier in the chapter. However, in certain scenarios, such an analysis is overly reductive. For example, in the hallway scenario from Chapter 5, $\hat{y}$ was roughly constant, since the robot and person had to pass by each other, so regardless of the parameters, the two would have to come into very close proximity to each other.

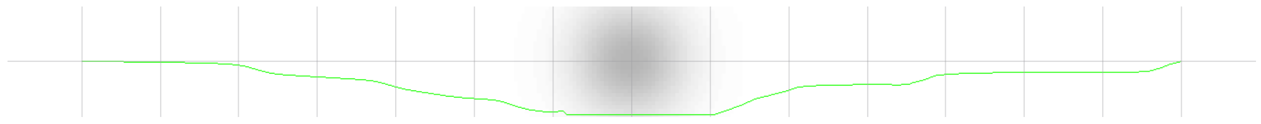
This is illustrated in Figure 6.5. In Figure 6.5a, with a relatively low values for σ and A, the planned path reaches the maximum possible $\hat{y}$. As the path passes the person, it is already as close to the wall as is possible. Hence, when the parameters are increased in Figure 6.5b, $\hat{y}$ remains constant.

Two paths with the same metric value does not mean that these paths are functionally equivalent. An additional metric is needed to encompass how the two paths are different in constrained environments such as this. In this environment, a key metric is how long the path stays butted up against the wall. In a social navigation situation, this corresponds with how quickly the robot moves to the side of the hallway. The ideal behavior is that once the robot detects that a person, it will indicate which side of the hallway it will pass the person on. In Figure 6.5a, the path only moves all the way to the side when it gets quite close to the person. With the increased parameters in Figure 6.5b, more time is spent hugging the wall.

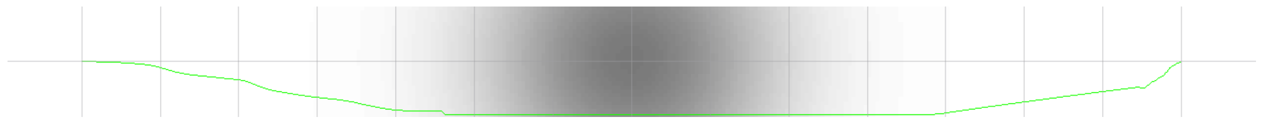
The behavior in Figure 6.5b is still not ideal, since it still spends a lot of time in the middle of the hallway. We can attempt to remedy this by increasing the parameters of the Gaussian. However, not surprisingly, this also results in the same behavior as we have seen throughout this chapter, in which the path reverts back to the shortest path possible when the parameters get too high. This can be seen in Figure 6.5c. When we expand our analysis to include additional metrics, the dead zone still exists. This is indicative of the real world problem encountered while tuning the parameters in Chapter 5.

6.6 Discussion

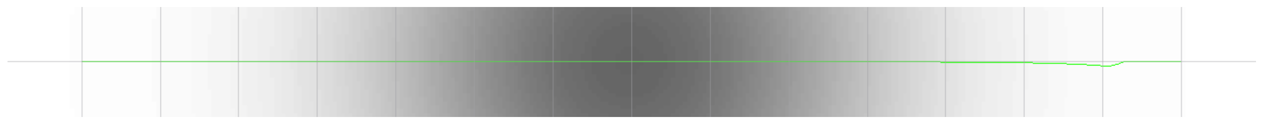
As robots require more and more social behaviors, it will be necessary to precisely design systems where the robot will navigate in a particular way. With this in mind, it is worth considering the placid assumption that the Gaussian costmap addition is the ideal tool for the job. In particular, the existence of the large dead zone where small changes to the parameters will have no effect on the path and the discontinuities when configuring just one parameter, together present numerous challenges to tackle when configuring a system. Additional and



(a)



(b)



(c)

Figure 6.5: Tuning Costmaps in Confined Spaces - Using ROS navigation, the above graphs show path planning with multiple different parameters in a confined hallway setting.

more complicated cost functions may alleviate some of the problems. Figure 6.3c shows the heatmap when using the sum of *two* Gaussian functions. This particular example creates greater range for $\hat{y}$ for small variances, but has its own problems with discontinuities. This is clearly not the only other option, and we advise that designers of new cost functions thoroughly test their new functions with the phase plots shown in this chapter.

The choice of costmap approach is vitally important for social navigation. The Gaussian adjustment explored in this paper are certainly better than no costmap modifications. However, in both tight environments like hallways and wide open spaces, getting the optimal behavior from the available set of parameters is nontrivial, and the stakes are high. Small adjustments to these parameters can result in the worst case scenario, which is a large risk when dealing with people and their personal space.

The lesson from this work is applicable to many types of costmap adjustments. This chapter's discussion has largely been focused on a single method for modeling personal space and a handful of metrics, but the takeaways are applicable to additional environments. When introducing non-lethal constraints to a costmap, there will always be a trade-off between path length and other factors. Different costmap adjustments will result in dead zones of different shapes, with different parameter values, but all adjustments will have the possibility of making longer paths so costly that the shortest path will be just as costly.

This would also be true in scenarios with multiple people and multiple personal spaces to plan around. Adding complexity to the costmap increases the odds of the optimal path being not the one that the costmap designer had in mind. However, the analysis here only addresses the parameters for a single Gaussian and the robot's relationship to a single person, to say nothing of the interaction between people and how multiple changes to the costmap would need to combine and interact. Additional people raises the complexity of both the parameter space and the analysis, and further research is needed to explore this vital space.

The larger implication for the field of human-robot interaction is that there needs to be an additional level of rigor added to finding optimal parameters. For a single experiment or use case, it may be sufficient to find parameters using trial and error. However, too often merely the winning parameters are published, without reference to all of the other parameter values that can result in disastrous behavior. Despite the social and psychological goals of much of the work in this field, the models used should still be investigated with rigorous mathematical analysis to ensure that there are not hidden pitfalls lurking within.

As a final note, social robots need to take many metrics into consideration to find the paths that will be best suited for interacting with humans. We do not claim that the sole definition of "best" path relies upon just path length and closest distance to an obstacle. Finding the best cost functions, and planners that optimize a wide variety of metrics, will likely see much fruitful work in the coming years.

Chapter 7

Layered Costmaps for Context-Sensitive Navigation

Our initial work to implement the behaviors and costmaps used in Chapter 5 drove our development of the data structure that came to be known as the *layered costmap*. The development process revealed to us that the standard data structure used for maintaining costmaps, which we call the *monolithic costmap* was inadequate for incorporating the additional semantic information into costmaps. In this chapter, we examine the features that the costmap requires, both to replicate the functionality of the previous navigation algorithm and to allow our context-specific changes. Then we discuss the new data structure and update algorithm that we designed to handle these cases, and how they have improved upon the previous approach. Finally, we will discuss the details of our algorithm, and how they improved the ROS navigation stack.

7.1 The Monolithic Costmap

In a traditional costmap, all of the data is stored in a single grid of values, in what we term a *monolithic costmap*. The monolithic costmap has been the prevailing technique because of its simplicity, in that there is only one place to read values from and write values to. However, proper maintenance of the costmap from cycle to cycle becomes more difficult as the number of types of values in the costmap increases.

For example, consider the relatively simple example of a conflict between the sensor data and the values already in the global costmap. The sensor data indicates that a certain area is clear, while the costmap indicates there is an obstacle. The correct method for updating the costmap depends on the origin of the data and additional semantic information. One scenario might be that the previous values indicated a prior position for a person who has now moved. Then, the correct behavior may be to overwrite the lethal values in the costmap with the new free values, allowing the robot to pass through the newly vacated space. However, an equally valid scenario is that the values in the costmap originate from the static map, which was created to include obstacles that cannot be seen by the sensors, such as glass walls. In that case, the lethal values should stay in the costmap.

The key problem with monolithic costmaps is that it is impossible to differentiate these two cases, as both present as lethal values in the costmap. Any semantic data for what the values in the costmap represent is stripped from the data as soon as it is reduced to a single value in the costmap.

The previous example revealed the inadequacies of the monolithic costmap in differentiating between just two data sources. The problem is exacerbated when additional types of data are integrated. If we would like to declare precedence between two different sensors, the monolithic costmap also lacks the semantic information to update accordingly. If, for example, one wanted to integrate a sonar sensor with laser range finder data, to avoid driving through glass walls, the sonar data may be given precedence. However, in the update process, the context of the values previously in the costmap are obscured meaning that the sonar values can easily be overwritten by the free values from the laser range finder.

Storing all of the obstacle information in a two-dimensional grid also can be problematic for properly handling three-dimensional obstacle data. The original developers of the ROS implementation encountered this problem when they used three-dimensional sensors like a tilting laser range finder. If the obstacle data is stored only in the monolithic costmap, obstacles at different heights could be inappropriately removed by clearing observations. Thus, they introduced voxel grids to keep track of the additional information[62].

The problem also presents when multiple non-lethal data sources are integrated, each of which will have individual semantic meaning. Consider the complexities of differentiating between a nonlethal value generated from the inflation step and the nonlethal value based on someone's personal space, like in Figure 7.1. The value in a given cell in this example is the sum of the two costs. However, in order to change or remove the costs for one of the data sources, the update algorithm would need to know the value of the two individual

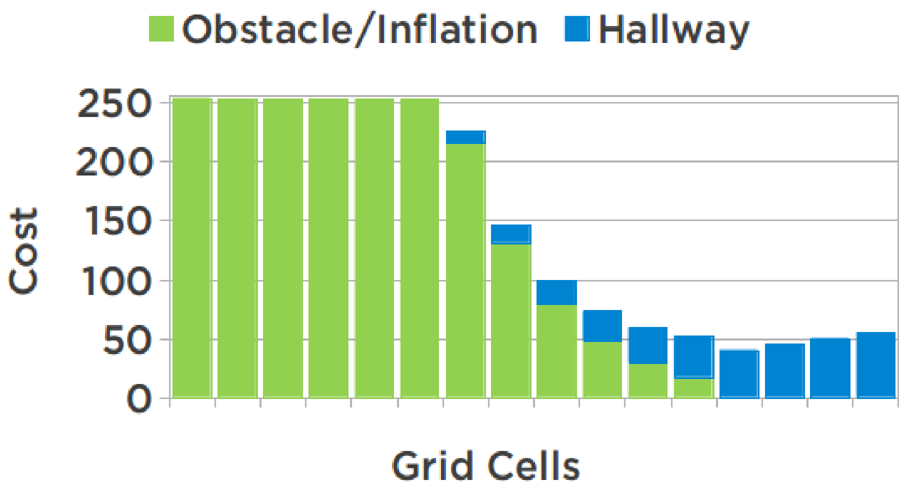


Figure 7.1: Overlapping Costmap Costs - This graph shows how to calculate the values in a row of grid cells using two overlapping costs. The first is the standard obstacle and inflation layer with its exponential drop off as distance increases from the obstacle, shown in green. The second is a linear hallway layer like the one used in Chapter 5, aimed at making the lowest cost be as far left as possible, shown in blue. The value in the monolithic costmap for each cell is the sum of the two costs.

costs, not just their sum. However, in a monolithic costmap, the only information that is maintained is the sum. Hence, a change in one of the values requires that both of the costs be recalculated. Such a recalculation would require additional semantic information, which would have to be maintained somewhere outside the monolithic costmap.

The monolithic costmap also only affords a single interpretation of the values in the costmap. Thus far the costs have represented the cost of being in that cell, but other use cases may require different interpretations. For example, it is common when using sonar data to create a probabilistic occupancy grid, where each cell's value is the probability that there is an obstacle in that cell. The real trouble lies in trying to combine a probabilistic map with a cost-based map. Since the values do not mean the same thing, they cannot both be stored in the monolithic costmap.

In addition to the limited storage capacity of the monolithic costmap, there is the problem of the exact update process to use to combine the disparate data sources. For a probabilistic occupancy grid with a single data source, the update process consists of the simple application of Bayes Theorem. It is similarly easy to define policies to update a costmap with relatively few data sources. However, as the number of sources increases, the policies needed to combine them become increasingly complex.

The lack of semantic information in the monolithic costmap also makes it difficult to tell how long any particular cost value has been in the costmap. Hence, if the updated area needs post-processing or to be published to some external source, there is no established way to determine the scope of the most recent updates.

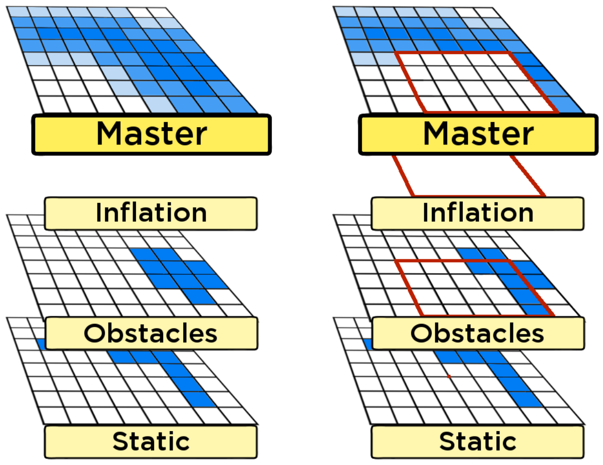
The lack of an established paradigm for maintaining and updating a costmap results in implementations that take an ad-hoc approach. This method has worked thus far due to the relatively small number of data sources used in practice, but it becomes infeasible as the number of sources increases. In order to ensure that the data is combined in the correct way, every data source needs to be aware of every other data source.

Even in the prior work that define useful algorithms for calculating costs, the process that they use to actually integrate them with their full costmap is usually opaque. Without the information about how precisely costmaps are updated, accurately replicating results becomes impossible.

