Multi-objective Intent-based Path Planning for Robots for Static and Dynamic Environments

Meher Talat Shaikh

Brigham Young University

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Physical Sciences and Mathematics Commons

BYU ScholarsArchive Citation
Shaikh, Meher Talat, "Multi-objective Intent-based Path Planning for Robots for Static and Dynamic Environments" (2020). Theses and Dissertations. 8510.
https://scholarsarchive.byu.edu/etd/8510

This Dissertation is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact ellen_amatangelo@byu.edu.
Multi-Objective Intent-Based Path Planning for Robots for Static and Dynamic Environments

Meher Talat Shaikh

A dissertation submitted to the faculty of Brigham Young University in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Michael A. Goodrich, Chair
Ryan Farrell
Marc D. Killpack
Kent Seamons

Department of Computer Science
Brigham Young University

Copyright © 2020 Meher Talat Shaikh
All Rights Reserved
ABSTRACT

Multi-Objective Intent-Based Path Planning for Robots for Static and Dynamic Environments

Meher Talat Shaikh
Department of Computer Science, BYU
Doctor of Philosophy

This dissertation models human intent for a robot navigation task, managed by a human and undertaken by a robot in a dynamic, multi-objective environment. Intent is expressed by a human through a user interface and then translated into a robot trajectory that satisfies a set of human-specified objectives and constraints. For a goal-based robot navigation task in a dynamic environment, intent includes expectations about a path in terms of objectives and constraints to be met. If the planned path drifts from the human’s intent as the environment changes, a new path needs to be planned.

The intent framework has four elements: (a) a mathematical representation of human intent within a multi-objective optimization problem; (b) design of an interactive graphical user interface that enables a human to communicate intent to the robot and then to subsequently monitor intent execution; (c) integration and adoption of a fast online path-planning algorithms that generate solutions/trajectories conforming to the given intent; and (d) design of metric-based triggers that provide a human the opportunity to correct or adapt a planned path to keep it aligned with intent as the environment changes.

Key contributions of the dissertation are: (i) design and evaluation of different user interfaces to express intent, (ii) use of two different metrics, cosine similarity and intent threshold margin, that help quantify intent, and (iii) application of the metrics in path (re)planning to detect intent mismatches for a robot navigating in a dynamic environment. A set of user studies including both controlled laboratory experiments and Amazon Mechanical Turk studies were conducted to evaluate each of these dissertation components.

Keywords: robot path-planning, human-robot interaction, multi-objective optimization, human supervisory control, user interfaces
ACKNOWLEDGMENTS

About three years back, when I was a little down I received this message “I have confidence in you. You have what it takes to succeed with your PhD.” This was from my advisor Dr. Michael Goodrich. With very few words he rekindled the spark that transformed into this thesis. I will be forever grateful to him for his teachings and his unwavering support. All through my research I was in great awe of how he had the insights to drive and delegate experiments that were meaningful and consequential.

I want to extend my gratitude to my dissertation committee: Dr. Farrell, Dr. Killpack, and Dr. Seamons. I thank them for their valuable feedback in shaping out the final dissertation. I would also extend my thanks to all the teachers in Brigham Young University Computer Science Department for making my learning and education at BYU a memorable experience.

I would have not been able to reach any milestone of my research unless my family would have supported me. My husband, Mujahed, was a great support to lean on. For much of the dissertation period he played the part of both a mom and a dad to my four girls. I’m greatly indebted to him. I’m also thankful for my four girls who missed their mom in their homework discussions, in their play-times and few trips that they made with their dad. I am also grateful to all my friends whose words always encouraged me. This list being long, I do not cite any. I am also grateful to have the blessings of my parents who from a distance always prayed for me and my family.
Table of Contents

List of Figures x

List of Tables xiv

1 Introduction 1

1.1 Importance of the Problem ........................................... 1
1.2 Problem Formalism .................................................... 2
   1.2.1 Path-Planning Task ......................................... 2
   1.2.2 Multi-Objective Path-Planning (MOPP) ....................... 2
   1.2.3 Collaborative Human-Robot Path-Planning .................. 3
   1.2.4 Collaborative Human-Robot Path-Planning in a Dynamic Environment 4

1.3 Related Work ....................................................... 5
   1.3.1 Research Area Overview ..................................... 5
   1.3.2 Research-Related Work ...................................... 8

1.4 Thesis Statement ................................................... 11

1.5 Intent-Based Multi-Objective Path-Planning ....................... 11
   1.5.1 Creating Meaningful Cost Functions .......................... 12
   1.5.2 Generating Trajectories on the Pareto Front ................ 13
   1.5.3 Representing Intent .......................................... 13
   1.5.4 GUI Design .................................................. 16

1.6 Intent-Based Multi-Objective Path-Planning in a Dynamic Environment 20
   1.6.1 Evaluate Replanning Triggers from the Literature .......... 20
2 Interactive Multi-Objective Path Planning through a Palette-based User Interface

2.1 Introduction .................................................. 26
2.2 Related Work .................................................. 29
2.3 Adverb Palette .................................................. 30
  2.3.1 Palette .................................................... 31
  2.3.2 Sliders ..................................................... 33
  2.3.3 Waypoints ................................................ 35
2.4 Cost function for adverb ‘Stealthily’ ................................ 36
  2.4.1 Seen by Any Enemy ....................................... 37
  2.4.2 Detection Likelihood of a robot by an enemy $e_i$ ............ 39
  2.4.3 Stealthily cost for a path .................................. 42
2.5 Cost function for adverb ‘safe’ ................................... 43
  2.5.1 Safety cost for a path ...................................... 44
2.6 Cost function for adverb ‘quickly’ ................................ 44
2.7 Cosine similarity for path selection ................................ 45
  2.7.1 Path vector ................................................ 45
  2.7.2 Cosine Similarity ........................................... 46
2.8 Subjective Evaluation ........................................... 47
2.9 Summary and Future Work ...................................... 48

3 Design and Evaluation of Adverb Palette:

A GUI for Selecting Tradeoffs in Multi-objective Optimization Problems 49

3.1 Introduction .................................................. 50
3.2 Related Work .................................................. 52
3.3 Adverb Palette ........................................ 53
  3.3.1 Conditioning Objectives .......................... 54
  3.3.2 Palette ........................................... 55
  3.3.3 Sliders ........................................... 57
  3.3.4 Prism ............................................. 57
  3.3.5 Waypoints ....................................... 59
3.4 Mapping between strategic intent and pay-offs ....................... 59
3.5 Objective Functions ...................................... 61
3.6 User Study ............................................ 63
  3.6.1 User Study Procedure ............................ 63
  3.6.2 Hypothesis testing ............................... 65
3.7 Results ................................................ 66
  3.7.1 F statistics ...................................... 66
  3.7.2 User Preference ................................. 69
  3.7.3 Discussion ...................................... 70
3.8 Summary and Future Work .................................. 71
3.9 Acknowledgments ......................................... 72

4 When does a Human Replan? Exploring Intent-Based Replanning in
  Multi-Objective Path Planning .......................... 73
  4.1 Introduction ......................................... 74
  4.2 Related Literature .................................... 75
  4.3 Operational Definition of Intent ..................... 76
  4.4 Replanning Architecture ............................ 78
    4.4.1 Adverb Palette .................................. 78
    4.4.2 Planning, Execution, and Replanning ............ 80
  4.5 Path Planning ....................................... 82
    4.5.1 Problem ...................................... 82
5.6.2 Hypothesis ......................................................... 107
5.7 Results .............................................................. 108
  5.7.1 Summary Statistics ........................................... 108
  5.7.2 Comparing Trigger Types ................................... 110
  5.7.3 Change Path Recommendation ............................. 110
  5.7.4 Correlation between Q1 and Q2 responses ............... 111

6 A Measure to Match Robot Plans to Human Intent:
A Case Study in Multi-Objective Human-Robot Path-Planning* 113
  6.1 Introduction ..................................................... 113
  6.2 Related Literature .............................................. 116
  6.3 Representing Path Tradeoffs, Human Intent, and Tradeoffs ..... 117
    6.3.1 Representing Tradeoffs as Vectors ...................... 117
    6.3.2 Representing Human Intent as a Vector ................. 118
    6.3.3 Intent-Matching Metrics .................................. 119
  6.4 Intent Threshold Margin ..................................... 121
    6.4.1 Definition .................................................. 121
    6.4.2 Ranking Solutions ......................................... 123
  6.5 Evaluations .................................................... 125
    6.5.1 Hypotheses ............................................... 127
    6.5.2 Data ....................................................... 127
  6.6 Results and Discussion ....................................... 128
    6.6.1 Comparing InThresh to Cosine Similarity ............... 128
    6.6.2 Difference Between Hypothesized and Participant Ranks ..... 128
    6.6.3 Spearman’s Rho for User Ranking ....................... 131
    6.6.4 Correlation Between Intent Threshold Margin and User Ranks ... 132
  6.7 Summary and Future Work .................................... 133
7 Path Replanning in a Dynamic Environment using Intent Threshold Margin

7.1 Introduction ........................................... 135

7.2 CS Metric Limitations .................................. 137
  7.2.1 Pure Intent ..................................... 137
  7.2.2 Mixed Intent ..................................... 138

7.3 Replanning using ITM ..................................... 140

7.4 Normalizing When a Path is Being Followed ......................... 143

7.5 Human Factors Study ..................................... 145
  7.5.1 Hypotheses ..................................... 146
  7.5.2 Methodology ..................................... 146

7.6 Results .................................................. 149
  7.6.1 Summary Statistics ................................ 149
  7.6.2 Summary Statistics across Worlds ................. 151

7.7 Summary and Future Work .................................. 153

8 Summary and Future Work .................................... 156

8.1 Summary ................................................ 156

8.2 Future Work ............................................. 156
  8.2.1 Spider Web Interface .............................. 156
  8.2.2 Path Planning Algorithms ......................... 157
  8.2.3 Multi-objective Applications ....................... 157
  8.2.4 Other Replanning Domains ......................... 158
  8.2.5 Clustering ......................................... 158

A Spider Web Interface ....................................... 159

References .................................................... 162
# List of Figures

1.1 Life cycle of human-robot collaboration task. ........................................ 11  
1.2 Pareto front of two objective path planning problem. ............................. 14  
1.3 Example of a quick path (left) and a stealthy path (right). ....................... 16  
1.4 Left Panel: map showing multiple solutions. Right Panel: palette interface. 17  
1.5 Adverb palette GUIs. ........................................................................... 18  
1.6 Spider web metaphor. ............................................................................ 19  

2.1 Adverb Palette: Left Panel: Environment showing multiple solutions. Right  
Panel: Command interface. ................................................................. 27  
2.2 Path Planning with MORRF* for two objectives. ................................. 28  
2.3 Quick and safe command: lowermost dab in CI represents the command, and  
the map shows the corresponding path (highlighted magenta). .................... 33  
2.4 Adverb Palette: Slider interface. ......................................................... 34  
2.5 Waypoints Interface. ........................................................................... 36  
2.6 Probability of $x_{rob}$ being seen by any enemy (modelled by Noisy OR). .... 38  
2.7 Detection range curve. ........................................................................ 41  
2.8 Stealthily objective function. ................................................................. 43  
2.9 Safety cost for every robot location. ..................................................... 44  
2.10 Path **ALMNOB** is quicker than the path **AXYB**. ......................... 45  
2.11 Path Comparison w.r.t example human intent vector $h_{ex}$. .................... 47  