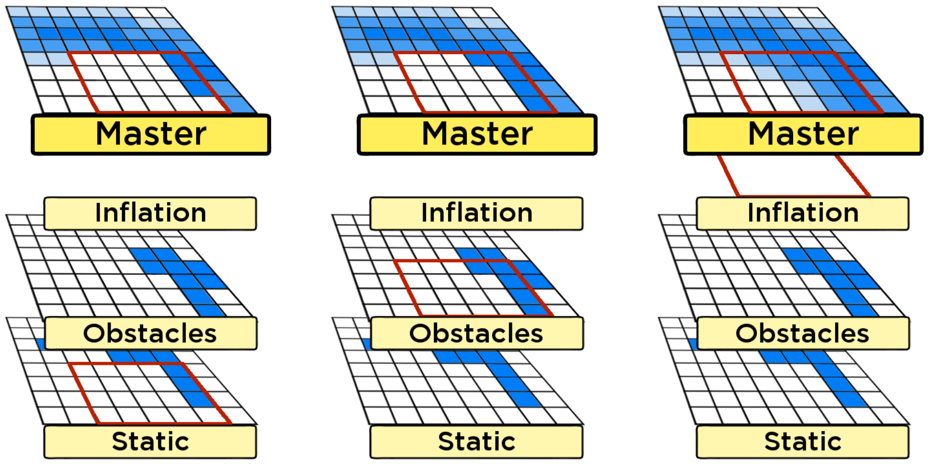
Storing all of the compiled cost data into this single simple data structure loses all the semantic information about where each value came from and what it represents. Once in a monolithic costmap, the value lacks context.

7.2 Layered Costmaps Data Structure and Update Algorithm

To counteract the limitations introduced in the previous section, we devised the *layered costmap*. The data structure still contains a two-dimensional grid of costs that is used for path planning. The key difference is how the values of this *master costmap* are populated. Instead of storing data directly in the grid, the layered costmap maintains an ordered list of *layers*, each of which tracks the data related to a specific functionality. The data for each of the layers is then accumulated into the master costmap in two distinct passes.



(a) Initial Costmap Values (b) Update Bounds



(c) Update Values: Static (d) Update Values: Obstacles (e) Update Values: Inflation

Figure 7.2: Update Algorithm - In (a), the layered costmap has three layers and the master costmap. The obstacles and static layers maintain their own copies of the grid, while the inflation layer does not. To update the costmap, the algorithm first calls the `updateBounds` method (b) on each layer, starting with the first layer in the ordered list, shown on the bottom. To determine the new bounds, the obstacles layer updates its own costmap with new sensor data. The result is a bounding box that contains all the areas that each layer needs to update. Next, each layer in turn updates the master costmap in the bounding box using the `updateValues` method, starting with the static layer (c), followed by the obstacles layer (d) and the inflation layer (e).

In the first pass, the `updateBounds` method, each layer is polled to determine how much of the costmap it needs to update. The layers are iterated over, in order, providing each layer with the bounding box that the previous layers need to update (initially an empty box). Each layer can expand the bounding box as necessary. This first pass results in a bounding box that determines how much of the master costmap needs to be updated. During the second pass, the `updateValues` method is called, during which each successive layer will update the values within the bounding box's area of the master costmap. Figure 7.2 illustrates the update algorithm using a set of layers that replicate the behavior of a basic monolithic costmap.

Some layers will maintain their own version of the costmap for caching results. This is one of the primary ways the data structure maintains semantic information about the data. For example, an obstacles layer keeps a private costmap of the same size as the master costmap to store the results of all previous ray-tracing and marking steps. During the updateValues step, the values from the private costmap are copied into the master costmap as needed. Since the values in the private costmap are only accessible to the particular layer, the information store within cannot be lost by another data source writing over it. Even if a value is not written into the master costmap, the semantic information is not lost since it is maintained in the individual layer. The real benefit can be seen in the example shown in Figure 7.1. The two data sources will be maintained individually, and thus each of the costs going into the sum will persist. This allows changes in one of the cost functions to not require the recalculation of all other values at that particular point.

Other layers do not require that much data to be kept from cycle to cycle and will update the master costmap with their data on each turn, or will simply operate on the data that other layers have already written into the master costmap.

The example in Figure 7.2 shows how the previous ad hoc approach used to generate the costmap can be refined into a neat, well-defined process.

The layered costmap implements the chain of responsibility design pattern[30]. Rather than have one class that is solely responsible for updating the values of the costmap, the layered costmap approach instantiates an ordered set of layers that can each update the master costmap in turn. This is a slightly modified version of the original design pattern, in that each layer has the chance to update the costmap, even if other layers have already updated the values. This stems from the need to encapsulate the interaction between the layers.

Some layers may need to change the values in the costmap placed there by other layers. For example, obstacles written by the static map may need to change based on new sensor data.

The `updateBounds` step of the algorithm provides each layer with the bounding box containing the update area of all the previous layers. This is counter to what may be the more intuitive process, which would be to simply collect each layer's bounding box, and then draw one bounding box that contained them all. This derives from the needs of the inflation layer, which needs to update the areas updated by other layers *and* an additional border to that area for obstacles that are outside the original updating area that would inflate into the updated area.

7.3 Benefits

The layered costmap approach specifically addresses the limitations of the monolithic costmap.

Clearer Update Step

Different types of costmap information are added to separate layers in the layered costmap approach, making the update step much easier. If the desired behavior included the ability to treat static obstacles, sensed laser obstacles and sensed sonar obstacles differently, storing those obstacles in their own layers simplifies the bookkeeping substantially. Each layer only needs to keep the information consistent with other information of the same type.

The layered costmap also eliminates contention between the competing costmap information sources. Each layer only needs to be updated as new information of that type comes in. If a layer remains largely static, it does not need to be recalculated each time another layer updates some sub-area. The static layer merely needs to update into the master costmap and the update can move on to the next layer.

This clearer separation of concerns also makes the individual components of the costmap easier to tune. Initial users can introduce one layer at a time and debug each in turn.

Ordered Update Process

As opposed to the undefined order in which elements in the monolithic costmap were updated, the layered costmap has an explicit ordering. In our example, it is clear that the inflation layer will inflate values from both the obstacles and static layers by virtue of the inflation layer coming after the other two in the ordered list. Furthermore, the interactions between layers are explicitly specified. Each costmap can be configured to combine the previous value and the layer's value as a maximum, minimum or some other mathematical function of the two.

Flexible Configurations

Finally, and most importantly, the capabilities of the layered costmap approach are endless. The layers needed to implement an equivalent set of behaviors to the previous implementation are only the beginning. As many layers as the robot operator desires can be added to the layered costmap. The result is that the individual layers can implement arbitrarily complex logic for updating the costmap, expanding the costmap's semantic possibilities. Each of the layers can also have its own independent representations of the data, such that probabilistic occupancy grids can exist in their own layers alongside cost-based layers.

7.4 Implementation

While there is much to admire in the way the costmap is implemented in ROS, ultimately it has the problem of being designed to achieve a single objective: enable collision-free efficient path planning. It was thus designed to maintain a monolithic costmap with only a handful of data sources. In order to implement the context-sensitive additions to the costmap, it was necessary to implement the layered costmap to work within the ROS framework.

7.4.1 Problems with the Previous Implementation

Fixed Update Areas

The area scanned for inflated obstacles is constant on every update cycle, regardless of how much of the costmap is actually updated by the sensors. Once information is placed into the map, it loses the information about when it was put there in addition to what kind of data it is. Thus, there is no established way to differentiate the old values from the new. The conservative update area is often larger than it needs to be, requiring the algorithm to recalculate the inflation values for old obstacles that were already inflated. In practice this means re-inflating a roughly 6m x 6m square around the robot, even when none of the values in the costmap have changed.

Inflexible Policy

When the robot receives sensor data that conflicts with values already in the costmap, the ROS implementation assumes that the clearing observations should take precedence over lethal obstacles, reasoning that the lethal obstacle was either a obstacle that moved or an error in the static costmap. This policy works in many scenarios, but can also cause problems. If there is noise in the data or improper localization, a static wall can be erroneously cleared, resulting in the robot planning a path through the wall. Practical experience with running the ROS navigation system on a PR2 robot has confirmed this tendency and the result makes the robot look quite unintelligent, since it tries to exit the room by driving to a corner of the room with no door. This stems partially from the limitations of monolithic costmaps, and ROS does not provide any additional mechanisms to change the robot's policy of how to deal with different contexts. ROS only allows for one specific use case and policy.

Mostly Binary Costmaps

The ROS costmap implementation has additional problems since the only type of information it accepts is binary obstacle data, i.e. where there are definitely obstacles or there is definitely free space. There is no way to introduce non-lethal values into the costmap in arbitrary locations, as only marking and clearing observations can be specified. The only non-lethal values that exist in the costmap are introduced through the inflation step as seen in Figure 4.3. The exponential decay used to calculate those costs depends on the inflation radius parameter. The documentation for the implementation notes that "this is a sort of default user preference."[14] There are no provided ways to change this preference, other than to change the value of the parameter. (For example, there is no way to change the nonlethal values to linearly decay.) Beyond this relatively inflexible buffer zone, there are not ways to add nonlethal values.

Muddled Update Process

From a software engineering perspective, the ROS implementation was quite complex to extend. All of the logic for handling the multiple data sources (static map, sensor data, inflation) is scattered across the `Costmap2DROS` and `Costmap2D` classes. Thus the logic for any particular data source is not encapsulated in any one class or location, but distributed into the core structure of both classes.

Furthermore, the update process described has many hidden intricacies. First, the step which clears all of the previous inflation data also wipes out any other nonlethal values that has been written to the costmap. Initially, we could only get nonlethal values we wrote to the costmap to exist for a single update cycle, since they would be cleared with the inflation data on the next cycle. Once we solved that problem, we found that the non-lethal values were not persisting in the area of the costmap closest to the robot. This was the result of the footprint clearing step, necessitating that our added values be calculated twice for the area within the footprint.

Additionally, all of these operations had to occur within the core navigation code, requiring a recompile of the entire navigation system with each change. This makes the navigation system very rigid to minor adjustments and makes debugging a longer process.

[14] `http://ros.org/wiki/costmap_2d`

7.4.2 Layered Costmap Implementation

Although the algorithm and data structure for layered costmaps are system-agnostic, due to the ubiquity of the platform, we focused on implementing the system to work with the ROS Navigation stack in order to demonstrate the capabilities of the approach. The layered costmap implementation keeps the `Costmap2DROS` API mostly intact and like the rest of the navigation code, is implemented in C++, as are each of the layers. `Costmap2DROS` remains the facade and provides access to the master costmap for path planning. The `Costmap2D` class no longer has logic for interpreting the sensor data or any other data source, and instead just acts as the data structure for holding the grid of values. All of the data source-specific logic now resides in the individual layers.

Implementing a layer requires a few simple steps. First, a new class must be created which extends the `costmap_2d::Layer` class. This means implementing the `initialize` function (where the layer can independently subscribe to any data sources in the ROS ecosystem), the `updateBounds` function and the `updateCosts` function. Independently compiled layers can be plugged in to the layered costmap with simple run-time parameter changes. This removes the need for having to recompile the entire navigation stack with every change to how the costmaps work.

Each of the layers can be implemented individually, but since many of them share the functionality of maintaining a private costmap which is then combined with the master costmap in some fashion, many of the layers are implemented using the template class `CostmapLayer`, which handles the basic bounds bookkeeping and the combining of the costmaps, leaving the subclassed layer to only need to populate the private costmap.

Whereas previously the costmap class had special cases to deal with whether there was a static map or not, and whether to track the obstacles in three dimensions, these cases are instead handled by configuring the global and local costmaps with different layers. Such changes can be made by either instantiating the static layer or not, or by instantiating either the obstacles layer or the voxels layer. This greatly reduces the amount of special case code within the `Costmap2DROS` class.

The implementation fixes the problems enumerated in the previous section, some just by virtue of having the layered costmap. With the layered costmap's update algorithm, the exact bounds of the updated areas of the costmap are tracked, allowing the inflation layer to only update the required areas of the costmap. This also reduces the bandwidth of publishing the costmap, since only the changes are published. Since how each layer combines with the previous layers is determined on a layer-by-layer basis, the policy is customizable per layer, allowing for any number of combination policies to be enacted. The previously rigid types of data the costmap could take as input become looser since the layers can be customized for any particular data type, thus allowing us to introduce nonlethal values wherever we like. Also, encompassing each data types functionality in a layer forces an enhanced level of encapsulation of related logic, and as a result, the update process becomes much clearer.

7.5 Discussion

In this chapter, we have discussed the benefits of the new layered costmap model over the previous monolithic model. Due to its efficiency and extensibility, an implementation of it has been adopted as the default navigation algorithm for the latest Hydro release of ROS, the source code for which can be found at `http://github.com/ros-planning/navigation`. All of the code for the additional layers can found linked to from `http://wiki.ros.org/costmap_2d`. Furthermore, based on the plugin-based layer structure, we hope that the new developments in creating additional costmap rules will be implemented as layers and tested within the ROS navigation framework, allowing for more open exchange of algorithmic behavior and more accessible comparisons between them.

The layered costmap and the associated layers open up the possibility for a wide range of additional robot behaviors. As more layers are assimilated into the planning algorithms, the robots will become more aware of different facets of their environment, and take those contexts into account while navigating. The current state of the practice is just to ignore the additional contexts, or tackle them one at a time. While the layered costmap does enable the contexts to be integrated, we predict that the future challenge will be to find a way to dynamically manage the collections of layers in order to ensure the right contexts are prioritized at the right times. Proxemic behavior dictates that personal space should be respected, but precisely how much less efficient the robot's path should be as a result is an open question. Half of the problem is designing costmap layers such that the mathematically optimal path is the desired distance away. The other half is a social question with unclear answers, of how to balance the needs of robots against the needs of people. While we can offer no concrete answers to such a question, we believe that having a highly customizable data structure for customizing robot behavior will make answering such a question much easier.

Chapter 8

Layers in the Costmap

In this chapter, we examine the different layers that can be added to the costmap. This includes layers whose values were already included in the ROS costmap, new layers to replicate the costmap adjustments of previous works and entirely new layers. Each layer gives us the ability to integrate environmental contexts into the costmap and have them treated in an equal manner to the other layers. With the power of the layered costmap, we can enable and disable any subset of these layers to produce the desired behavior in the robot.

8.1 Standard Layers

Static Map Layer

In order to perform global planning, the robot needs a map that reaches beyond its sensors to know where walls and other static obstacles are. The static map can be generated with a SLAM algorithm a priori or can be created from an architectural diagram. Figure 8.1 shows a static map generated with a SLAM algorithm.

When the layer receives the map, the updateBounds method will need to return a bounding box covering the entire map. However, on subsequent iterations, since it is static after all, the bounding box will not increase in size. In practice, the static map has been the bottom layer of the global costmap, and thus it copies its values into the master costmap directly, since no other layers will have written into the master before it.

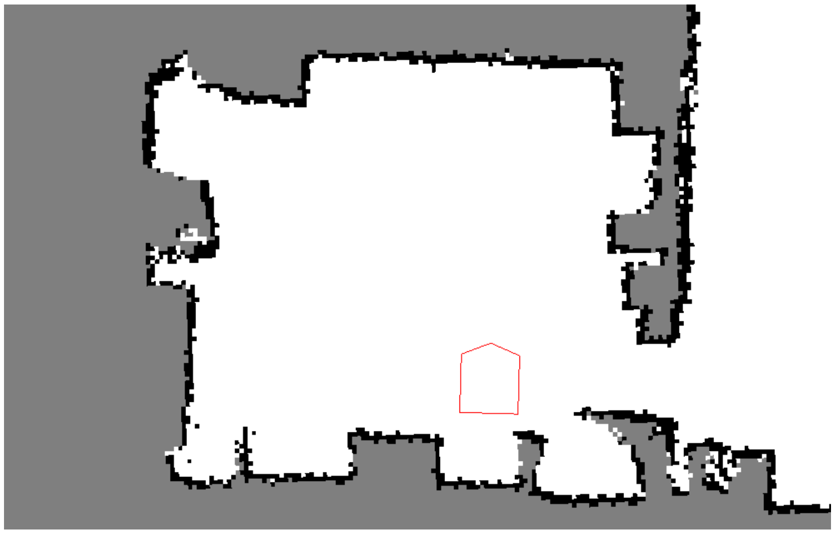


Figure 8.1: Static Map Layer

One additional behavior that the layered costmap enables is updating the static map without losing information. If the robot is running SLAM while using the generated map for navigation, the layered costmap approach allows the static map layer to update without losing information in the other layers. The monolithic costmap implementation that ROS used would overwrite the entire costmap with the new static map, thus abandoning all previous costmap data.

Obstacles Layer

This layer collects data from high accuracy sensors such as lasers and RGB-D cameras and places it in a two dimensional grid. The space between the sensor and the sensor reading is marked as free, and the sensor reading's location is marked as occupied. During the updateBounds portion of each cycle, new sensor data is placed into the layer's costmap, and the bounding box expands to fit it.

The precise method that combines the obstacles layer's values with those already in the costmap can vary, depending on the desired level of trust for the sensor data. We previously described this problem in 7.1. The default behavior that ROS used was to overwrite the static map data with the sensor data. This was most effective in scenarios where the static map may be inaccurate, such that you would rather trust the information that the sensors are

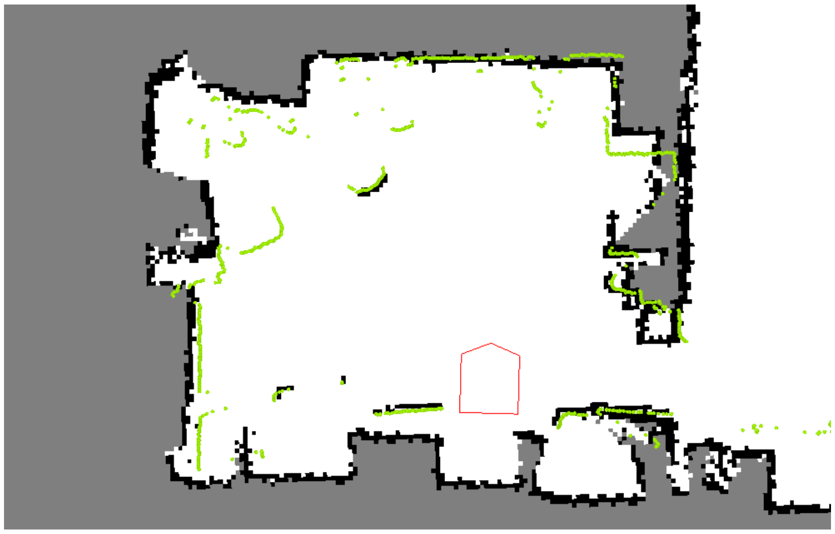


Figure 8.2: Obstacles Layer - Sensor data is shown in green

feeding you at the moment. This policy is still available in the layered approach. However, with the scenario from 7.1, where the static map is more trustworthy, then the layer can be configured to only add lethal obstacles to the master costmap.

The obstacles layer can also be tuned to improve the run time of the costmap update process. Both the layered and non-layered parts of the costmap that update the obstacles have parameters for limiting the spatial extent of the sensors. The costmap can be configured to, for example, ignore all sensor readings further than six meters away. While it does exclude some information from the costmap that could be useful, it also shrinks the area of the costmap that needs to be updated each cycle, thus improving the run time.

Voxels Layer

This layer has the same function as the Obstacles Layer, but tracks the sensor data in three-dimensions. The three dimensional voxel grid, introduced in Marder-Eppstein et al. [62] allows for more intelligent clearing of obstacles to reflect the multiple heights at which they can be seen. For example, in 8.3, both costmap layers receive sensor data about the legs and surface of the table. However, since the obstacles layer also sees many clearing observations that go under and over the table surface, the costmap marks those cells as free, which can lead to trajectories that try to drive through the tabletop. However, the voxels layer notes

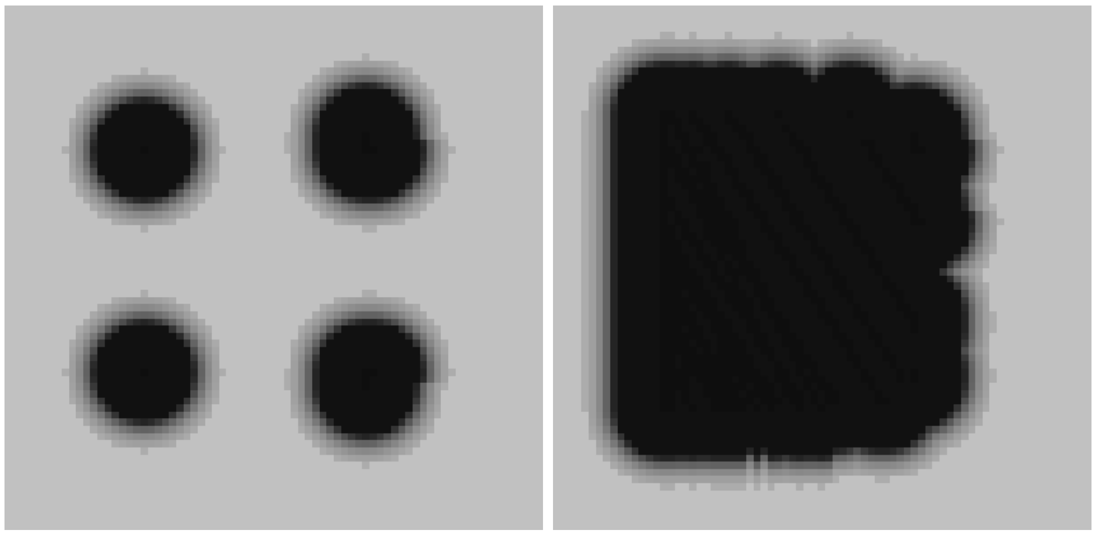


Figure 8.3: Obstacles Layer vs. Voxels Layer - The costmap generated by sensor data of a table, on the left from a two dimensional obstacles layer, and on the right from a three dimensional “voxels” layer. Both are shown with the inflation layer also active.

that the clearing observations are at a different level than the marking observations, and thus does not clear the marking observations, resulting in the correct costmap on the right.

Inflation Layer

As discussed earlier, the inflation process inserts a buffer zone around each lethal obstacle. Locations where the robot would definitely be in collision are marked with a lethal cost, and the immediately surrounding areas have a small non-lethal cost. These values ensure the robot does not collide with lethal obstacles, and prefers not to get too close. The updateBounds step increases the previous bounding box to ensure that new lethal obstacles will be inflated, and that old lethal obstacles outside the previous bounding box that could inflate into the bounding box are inflated as well. The updateValues step operates directly on the master costmap, without storing a local copy. Figure 8.4 shows the same costmap as 8.2 with the inflation layer added.

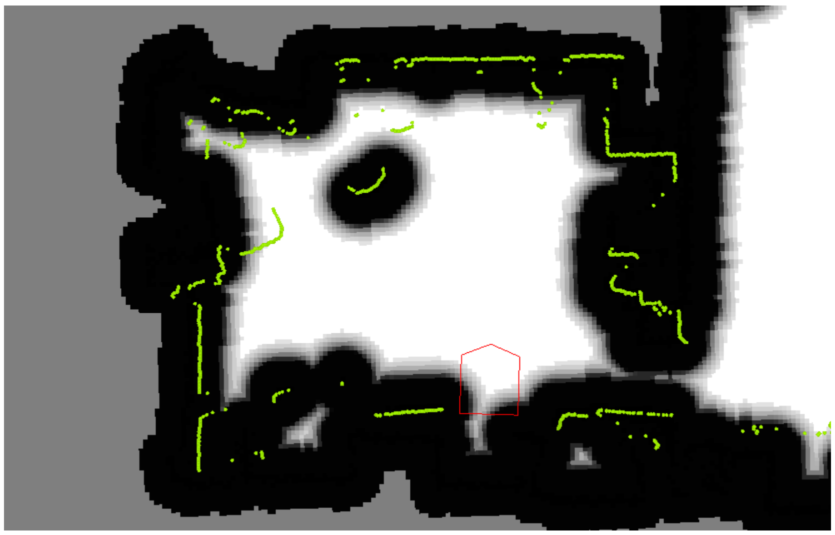


Figure 8.4: Inflation Layer

8.2 New Functionality

Sonar Layer

Monolithic costmaps are capable of handling sonar data, but layered costmaps increase the options for how to deal with it. Dedicating a layer to sonar readings can avoid problems with glass walls being cleared out by laser observations. Furthermore, we can also use this layer to treat sonar data differently than the high accuracy obstacles layer. The sonar layer we built implements a probabilistic sonar model and updates the costmap using Bayesian logic. We can then set a cutoff probability in which we only write data that we are relatively sure about into the master costmap. Note that this approach allows us to maintain the semantic meanings of the probabilities without having to directly combine them with the costs. An example can be seen in Figure 8.5.

Caution Zones Layer

This layer gives us the ability to specify areas of the robot environment with greater detail than free/occupied. Despite being free of obstacles, most robots will want to avoid navigating into stairwells leading down. Or perhaps the robot should never navigate into a particular

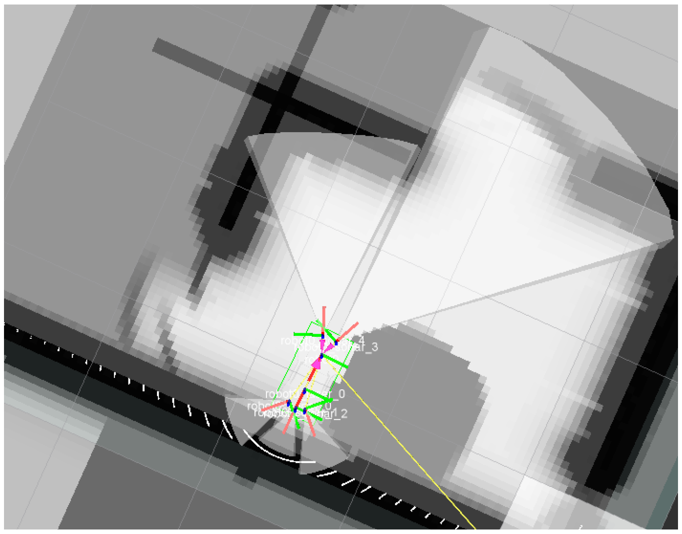


Figure 8.5: Sonar Layer - Shown running on one of Stanford's Autonomous Vehicles

person's office. There are numerous scenarios where operators will want to restrict where the robot can safely drive, despite appearing navigable. One technique seen in practice for these restrictions is to mark obstacles on the static map. This technique can work, but removes information from that map that might be needed for other applications, such as AMCL. This layer also affords us the ability to mark zones that are not necessarily forbidden, but not desirable. Adding a non-lethal cost to a kitchen can ensure the robot does not drive near hazardous liquids unless there is no other option. These zones can also be used for areas where it would be socially less acceptable to be, such as the space between a person and an object they are interacting with, like a TV, as seen in Ferguson and Likhachev [26].

Claustrophobic Layer

The inflation layer adds a small buffer around lethal obstacles, but the claustrophobic layer adds a larger buffer to increase the relative cost of driving close to obstacles. As a result, the robot would prefer to move in wide open spaces as far from obstacles as possible, thus maximizing the clearance to any sensed obstacles. This layer would be useful for scenarios with more uncertainty about the exact location of the robot relative to obstacles and the odds or costs of driving into an obstacle are high. An example can be seen in 8.6.

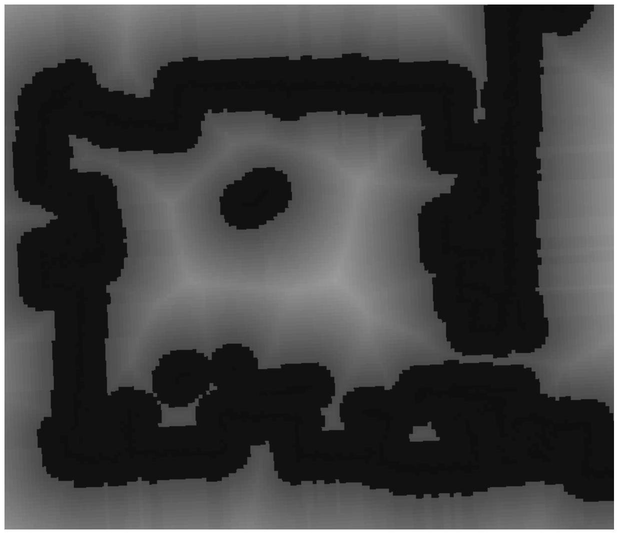


Figure 8.6: Claustrophobic Layer

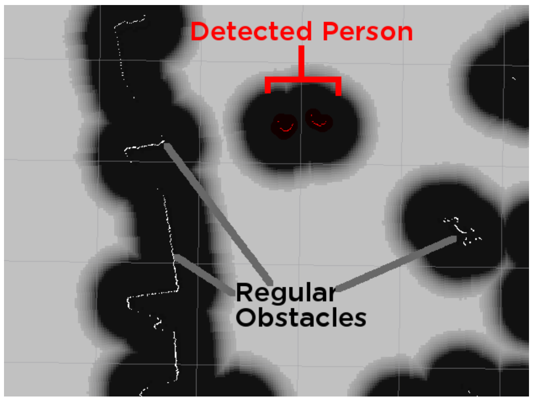
8.3 Human-Robot Interaction Layers

As seen in Section 4.3, one of the primary motivations for adding more complex costs to the costmap is for modeling constraints introduced by human-robot interaction.