3.1 Path Planning with MORRF* for two objectives. ................................. 51
3.2 Quick and safe command in lowermost dab: map shows the corresponding highlighted path. ................................. 56
3.3 Adverb Palette: Slider interface. ............................................. 56
3.4 Adverb Palette: Prism interface. ............................................. 58
3.5 Barycentric coordinates on an equilateral triangle. ......................... 58
3.6 Waypoints interface. ........................................................... 60
3.7 Path Comparison w.r.t example human intent vector $h_{ex}$. ................ 61
3.8 Objective functions. .................................................................. 62
3.9 Difficulty level: (a) Easy and (b) Hard. ......................................... 64
3.10 Accuracy of different interfaces. ............................................... 67
3.11 Answer Time in seconds on different interfaces. ............................. 67
4.1 Intent framework for human-robot teams. ....................................... 77
4.2 Map showing robot’s potential paths. Robot: black entity, Enemy: orange haloed entities, Obstacles: gray blocks (trees or buildings) ................ 79
4.3 Example of a tradeoff that satisfied both quickly and stealthily. ............... 80
4.4 Robot’s current and new path. Current path: dashed green, New path: solid green ................................................................. 81
4.5 Objective space, $o(\sigma_p)$, and color space, $h$. .............................. 86
4.6 Color and costs space for two objectives. ....................................... 88
4.7 Branching from the original blue path. ......................................... 89
4.8 Time trigger at point $s_{t1}$. ....................................................... 90
4.9 Time trigger at point $s_{t2}$. ....................................................... 90
4.10 Change in intent trigger example. .............................................. 91
4.11 Homotopy trigger example. ..................................................... 91
5.1 Life cycle of human-robot collaboration task. ................................. 99
5.2 Two objectives Pareto front of trajectories and intent/trajectory mapping. .... 101
5.3 Path vector $O(\sigma_h)$, intent vector $h$, and cost parts. ............................ 102

5.4 Changing costs resulting into intent mismatch. ................................. 104

5.5 Example survey question, obstacles, enemies, original dashed path, and re-
planned solid path. .......................................................... 106

6.1 Example paths; obstacles are gray and enemies orange. .................. 114

6.2 Vector associated with each path and human intent. ......................... 118

6.3 Paths that make to intent-satisfying-region. .................................... 122

6.4 Average participants ranks across configurations. ............................ 130

6.5 Distribution analysis of Spearman’s rank correlation coefficients for 423 (that
is, 47 * 9) user rankings. ....................................................... 132

6.6 Correlation between epsilon distances and average user ranks. Pearson’s
Correlation coefficient 0.823, $p$-value < 0.001. ................................ 133

7.1 Decision scenario to replan. .................................................. 140

7.2 The 3 best paths used for normalization during robot walk. .............. 145

7.3 Task example for participant evaluation. ...................................... 148

7.4 Box-and-whiskers plots for the distribution of ratings across trigger type.
Median is indicated by the horizontal line, the mean is represented by the
diamond, the 25% – 75% quartile range is represented by the shaded region,
the 90% range is enclosed in the whiskers, and outliers are indicated by black
circles. ................................................................................. 150

7.5 User rating statistics for CS-based replanning. x is mean, horizontal line is
median. ................................................................................. 152

7.6 User rating statistics for ITM-based replanning. x is mean, horizontal line is
median. ................................................................................. 153

7.7 User rating statistics for NOM-based tasks. x is mean, horizontal line is median. 154

A.1 Example of a Covert Intent Set on Spider Web. ............................... 159
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Conditional Probability Table for three enemies: T=true, F=false.</td>
<td>38</td>
</tr>
<tr>
<td>3.1</td>
<td>Effect of interface, difficulty level, and interaction of interface and difficulty level on various measures.</td>
<td>66</td>
</tr>
<tr>
<td>3.2</td>
<td>Pairwise differences in Answer Time.</td>
<td>68</td>
</tr>
<tr>
<td>3.3</td>
<td>Pairwise differences of Interface Time.</td>
<td>68</td>
</tr>
<tr>
<td>3.4</td>
<td>Interface Time on different interfaces. t value computed as E minus H.</td>
<td>68</td>
</tr>
<tr>
<td>3.5</td>
<td>Pairwise differences for subjective workload computed using NASA TLX. I/F = Interface.</td>
<td>69</td>
</tr>
<tr>
<td>3.6</td>
<td>Pairwise differences between interfaces for appeal variables.</td>
<td>69</td>
</tr>
<tr>
<td>5.1</td>
<td>Different trigger type response statistics.</td>
<td>109</td>
</tr>
<tr>
<td>5.2</td>
<td>Triggers comparison.</td>
<td>110</td>
</tr>
<tr>
<td>5.3</td>
<td>Mean recommendation at different triggers.</td>
<td>111</td>
</tr>
<tr>
<td>5.4</td>
<td>Q1/Q2 responses: Pearson Correlation.</td>
<td>111</td>
</tr>
<tr>
<td>6.1</td>
<td>Mean (and median, in parentheses) user ranks for the 9 configurations compared to the hypothetical ranks. The † superscript by the configuration indicates a mixture intent of ‘quick and stealthy’, and the ‡ superscript indicates a mixture of ‘stealthy and safe’; intent for all other configurations are for single attributes.</td>
<td>129</td>
</tr>
<tr>
<td>6.2</td>
<td>22 out of 36 responses indicate that the user ranks were not significantly different from the hypothetical ranks.</td>
<td>131</td>
</tr>
</tbody>
</table>
7.1 Cosine similarity values for various paths relative to the pure intent \( \mathbf{I} = [0, 1, 0]^T \).

7.2 Cosine similarity values for various paths relative to the mixed intent \( \mathbf{I} = [0.01, 0.68, 0.73]^T \).

7.3 ITM differences for the payoff vector of a path at various times for the intent vector \( \mathbf{I} = [0, 1, 0]^T \). When \( \theta = [0.01, 0.1, 0.01] \) no replanning is triggered.

7.4 Summary statistics of CS, ITM, and NOM enabled tasks.

7.5 One-way ANOVA for the pure intent: The means of CS, ITM, and NOM differed significantly.

7.6 One-way ANOVA for the mixed intent: The means of CS, ITM, and NOM differed significantly.
Chapter 1

Introduction

Realistically speaking, as humans, we want robots to do what we want them to do [28, 63]. Regardless of a robot’s level of autonomy [121], humans want the robot to be in sync with the human’s objective. A human’s intent for a robot includes the robot’s activity — what the robot should do — as well as the objectives associated with the robot’s activity — how the robot should do it.

The previous century witnessed mobile robots designed to navigate on the sea, on the land, in the air, under water, and in space. What all these mobile robots have in common is the challenge of navigation, including the need to travel in unknown and unstructured environments. In most of the problems, the navigation challenge includes human desires, constraints, and goals. However, making a mobile robot comply with what a human wants can be non-trivial. This dissertation applies the definition of human intent to the problem of multi-objective robot path-planning and replanning in dynamic environments. For a ground-based robot, the human’s intent is encoded in a trajectory or a plan that reaches a desired end state while appropriately balancing tradeoffs between objectives.

1.1 Importance of the Problem

Robot navigation is one of the very important activities in many of the current state-of-the-art human-robot interaction (HRI) applications. And the field is still not mature enough. For example, NASA gave up on Mars Rover Spirit after attempting its revival for about a year when it got stuck in a patch of Martian sand.
Among many, three critical human-robot interaction (HRI) tasks emerge: (i) creating an appropriate interface to command the robot according to intent, (ii) managing multiple objectives for and during task-execution, and (iii) identifying critical timings/events that can indicate the need to replan collaboratively. This research contributes in all three of these areas as follows. First, it presents and analyzes graphical user interfaces that help intent to be expressed, communicated and monitored for robot navigation. Second, most multiple objective optimization literature stops at generating solutions. The interfaces help explore multi-objective tradeoffs by bridging the gap between the set of Pareto front solutions and the real solution/plan/actions. Third, while the replanner running on the robot may generate an updated plan for a given intent every few seconds, it is not reasonable to seek an operator’s attention after each such update. This work identifies and evaluates triggers that can help the operator to replan at critical times/events that should improve the performance of human-robot collaborative tasks with minimum operator workload.

1.2 Problem Formalism

1.2.1 Path-Planning Task

The kinematic motion-planning task in this dissertation is to find a path, \(\sigma\), starting from an initial state (or robot configuration) and terminating at a specified goal state (or robot configuration) bounded by time. Let \(X_{\text{init}}\) be the initial state, and let \(X_{\text{goal}}\) be the goal state. A trajectory, \(\sigma\), is a sequence of states, \((x_0, x_1, \ldots, x_{n-1}, x_n)\), and a solution is a trajectory such that \(x_0 = X_{\text{init}}\) and \(x_n = X_{\text{goal}}\).

1.2.2 Multi-Objective Path-Planning (MOPP)

Denote a finite set of possible trajectories from \(X_{\text{init}}\) to \(X_{\text{goal}}\) as \(\Sigma = \{\sigma_i\}\). Each trajectory \(\sigma_i\) differs with respect to different objectives it fulfills. Let \(c_j\) denote a cost function that maps trajectories into a real number, \(c_j : \sigma \to \mathbb{R}\), and suppose that there are \(J\) different
cost functions. Each $\sigma_i$ is associated with a cost vector that has $J$ elements defined as 
$c(\sigma_i) = [c_1(\sigma_i), \ldots, c_J(\sigma_i)]^T$, where each $c_j$ represents the $j^{th}$ objective.

The multi-objective path-planning problem is to find a trajectory $\sigma$ such that the resulting cost vector $c(\sigma)$ satisfies some trajectory predicate, $P(\sigma)$. $P(\sigma)$ is true if a set of given objectives are satisfied, otherwise false. For example, a trajectory predicate could be to find the path that minimizes the cost for objective $j$, in which case the solution to the multi-objective path-planning problem would be $\sigma^* = \arg \min_{\sigma \in \Sigma} c_j(\sigma)$. Similarly, a trajectory predicate could be to find the path that uses a weighting vector $w = [w_1, \ldots, w_j]^T$ to find a tradeoff among objectives that satisfies $\sigma^* = \arg \min_{\sigma \in \Sigma} w^T c(\sigma)$.

1.2.3 Collaborative Human-Robot Path-Planning

Assume that the robot’s initial configuration is given. A collaborative human-robot path-planning problem requires (a) a human to specify two decision-making elements, the goal state, $X_{\text{goal}}$, and the trajectory predicate, $P$, and (b) a robot to generate and follow trajectory solutions, $\sigma$, that reach the goal state and satisfy the trajectory predicate. We assume that the human will only be interested in trajectory predicates that allow optimal trajectories, that is, trajectories that are on the Pareto front, as in other multi-objective problems [42]. We further assume the human will rarely be interested in specifying a very specific weight vector, but will rather use natural language-like descriptors like “find a path that is both safe and stealthy”.