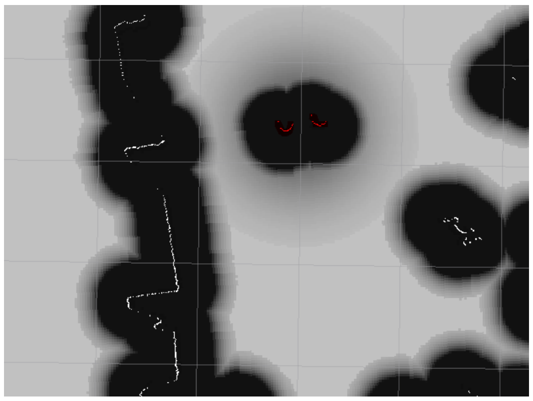
Proxemic Layer

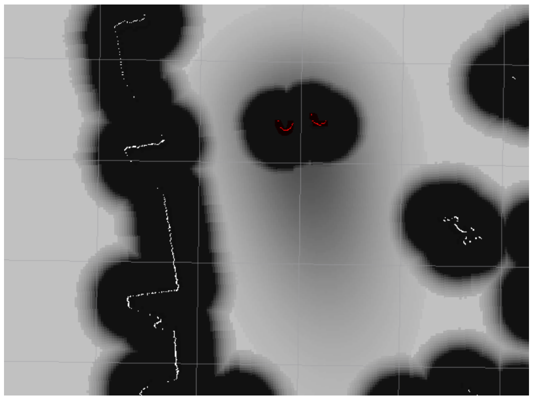
There has been a steady rise in the study of the spatial relations between people and robots, as well as methods for ensuring robots do not violate the expected relations. The most common way this is done is by adding Gaussian distributions, or mixtures of multiple Gaussian distributions, to costmaps, as in Kirby et al. [44]. These adjustments create areas around detected people that makes paths passing closer to people more costly, respecting their proxemic concerns.

We created a proxemic layer which implements Kirby's mixture of Gaussians model. Using the location and velocity of detected people (extracted from laser scans of the person's legs), the layer writes the Gaussian values for each person into the layer's private costmap, which are then added into the master costmap. Figure 8.7a shows the standard costmap with no proxemic values added, while Figures 8.7b and 8.7c show Kirby's proxemic model, while the person is stationary and moving respectively.



(a) Standard Costmap


(b) Proxemic Costmap (stationary person)


(c) Proxemic Costmap with Velocity

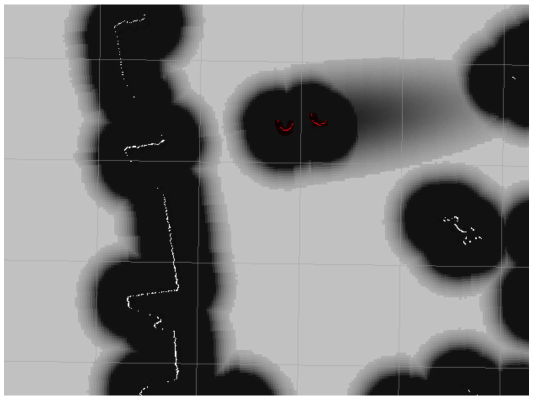

(d) Passing Costmap

Figure 8.7: Social Costmap Layers

Hallway Layer

In some cultures, there is the custom of walking on the right side of pathways, much in the same way drivers in many countries stay to the right side of the road. We implemented a layer that determines whether the robot is in a hallway and dynamically will increase the cost on the left side of the hallway to have the robot prefer the right side. We used a similar model in a recent user study [58] where the layer changed costs to make the robot to prefer navigating on the opposite side of the corridor as the closest person to it (which was often the right side). The addition of this layer was shown to not only effectively move the robot to one side of the hallway, but also to make the person behave more effectively during the interaction.

Another approach to determining the correct side of the hallway is to add costs on one side of each detected person, to ensure that the robot passes on the correct side, as demonstrated by Kirby et al. [44]. Figure 8.7d shows just such a costmap.

Wagon Ruts Layer

If the robot aims to avoid being socially invasive and minimize unexpected obstacles, one effective strategy could include mimicking human traffic patterns. This layer can decrease the cost of paths that people have traveled on, resulting in the robot's optimal path to follow them as well. You could also reverse the polarity of the costs, and increase the value in areas where people often are in order to minimize social disruption.

8.4 What There Are Not Layers For

Layered costmaps can encapsulate a wide range of new behaviors, but they cannot possibly be used for creating all sorts of behavior.

The primary limitation on the costmaps presented in this paper is that they cannot represent the passage of time well. The general approach that the robots use to dynamically avoid obstacles is to update the costmap and replan with a high frequency. This is adequate for obstacle avoidance but does not work well for higher order logic. For example, there is no specific logic in the costmaps that can represent a blocking obstacle that is about to be cleared. For example, there is no way to represent a door that is temporarily closed that will open again. The costmap can only represent it as being blocked at the current moment.

Besides constantly replanning, the other way that it is possible to take dynamic obstacles into account is by predicting where they will go. This is what the proxemic layer does when it tracks the velocity of people and increases the costs in front of person, in the area that they are about to occupy. This will create a preference for the robot not to navigate in front of people, but is still quite limited in its predictive ability.

Chapter 9

Engineering the Costmap

The previous chapter described the functional limitations and desired capabilities that lead to the overall design of layered costmap data structure and algorithm. However, the abstract data structure alone will not instruct the robot to move. As discussed in Section 1.4, a well-designed implementation is needed to fully demonstrate the capabilities of the system. The implementation also needs to be freely available to allow others to use and test the claims. This section examines the practical necessities of fleshing out such an implementation, featuring analysis from a software engineering perspective.

9.1 General Requirements

9.1.1 What is a costmap?

There are three major design forces that led to the basic costmap as a data structure. The first two are that the robot can only navigate in open space, and that the space around the robot can change. Hence, it must keep track of where obstacles are (and are not) over time. The next design force is that the robot needs the ability to use standard graph theory algorithms for path planning. One of the most common solutions used by the field (and us) is to represent these obstacles' locations as connectedness graphs.

There are two different approaches that navigation algorithms can take with regard to how complex of a graph to create. There are situations in which the locations represented by the graph are non-uniformly distributed, and each of the nodes represents a space of variable size, in what is called a *topological graph*[63, 101]. This allows for efficient graph algorithms and less space needed to store the graph. However, such an approach is complex, is difficult to update, and can result in suboptimal paths[101]. The alternative approach utilizes a uniformly distributed graph, where each node has a fixed area, and the nodes are arranged in a two-dimensional grid with the geometry of the grid reflecting the geometry of the world. These graphs are generally referred to as *grid-based* or *metric* maps. In this chapter, fixed-area locations in the graph are referred to as *cells*. A value associated with each cell corresponds with the robot's knowledge about obstacles at a particular location. This value is referred to as the *cost*. The entire graph data-structure maps these costs to particular locations, and is known as a *costmap*.

Observation 1. *Robots need to track information about obstacles in a uniform way that is quick to update and easily changed.*

Conclusion 1. *Information about obstacles is stored in a two-dimensional grid with each cell representing the robot's knowledge of obstacles at a particular location.*

There are at least three different pieces of knowledge that the robot needs to be able to store in a cell:

1. No knowledge (a.k.a. unknown)
2. Obstacle (a.k.a. lethal cell)
3. No obstacle (a.k.a. free space)

In general, the planning algorithms need to avoid collisions between the robot and the lethal cells, primarily by staying in free space. Whether the robot is allowed to plan paths through unknown space is a user preference, determined by their knowledge of context in which the robot is navigating in. For instance, there could be a scenario where the robot knows there is a hallway to the left that leads to the goal. The hallway on the right could have a path that gets to the goal quicker, but the length of that hall is unknown. The user must decide whether to allow the robot to venture into the unknown hallway based on the particulars of the task, whether it is worth it to try the shorter path at the risk of failing.

These three categories can be further subdivided into values with differing semantics. Specifically, the two different known values (obstacle and no obstacle) can be represented by a range of values with a specific cutoff value, where values below the cutoff are considered free space and those above are considered lethal. Having multiple lethal values allows for distinguishing between multiple types of lethal values, which is necessary for certain operations discussed in section 9.1.2. Multiple different values for free space allow for associating additional costs with each cell, to enable behaviors like the robot's hallway behavior seen in Chapter 5, the proxemic behavior seen in Chapter 6, or a probabilistic approach to representing the cost as seen in Section 8.2. These values are referred to as non-lethal values. Semantically, if the cost of a given cell is high enough, the robot should never plan paths there, which is why these values exist on a continuum with a cutoff point.

Observation 2. *Robots need the ability to track three different states of knowledge about a particular point: no knowledge, knowledge of an obstacle, and knowledge of no obstacle.*

Observation 3. *The robot may need to associate different costs with each point, with a relative measure of the desirability of traversing a particular cell.*

Conclusion 2. *The costmap stores the cost c of being in each cell as a number with one of the following semantic interpretations:*

- $c = 0$*: No obstacle, no cost*
- $0 < c < c_{lethal}$*: No obstacle, some cost*
- $c \geq c_{lethal}$*: Obstacle, or other high cost, and cell should not be traversed.*
- $c = c_{unknown}$*: Special case label for unknown information.*

In the ROS implementation, these values are stored as an `unsigned char`, with $c_{unknown} = 255$ and $c_{lethal} = 253$. The `unsigned char` data-type was chosen for its small memory requirement and its fast integer operations. It is unsigned to maximize the range of positive costs available.

Particular attention must be paid during the update step for a costmap to how these different values are combined, and what the semantic result of the operation is. We discuss the particulars more in the upcoming sections.

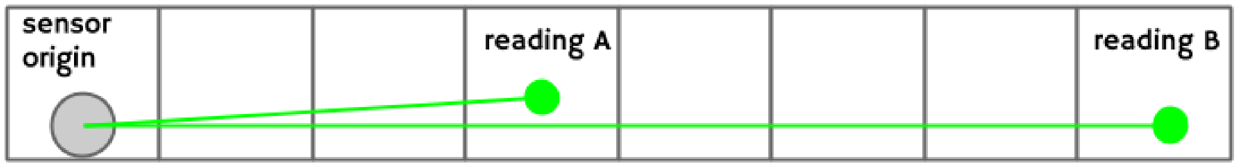


Figure 9.1: Marking and clearing obstacles - Each sensor reading is cleared in the costmap, and then each sensor reading is marked. Otherwise, in the above case, if reading A was marked and cleared, and then reading B was marked and cleared, the mark from reading A would be overwritten, thus creating an unmarked obstacle.

9.1.2 Data Sources

There are three primary sources of data that are used for constructing costmaps: live sensor data, prior data and the robot's footprint. In this subsection we discuss each and how they influence the costmap.

Sensor Data

There are two pieces of information that can be derived from high accuracy sensors such as laser range finders. First, the robot can determine where there *is* an obstacle by determining the location of the end point of the sensor reading, i.e., where the obstacle caused the laser beam to be reflected back. Second, given the location of the sensor, the robot can also determine that there is no obstacle between the sensor and the end point of the sensor reading. To integrate this information into the map, the typical implementation is to mark the grid cell corresponding with the end point as a lethal obstacle, and then use a ray tracing algorithm to mark all of the cells between the sensor and the sensor reading as free space. These two processes are called marking and clearing respectively.

As new sensor readings arrive, the information is buffered and integrated into the costmap in two passes, first for the clearing step, then for the marking step. If it was performed in a single pass of each sensor reading, the clearing might erase obstacles it should not, as shown in Figure 9.1. As a result, the approach is cautious, prioritizing marking information over clearing information.

Observation 4. *Sensor data provides two types of information, cells that should be marked as lethal, and those that should be cleared of lethal values.*

Observation 5. *Due to the limited resolution of costmaps, and the fallibility of sensors, the two types of information can disagree with each other for a given sensor reading.*

Conclusion 3. *To minimize collisions with obstacles, marking data should be prioritized over clearing data.*

Prior Data

The planning algorithms also need to be able to reason about areas of the world that are outside of the robot's current sensor view. This is most frequently provided as a static map (i.e., not live data) created with a mapping algorithm. Semantically, the static map can be interpreted in two different ways. First, it could be used as the initial values of the costmap, any of which could change with updated sensor data. Alternatively, the static map could represent the ground truth, to be maintained even when it disagrees with sensor data. The choice of representation is made by the costmap designer and is discussed in section 7.1.

The Robot's Footprint and Inflation Data

The inflation process adds costs around each obstacle as a computational optimization that allows the robot's path planning algorithm to represent the robot as a point, and not a two-dimensional object that spans multiple costmap cells. This approach performs the planning in the *configuration space* of the robot[54].

The basic geometrical reasoning of the inflation process is shown in Figure 9.2. The two key parameters that affect the spatial extent of the inflation data are robot's radius, and what is known as the inflation radius. Cells whose distance to the nearest obstacle is less than the robot radius are marked as having a lethal cost. It is also possible to add costs outside of that radius, but within the radius defined by the inflation radius parameter. The standard ROS behavior inserts non-lethal costs to have the robot prefer to not be too close to the obstacle.