Given these assumptions, the algorithmic problem to be solved is for the robot to generate a finite sample of trajectories from the Pareto front given $X_{\text{init}}$, $X_{\text{goal}}$, $P$, and the robot’s knowledge of the environment. Let $\hat{\Sigma} = \{\sigma_1, \sigma_2, \ldots, \sigma_T\}$ represent $T$ trajectories sampled from trajectories on the Pareto front. The human problem to be solved is to allow the human analyze $T$ with respect to the $J$ objectives and select a trajectory that matches his or her intent. Please note that, $\hat{\Sigma}$ is assumed to be a convex Pareto front [89], with solutions well spread across it. This assumption is important because this dissertation does not address
the well-acknowledged problem of generating uniform and well-populated samples from a Pareto front.

For three adverbs or objectives, let \( \mathbf{h} = [h_1, h_2, h_3]^T \), be a three-element vector that represents human intent. Each \( h_i \in [0, 1] \), meaning that a value of 1 indicates utmost preference of the corresponding objective. Let \( \sigma_H \in \hat{\Sigma} \) represent the trajectory selected by the human (\( H \) for Human) from the sample of trajectories on the Pareto front, \( T \). When cost functions, \( c_j(\cdot) \), produce a convex Pareto front, selecting a trajectory \( \sigma_H \) is equivalent to selecting a preferred cost vector, \( c(\cdot) \), associated with the trajectory that dictates preference among the \( J \) objectives. One contribution of this dissertation is creating a mechanism of selecting the path \( \sigma_H \) for the intent vector, \( \mathbf{h} \), in an appropriate metric space. This unique vector is expressible using appropriate metaphors in a graphical user interface.

1.2.4 Collaborative Human-Robot Path-Planning in a Dynamic Environment

In a dynamic environment, things can change while the robot is following its trajectory. The human’s intent stays constant as the environment changes, even as the cost vector varies over time.

Thus, a new problem emerges, namely adapting the robot’s trajectory so that it satisfies the human’s intent. While following the selected trajectory, the one that initially matched human intent, the robot simultaneously computes a new trajectory, one that maintains intent as the world changes. The replanned trajectory, \( \sigma_R \), is the result of ongoing perception and updated knowledge of the environment that the robot receives. The design question is therefore, under what circumstances should the robot switch from it’s current trajectory \( \sigma_H \) to \( \sigma_R \)? A trigger is an event that gives the human an opportunity to switch to the replanned (newly-adapted) path. Adapting new paths in response to environmental changes is referred to as replanning.

Four classes of triggers are considered: (a) time-based: replanning at regular time intervals, (b) intent-based: replanning when the executing path no longer matches intent,
(c) region-based: replanning, for example, when there is reason to believe that a better path can be obtained from a different homotopy class\(^1\), and (d) user-initiated: replanning at the complete discretion of a human.

1.3 Related Work

1.3.1 Research Area Overview

There are four major areas of related work: (a) path-planning and replanning algorithms, (b) multi-objective planning, (c) human supervisory control and intervention, and (d) models of intent.

Planning/Replanning Algorithms

In goal-based navigation, a robot has to move from a start configuration to a goal configuration. Researchers have created several path-planning algorithms, both for static and dynamic environments [33, 55, 69, 75, 86, 94, 95, 98, 110, 152]. Path-planning algorithms can generally be categorized into four types: sampling-based approaches, graph-based approaches, field-based approaches, and parametric curve-based approaches. Sampling approaches include probabilistic roadmaps (PRM) [87], rapidly-exploring random tree (RRT) [97], PRM*, and RRT* [85]. Graph-based approaches include A* [68], D* (A* for dynamics worlds) and its variants [94, 152], and MOD* Lite [119]. Field-based approaches include potential fields of various forms [32, 88, 168]. Genetic algorithms can be combined with other planners or used to find near-optimal paths using parameterized curves [47, 58, 72, 107].

Many replanning algorithms are triggered by environment changes such as the emergence of obstacles [49, 57, 120, 171, 174]. For replanning [94, 152, 164], it is often computationally infeasible to generate a new path from scratch. Rather, the current path is fixed or repaired. Most existing replanning algorithms find shortest paths [94]. Replanning is sometimes called “reactive navigation” or “reactive motion navigation” [54, 128].

\(^1\)The path \(\sigma_1\) is said to be homotopic to \(\sigma_2\) if \(\sigma_1\) can be mapped to \(\sigma_2\) without encroaching on any obstacle.
Planning and replanning are applicable to many types of robotic applications besides robot navigation including robot arm manipulators [174], unmanned aerial vehicles [90, 171], and unmanned water [18, 124] and surface vehicles [96].

**Multi-Objective Planning**

In real-time navigation, multiple objectives are often important. Objectives include path length, energy consumed, coverage, smoothness, traversal risk, safety, stealth, etc. [55, 58, 99, 119, 164]. Multi-objective path-planning is typically applied only to static environments [42, 79, 106].

Research on combining multi-objectives and replanning is rare [161, 164]. An example of multi-objective path-replanning is found in [161]. Their Incremental Search Engine produced plans that are optimal with respect to weighted combinations of minimum plan length and energy cost. Other work uses a PRM-based approach, that continuously correct paths in response to a changing environment [164]. These authors view the cost of a trajectory as a function of time for traversal, traversal risk, stealth, and visibility. As the agent navigates through the changing environment, it receives updated information and improves its trajectory to reflect these changes.

**Human Intervention**

This work combines algorithmic (re)planning and human supervision. This puts the work in the category of human supervisory control [146]. According to Sheridan [146], supervisory functions include monitoring the automatic action to detect failures and intervening to specify a new goal in the event of trouble.

Human monitoring and intervention is important because, as Cummings notes, “... automated planners cannot always generate accurate solutions...” [38]. Cummings also asserts that automation will occasionally recommend a plan worse than the current plan. The triggers defined in this research serve the monitoring purpose. The trigger mechanism is
analogous to the term *intervention* as used by Scholtz [137], which means identifying when the expected actions of the robot are not appropriate given the current situation. By means of this intervention, the user either revamps a plan or gives more specific commands to the robot to modify behavior.

**Intent**

Intent and intention have been studied in myriad ways in many disciplines; art, psychology, science, animation and technology. Most of this literature expresses intent as a mental state that encodes a commitment to achieve something in the future through a plan or partial plan. Many theories suggest that intentions are precursors to action [6, 20, 35, 40, 44, 62, 105]. Following Bratman [21], Tomasello et al. [160] describe intent as an action plan that the organism chooses and commits itself to in pursuit of a goal. Intention therefore includes both a means (action plan) as well as a goal.

In human-robot interaction (HRI) applications, intent is generally “owned” by the human and expressed through a command and/or correction. Commands dictate (a) *what* the robot should do and (b) *how* to do it. What a human *wants* is often tightly coupled to decision-making under multiple conflicting objectives/constraints.

In HRI enabled robot navigation, human intent should be communicated to the robot, either explicitly or implicitly [8, 12, 24, 93]. Intent stated directly is *explicit*, such as in the case of the Curiosity Rover where engineers on earth send streams of commands of the nature *FORWARD 5 M then TURN 90 DEG* to the robot\(^2\). On the other hand, intent that is not communicated overtly but rather inferred from behavior and actions of the agent (human or the robot) is *implicit* intent. In addition to estimating intent from observations of human actions, implicit intent can be inferred using measurement of physiological signals such as ECG (Electrocardiogram), EMG (Electromyogram) [136], EEG (Electroencephalogram) [17, 37], skin conductance, pupil-dilation, and eyeball movement [3, 29, 76].

\(^2\)http://www.extremetech.com/extreme/143884-how-nasa-drives-mars-rover-curiosity
From a broader perspective, modes of communicating intent can be classified as verbal [112, 118] or non-verbal [12, 24, 52, 109, 114]. Bauer, et al.’s survey of human-robot collaboration [12] identifies five ways of communicating intent: speech, gesture, action, haptic signals, and physiological signals. The most common forms of verbal communication are speech/voice [116, 118, 127] and natural language commands [19, 71, 108, 112, 157]. Well-established non-verbal interaction techniques include gesture [129, 158, 170]. For remote robots, interaction is generally through conventional graphical user interfaces such as display, keyboard, mouse, etc., so intent must be inferred from human activities using these input devices. For example, Kadous et al. [83] present the design and implementation of a display/keyboard interface for teleoperating an urban search and rescue robot using metaphors based on computer games, flight simulators and mobile phones. Conventional gadgets such as Personal Digital Assistants (PDAs) [50, 123] and tablets [11] are also used as interfacing devices to communicate intent. One such supervisory control interface for collaborative human-robot exploration was achieved using a PDA [50].

Multimodal communication is often used in robotic applications [1, 11, 122, 123, 132, 144, 162]. Abioye et al. [1] demonstrated a multimodal speech and visual gesture interface to control aerobots for patrol, speech and rescue applications. In their work, speech is combined with visual gestures to generate control output for the UAV.

1.3.2 Research-Related Work

Planning and Replanning

This work requires a planner that approximates the Pareto front for static environments and a planner that does real-time replanning. Two planners were previously developed in our lab: MORRF* [172], which generates paths on the Pareto front, and online fast marching tree (O-FMT*) [30], used for real-time replanning. The design and evaluation of interfaces in this work uses MORRF* planners. The replanning component of this work employs O-FMT* planners.
When to replan depends upon the intent one wants to achieve. In Hyperion robot navigation [161], a replan was triggered if the rover did not reach the expected navigation waypoint at the scheduled arrival time. The Hyperion replanning trigger is thus connected to rover speed and path progress. In contrast to progress-based replanning, Cummings et al. [38] studied how time-based replanning triggers and their associated replanning rates affected operator performance and workload when supervising a decentralized network of heterogeneous unmanned vehicles. The author concluded that operator workload and situational awareness are affected by the frequency of replanning.

Klanke et al. [92] present a real-time path planner for the 7-DOF Mitsubishi PA-10 arm. Their algorithm, responsible for the end-effector’s position and orientation, generates paths online by collecting obstacle information on-the-fly. Yoshida et al. [174] explored replanning for a robot manipulator arm, with and without environment changes. The authors use two threads, one for execution and the other for planning. When a collision is expected along the current path, the execution thread queries the planning thread for a better plan. The planning thread then starts replanning. The execution continues to execute after sending the query signal till a point where it can stop successfully if replanning is not finished; otherwise, it will switch to the new plan from the planning thread. This can be classified as collision-based replanning.

Wzorek et al. [171] illustrated three replanning strategies: full replanning, partial replanning, and plan repair. In full replanning, the path is always replanned from the next waypoint to the final waypoint. In partial replanning, replanning starts with the waypoint where a collision is anticipated, leaving all previous waypoints intact but potentially changing all subsequent waypoints. In plan repair, replanning is done only for all waypoints where a collision is anticipated, attempting to retain waypoints where no collisions occur.

Thus, the replanning literature identifies progress-based, time-based, and collision-based replanning. Furthermore, the literature allows replanning to occur in parallel with path execution, or, only when problems are anticipated or detected. Our work uses time-based
triggers and also includes a replanning trigger when the current path deviates from human intent.

**Communicating Intent**

Intent in this dissertation leans towards Bratman’s [20] and Malle et al.’s [105] theories of intentions. Bratman approaches intentions by way of planning theory. Accordingly, intentions are *partial plans* brought about by deliberation and practical reasoning in consideration of resources and coordination. Plans may get updated over time in such a way that they eventually bring about the desired outcome. In this dissertation’s intent framework, the robot has a knowledge of the environment and updates its partial plans as needed. These plans, along with other spatial information such as robot’s position, enemy position, etc., are assumed to be known and can be presented to the user.

Here, in this research we explore robot path-planning that works in a dynamic environment and that is in sync with human intent. Specifically, the start and end states of the path are given, and the human then expresses desired qualities for the trajectory, specifying a tradeoff between conflicting objectives. We created and compared a series of graphical user interfaces for command. Commands to the robot can be in the form of plans, images, sketches, etc. on the display/keyboard control interfaces that explicitly dictate the intent. Such interfaces are convenient in HRI applications where the robot is working remotely in difficult, dangerous, and unstructured environments [147].

For intent communication, this work is close to a teleoperation controller for mobile robots [51]. Important elements of an effective teleoperation controller in a dynamic environment include visual feedback and spatial feedback (images and/or video, and maps), and situational feedback such as a robot’s current position [51].
1.4 Thesis Statement

In multi-objective robot path-planning with a known start and end state, human intent can be expressed as a predicate using a graphical user interface that uses an appropriate visual metaphor. This predicate can be associated with a tradeoff among a set of Pareto-optimal paths. The selected tradeoff is a specific trajectory/path that reaches a desired end state while balancing desired objectives. The trajectory satisfying the predicate is managed by a human and is executed by the robot. In dynamic worlds, appropriately-defined replanning triggers enable a robot to satisfy the human’s intent over the duration of a mission with acceptably low tolerable levels of human interaction and workload.

1.5 Intent-Based Multi-Objective Path-Planning

Figure 1.1 illustrates the execution phases for intent-based planning and replanning. The life cycle starts when the human formulates and expresses intent and the cycle ends when the goals associated with the intent are accomplished. In between, the robot follows the planned or collaboratively replanned trajectory. This section describes how intent is expressed and executed in a static world, and the next section addresses replanning.

![Life cycle of human-robot collaboration task.](image)

During execution, the robot may follow an initial plan, an adaptable plan, and a closing plan. In the initial plan phase, the robot follows the original planned trajectory, $\sigma_H$, transitioning from one configuration to another until a replanning event. In the adaptable plan phase, the robot needs to be aware of the environment and adapt plans accordingly. The figure illustrates that this phase is metaphorically wider than the initial phase to emphasize
that the robot may have to replan multiple times. When the robot is close by the goal state
the human and robot may decide to ignore intent rolling into the closing plan. Such disregard
towards intent may be based on the following two factors. First, the replanned trajectory
may require too many state transitions from the current state to the goal state that may
have the risk of recursive replanning. Second, there may be no significant difference in the
current trajectory and the replanned trajectory.

The remainder of this section formulates multi-objective path-planning for static
environments. This includes formulating meaningful cost functions for robot path-planning,
selecting algorithms for generating trajectories on the Pareto front in static worlds, represent-
ing intent, selecting paths that match intent, and designing GUIs to test whether humans
can select paths that match intent. We have published some of the work for static worlds
in [143] and [140].

1.5.1 Creating Meaningful Cost Functions

Since this work is motivated by path-planning in hostile environments such as military,
the initial work [143] multi-objective optimization problem considered three costs, quickly,
stealthily and safely, and current work is exploring efficiently. Each trajectory, $\sigma$, is represented
as a sequence of edges. The “quickly” cost is the sum of the Euclidean distances of each
edge in the trajectory. The “stealthily” cost function is loosely modeled as the probability of
the robot being seen by the enemy. It is the sum of costs for each point on the trajectory,
computed as a function of two factors: the distance of the robot from each enemy and the
visibility of the robot from all enemies. The safety of a collision-free path is the sum of the
inverse distance between the robot position and the nearest obstacle in the environment.
This type of “safely” objective is also referred to as “clearance”, defined as the maximum
possible distance from obstacles [42].
1.5.2 Generating Trajectories on the Pareto Front

Given a set of meaningful objectives, the next step is to generate several possible paths from start to goal. Two path-planners have been explored. First, we evaluated the MORRF* algorithm [172] because it automatically generates a set of trajectories from the Pareto set. MORRF* blends two concepts: optimal rapidly exploring random tree (RRT*) [84] for efficient path finding, and a decomposition-based approach to multi-objective optimization [176].

MORRF* can be slow and is therefore not appropriate when considering replanning. To deal with replanning in a dynamic environment, we evaluated the online fast marching tree* (O-FMT*) [30]. The algorithm generates a new path in less than two seconds given a weighted combination of the speed, stealth, and safety objectives. O-FMT* allows a robot to consider alternative paths for various weightings in approximately real-time. Selecting appropriate weightings so that O-FMT* generates multiple paths uniformly distributed over the Pareto front requires iteration.

We used MORRF* to generate paths for static environments and O-FMT* for planning and replanning in dynamic environments. While O-FMT* is used for replanning, any such fast algorithm could be used in the architecture. Evaluating different algorithms is not, however, in the scope of this work.

1.5.3 Representing Intent

It is useful to illustrate the Pareto optimal set before describing the trajectory portion of intent. For a given path-planning problem, there is a set $\Sigma = \{\sigma_i\}$ of possible trajectories that satisfy the goal portion of intent, that is, the $T$ trajectories travel from $X_{\text{init}}$ to $X_{\text{goal}}$. For a set of objectives, the Pareto optimal trajectories are those trajectories for which further improvement in one objective cannot be obtained without sacrificing some other objective. This can be explained as follows:

Consider two non-trival objectives, $O_1$ and $O_2$ for path planning. Objectives are non-trivial if it is not possible to get the most of objective $O_1$ without sacrificing $O_2$ and vice
versa. Given $O_1$ and $O_2$, the tradeoff between trajectories is illustrated in Figure 1.2. In Figure 1.2 (a), objectives are encoded as payoffs, meaning higher values are preferred to lower values. Each of the red and blue circles in Figure 1.2 (a) denote a trajectory represented by its payoff vector. The extreme right blue circle in Figure 1.2 (a) corresponds to a trajectory that is of highest for objective $O_1$, and similarly, the extreme left blue circle corresponds to a trajectory that maximizes $O_2$. All other blue circles on the blue curve indicate the best trajectories for different tradeoffs between $O_1$ and $O_2$. Notice that each of the red trajectories are “dominated”, meaning, there is another trajectory in which all payoffs are higher. The Pareto front, the blue curve, is made up of non-dominated trajectories. In the chapters that follow, the entire Pareto front is not fully specified because that would require finding too many possible paths. However, each path found corresponds to an actual point on the Pareto front because each path is found by maximizing a weighted sum of objectives.

**Normalization**

For static worlds, we assume that the user will associate objectives with payoffs so that, “more stealth” or “more safety” are associated with increases in the objective function. However, the objectives used by the path planner are expressed as cost functions, and each cost function is...
expressed in its own units, units that are not necessarily commensurate with each other. We use normalization to change costs into payoffs and to make payoffs commensurate as follows.

In \( c(\sigma_i) = [c_1(\sigma_i), \ldots, c_J(\sigma_i)]^T \), each cost function, \( c_j \) is translated into commensurate payoffs, that fall between 0 and 1 as follows. Let \( \Sigma_P \) denote the set of trajectories on the Pareto front. The normalized payoff objective \( o(\sigma_i) \) is obtained as follows:

Each of the cost term \( c_j \) is converted to payoff term as \( p_j(\sigma_i) = (-1) * c_j(\sigma_i) \). The normalized payoff objective for trajectory \( \sigma_i \) is then:

\[
o_j(\sigma_i) = \frac{p_j(\sigma_i) - \min_{\sigma \in \Sigma_P} p_j(\sigma)}{\max_{\sigma \in \Sigma_P} p_j(\sigma) - \min_{\sigma \in \Sigma_P} p_j(\sigma)}
\]

resulting into trajectory payoff vector \( o(\sigma_i) = [o_1(\sigma_i), \ldots, o_J(\sigma_i)]^T \).

**Intent on the Pareto Front**

For three adverbs or objectives, let \( h = [h_1, h_2, h_3]^T \), be a three-element vector that represents human intent. Each \( h_i \in [0, 1] \), meaning that a value of 1 indicates utmost preference of the corresponding objective. Higher values mean “more stealth” or “more safety”, and a value of \( h_i = 0 \) means “less stealth”, etc. The vector \( h \) represents human intent as a point in the tradeoff space between objectives. For example, if \( h = [1, 0, 0]^T \) then the human wants trajectories that pay attention to only the first objective, and if \( h = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]^T \) means that the human wants each objective weighted equally.

Figure 1.2(b) illustrates how the normalized objective vectors on the Pareto front, \( o(\sigma_i) \) and the human intent vector, \( h \) are represented in the same payoff space. For simplicity, this is illustrated when there are two objectives. The solid blue dots on the curved blue line represent the Pareto front. The vectors emanating from the origin represent possible human intent vectors.
1.5.4 GUI Design

This subsection summarizes completed work on how using color as a visual metaphor enables a human to express the intent vector, $\mathbf{h}$.

Color-Based Trajectory Predicates

If paths are represented by payoff vectors, $\mathbf{o}(\sigma_i)$, and human intent is represented as a tradeoff vector, $\mathbf{h}$. An important research task is to find a way to present intent graphically as a predicate over entire trajectories. We present here a color based metaphor to represent intent. For example, color allows a user to express the safest path using the “blue path” trajectory predicate or a tradeoff between quickly and stealth as the “brown path” (mix of red and green) trajectory predicate.

Our previously published work considered three objectives/adverbs for path planning [140, 143]. The intent predicate, $\mathbf{h}$ and the objective vectors, $\mathbf{o}(\sigma_i)$, are scaled so that a) each element $h_i$ and $o_j(\sigma_i)$ fall between 0 and 1 and (b) $h_1 + h_2 + h_3 = 1$, that is, $\sum_{i=1}^{3} h_i = 1$. Each intent component $h_i$ was mapped uniquely to one of the RGB colors, which is equivalent to restricting the color palette to $R + G + B = 1$. Red was associated with ‘quickly’, green with ‘stealthily’, and blue with ‘safely’. Figure 1.3 illustrates.

![Figure 1.3: Example of a quick path (left) and a stealthy path (right).](image)

Matching Intent to Robot Paths  The trajectory predicate, $\mathbf{h}$, is expressed as a color, so a method is needed to determine the color of each trajectory, $\sigma_i$. We compared multiple ways for matching the payoff vector with human intent including cosine similarity, TOPSIS,
WPM [9], and Euclidean distance. The cosine similarity between a path vector, \( \mathbf{o}(\sigma_i) \), and the human intent vector is \( \mathbf{h} \) is:

\[
CS(\mathbf{h}, \mathbf{o}(\sigma_i)) = \frac{\mathbf{h} \cdot \mathbf{o}(\sigma_i)}{\|\mathbf{h}\| \|\mathbf{o}(\sigma_i)\|},
\]

and the trajectory that best matches human intent is given by \( \sigma_H = \arg \max_{\sigma_i \in T} CS(\mathbf{h}, \mathbf{o}(\sigma_i)) \).