Given a newly updated obstacle, it is clear that the algorithm will need to update the values of all of the adjacent cells within the inflation radius. However, the value of these adjacent cells depends on more than just the newly updated obstacle. In Figure 9.2, if the algorithm

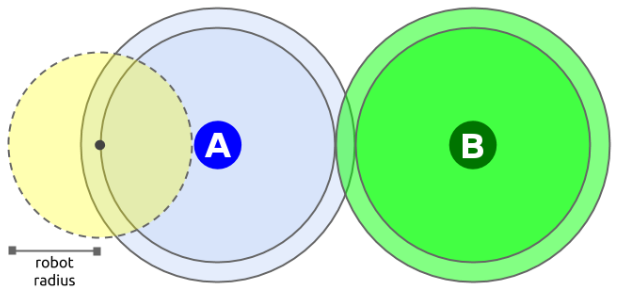


Figure 9.2: Inflation Geometry - Pictured in this figure are a circular robot (shown in yellow with a dotted outline) and two obstacles A and B. The robot will be in collision if the center of the robot is less than the robot's radius from an obstacle. In this figure, this is represented by the first circle around each obstacle (medium shading). The slightly larger circle (with the lightest shading) represents a buffer zone where the cost is increased slightly to bias the robot toward not driving too close to the obstacles.

adds obstacle A to the costmap, then all of the cells within the robot's radius will need to be updated (marked by the largest circle surrounding A). However, that requires checking for lethal obstacles like B within a radius twice the inflation radius.

Observation 6. *The inflation process changes the values of cells based on the distance to the nearest obstacle.*

Observation 7. *If the value of a cell changes whether it is lethal or not, the surrounding cells need to be recalculated.*

Conclusion 4. *Updating the lethality of any particular cell can require looking at all of the cells in the surrounding area at distances as large as twice the inflation radius.*

It should be noted that in ROS, the lethal value of inflated obstacles is different than that of the actual obstacle. The actual obstacle has value 254, whereas inflated obstacles are 253. Both are greater than or equal to the lethal threshold 253, so both are considered lethal, but since they have different values, the update process can differentiate them and update them accordingly.

9.2 Starting Point

All of the aforementioned design requirements were implemented in the ROS Navigation Stack. This section describes the approach that was taken and some of its limitations.

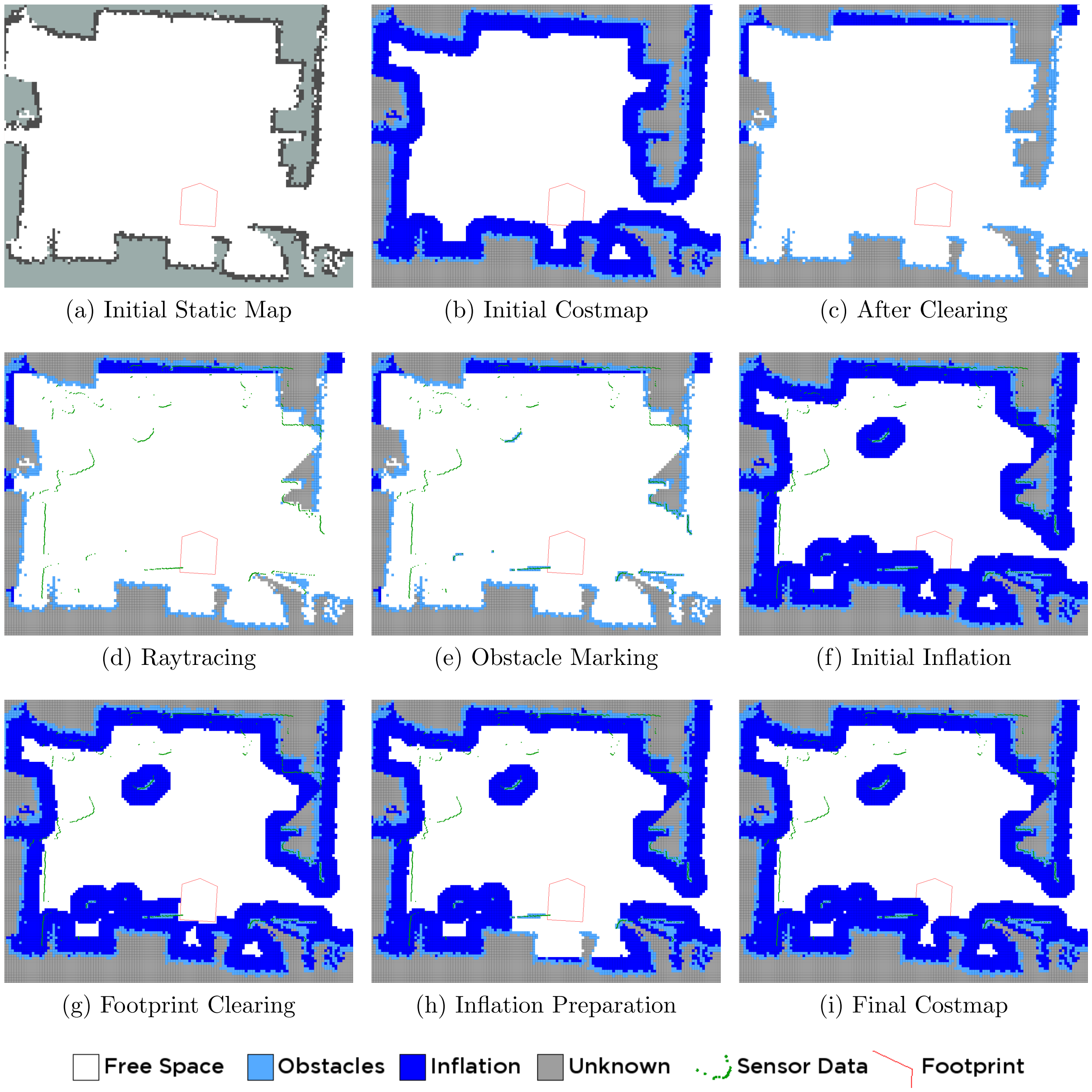
9.2.1 Original ROS Implementation

The costmap was implemented in C++ as two classes: `Costmap2D` and `Costmap2DROS`. The costmap object as it is seen in Figure 4.1 acts as a bridge for all planning operations. In the ROS implementation, this bridge is the `Costmap2DROS` class, which contains within it the `Costmap2D` class which holds the actual cost data. The separation of the two classes is ostensibly for dividing the ROS-specific logic from the a ROS-independent data structure, which could theoretically be compiled into a non-ROS-based C++ project. However, the actual implementation shows that the separation is fuzzy. The `Costmap2DROS` class uses the ROS messaging system to subscribe to the static costmap and sensor data streams and then passes it on to the `Costmap2D` in non-ROS-based data structures for integration.

The Update Process - Preparation and Sensors

The static map's values (Figure 9.3a) are filled into the `Costmap2D` at initialization and are not used in the update process.

The actual update process is as follows. The update starts with values already in the map (including the inflated areas) as in Figure 9.3b. The first step is to clear our all of the inflated values around the robot, as seen in Figure 9.3c. This is done on a large swath of the map, the exact size of which we will discuss shortly. Next, in Figure 9.3d, the sensor data is used for raytracing purposes, clearing out many of the previous obstacles and marking previously unknown areas as free space. Then the sensor data within a particular range is marked as lethal obstacles, as seen in Figure 9.3e. The range is specified by the *max_obstacle_range* parameter, and this limit explains why not all of the sensor readings in Figure 9.3e are input into the costmap.



(a) Initial Static Map (b) Initial Costmap (c) After Clearing

(d) Raytracing (e) Obstacle Marking (f) Initial Inflation

(g) Footprint Clearing (h) Inflation Preparation (i) Final Costmap

(j) Key

Figure 9.3: Original Costmap Update Process

By default, the ROS implementation tracks the obstacles in two-dimensions, but it also can track obstacles in three-dimensions to achieve the behavior described in Marder-Eppstein et al. [62]. This is done by marking and clearing obstacles in a three-dimensional voxel grid. This is implemented in the class `VoxelCostmapROS`, a subclass of `Costmap2DROS`. However, ultimately these obstacles get projected into two dimensions in the costmap.

The Update Process - Inflation

Next, the update algorithm needs to inflate the cells. The inflation needs to be updated for all the new obstacles, which could be up to *max_obstacle_range* away. The ROS implementation makes two assumptions for simplicity. First, the algorithm assumes that there might be obstacles *max_obstacle_range* away, updating a possibly larger region of the map than is necessary. Secondly, the update algorithm actually updates the values in a grid-aligned square portion of the costmap, which is also more of the costmap than strictly necessary. The actual size of the updated area is influenced by the geometry discussed in Figure 9.2. The inflated values can appear up to $max\_obstacle\_range + inflation\_radius$ away, and the algorithm must scan for obstacles up to $max\_obstacle\_range + 2 * inflation\_radius$ away, resulting in a $2 * (max\_obstacle\_range + inflation\_radius)$ square region that needs to be updated. It is this area that is cleared in Figure 9.3c.

The actual inflation process starts by scanning for obstacles and inserting those cells into a queue, and then performing a breadth-first search from all the lethal obstacles. As each cell is visited, a value $f(d)$ is written into the cell based on the distance d to the nearest obstacle (with R_r is the robot radius and R_i is the inflation_radius).

$$f(d) = \begin{cases} 254 & d = 0 \\ 253 & 0 < d \leq R_r \\ exp(-k(d - R_r)) & R_r < d \leq R_i \\ unchanged & d > R_i \end{cases}$$

The resulting values are seen in Figure 9.3f.

The Update Process - Footprint

The update process could be finished here, except for one quirk: sensor readings that indicate the robot is already in collision. This edge case can be caused by the robot sensing part of itself or a lack of precision in the costmap. Either way, if the robot thinks that it is already in collision, then there will be no valid plans (as all plans would start with a collision). Hence, the ROS costmap removes all sensor data found to be within the robot's defined footprint. This is done after the inflation step by first clearing all data from the costmap within the robot's footprint (Figure 9.3g). This also clears the inflation data, so the area needs to be re-inflated, so the square area around the robot is cleared (Figure 9.3h) and then reinflated (Figure 9.3i).

9.2.2 Limitations of Implementation

There are several limitations of the original ROS implementation. In Chapter 7 we discuss the limitations of the monolithic costmap as a data structure. However, there are additional implementation specific problems in the implementation of the data structure that we will discuss here.

Size of Update Area

One of the largest problems is the way that this update algorithm conservatively updates areas of the map. No bookkeeping is performed to figure out what area of the costmap is actually updated. Instead, the algorithm conservatively figures that it could have seen obstacles up to *max_obstacle_range* away, which is already likely more of the costmap than needs to be updated, and then updates an even larger square area. Given that this approach requires each cell in the update area to be iterated over at least twice (once for the initial clearing, once during the scanning for lethal obstacles prior to inflation, and possibly a third time for updating the inflation data), conservatively updating too-large areas of the costmap recalculates the values of many cells, only to have their values not change.

Observation 8. *Without tracking which parts of the costmap were updated, the inflation process requires updating conservatively large areas of the costmap.*

Observation 9. *The conservatively large update area means that linearly increasing the maximum obstacle distance would quadratically increase the update area.*

The fact that the processing time is proportional to the update area is one of the reasons why the *max_obstacle_range* parameter was implemented in the first place. Laser range finders can sense obstacles tens of meters away, while in practice, the *max_obstacle_range* is limited to a couple of meters. For the standard PR2 setup, the laser can see 60 meters away, but the costmap is limited to updating 2.5 meters away. Confining the updates to that area can result in quicker updates, but at the expense of possibly useful information. Information about obstacles outside of the range could be useful for long-term planning, but in this system, is ignored for the sake of update expediency.

Conclusion 5. *To avoid large update times, the size of the update area should be limited.*

Order of Updates

Another problem with the implementation is the piecemeal ordering to the costmap updates. The first step (clearing the nonlethal values) is related to the inflation process. This is followed by updating the obstacle information (raytracing and obstacle marking), then another inflationary step. At this point the footprint is cleared followed by yet another round of inflation. While this implementation worked, the design was far from elegant. Trying to extend the functionality of the costmap beyond its initial capabilities required careful integration with the existing components.

As an example, consider the process of clearing the robot's footprint. The need to remove lethal obstacles from the robot's footprint was likely discovered after the initial design of the costmap. Thus, the decision was made to remove the lethal obstacles after the costmap had been updated, and before the path planning began. With the obstacles and primary inflation happening in the `Costmap2D` method $updateWorld$, the footprint clearing was added in `Costmap2DROS` after $updateWorld$ was completed. This required recalculating the inflation values in the area around the robot's footprint as well.

Again, this solution works, but fails to efficiently deal with the relationships between the key portions of the update process. The inflation data depends on the locations of the lethal obstacles. The given approach changes the number of lethal obstacles within the footprint, thus necessitating reinflation. However, using the gift of hindsight, it is apparent that the more direct approach is to clear the footprint before the initial inflation step, thus eliminating the need for the second inflation. The lack of awareness about the other components of the update process led to the convoluted ordering of steps.

Observation 10. *Updating the costmap relies on knowledge of the dependencies between the different components of the costmap.*

Conclusion 6. *Adding new components to the costmap requires careful consideration of the other components of the costmap.*

9.2.3 Extending the Approach

The other key limitation of this implementation is the restrictions on the type of data that can be integrated into the costmap. This was motivated by our desire to change the robot's navigation around people.

Observation 11. *Social navigation requires that people be treated differently than other obstacles.*

Conclusion 7. *Social navigation can be achieved by increasing the cost in the costmap specifically around people.*

In the implementation discussed so far, the only ways to increase the cost are to add obstacles or change the inflation process. Since changing the inflation changes all obstacles and not just the people, that is not an option. Adding additional lethal obstacles around people does accomplish the goal of increasing the cost, but also disallows some paths which could be needed to accomplish the goal. Hence, there needed to be a way to insert non-lethal costs into the map, other than through the inflation process.

Our initial attempts to insert non-lethal values into the costmap underscored the lessons enumerated in the previous subsection, that adding new components to the costmap requires careful consideration of all of the other components. First we tried inserting the values by accessing the `Costmap2D` object and writing the values into the costmap directly. However, these values were quickly overwritten by the actual update algorithm, which cleared all non-lethal obstacles to prepare for inflation. We then added a step in `Costmap2D`'s $updateWorld$ step which wrote the non-lethal values into the costmap. However, these values were being cleared in the vicinity of the robot by the robot's footprint clearing step. The solution we chose at the time was to recalculate the non-lethal values near the robot's footprint after the footprint clearing step.