We subjectively evaluated each measure from the above list. Except for cosine similarity, the other measures produced numeric values based on which it could not be determined to what degree certain path matched the intent. Moreover, cosine similarity is a simple computation that uses only the dot product of two vectors, and given the positive path and objective vectors, produced similarity values between 0 and 1. TOPSIS and WPM are relatively computationally expensive.

**GUI Designs** We designed an interactive graphical human-robot interface called the *Adverb Palette (AP)*. The AP has two panels: the left panel displays possible paths and obstacles, a depiction of the robot, and the start and end states. The right panel includes three different ways that the color metaphor can be presented to the human: *the palette, the sliders, and the prism*. Figure 1.4 shows the AP with the *palette* interface on the right panel. Figure 1.5 (a) and Figure 1.5 (b) show the sliders and the prism GUIs, respectively. Using any of these GUIs, the human can express the intent predicate, \( \mathbf{h} \) using the tools in the right panel.

![Image of GUI designs](image-url)
Problems with Color  There are problems with using color as the metaphor for the intent predicate. One important problem is that it doesn’t scale well beyond three colors. Another problem is that color only works well if there are no dominant paths or nearly dominant paths, meaning that one or two paths are superior to all others. A third problem is that there can be “holes” in the Pareto front, meaning that there can be portions of the convex closure of the Pareto front that for which there is no feasible path. A fourth problem, addressed in the next section, is that color may not work well when objectives change over time.

Consider the problem caused by dominant or nearly dominant paths. Suppose that there is a single path that is more stealthy, more safe, and quicker than any other path. Suppose further that the human only cares about one objective, say stealth. The human specifies the blue path, but the most blue path is also the most green path and the most red path. The intent trajectory predicate is expressed as a blue path, but a green path and a red path and a brown path all match the human’s intent as well. Color allows ambiguity.

Region Metaphor  Unlike the palette and the prism interfaces, the sliders interface allows the user to visually see that the most stealthy path might also be the quickest and safest path. In the sliders interface, it is possible for a single path to produce maximum values of red, green, and blue. In addition to colors, sliders also use regions to define which paths match which intents, allowing for some paths to satisfy multiple intents. Sliders include a
spatial region, namely being near the top of the slider scale, to indicate that a single path might maximize each objective.

Another approach that uses regions is the use of so-called “cluster analysis” graphs [154], which show how particular solutions to multi-objective optimization problems can cluster in regions delineated by each objective. Figure 1.6 illustrates how the objective vector, \( o(\sigma) \), can form unique shapes for different paths; the figure shows three objectives. Path \( \sigma_i \) maximizes objective 1 and has lower values for objectives 2 and 3. Path \( \sigma_k \) doesn’t maximize objective 1, but it does produce a high value for this objective while also producing high values for the other objectives.

![Figure 1.6: Spider web metaphor.](image)

**Solution to Cosine Similarity Intent-Path Match** We defined and evaluated an alternative measure, *intent threshold margin*, that indicates when a path matches intent. The metric was motivated by the limitations of cosine similarity. In simple words, if a pure intent is expressed, cosine similarity would fail to match a path that has better payoffs for multiple objectives. It would seek a path that has good payoffs only for the specified objectives, even though better payoffs for the unspecified objectives would be more desirable to the human. Note that, if the user specifies that objective 1 is most important in the problem illustrated in Figure 1.6, then the cosine similarity metric would select \( \sigma_i \) as the best match. However, path \( \sigma_k \) produces a low cosine similarity value even though it nearly maximizes objective 1 while simultaneously producing high values for the other objectives.
1.6 Intent-Based Multi-Objective Path-Planning in a Dynamic Environment

This section considers intent-based path-planning when the environment changes with time. When objectives change over time, the initial chosen trajectory may fail to meet the objectives while the robot moves to the goal. For example, suppose that the selected trajectory was to evade enemies in the environment (maximizing stealth) but during execution the enemy moves really close to the initially planned trajectory (reducing stealth). Under such conditions, an alternative path needs to identified. In terms of Figure 1.1, a changing environment means that the plan must be adapted over time.

1.6.1 Evaluate Replanning Triggers from the Literature

The first research task for a changing environment is to understand which replanning triggers from the literature make sense for a human. Triggers are events that signal the human to consider replanning. While this research defines replanning moments as triggers, in the past literature a trigger is identified with terms such as ‘intervention’, ‘correction’, ‘automation’ etc. On a trigger, a GUI communicates (a) the robot’s current location, (b) the current path, (c) one or more replanned paths, and (d) tradeoffs associated with different path choices. On a trigger, the interface pops up buttons that allow the user to either ‘Stay with the current path’ or ‘Switch to the new path’.

Before describing the four triggers of this document, we briefly review ideas related to triggers from the literature. Time triggers are often used in supervisory control to help the human monitor progress at regular timings to “check in” to see if something is wrong (see, for example, [38]). Next, many complex human-machine systems rely on alerts or alarms to signal important changes occurring in a system (see, for example, [150]). An alarm indicates that system behavior does not match expected execution. A trigger can be thought of as a type of alarm, one indicating that replanning may need to occur.

One potentially important deviation from expected behavior is if a replanned path goes through different regions of the environment with respect to the current path, referred
to as belonging to a different homotopy/topology class than the original path. For example, an original path might pass to the left of a building and a replanned path might pass on the right. Finally, instead of being automatically prompted, a user may discretionarily choose to replan for whatever reason — resulting in a user-initiated trigger. This is similar to the work of Cummings [38] where an operator can elect to select a ‘replan button’ anytime, even when it is not prompted by automation.

The work here describes and evaluates four different types of triggers: time, intent-mismatch, path-divergence, and user-initiated. We now discuss each trigger type.

**Time Trigger**

Nominally, a time trigger occurs at deterministic (and possibly varying) time intervals to make the human aware of the current environment and the two paths: the current one and the replanned one.

**Intent-Mismatch trigger**

If the costs of the current path trajectory change beyond a certain predefined threshold, the trajectory might no longer meet the user’s objective(s) and an intent mismatch has occurred. Intent mismatches can be detected by evaluating how well the original trajectory satisfies the trajectory predicate. We considered two types of intent mismatch.

**Cosine Similarity metric.** The color of a path (visible on the GUI) can change in dynamic worlds, such as when an enemy agent moves. When intent is represented using the color metaphor, an intent mismatch occurs when costs change in such a way that the color of the original path changes. If color changes beyond a certain predefined threshold, then the path should be replanned. Changes in cosine similarity can be used to measure changes in color.

**Intent Threshold Margin.** When intent is represented using a region-based metaphor, the intent threshold margin detects when a change in an objective exceeds
an acceptable threshold and triggers replanning. Since there can be correlations in how the objectives change over time, cosine similarity is not always effective for identifying changes in region, thereby motivating the need for the intent threshold margin.

**Homotopy Trigger**

A homotopy trigger takes into consideration diversity in paths that reach the goal in diverse ways. A change in homotopy occurs when the current and replanned paths go around obstacles in different ways. That is, a homotopy trigger occurs when the robot’s replanned path and original path are not in the same homotopy class.

**User-Initiated Trigger**

The human can pause the robot anytime and put it on a replanned path.

### 1.6.2 Evaluation of When Humans Replan or Rationale behind Triggers

Having given a rationale for the above triggers, it is not fully understood if the triggers are sufficient to keep track of the navigation intent. Hence, this work conducted user studies to explore the usefulness of the above described triggers that enable intent-based navigation in dynamic environments.

The studies include recording a set of robot navigation tasks executed on the GUI from the point in time that the intent is initially made to the time that the robot reaches its goal stepping through the above triggers. The recordings are in the form of both (i) a time lapse image sequence, and (ii) videos. These navigation tasks were presented to the users of Amazon’s Mechanical Turk. The objective was to determine the usefulness of the time trigger, the intent-mismatch trigger, and the homotopy trigger.
1.7 Dissertation Chapters

This dissertation is compiled as a set of papers published as independent units in different venues. Each chapter presents a paper describing one or more elements of the intent framework for robot path planning and replanning in both static and dynamic environments. The second-to-last chapter is work that has not been submitted to a conference yet. The following section gives a brief description of each chapter.

Chapter 2, which was published in [143], introduced four designs of a graphical user interface (GUI) called the Adverb Palette (AP) to support interactive path-planning under multiple conflicting objectives; quickly, stealthily, and safely. The human-robot interface helped the operator to issue commands to the robot such as ‘go quickly’, or ‘go quickly and stealthily’ through a set of interfaces: palette, sliders, and prism. The chapter describes in detail how the Adverb Palette represents tradeoffs between the robot paths with three objectives. Next, this chapter models the cost functions that are used for robot path planning that are used in both static and dynamic environments. This chapter also introduced the cosine similarity metric that was used to map a path to the expressed human-intent.

Chapter 3, which was published in [140], evaluated the GUI designs of palette, sliders, prism, and waypoints for selecting tradeoffs among Pareto optimal solutions. The controlled laboratory user study indicated the ease of palette and sliders over the other two interfaces. This work in turn evaluated the representation of intent through different interfaces.

Chapter 4, which was published in [141], discusses three system-initiated triggers (prompts) for path-replanning. The triggers are to replan (a) at regular time intervals, (b) when the current robotic path deviates from the user intent, and (c) when a better path can be obtained from a different homotopy class. Further, it considers one user-generated replanning trigger that allows the user to stop the robot anytime to put the robot onto a new route.

Chapter 5, which was published in [142], presents a replanning framework with three elements: (a) the integration of fast online path-planning algorithms that generate trajectories
conforming to the given intent; (b) a mathematical model that says when replanning must happen; and (c) an evaluation of events that trigger replanning. It presents a study of 50 MTurk participants to assess what replanning triggers best enable a human-robot collaboration to persistently satisfy intent.

Chapter 6; “A Measure to Match Robot Plans to Human Intent: A Case Study in Multi-Objective Human-Robot Path-Planning”, is submitted to the RO – MAN 2020 conference. The paper introduces the intent-threshold margin and presents its initial evaluation. An MTurk study of 50 participants reveals that, for a given intent, there is a close match between the ranks given by the participants and the ranks induced by the metric.

Chapter 7 presents an MTurk study that compares various triggers for replanning, including the intent threshold margin. This chapter can be seen as a continuation and extension of Chapter 6 to include the intent threshold margin.

Chapter 8 concludes the research and summarizes the major contributions. It talks about future work of evaluating a so-called “spider web interface” that would be an alternative to palette, sliders and prism. Since, this work emphasizes multi-objective path planning, a high-level perspective of its potential for some other applications is discussed.
Chapter 2

Interactive Multi-Objective Path Planning through a Palette-based User Interface


Abstract

In a problem where a human uses supervisory control to manage robot path-planning, there are times when human does the path planning, and if satisfied commits those paths to be executed by the robot, and the robot executes that plan. In planning a path, the robot often uses an optimization algorithm that maximizes or minimizes an objective. When a human is assigned the task of path planning for robot, the human may care about multiple objectives. This work proposes a graphical user interface (GUI) designed for interactive robot path-planning when an operator may prefer one objective over others or care about how multiple objectives are traded off. The GUI represents multiple objectives using the metaphor of an artist’s palette. A distinct color is used to represent each objective, and tradeoffs among objectives are balanced in a manner that an artist mixes colors to get the desired shade of color. Thus, human intent is analogous to the artist’s shade of color. We call the GUI an “Adverb Palette” where the word “Adverb” represents a specific type of objective for the path, such as the adverbs “quickly” and “safely” in the commands: “travel the path quickly”, “make the journey safely”. The novel interactive interface provides the user an opportunity to evaluate various alternatives (that tradeoff between different objectives) by allowing her to visualize the instantaneous outcomes that result from her actions on the interface. In
addition to assisting analysis of various solutions given by an optimization algorithm, the palette has additional feature of allowing the user to define and visualize her own paths, by means of waypoints (guiding locations) thereby spanning variety for planning. The goal of the Adverb Palette is thus to provide a way for the user and robot to find an acceptable solution even though they use very different representations of the problem. Subjective evaluations suggest that even non-experts in robotics can carry out the planning tasks with a great deal of flexibility using the adverb palette.

**Keywords**  human-robot interaction, multi-objective decision making, user interface, supervisory control

2.1 Introduction

Consider a problem where a human uses supervisory control to manage robot path-planning by evaluating multiple paths generated by an algorithm and assigning a robot to execute one of the paths. Given a set of paths, the task of choosing among these multiple paths places a burden on a human operator, as the human may find it difficult to compare the paths against each other. This triggers the need of a robust and intuitive interface that can act on the output of well established path-planning algorithms, and allow the user to select the most desired path in a way that keeps human workload within acceptable bounds.

This paper proposes a novel human-robot interface called as an *Adverb Palette (AP)* to help the operator issue commands to the robot to take a specific path from the many available paths. Figure 2.1 shows the adverb palette. On the left side of the interface, the map shows in gray all potential paths that a robot can take, and the right side of the interface provides an area that can be used by the human to find tradeoffs among the paths. Based on the command issued on right side panel, one of the gray paths gets highlighted on the left panel.
An *adverb* encodes the objective associated by a verbal command given by the human to the robot. For example consider a command, “Go from point A to point B quickly.” The adverb “quickly” in the command indicates minimizing path length, in other words asking the robot to take the shortest path from point A to point B. For a command, “Go from point A to point B quickly and safely,” two objectives need to be minimized: path length and risk of being exposed to threats in the environment, respectively. *AP* is an interface where such commands can be interactively evaluated by a user. The execution of the command by the robot, and robot’s performance evaluation is not in the scope of this work and is left for future work. The goal of this paper is to present the adverb palette interface and subjectively compare it to two other interfaces.

Although there exist many algorithms for multi-objective optimization (see, for example, [4, 25, 27, 87, 97, 172]) we use the MORRF\* algorithm [172] as it has demonstrated both effectiveness and efficiency in generating Pareto optimal solutions. A solution is Pareto optimal if there is no other solution that is better for every objective. Figure 2.2 shows the Pareto optimal paths discovered by the MORRF\* algorithm in a simple world with two
objectives to minimized. Each point in the curve represents a path and its associated cost of 
*objective 1* and  *objective 2*. The blue square in the top left corner represents a path where the cost of objective 1 is minimum, and similarly the blue square at the bottom right corner represents a path for which the cost of *objective 2* is minimum.

![Diagram of path planning with MORRF*](image)

Figure 2.2: Path Planning with MORRF* for two objectives.

We quote the formal definition of the path-planning problem from [172] as follows: Consider a bounded, connected open set $X \subset \mathbb{R}^d$, an obstacle space $X_{\text{obs}}$, an initial state $x_{\text{init}}$, and a goal region $x_{\text{goal}}$. Consider the set of $K$ objectives determined by a vector function $c(\cdot) = [c_1(\cdot), \ldots, c_K(\cdot)]^T$ defined by $c : \mathbb{X} \rightarrow \mathbb{R}^K$. Denote the obstacle-free space by $X_{\text{free}} = X \setminus X_{\text{obs}}$. Note that $c$ is defined for all points in $X$ in free space.

Again quoting from [172], the solutions that satisfy the following equation are Pareto Optimal.

$$
\arg \min_{x} \max_{1 \leq k \leq K} \{ \lambda_k \left( |c_k(x) - z_{k}^{\text{utop}}| \right) \} \tag{2.1}
$$

where $\lambda = [\lambda_1, \ldots, \lambda_K]^T$ is a weighting vector such that $\sum_{k=1}^{K} \lambda_k = 1$, $z_{k}^{\text{utop}} = [z_{1}^{*}, \ldots, z_{K}^{*}]^T$ denotes the Utopia reference vector, and finally $x$ denotes a potential solution. For details see [172].

Given the Pareto optimal solution set that satisfy Equation 2.1, where each solution represents a path that goes from point A to point B, the goal is to enable a user to find a tradeoff that best expresses his or her intent. Expressing intent has two subproblems to be solved:}
1. Design an interface to construe human intent, and,
2. Design an algorithm that translates from the interface input to one of the Pareto optimal paths that most closely matches human intent.

2.2 Related Work

The design and use of Adverb Palette (AP) relates to user interface design, multi-criterion/attribute/objective decision making, human-machine systems, human-factors, human-robot interaction, ecological interface design, cognitive engineering systems etc.

Making trade-offs in decision-making is known as multiple criterion decision-making [166], multiple attribute decision-making [74], etc. The goal however remains the same as to make preference decisions over available alternatives, or in other words, to choose from among a finite set of discrete alternatives [56]. This paper uses three objectives: minimizing distance from the robot’s start location to a goal location, avoiding exposure of the robot to one or more enemies, and avoiding collisions with obstacles.

A great deal of emphasis has been on designing powerful and easy to use interfaces [2, 10, 14, 15, 31, 67, 83, 165]. Many in the field also elaborate on the challenges and complexity of practical design problems. The AP is closely related to ecological interface design [165], which is based on a taxonomy of skills, rules, and knowledge used in cognitive control [131]. An ecological interface should not contribute to the difficulty of task, and at the same time it should support the entire range of activities that the operators are faced with. The term ecological (relation between organism and the the environment) corresponds to the operator and the work environment. In our scenario, the work environment is the n-dimensional space that the robot is going to navigate from one location to another, and the AP is at the disposal of an operator to make effective path planning decision. To make the robotic-path planning task easier and intuitive for the operator, we have used the metaphor of a palette. In the current work, the three objectives that we consider are represented by the colors red, green
and blue respectively. Colors are a strong stimuli [103], though the interfaces in this paper would not work for color blind individuals.

Although teams of humans and robots working as peers may be forthcoming, most robots are managed using supervisory control [146]. For example, in search and rescue operation where the robot may be in unstructured and unfamiliar environments [83, 101], strategic decision-making may be necessarily performed by an operator. Designing interfaces for supervisory control is one element of the field of human-robot interaction (HRI) [63]. Designing intuitive and efficient interfaces has been a challenging issue in HRI [46, 48]. However, significant research on HRI is inspired by the principles, and levels of autonomy (LOA) given by [121]. According to types and levels of human interaction [121], a design involving human-machine interaction varies according to level of automation required. Studies has also been conducted in adjusting the autonomy responding to the environment and workload changes [64]. Considering the given LOA, AP allows the user to make decisions on paths, and then delegate the task to the robot. Note that recent work has identified a critical need to move past the limitations of LOAs on human-robot teaming [82].

As previously mentioned, robots are transitioning from functional tools to interactive teammates [53, 125, 126, 156]. Robust level of robot intelligence will cause HRI to evolve beyond command and control methods. Human mental models [126, 156] for human-robot teams dictate how humans expect a robot to plan and execute tactical movement commands under constraints like “navigate quickly”, “navigate stealthily”, and “navigate safely”. AP provides a medium to explicitly express the human expectations on the interface with the help of adverbs in order to plan the path for robot.

2.3 Adverb Palette

The Adverb Palette (AP) is a mouse-based interactive graphical user interface designed for motion-planning for robots. It provides selection and visualization of possible routes/paths that a robot can take to go from a start location $x_{init}$ to a goal region $x_{goal}$, given the
configuration space. The AP interface helps the user to blend objectives in a way a painter blends colors on a palette. A blend/mixture of objectives corresponds to one path from the available Pareto optimal paths. As shown in Figure 2.1, the Adverb Palette has two parts: the map in the left panel that aids visualization and the command interface (CI) in the right panel through which the user can balance different adverbs. In short, the right panel is the command area for the user actions, and the map is the area that gets updated by highlighting the path according to the user’s action on the CI. We will explore two types of AP interfaces.

Consider an AP interface that supports three adverbs: Quickly, Stealthily, and Safely, symbolized by colors red, green and blue, respectively on the CI.

- **Quickly**: A command to the robot to consider a path which has shortest distance.
- **Stealthily**: A command to the robot to consider a path that avoids being viewed by enemies.
- **Safely**: A command to the robot to consider a path that stays away from obstacles.

We have developed the adverb palette and a complementary interface that uses a different CI which we call the sliders interface. We have also implemented a baseline command interface waypoints input. The map is common to all these options. In each case, the map shows all the routes (paths) in gray, and with no user action a highlighted path is displayed that gives equal preference to all the adverbs. Each of the options has a different way of balancing the adverbs for a particular tradeoff path. The following sections briefly describe each option.

### 2.3.1 Palette

The palette AP displays three initial circles called the “primary dabs,” one for each adverb (objectives). The user can select a path that uses only one objective by clicking on one of the primary dabs; i.e. to take the shortest path, most stealthy path, or most safe path by clicking quickly, stealthily, or safely, respectively. The user can also issue a mixture of the
above adverbs by dragging and dropping adverbs into the white area of the CI to create smaller circles called “paint dab” that blend colors, similar to the way a painter mixes the colors in her paint dab to get a desired shade. Line segments connect the primary and paint dabs, creating a tree structure that allows the human to see the proportions of each objective.

For example, the user can command a robot to go from location A to location B using a path that is both quick and stealthy, which is represented numerically as “50% quickly, 50% stealthily, 0% safely”. This numerical mixture can be made by dragging the adverb *quickly* into a new paint dab, and later on dragging the adverb *Stealthily* onto it. Blending in multiple adverbs (colors) is thus equivalent to making trade-offs with multiple objectives. The default magenta paint dab in Figure 2.1 is thus an example of a mixture “33.33% quickly, 33.33% stealthily, 33.33% safely”. Such a mixture may be desired if there is a command for the robot to move from location A to B such that it should move fast, should stay away from both obstacles and the enemy.

The pie graph on the lower left area in the CI shows the proportion of each objective in a particular paint dab. The paint dab thus represents a human command. By mixing different paint dabs, a user can visualize the consequences of different commands. Figure 2.3 shows an example command where the user desires a path that is quick and safe but does not care about being seen by enemies.

We now discuss the algorithm used to translate the user commands into a tradeoff between objectives. Let $K$ denote the number of objectives and let the user’s action be denoted by $x$. The CI highlights the paths on the map for each potential solution $x$. Let $dab_d$ represent any paint dab on CI. Let $n_i$ be the number of times the user has dragged adverb $i$ on $dab_d$, where $0 > i \leq K$. The total number of drags a user makes for $dab_d$ is:

$$n = \sum_{i=1}^{K} n_i \quad (2.2)$$
Based on the number of adverb drags the user makes on the paint dab, the user’s intent, also referred to as human intent, can be represented as a vector \( \vec{h}_n \) as:

\[
\vec{h}_n = [h_{1n}, h_{2n}, ..., h_{Kn}]^T
\] (2.3)

where \( h_{in} \) is computed as \( n_i/n \). Therefore, \( \sum_{i=1}^{K} h_{in} = 1 \).