This solution worked, properly writing the values into the costmap, which were appropriately used by the planning algorithms. This methodology worked well enough to be used for the experiment in Chapter 5. However, the process for getting it working required knowledge of precisely how the obstacles were updated (dictating the area of the update), how the inflation values updated (to not get overwritten) and how the footprint clearing step worked (to not get cleared). The unstructured nature of the update process will only grow more complex as additional features are added.

9.3 Convergence on Layered Costmaps

The complexities of multifaceted costmaps led us to examine a more structured solution. It is possible to keep extending the `Costmap2DROS` object to include functionality for integrating additional sources of data. This, however, grows more complex since each source updates the costmap in its own way. Even switching out one component to another of a similar type can be tricky. This led to a thorough examination of the properties of how each source changes the costmap.[15]

There are two key properties that define how a data source changes the costmap: how the values change, and how many of the values change.

9.3.1 How the Values Change

First, how each data source changes the values can be classified into loose groups.

Observation 12. *Some data sources always overwrite the previous values in the costmap. Others are combined with the values that are already there.*

For example, the obstacles component always overwrites the previous values with new lethal and free information. On the other hand, the inflation component needs to know where the obstacles are in order to inflate the right areas, and the non-lethal data sources may want to add to/combine with the values already in place. Due to the different ways the sources change the values, the update algorithm needs the ability to combine values from the various data sources to calculate the new values.

Observation 13. *Some data sources raise the values of the costmap, some lower the values, and some do both.*

The inflation process always raises the values of cells in the costmap, while the footprint clearing process always lowers the values. The obstacle tracking raises and lowers the values. Due to these changes, combining the costmap values cannot be implemented as a simple operation like taking the maximum of each data source's value for a cell.

[15]This was initially motivated by attempting to find a generalized method for updating the non-lethal values using different methods.

9.3.2 Which Values Change

There are also recurring patterns in which values change.

Observation 14. *For most data sources, the amount of the costmap that needs to be updated can be determined solely by the data itself.*

For example, the obstacles update an area based on where the sensor data is; the footprint clearing depends on where the footprint of the robot is.

However, this is not true for the inflation process. The inflation process is unique for ways explained in the next two observations.

Observation 15. *The amount of the costmap that the inflation process needs to update is determined by how much of the costmap the other data sources updated.*

If it were not for this observation, the portion of the costmap that needed to be updated could simply be a union of each data source's update portion.

Observation 16. *The inflated value of a cell not only depends on the previous value of that cell, but other cells around it.*

This observation means that each cell's value can not be considered in isolation. When including the inflation process, the costmap cannot be updated by combining all of the data sources for one cell, and then moving on to the next cell.

9.4 The Layered Costmap Pattern

We designed the layered costmap to reflect all of the previous observations.

Conclusion 8. *Each data source for the costmap can be encapsulated into a costmap layer by defining the following operations:*

1. *Specify a portion of the costmap which it needs to update on this update cycle, given the portion needed by lower layers.*

2. *For a particular portion of the costmap, given the values already written by lower layers, specify the new values for that portion of the costmap.*

This pattern aims to achieve maximum generality while maintaining a fixed structure for clarity and bookkeeping information for efficiency. The pattern encompasses all of the data sources implemented in the previous ROS costmap, as well as the additional layers discussed in Chapter 8, and potentially many more. The data need not fit into the previous definitions of lethal obstacles, or even follow the same inflation process. There is enough flexibility to implement all sorts of changes to the costmap.

Furthermore, it maintains that flexibility even as the rigid costmap layer structure is imposed onto each data source. This results in a much more clearly structured update process. With each data source being treated as a layer, it becomes much easier to switch the way the costmap values are changed.

Furthermore with the unified interface, the complexities of integrating additional data sources becomes simpler. The previous implementation required considering the update process for all of the other data sources. With the layered costmap pattern, the updates proceed in a linear fashion. The result is that each layer only has to integrate its values with the previous values already in the costmap, without regard to their particular sources.

9.4.1 Implementation Changes

Encapsulating each data source as a layer also simplified the process of customizing the costmap on the developer side. Previously, the only way to add a data source was to compile the changes into the `Costmap2DROS` and `Costmap2D` objects themselves. This meant that any changes to the data sources had to be done at compile time. Instead, given the layer interface shared by all the data sources, each layer could be integrated through dynamically linking to a library that implemented the interface. Using ROS' pluginlib interfaces, the new implementation allows each layer to be compiled independently and/or downloaded independently, and configured at runtime by simply changing a parameter.

This capability not only allows for new layers to be integrated, but it also cleans up the interactions between the existing components. In the previous implementation, the three-dimensional costmap data was handled with `VoxelCostmapROS` as a subclass of the `Costmap2DROS` class, resulting in unneeded code duplication between the two. In the layered costmap implementation, the three-dimensional obstacle tracking is a separate layer and can be easily swapped in to replace the two-dimensional obstacle tracking layer. Also, in the previous implementation, the option to integrate a static costmap or not was controlled by a boolean variable, with large blocks of code to deal with each case. In the layered costmap implementation, this also is customized simply by using the layer or not.

Lastly, the division of labor between the `Costmap2D` and `Costmap2DROS` class is clarified. Instead of the logic for updating the costmap being spread across the two classes, the `Costmap2D` class now only contains the data structure for keeping the values, and all of the functionality for updating the costmap are neatly divided into each of the component layers. That leaves the `Costmap2DROS` class responsible for the integration with the rest of the ROS framework and maintaining which layers to use.

Chapter 10

Contextualized Navigation

Our ultimate goal has been to have the ability to change *how* robots navigate. The existing algorithms provide only one way to navigate: the efficient collision-free route. While that solution is functional, it fails to take into account the wide range of different contexts that the robot can be in.

The work we performed with theatre dictated that we analyze the robot's motions and justify each one of them if possible. This led to a thorough examinations of where the robot drove and how it navigated there. This led to us developing several objectives beyond the efficient collision-free path. First, the robot should move in a clear predictable way, maintaining the same "character" throughout its actions. Secondly, the robot should interact with the people that it encounters while navigating in a socially acceptable way. Finally, the robot should incorporate a wide array of contextual factors into where it drives. All of these factors present additional, multifaceted objectives that the robot needs to consider.

These additional objectives need to be represented in the robot. The work presented in this dissertation to alter the costmaps provides one way to include these additional cost functions into the representations that robots already use to create their behavior. With the ability to incorporate additional layers into the costmap, robot operators can design additional behavior without having to fundamentally change the core navigation algorithms of their robot. The robot behavior can even be changed dynamically as the context changes around the robot, by simply changing the weight of a particular layer, or enabling or disabling it entirely. This presents a platform in which many additional contextual behaviors can be utilized, which in turn enables robots to effectively deal with a wider range of scenarios than those where they just have to move efficiently.

The costmaps are, of course, only one part of the navigation algorithm as a whole. In order to create the behavior we wanted, we had to modify both the global and local planners. The modifications to the local planner were extensive, since there were many cost functions that did not work as intended, as documented in Appendix A. The work with the planners allows us to incorporate additional contextual cost functions into the robot's overall navigation algorithm. We also have some preliminary work with the high-level executive of the navigation algorithm to be able to incorporate gestures natively into the robot's behavior.

As mentioned in Section 1.4, this work exists as more than just a scholarly exercise. Throughout the course of this research, we have worked tirelessly to make the products of the work available to the open-source community. All of the code for the layered costmaps, each of the mentioned layers, our improved planners, and the testing software we used are all published on github.com for others to make use of. The work here aims to satisfy two goals, in a way that mirrors the philosophical discussions of robot acting in Chapter 2. From the Chinese Room perspective, all that matters is the performance, and for people interacting with the robot, our goal has been to improve the robot's navigation in an intelligent way. However, as computer scientists, we must acknowledge that the same robot performance could be achieved using a Wizard-of-Oz scenario with everything controlled by humans. Thus, in the ontological sense, we need to also have a robust system which backs up the performance with a robust system for autonomously generating the behaviors.

Ultimately, we return to the conception of "robotic motion" from Section 1.2. With additional layers to encode the world around them, the robots will be able to respond to and interact with their environment in a more intelligent way. With contexts and objective beyond efficient travel, the scope of the robot's behavior widens to be more specific and more sharply motivated. With the Viewpoints in mind, we can create more variety in the robot's behavior, thus dissuading the notion of the robot "robotically" doing the same things over and over again. With the additional contexts, the robot can better fit people's theory of mind for it and thus improve some of the interactions between humans and robots.

References

[1] P. Althaus, H. Ishiguro, T. Kanda, T. Miyashita, and H.I. Christensen. Navigation for Human-Robot Interaction Tasks. In *Proc. ICRA'04*, volume 2, pages 1894–1900, New Orleans, LA, 2004.

[2] Michael Argyle and Mark Cook. Gaze and Mutual Gaze. 1976.

[3] K.O. Arras, O.M. Mozos, and W. Burgard. Using Boosted Features for the Detection of People in 2D Range Data. In *Proc. ICRA 2007*, pages 3402–3407, San Diego, California, 2007.

[4] W.A. Bainbridge, J. Hart, E.S. Kim, and B. Scassellati. The effect of presence on human-robot interaction. In *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, pages 701–706. IEEE, 2008.

[5] BBC News. Actor robots take japanese stage. 26 November 2008. `http://news.bbc.co.uk/2/hi/asia-pacific/7749932.stm`.

[6] Morgan Bettex. Disembodied performance. *MIT News*, 10 September 2010. `http://web.mit.edu/newsoffice/2010/opera-machover-0910.html`.

[7] Anne Bogart and Tina Landau. *The Viewpoints Book: A Practical Guide to Viewpoints and Composition*. Theatre Communications Group, 2005.

[8] Jean-David Boucher, Ugo Pattacini, Amelie Lelong, Gerard Bailly, Frederic Elisei, Sascha Fagel, Peter Ford Dominey, and Jocelyne Ventre-Dominey. I reach faster when i see you look: gaze effects in human–human and human–robot face-to-face cooperation. *Frontiers in neurorobotics*, 6, 2012.

[9] Ben Brantley. In robot world she turns more Hedda than Hedda. *The New York Times*, 18 February 2006. `http://theater2.nytimes.com/2006/02/18/theater/reviews/18hedd.html`.

[10] C. Breazeal. Emotion and sociable humanoid robots. *International Journal of Human-Computer Studies*, 59(1):119–155, 2003.

[11] Cynthia Breazeal, Aaron Edsinger, Paul Fitzpatrick, and Brian Scassellati. Active vision for sociable robots. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 31(5):443–453, 2001.

[12] Cynthia Breazeal, Andrew Brooks, Jesse Gray, Matt Hancher, John McBean, Dan Stiehl, and Joshua Strickon. Interactive robot theatre. *Communications of the ACM*, 46(7), July 2003.

[13] Oliver Brock and Oussama Khatib. High-speed navigation using the global dynamic window approach. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 1, pages 341–346. IEEE, 1999.

[14] A. Bruce, J. Knight, S. Listopad, B. Magerko, and I. Nourbakhsh. Robot improv: Using drama to create believeable agents. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2000)*, volume 4, pages 4002–4008, 2000.

[15] Allison Bruce, Illah Nourbakhsh, and Reid Simmons. The role of expressiveness and attention in human-robot interaction. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2002)*, volume 4, pages 4138–4142, 2002.

[16] Jonathan B Buckheit and David L Donoho. *Wavelab and reproducible research.* Springer, 1995.

[17] Robert Burns. *To a Mouse, on Turning Her Up in Her Nest with the Plough.* 1785.

[18] J. Cassell, T. Bickmore, L. Campbell, H. Vilhjálmsson, and H. Yan. More than just a pretty face: Converstational protocols and the affordances of embodiment. *Knowledge-Based Systems*, 14:55–64, 2001.

[19] Justine Cassell. Body language: Lessons from the near-human. In J. Riskin, editor, *Genesis Redux: Essays in the History and Philosophy of Artificial Intelligence*, pages 346–374. University of Chicago Press, Chicago, IL, 2007.

[20] D. Chi, M. Costa, L. Zhao, and N. Badler. The emote model for effort and shape. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 173–182. ACM Press/Addison-Wesley Publishing Co., 2000.

[21] H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun. *Principles of robot motion: theory, algorithms, and implementations.* MIT press, 2005.

[22] K. Dautenhahn, M. Walters, S. Woods, K.L. Koay, CL Nehaniv, A. Sisbot, R. Alami, and T. Siméon. How May I Serve You?: A Robot Companion Approaching a Seated Person in a Helping Context. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 172–179. ACM, 2006.

[23] Steve Dixon. Metal performance: Humanizing robots, returning to nature, and camping about. *The Drama Review*, 48(4), 2004.

[24] Anca Dragan and Siddhartha Srinivasa. Generating legible motion. In *Robotics: Science and Systems*, June 2013.

[25] B.A. Duncan, R.R. Murphy, D. Shell, and A.G. Hopper. A midsummer night's dream: social proof in HRI. In *Proceeding of the 5th ACM/IEEE international conference on Human-robot interaction*, pages 91–92. ACM, 2010.

[26] D. Ferguson and M. Likhachev. Efficiently using cost maps for planning complex maneuvers. In *Proceedings of the ICRA 2008 Workshop on Planning With Costmaps*, 2008.

[27] Tully Foote. ROS Community Metrics. August 2013. `http://download.ros.org/downloads/metrics/metrics-report-2013-08.pdf`.

[28] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *Robotics & Automation Magazine, IEEE*, 4(1):23–33, 1997.

[29] S.R. Fussell, L.D. Setlock, and E.M. Parker. Where do Helpers Look?: Gaze Targets During Collaborative Physical Tasks. In *Proc. CHI 2003*, pages 768–769, Fort Lauderdale, Florida, 2003.

[30] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 1994.

[31] Brian Gerkey. personal communication, 2013.

[32] M.J. Gielniak and A.L. Thomaz. Enhancing interaction through exaggerated motion synthesis. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 375–382. ACM, 2012.

[33] R. Gockley, A. Bruce, J. Forlizzi, M. Michalowski, A. Mundell, S. Rosenthal, B. Sellner, R. Simmons, K. Snipes, A.C. Schultz, et al. Designing robots for long-term social interaction. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1338–1343. IEEE, 2005.