In Section 2.7, we will discuss the mapping between the human intent and the path that best matches the intent.

### 2.3.2 Sliders

Figure 2.4 shows the slider interface of the CI. Here the user adjusts the trackbars to get to a desired mixture, and the corresponding path from the left panel is selected. The three sliders represent the three adverbs. The user can issue any of the three primary commands to the robot i.e. to take the shortest path, most stealthy path, or most safe path by sliding the red, green, or blue slider to the maximum units, respectively.
If $max_{scale}$ is the maximum number of units considered for each slider, then at any point of time, the sum of the units on each slider do not exceed the value $max_{scale}$. Therefore, if $max_{scale}$ is 100, and if the red, green, or blue sliders are at say 33, 33, 34 units respectively, then moving the blue slider to 60 units will cause a change to the slider units to 20, 20, 60 units respectively. The adjustment thus guarantees that at any point of time the mixture represents a percentage of each of the adverbs. Unlike the palette, the user can discover paths while moving a slider, and settle down to certain position if she desires it; if the user moves one slider the other two sliders get updated automatically and the corresponding path gets shown on the map.

Let $s_i$ is the number of units on slider $i$, and let $max_{scale}$ be the maximum number of units considered for each slider. The maximum units are determined by the objective values for the set of paths returned by the MORRF* algorithm; the maximum unit is the cheapest path for that objective returned by MORRF* and the minimum unit is the most expensive path returned by MORRF*. The human intent can be represented as a vector $\vec{h}_s$ as:

$$\vec{h}_s = [h_{1s}, h_{2s}, \cdots, h_{Ks}]^T$$ (2.4)
where $h_i$ is computed as $s_i/\text{max}_\text{scale}$. Therefore, $\sum_{i=1}^K h_i = 1$.

Let $\mathcal{M}(H, K)$ represent a matrix of slider values such that:

- each row represents slider values, a unique combination of $K$-slider values treated as a vector $\vec{s}$
- each element in a row represents a slider value, where the slider value $s_i$ is $s_i \in \mathbb{Z} : 0 \leq s_i \leq \text{max}_\text{scale}$.
- the elements in each row add to the value $\text{max}_\text{scale}$
- $H$ is the total number of rows in the matrix, where each row represents a unique combination of the $K$ slider values (in other words there are $H$ combinations), and
- $K$ is the number of columns in the matrix. Each column represents one objective/slider.

In Section 2.7, we will discuss the mapping between the human intent and the path that best matches the intent.

### 2.3.3 Waypoints

The *waypoints* interface assists a user to construct her own path on the map by allowing her to provide location guidelines that the robot should visit while taking a path. Unlike the other two interfaces, the user here does not make a tradeoff among the available paths from the algorithm but instead makes her own path on the map. She can however compare her path with the best or worst with respect to an adverb based on the Pareto optimal paths’ best and worst for that particular adverb. Figure 2.5 shows one of the paths constructed using the waypoints interface.

The interface allows to point to locations using ‘Submit Waypoints’ button. Then when the user is done submitting the main location points that she desire, she can commit these locations to form a path using ‘Commit Waypoints’ button. Any number of paths can be created using the mentioned button pair. Clicking on a paths’ waypoint will give its corresponding path score. All paths can be cleared off to remove mess using ‘Clear
Waypoints’ button. The path score is calculated in a way similar to how the MORRF* algorithm calculates the score for each adverb for a path.

The following sections detail the approach towards finding the costs associated with paths generated by the computer algorithm, and the one generated by the waypoints interface.

2.4 Cost function for adverb ‘Stealthily’

In the introduction, we discussed in brief the adverb ‘stealthily’. Choosing a path that is stealthy means taking a path that is less likely to be detected or seen by the enemies that are posted in the region of interest. Therefore, we express the cost function in terms of the probability of the path being seen by enemy.

Let $X_E$ be the set of the locations of $n$ enemies, $X_E = \{x_{e_i} | x_{e_i} \in X_{\text{free}}\}$, and let the location of the robot be denoted by $x_{\text{rob}}$. We define the cost of stealthiness for any location of the robot $x_{\text{rob}} \in X_{\text{free}}$, in terms of the probability of this point being seen. Let
The stealthily cost given the positions of the robots and the enemies. Defining this as the probability of being seen by any enemy gives:

\[
C_{\text{stealth}}(x_{\text{rob}}, X_{\text{E}}) = P_{\text{Seen}}(\text{true}).
\] (2.5)

Equation 2.5 defines the cost in terms of the probability that the robot is seen by at least one enemy. We will now create a Bayesian network that allows us to compute \( P_{\text{Seen}}(\text{true}) \). Formally, we say that the agent has been seen if it has been seen by one or more enemies. Thus, we have a family of boolean random variables \( \text{Seen}_i \), one for each enemy, and the \( \text{Seen} \) random variable is an accumulation of these.

This means that we will compute \( P_{\text{Seen}}(\text{true}) \) as the marginal distribution from the joint probability of all random variables as follows:

\[
P_{\text{Seen}}(\text{true}) = \sum_{s_1, \ldots, s_n} P_{\text{Seen,Seen}_1,\ldots,\text{Seen}_n}(\text{true}|s_1, \ldots, s_n).
\] (2.6)

We now construct the Bayesian network by which this joint probability will be computed. We adopt a Noisy OR network because it matches our intention that the robot is seen if it is seen by any enemy [134, Chapter 14].

Computing the joint distribution will be done in two steps: First, we propose a simple Boolean network that models how the \( \text{Seen} \) random variable relates to the set of \( \text{Seen by enemy } i \) random variables. Second, we propose a second simple Boolean network that models how the \( \text{Seen by enemy } i \) random variable can be created from component parts.

### 2.4.1 Seen by Any Enemy

Figure 2.6 illustrates a Bayesian network that models the probability of being seen as a function by any of the enemies. The Bayesian network uses a Noisy OR model. In the Noisy OR model, we construct a conditional probability table for \( P_{\text{Seen}\mid\text{Seen}_1,\ldots,\text{Seen}_n}(\text{true}|s_1, \ldots, s_n) \) for each \( s_i \in \{\text{true, false}\} \).
Figure 2.6: Probability of $x_{rob}$ being seen by any enemy (modelled by Noisy_OR).

The algorithm for constructing this table assigns the probability of $n + 1$ rows in the table, computing the probability of every other row from these initial assignments. The rows that are assigned values correspond to situations where one and only one enemy, say enemy number $i$, sees the robot and all other fail to see the robot. This means that values are assigned for

$$P_{\text{Seen}|\text{Seen}_1,\ldots,\text{Seen}_n} (\text{true} | \text{false}, \ldots, \text{false}, \text{true}, \text{false}, \ldots, \text{false}) = p_i \quad (2.7)$$

where the true on the right side of the conditioning bar occurs at the position corresponding to random variable $\text{Seen}_i$. For example, Table 2.1 shows an example conditional probability table when there are 3 enemies in the environment. As illustrated in the figure, when there are $n$ enemies then we need to specify $n$ values.

<table>
<thead>
<tr>
<th>Seen$_1$</th>
<th>Seen$_2$</th>
<th>Seen$_3$</th>
<th>$p_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>$p_1$</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>$p_2$</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>$p_3$</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>$1 - [(1 - p_2) \times (1 - p_3)]$</td>
</tr>
</tbody>
</table>

Table 2.1: Conditional Probability Table for three enemies: T=true, F=false.

By convention, the probability of the Seen random variable being true given that all of its parent random variables are false is zero. Values at other positions are assigned as
follows:

\[ P_{\text{Seen}|\text{Seen}_1, \ldots, \text{Seen}_n}(\text{true} | s_1, \ldots, s_n) = 1 - \prod_{\{i: s_i = \text{true}\}} (1 - p_i) \]  

(2.8)

where the \( p_i \) is defined in Equation 2.7. The last row of Table 2.1 illustrates this situation.

Equations 2.7-2.8 define a noisy-OR in terms of tunable parameters \( p_i \). For simplicity, we let \( p_i = 1 \) for all \( i \). In the context of the cost associated with the “stealthily” adverb, this produces the effect of saying that there is an equal cost to the robot if one, two, or more enemies see the robot. Formally, when \( \forall i p_i = 1 \) implies that

\[
P_{\text{Seen}|\text{Seen}_1, \ldots, \text{Seen}_n}(\text{true}|s_1, \ldots, s_n) = \begin{cases} 
1 & \text{if any } s_i = \text{true} \\
0 & \text{otherwise}
\end{cases}
\]  

(2.9)

Equation 2.9 leads to a convenient form for computing the marginal probability \( P_{\text{Seen}} \).

\[
P_{\text{Seen}}(\text{true}) = \sum_{s_1, \ldots, s_n} P_{\text{Seen}|\text{Seen}_1, \ldots, \text{Seen}_n}(\text{true} | s_1, \ldots, s_n) \\
= \sum_{s_1, \ldots, s_n} P_{\text{Seen}|\text{Seen}_1, \ldots, \text{Seen}_n}(\text{true} | s_1, \ldots, s_n) \prod_{i=1}^{n} P_{\text{Seen}_i}(s_i) \\
= \left[ \sum_{s_1, \ldots, s_n} \prod_{i=1}^{n} P_{\text{Seen}_i}(s_i) \right] - \prod_{i=1}^{n} P_{\text{Seen}_i}(\text{false}),
\]  

(2.10)

where the first line is how you compute a marginal distribution from a joint distribution, the second line exploits the conditional independence assumptions of the noisy-OR Bayesian network, and the last line follows from the fact that the conditional is only one or zero.

2.4.2 Detection Likelihood of a robot by an enemy \( e_i \)

Consider three factors that affect whether the robot at location \( \mathbf{x}_{\text{rob}} \) can be seen by an enemy: the distance of \( \mathbf{x}_{\text{rob}} \) to \( \mathbf{x}_{e_i} \) encoded as detection range, the visibility of \( \mathbf{x}_{\text{rob}} \) from \( \mathbf{x}_{e_i} \) considering objects in the world, and visibility of \( \mathbf{x}_{\text{rob}} \) from \( \mathbf{x}_{e_i} \) considering the environment as terrain. The resulting effect after considering all the three factors for an individual enemy yields \textit{detection likelihood} of \( \mathbf{x}_{\text{rob}} \) by \( \mathbf{x}_{e_i} \). This detection likelihood serves as the \( P_{\text{Seen}_i} \), i.e. the
probability of $x_{rob}$ being detected or seen by $x_e$. The factors contributing to this detection likelihood are computed as follows:

**Detection range**

A distant enemy is less likely to have an harmful effect at $x_{rob}$ than an enemy that is standing next to it. This aspect is captured in detection range as $P_{d\text{Range}_i}(\text{Seen}_i; x_{rob}, x_e)$ which is calculated as a function of euclidean distance between the $x_{rob}$ and $x_e$, $\|x_{rob} - x_e\|$, in d-dimensional space, where $x_{rob} \neq x_e$.

\[
P_{d\text{Range}_i}(\text{Seen}_i; x_{rob}, x_e) = \begin{cases} 
1 & \text{if } 0 \leq \|x_{rob} - x_e\| \leq \delta \\
m(\|x_{rob} - x_e\|) + b & \text{if } \delta < \|x_{rob} - x_e\| \leq D \\
0 & \text{otherwise}
\end{cases} 
\]  

(2.11)

where $\delta$ is the minimum distance at or below which the detection of $x_{rob}$ by $x_e$ is maximum, hence 1, and $D$ is the maximum possible distance between two points across the configuration space. As $\|x_{rob} - x_e\|$ reaches $D$, the likelihood of detection approaches zero. Therefore, given this definite detection range, the probability of detection for any distance that lie between $\delta$ and $D$ follows the equation of line formed between points $(\delta, 1)$ and $(D, 0)$ with $O$ as origin, $m$ representing its slope, and $b$ being the y-intercept. Figure 2.7 illustrates the detection range curve.