[34] Jerzy Grotowski. *Towards a Poor Theatre*. Routledge, 2002.

[35] E.T. Hall. *The Hidden Dimension*. Doubleday, 1966.

[36] Kotaro Hayashi, Masahiro Shiomi, Takayuki Kanda, and Norihiro Hagita. Friendly Patrolling: A Model of Natural Encounters. In *Proc. RSS 2012*, page 121, Sydney, Australia, 2012.

[37] Guy Hoffman, Rony Kubat, and Cynthia Breazeal. A hybrid control system for puppeteering a live robotic stage actor. In *Proceedings of the 17th International Symposium on Robot and Human Interactive Communication (RoMan'08)*, pages 354–359, 2008.

[38] Kuo-Chen Huang, Jiun-Yi Li, and Li-Chen Fu. Human-oriented navigation for service providing in home environment. In *SICE Annual Conference 2010, Proceedings of*, pages 1892–1897. IEEE, 2010.

[39] E. Jessop, P.A. Torpey, and B. Bloomberg. Music and Technology in Death and the Powers. *New Interfaces for Musical Expression*, pages 349–354, 30 June 2011.

[40] T. Kanda, D.F. Glas, M. Shiomi, H. Ishiguro, and N. Hagita. Who will be the customer?: a social robot that anticipates people's behavior from their trajectories. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 380–389. ACM, 2008.

[41] Adam Kendon. Some functions of gaze-direction in social interaction. *Acta psychologica*, 26:22–63, 1967.

[42] C.D. Kidd and C. Adviser-Breazeal. Designing for long-term human-robot interaction and application to weight loss. 2008.

[43] Cory D. Kidd and Cynthia Breazeal. Effect of a robot on user perceptions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 4, pages 3559–3465, 2004.

[44] R. Kirby, R. Simmons, and J. Forlizzi. COMPANION: A Constraint-Optimizing Method for Person-Acceptable Navigation. In *Proceedings of the 18th IEEE Symposium on Robot and Human Interactive Communication (Ro-Man)*, pages 607–612, Toyama, Japan, 2009.

[45] Rachel Kirby. *Social Robot Navigation*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2010.

[46] Heather Knight. TEDWomen: Silicon-based comedy. www.ted.com, December 2010. URL `http://www.ted.com/talks/heather_knight_silicon_based_comedy.html`.

[47] Heather Knight, Scott Satkin, Varun Ramakrishna, and Santosh Divvala. A savvy robot standup comic: Online learning through audience tracking. In *International Conference on Tangible and Embedded Interaction*, Jan 2011.

[48] Kurt Konolige. Improved occupancy grids for map building. *Autonomous Robots*, 4 (4):351–367, 1997.

[49] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 2013.

[50] Chi-Pang Lam, Chen-Tun Chou, Kuo-Hung Chiang, and Li-Chen Fu. Human-centered robot navigationtowards a harmoniously human–robot coexisting environment. *Robotics, IEEE Transactions on*, 27(1):99–112, 2011.

[51] Brenda Laurel. *Computers as Theater.* Addison-Wesley Professional, 1993.

[52] Philip Lazebnik (Teleplay) and Tom Benko (Director). Devil's Due. Star Trek: The Next Generation, February 1991. 187.

[53] S. Lemaignan, M. Gharbi, J. Mainprice, M. Herrb, R. Alami, et al. Roboscopie: A theatre performance for a human and a robot. In *Proceeding of the 7th ACM/IEEE international conference on Human-robot interaction*, HRI 2012 (video). ACM, 2012.

[54] Tomas Lozano-Perez. Spatial planning: A configuration space approach. *Computers, IEEE Transactions on*, 100(2):108–120, 1983.

[55] David V. Lu. Researchin'. In *1st Annual Robot Film Festival*, 2011. `http://vimeo.com/davidlu/researchin`.

[56] David V. Lu. Ontology of robot theatre. In *Proc. ICRA Workshop on Robotics and Performing Arts: Reciprocal Influences*, 2012.

[57] David V. Lu and Ross Mead. Introducing students grades 6-12 to expressive robotics. In *Proceeding of the 7th ACM/IEEE international conference on Human-robot interaction*, HRI 2012 (video). ACM, 2012.

[58] David V. Lu and William D. Smart. Towards More Efficient Navigation for Robots and Humans. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[59] David V. Lu, Annamaria Pileggi, and William D. Smart. Multi-person Motion Capture Dataset for Analyzing Human Interaction. In *RSS Workshop on Human-Robot Interaction*, Los Angeles, California, July 2011.

[60] J. Mainprice, E. Akin Sisbot, L. Jaillet, J. Cortés, R. Alami, and T. Siméon. Planning human-aware motions using a sampling-based costmap planner. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5012–5017. IEEE, 2011.

[61] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige. The office marathon: Robust navigation in an indoor office environment. In *Proceedings if the IEEE International Conference on Robotics and Automation (ICRA)*, pages 300–307. IEEE, 2010.

[62] Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian P. Gerkey, and Kurt Konolige. The Office Marathon: Robust Navigation in an Indoor Office Environment. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, Alaska, 2010.

[63] Maja J Mataric. A distributed model for mobile robot environment-learning and navigation. 1990.

[64] L. Matthies and A. Elfes. Integration of sonar and stereo range data using a grid-based representation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 727–733, 1988.

[65] Ross Mead and Maja J Matarić. Automated caricature of robot expressions in socially assistive human-robot interaction. In *The 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI2010) Workshop on What Do Collaborations with the Arts Have to Say about HRI*, 2010.

[66] Joe Menosky (Teleplay) and Cliff Bole (Director). Emergence. Star Trek: The Next Generation, May 1994. 275.

[67] Ronald D. Moore (Writer) and Robert Scheerer (Director). The Defector. Star Trek: The Next Generation, January 1990. 158.

[68] Hans P Moravec. Sensor fusion in certainty grids for mobile robots. *AI magazine*, 9 (2):61–74, 1988.

[69] Masahiro Mori. The Uncanny Valley. *Energy*, 7(4):33–35, 1970. Translated by K. F. MacDorman and T. Minato. Original title: Bukimi no tani.

[70] Bilge Mutlu and Jodi Forlizzi. Robots in organizations: the role of workflow, social, and environmental factors in human-robot interaction. In *Human-Robot Interaction (HRI), 2008 3rd ACM/IEEE International Conference on*, pages 287–294. IEEE, 2008.

[71] Bilge Mutlu, Toshiyuki Shiwa, Takayuki Kanda, Hiroshi Ishiguro, and Norihiro Hagita. Footing in human-robot conversations: how robots might shape participant roles using gaze cues. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 61–68. ACM, 2009.

[72] S. Nakaoka, A. Nakazawa, F. Kanehiro, K. Kaneko, M. Morisawa, H. Hirukawa, and K. Ikeuchi. Learning from observation paradigm: Leg task models for enabling a biped humanoid robot to imitate human dances. *The International Journal of Robotics Research*, 26(8):829–844, 2007.

[73] C. Nass, Y. Moon, and P. Carney. Are People Polite to Computers? Responses to Computer-Based Interviewing Systems. *Journal of Applied Social Psychology*, 29(5): 1093–1109, 1999.

[74] E. Pacchierotti, H.I. Christensen, and P. Jensfelt. Human-Robot Embodied Interaction in Hallway Settings: a Pilot User Study. In *Proc. RO-MAN 2005*, pages 164–171, Nashville, Tennessee, 2005.

[75] S. Paepcke and L. Takayama. Judging a bot by its cover: An experiment on expectation setting for personal robots. In *Human-Robot Interaction (HRI), 2010 5th ACM/IEEE International Conference on*, pages 45–52. IEEE, 2010.

[76] Amit Kumar Pandey and Rachid Alami. A framework for adapting social conventions in a mobile robot motion in human-centered environment. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1–8. IEEE, 2009.

[77] Amit Kumar Pandey and Rachid Alami. A framework towards a socially aware mobile robot motion in human-centered dynamic environment. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5855–5860. IEEE, 2010.

[78] Miles L Patterson. A sequential functional model of nonverbal exchange. *Psychological review*, 89(3):231, 1982.

[79] Roland Philippsen and Roland Siegwart. Smooth and efficient obstacle avoidance for a tour guide robot. In *ICRA*, pages 446–451. Citeseer, 2003.

[80] M. Phillips and M. Likhachev. SIPP: Safe Interval Path Planning for Dynamic Environments. In *Proc. ICRA 2011*, pages 5628–5635, Shanghai, China, 2011.

[81] Pig Iron Theatre Company. The Robot Etudes. `http://www.pigiron.org/the-robot-etudes`, 2010.

[82] Tim Pryde. Meet don-8r, a coin powered robot raising money for charities! `http://timpryde.com/work/don-8r/`.

[83] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, Kobe, Japan, 2009.

[84] Byron Reeves and Clifford Nass. *The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places.* Cambridge University Press, New York, NY, 1996.

[85] J. Rett, J. Dias, and J.M. Ahuactzin. Bayesian reasoning for Laban Movement Analysis used in human-machine interaction. *International Journal of Reasoning-based Intelligent Systems*, 2(1):13–35, 2010.

[86] Thomas Richards. *At Work with Grotowski on Physical Actions.* Routledge, 1995.

[87] Raquel Ros, Séverin Lemaignan, E Akin Sisbot, Rachid Alami, Jasmin Steinwender, Katharina Hamann, and Felix Warneken. Which One? Grounding the Referent Based on Efficient Human-Robot Interaction. In *Proc. RO-MAN 2010*, pages 570–575, Viareggio, Italy, 2010.

[88] Paul Saulnier, Ehud Sharlin, and Saul Greenberg. Exploring minimal nonverbal interruption in social hri. Technical report, Research Report 2010-977-26, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, September. http://grouplab. cpsc. ucalgary. ca/Publications, 2010.

[89] L. Scandolo and T. Fraichard. An anthropomorphic navigation scheme for dynamic scenarios. In *Proceedings of the IEEE Internation Conference on Robtoics and Automation (ICRA)*, pages 809–814, 2011.

[90] B. Scassellati. Theory of mind for a humanoid robot. *Autonomous Robots*, 12(1):13–24, 2002.

[91] J.R. Searle. Minds, brains, and programs. *The Turing Test: Verbal Behavior As the Hallmark of Intelligence*, 2004.

[92] Candace L. Sidner, Christopher Lee, Louis-Philippe Morency, and Clifton Forlines. The effect of head-nod recognition in human-robot conversation. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, pages 290–296, 2006.

[93] E.A. Sisbot, L.F. Marin-Urias, R. Alami, and T. Simeon. A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 23(5):874–883, 2007.

[94] Constantin Stanislavski. *An Actor Prepares.* Theatre Arts Books, 1989.

[95] Maria Staudte and Matthew W Crocker. Visual Attention in Spoken Human-Robot Interaction. In *Proc. HRI 2009*, pages 77–84, San Diego, California, 2009.

[96] M. Svenstrup, S. Tranberg, H.J. Andersen, and T. Bak. Pose estimation and adaptive robot behaviour for human-robot interaction. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3571–3576, 2009.

[97] M. Svenstrup, T. Bak, and H.J. Andersen. Trajectory planning for robots in dynamic human environments. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4293–4298. IEEE, 2010.

[98] L. Takayama and C. Pantofaru. Influences on Proxemic Behaviors in Human-Robot Interaction. In *Proc. IROS 2009*, pages 5495–5502, St. Louis, Missouri, 2009.

[99] L. Takayama, D. Dooley, and W. Ju. Expressing thought: improving robot readability with animation principles. In *Proceedings of the 6th international conference on Human-robot interaction*, pages 69–76. ACM, 2011.

[100] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, et al. MINERVA: A Second-Generation Museum Tour-Guide Robot. In *Proc. ICRA 1999*, volume 3, Detroit, Michigan, 1999.

[101] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[102] Sebastian Thrun. Learning occupancy grids with forward models. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 1676–1681, 2001.

[103] Karel Čapek. *R.U.R. (Rossum's Universal Robots)*. Samuel French, English edition, 1923. Translated by Paul Selver and Nigel Playfair.

[104] AJN Van Breemen. Bringing robots to life: Applying principles of animation to robots. In *Proceedings of Shaping Human-Robot Interaction workshop held at CHI*. Citeseer, 2004.

[105] Barry Brian Werger. Profile of a winner: Brandeis University and Ullanta Performance Robotics' "Robotic Love Triangle". *AI Magazine*, 19(3), 1998.

[106] Chris Wilson. Robots on stage. M.S. thesis, Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO, August 2008.

[107] Karl R Wurst. I comici roboti: Performing the Lazzo of the Statue from the Commedia Dell'Arte. In William D. Smart, Tucker Balch, and Holly Yanco, editors, *AAAI Mobile Robot Competition and Exhibition (AAAI Technical Report WS-02-18)*, pages 124–128. AAAI Press, 2002.

[108] M. Yoda and Y. Shiota. The Mobile Robot Which Passes A Man. In *Proc. RO-MAN'97*, pages 112–117, Japan, 1997.

[109] Kimi Yoshino. These wireless robots try not to act remote. *Los Angeles Times*, 2 March 2007. `http://articles.latimes.com/2007/mar/02/business/fi-disney2`.

Appendix A

Debugging the Local Planner

The Scottish poet Robert Burns wrote “The best-laid schemes o’ mice an’ men Gang aft agley,”[17], commonly paraphrased as “The best-laid plans of mice and men oft go astray.” The same can be said of planning algorithms.

The work discussed in previous chapters explains our manipulations of the value of the robot being at any particular point in space. However, there are many components of creating the robot’s path that cannot be expressed in a spatial representation such as a costmap. For that, we turn to the two planners in our system, the global planner and the local planner. However, while the core ideas of a planner can be expressed simply (move the robot toward the goal efficiently, do not collide with obstacles, etc.) the actual implementation of planners can be deceptively complex.

There are additional facets to the desired behavior within our navigation task, especially as humans are factored into the equation. Humans sharing the same space as the robot want the robot’s movements to be predictable and legible, so that its behavior is clear and readable. Furthermore, they want the robot to follow certain social conventions, such as respecting their personal space. In order to accomplish these goals, planners have been developed which incorporate the various factors. The algorithms are often composed of relatively simple elements that seem straight-forward enough, and result in paths that accomplish the given goals frequently.

However, these “best-laid” planning algorithms do not always work. In this chapter, we examine a number of places where the planning algorithm for the ROS navigation stack generate behavior that is peculiar at best, and catastrophic at worst. The pitfalls of the global planner were discussed in Chapter 6. In this chapter, we focus on how to get the precise behavior we want out of the local planner.

A.1 Related Work

Much of the work in local planning stems from the Dynamic Window Approach[28] (aka DWA). It generates trajectories using the robot's position and velocity, based on its dynamics, and then chooses a trajectory based on the optimization of a cost function. In the original paper, the optimization had three factors, alignment with the target heading, maximizing clearance to obstacles and using maximal velocity.

Since the introduction of DWA, many have introduced variants which expand the optimization functions. Brock and Khatib [13] use additional data from the global plan to track progress toward the goal, as well as having a binary cost for trajectories that take the robot near to the goal. Philippsen and Siegwart [79] optimize over obstacle clearance, path alignment and a variable velocity function, which can be tuned to make the robot move in certain directions.

Additional factors have been added to the local planning optimizations based on the presence of people, including social formations[1], hallway passing behavior[44, 58, 74, 76], body pose (i.e. sitting vs. standing) [93], human visibility constraints [76, 89, 93], and, most frequently, personal space [38, 44, 50, 76, 77, 88, 89, 93, 97].

A.2 Local Planning

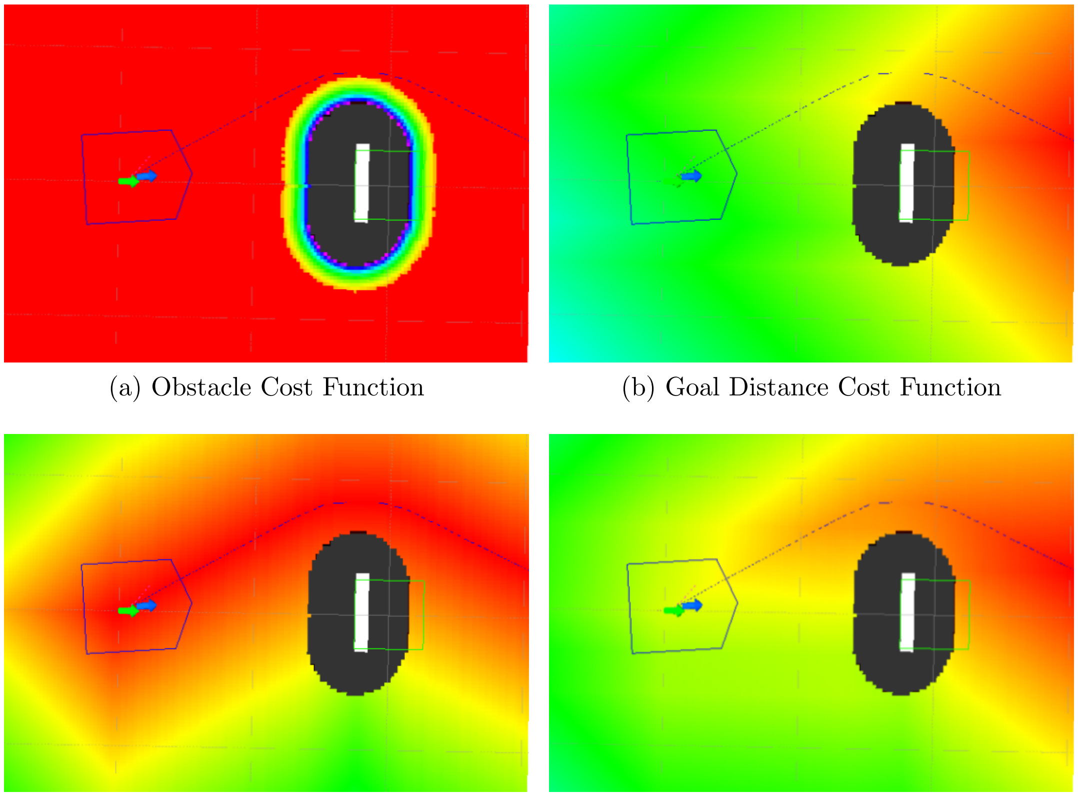
For our discussions of local planning, we use the following notation. At each iteration of planning, the robot's location is defined as x, y, θ, with velocity v_x, v_y, v_θ, and chooses a velocity command, $\dot{v_x}, \dot{v_y}, \dot{v_\theta}$. The predicted position of the robot based on a given velocity at a time t seconds into the future is notated as x_t, y_t, θ_t.

A.2.1 DWA Planner

The optimization for local planning from Fox et al. [28] can be generalized to the following.

$$\underset{\dot{v_x},\dot{v_y},\dot{v_\theta}}{\arg\min} \sum_i w_i f_i(x, y, \theta, \dot{v_x}, \dot{v_y}, \dot{v_\theta})$$

for some set of cost functions f_i and corresponding weights w_i. The set of velocities that the cost is minimized over is generated using the DWA approach based on the robot's position and velocity (v_x, v_y, v_θ). Each of the cost functions can be defined to score the trajectory in any manner. The cost functions discussed here all use the same general framework. Many of the cost functions rely on a parameter τ to find the position $x_\tau, y_\tau, \theta_\tau$.



(a) Obstacle Cost Function (b) Goal Distance Cost Function

(c) Path Distance Cost Function (d) Weighted Sum of Three Costs

Figure A.1: Local Planning Cost Functions - In each of these diagrams, the color indicates the cost at each position with the robot traveling from left to right around a sensed obstacle. Red indicates low cost, and green indicates higher cost, with the black areas being invalid costs.

Obstacle Cost Function

One of the highest priorities for the local planner is to avoid obstacles. Obstacle avoidance is integrated by using a costmap-based cost function. This cost function uses a discretized version of the trajectory that samples the space between $x, y\theta$ and $x_\tau, y_\tau, \theta_\tau$ at some regular distance interval. If the robot is in collision according to the costmap at any of the sampled poses, the trajectory is rejected. Otherwise the cost of the robot's footprint on the costmap is returned.

Goal Distance Cost Function

The other top priority is moving toward the goal. The GoalDistance cost function rewards trajectories that move closest to the goal. The cost is zero when x_τ, y_τ is at the goal, with costs increasing as the distance from x_τ, y_τ to the goal increases. As seen in Figure A.1b, the goal distance propagates around obstacles, such that the costs do not lead into a local minimum.

Path Distance Cost Function

The desired local planner behavior also stays reasonably close to the global plan. The PathDistance cost is thus defined as the Manhattan distance from the closest point on the global path to x_τ, y_τ, as seen in A.1c.

Goal Alignment and Path Alignment

The planner also considers the robot's alignment to the goal and the global path. These components assume there is a front side to the robot that we want to point in a particular direction. The GoalAlignment cost uses the same score gradient as the GoalDistance cost (Figure A.1b), and scores trajectories based on the score at the front of the robot at $x_\tau, y_\tau, \theta_\tau$. This process points the robot along the gradient toward the goal. The PathAlignment cost functions similarly with the same gradient as the PathDistance cost, and functions to orient the robot toward the global path when the robot is not on the global path. Both of these costs aim to have the front of the robot point in the direction of travel, which increases the legibility of where the robot is going to go.

Implementation Details

In the ROS implementation, all of the cost functions are cached in grids. For the costmap, it is only logical. For the rest, calculating the distance from a given point to the goal or to the nearest point on the path is computationally expensive, . Furthermore, due to the large number of trajectories checked, the calculations are going to happen frequently for each cycle, and some of them will duplicated. Hence the distances for GoalDistance and PathDistance are cached in a grid, matching the size of the costmap's grid. Since the distance is independent of the robot's orientation, the grid only needs to be two-dimensional.

To quickly calculate the Alignment costs, the ROS implementation uses the same distance grids as the Distance costs, but instead of checking where the robot's center is and using the cached distance from there, the alignment costs use a "forward point" a set distance (d_{fp})in front of the robot to check the alignment. For a given (x, y) location, the GoalDistance and PathDistance costs are always going to be the same, regardless of the orientation θ, but the GoalAlignment and PathAlignment can vary.

A.3 Discontinuities

A.3.1 Continuous Obstacle Checking Problem

The cost function that identified if the robot would collide with any of the obstacles worked most of the time, but on occasion would hit obstacles with the back part of the robot while coming around turns. This stemmed from a bug in the cost function that would calculate the cost of the robot's footprint at a number of arbitrary positions along a given trajectory, but only mark the trajectory as invalid if the final position was in collision. The positions that resulted in a collision were the intermediate positions between the original position and the final predicted position.

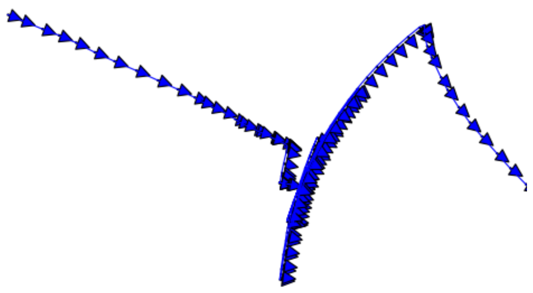


Figure A.2: End Path Scenario - Each arrowhead marks the robot position as it moves from left to right. As the robot approaches, it slows down until it reaches just outside the goal radius. Then it moves laterally without being able to enter the circle until it happens to cross the radius due to noise.

A.3.2 The End Path Problem

Using the forward point method, the alignment methods work reasonably well in the general case. However, when the robot nears the goal, these cost functions become trickier. Once the robot is within d_{fp} of the goal, the forward point is beyond the goal, meaning higher costs for points closer to the goal.

Let us consider the GoalAlignment cost specifically. In order to get around the problem, the ROS local planner sets the weight of the GoalAlignment cost to zero when the point x, y is within d_{fp} of the goal. This eliminates the problem within that radius, but causes problems just outside of it, stemming from the difference between the robot's location at planning time (x, y), and the predicted location (x_τ, y_τ). The weight is only set to zero when the starting location is within the radius, not when the predicted location is within the radius. Hence, paths that would move the robot into the radius from outside of it are more costly.

The resulting behavior ranges from annoying to failure-ridden. The worst case scenario is that the robot is not able to reach its goal. When the robot is just outside the radius, any path that enters within the radius is going to be scored higher than those that remain outside the radius. This leads to the behavior seen in Figure A.2. The robot moves laterally around the goal, only moving closer to the goal if the robot happens to enter the radius.

The annoying feature of this is that the robot slows down as it approaches the radius. This can also be seen in Figure A.2. Trajectories at full speed would move the robot's front point into the radius, and thus are scored higher, so that the slower trajectories cost less.

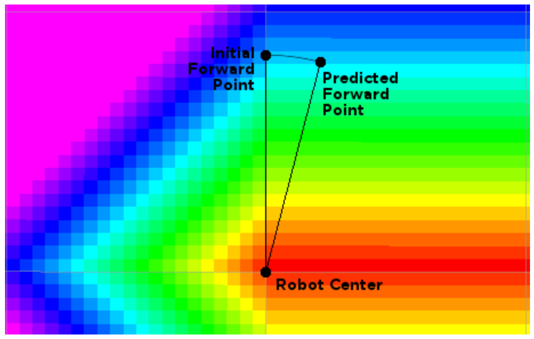


Figure A.3: Angular Problem - The path planner should score a plan that turns toward the goal more preferably, but because of the discretization of the grid, it does not. The initial forward point has the same PathAlignment score as the predicted point, with both falling in the same horizontal band of equal costs.

The Angular Problem

Consider a scenario where the robot must plan a path to a point initially to its right. Based on the alignment metrics used, one would expect the robot to turn right such that the front of the robot is aligned with the shortest path and the goal location. Instead, the robot does some combination of turning and translating. Even with configurations with very high weights on the PathAlignment will not recognize turning toward the goal as being less costly.

The problem lies in the discretization of the search space for computational efficiency. If the robot initially starts out with no velocity, then due to the dynamic constraints, the DWA trajectory generation will only consider very small velocities within the robot's acceleration limits. As a result, the robot's predicted location at the end of the simulation period τ will be at a point roughly parallel to the robot's path (when starting 90 deg away). This point can have the same score as the initial position (without any rotation) in the discretized grid. As a result, the robot sees no initial benefit to turning, despite there being actual progress toward reducing the PathAlignment cost.

The Driving Straight Problem

Sometimes even the simplest scenarios present interesting edge cases. Consider a holonomic robot aiming toward a goal directly in front of it. One would expect the robot to move straight forward toward its goal, i.e. v_x is the maximum allowed speed and v_y and v_θ are zero. It seems simple enough. However, the actual behavior includes using the holonomic base to move with maximum values of $\dot{v_x}$ and $\dot{v_y}$ with additional rotational velocity $\dot{v_\theta}$ to ensure the robot stays on path.

It seems bizarre that the robot would chose this complicated maneuver when a simple one would suffice. However, the optimization of costs does not optimize for simplicity. Instead, it moves toward the goal as quickly as it can, due to the GoalDistance cost. In this particular instance, the cause of the problem is the way in which the space of possible trajectories is explored. To encapsulate the robot's dynamics, there are parameters that specify the minimum and maximum values of $\dot{v_x}, \dot{v_y}$ and $\dot{v_\theta}$ and iterating over all combinations of those ranges with some resolution. However, the overall velocity of the robot when these three velocities are combined can be greater than the magnitude of any individual component velocity. Thus the quickest way to get to the goal (and minimize GoalDistance cost) is to move at a slight angle, translating in both the x and y directions.

This is a valid trajectory, and does in fact move the robot toward the goal in less time than the straight forward approach. However, what it gains in terms of speed efficiency, it loses in legibility. The robot moving at such an angle looks like it may start moving in a different direction. This modifies the robot's floor pattern and its shape, since driving at an angle change the profile of the robot. The effect on the robot's tempo/speed is minimal.

A.4 Wrap Up

In order to debug the individual cost functions mentioned above, we modified the structure of the local planner much in the same way that the costmap was modified to be more extensible. The local planner was extended to allow for arbitrary cost functions to be added to the weighted sum that scores each of the trajectories.