**Visibility**

Two points in the world are said to be mutually visible to each other if a straight line segment can be drawn between them and none of the obstacle points lie on the line segment. Thus, if
there lies no obstacle in between $x_{rob}$ and the $x_e$ then $x_{rob}$ is visible to $x_e$. Visibility between the two points is hence expressed as $P_{visible_i}(\text{Seen}_i; x_{rob}, x_e, X_{obs}) \in \{0, 1\}$ given by:

$$P_{visible_i}(\text{Seen}_i; x_{rob}, x_e, X_{obs}) = \begin{cases} 1 & \text{if } x_{rob} \text{ is visible from } x_e \\ 0 & \text{otherwise} \end{cases}$$

(2.12)

**Viewshed**

The navigation environment for a robot may be a rough natural terrain instead of a flat surface. Because of the characteristics of terrain such as hill tops or valleys, to an enemy the robot’s position may be visible or it may be occluded by terrain. In a configuration space that has terrain characteristics, which points are visible from a given point is captured by a concept called *viewshed analysis* in literature [91, 153]. The viewshed is computed based on a digital representation of the terrain called a *Digital Elevation Model*. Viewshed, $VS_{x_p}$, of any point $x_p$ is the set of all the points on the terrain that are in the line-of-sight of $x_p$.

For a stealthy path we desire $x_{rob}$ to be **not** in the viewshed of the enemy point. We assume that the viewshed of each of the enemy points is available with us, and hence do not digress to show viewshed calculations [167]. Given the viewshed of $x_{e_i}$, $VS_{x_{e_i}}$, we represent the viewshed component of $x_{rob}$ w.r.t. $x_{e_i}$ as:
\[ P_{\text{viewshed}}(\text{Seen}_i; \text{VS}_{x_i}, \text{x}_{\text{rob}}) = \begin{cases} 1 & \text{if } \text{x}_{\text{rob}} \in \text{VS}_{x_i} \\ 0 & \text{otherwise} \end{cases} \] (2.13)

**Fusion of detection range, visibility, and viewshed**

Recall that we use Boolean network to model *Seen by enemy* \(i\) from component parts. Once we know the three components \(P_{\text{dRange}_i}, P_{\text{visible}_i}, \text{and } P_{\text{viewshed}_i}\) of \(\text{x}_{\text{rob}}\) being detected by \(x_e_i\), we combine them by using a NOISY-AND network. This allows us to compute \(P_{\text{Seen}_i}\) as the product of \(P_{\text{dRange}_i}, P_{\text{visible}_i}, \text{and } P_{\text{viewshed}_i}\). In other words, the robot is seen by \(x_e_i\) and has a probability greater than zero only when all the three components yields results greater than zero. Therefore,

\[ P_{\text{Seen}_i} = P_{\text{dRange}_i} \times P_{\text{visible}_i} \times P_{\text{viewshed}_i} \] (2.14)

### 2.4.3 Stealthily cost for a path

Substituting \(P_{\text{Seen}_i}\) of Equation 2.14 for each of the enemy in Equation 2.10 we obtain \(P_{\text{Seen}}(\text{true})\) that produces the stealthy cost \(C_{\text{stealth}}(\text{x}_{\text{rob}}, X_E)\). The stealthy cost can thus be computed for every possible point of robot location in the configuration space. The stealthy cost of a path (starting from the initial state to the goal state) can be determined as the sum of the costs of individual points constituting the path. In short, if the path has significant number of points that have a high probability of being seen by the enemy, then the robot should avoid such paths if one desires stealthiness.

Figure 2.8 illustrates a world with three enemies and its corresponding stealthily objective function. The figure assumes a flat earth (i.e., viewshed is all points). Referring to Figure 2.8b, if the robot has to travel from the top left corner to the bottom right corner of the configuration space, a path that goes between the obstacles and lower part of the space is more stealthy than a path that goes through the left side of the left obstacle in the space.
(a) World with three enemies. (b) $C_{\text{stealth}}(x_{\text{rob}}, X_E)$

Figure 2.8: Stealthily objective function.

### 2.5 Cost function for adverb ‘safe’

To go from an initial state to the goal state, we would want the robot to take a collision free path w.r.t obstacles. A path that goes very close to the obstacles is unsafe. With this intuition, we represent the cost of safety associated with any location of the robot $x_{\text{rob}}$ in the configuration space as a function of inverse distance between $x_{\text{rob}}$ and the nearest obstacle in that space. Let $C_{\text{safe}}(x_{\text{rob}}, X_{\text{obs}})$ denote the safety cost given the positions of the robot and the obstacles. Defining this as a function of distance to the nearest obstacle, we get:

$$
C_{\text{safe}}(x_{\text{rob}}, X_{\text{obs}}) = \begin{cases} 
1 & \text{if } \exists x_{\text{obs}} \text{ such that } \|x_{\text{rob}} - x_{\text{obs}}\| \leq \eta \\
1/(\min_{x_{\text{obs}} \in X_{\text{obs}}} \{\|x_{\text{rob}} - x_{\text{obs}}\|\}) & \text{if } \eta < \|x_{\text{rob}} - x_{\text{obs}}\| \leq D \\
0 & \text{otherwise}
\end{cases}
$$

(2.15)

where $\|x_{\text{rob}} - x_{\text{obs}}\|$ equals the euclidean distance between $x_{\text{rob}}$ and $x_{\text{obs}}$, $\eta$ is the minimum distance at or below which the safety cost for $x_{\text{rob}}$ is maximum, and finally $D$ is as defined in subsection 2.4.2. If a path has many points for which $C_{\text{safe}}(x_{\text{rob}}, x_{\text{obs}})$ is high, i.e many path points are in the close proximity of obstacles, then the path is considered as a unsafe path.
2.5.1 Safety cost for a path

Using Equation 2.15 the safety cost can be computed for every possible point of robot location in the configuration space. Figure 2.9 illustrates the safety cost for every point in the configuration space w.r.t the given obstacles. As in the case of stealthily cost, the safety cost of a particular robotic path is the accumulation of the safety cost of individual points that make the path. Referring to Figure 2.9, if a robot has to travel from the top left corner to the bottom right corner, then a path that goes through in between the two obstacles (the vertical middle region of configuration space) would be relatively unsafe compared to a path that goes either from the left side or the right side of the environment.

![Figure 2.9: Safety cost for every robot location.](image)

2.6 Cost function for adverb ‘quickly’

The adverb ‘quickly’ is associated with minimizing path length. Assuming that the robot traverses every point in the configuration space with an equal cost, the ‘quickly’ cost is the Euclidean distance between the start and the goal position such that the obstacles do not intercept the path. Consider a path that is formed by $k$ straight line segments, then the total path length is the summation of the euclidean distances of these individual line segments. Figure 2.10 shows two path options going from start location $A$ to goal location $B$. It can
been that the path distance of the path formed by points ALMNOB is less than the path distance formed by AXYB, hence the orange path is comparatively quicker than the blue path.

![Figure 2.10: Path ALMNOB is quicker than the path AXYB.](image)

2.7 Cosine similarity for path selection

The palette and sliders interfaces produce a human intent vector denoted by \( \vec{h} = [h_1, h_2, ..., h_K]^T \) where \( \sum_{i=1}^{K} h_i = 1 \). Each Pareto optimal path given by the MORRF* algorithm can also be represented as a vector, which we will soon discuss. Thus we have two vector representations, a human intent vector and a path vector. For a given human intent vector \( \vec{h} \) we compute the cosine similarity with each of the available path vector. The path that shows the highest similarity to the human intent is the path that most closely matches the user’s intent.

2.7.1 Path vector

Recall from Figure 2.2 that each point in the figure represents a Pareto optimal path, and the X-axis and the Y-axis show the cost for objective 1 and objective 2, respectively. The path at the top left corner has the minimum cost for objective 1 but is expensive in terms of objective 2. Similarly, a path that is in the middle of the curve provides a balance between both the objectives.
We convert the cost vector to a payoff vector so that we can compare the path with the positive human intent vector. Let there be $S$ total solutions/paths and let the costs associated with a path $s_j \in S$ and objective $k \in \{1, \ldots, K\}$ be denoted by $c_k(s_j)$. The cost vector for path $s_j$ is

$$\vec{c}(s_j) = [c_1(s_j), c_2(s_j), \cdots, c_K(s_j)]^T. \quad (2.16)$$

For each path vector, we convert the path costs to path payoffs by multiplying the path cost vector in Equation 2.16 by $-1$ yielding payoffs for each objective $p_k(s_j) = (-1)c_k(s_j)$ and a payoff vector of

$$\vec{p}(s_j) = (-1)\vec{c}(s_j). \quad (2.17)$$

Next, from among all the solutions $S$, we determine the minimum and the maximum payoff values for each of the objectives.

The payoff values in Equation 2.17 can be normalized to the bounds $[0.0, 1.0]$ using the formula

$$\hat{p}_k(s_j) = \frac{p_k(s_j) - \min_{s_\ell \in S}\{p_k(s_\ell)\}}{\max_{s_\ell \in S}\{p_k(s_\ell)\} - \min_{s_\ell \in S}\{p_k(s_\ell)\}}.$$

The corresponding normalized vector is given by

$$\vec{p}(s_j) = [\hat{p}_1(s_j), \hat{p}_2(s_j), \cdots, \hat{p}_K(s_j)]^T. \quad (2.18)$$

### 2.7.2 Cosine Similarity

The cosine similarity between a path vector, $\vec{p}(s_j)$, and the human intent vector is $\vec{h}$ is:

$$\text{CosineSimilarity}(\vec{h}, \vec{p}(s_j)) = \frac{\vec{h} \cdot \vec{p}(s_j)}{\|\vec{h}\|\|\vec{p}(s_j)\|} = \frac{\sum_{k=1}^{K} h_k p_k(s_j)}{\sqrt{\sum_{k=1}^{K} h_k^2} \sqrt{\sum_{k=1}^{K} p_k^2(s_j)}}. \quad (2.19)$$

For certain user command $\vec{h}$, if some path $\vec{p}(s_j)$ ends up with same orientation, then they have the cosine similarity of 1, and if they are at, say, 90° apart then they end up with
the cosine similarity of 0 indicating that they have nothing in common. Figure 2.11 shows comparison of path $p(s_1)$ and $p(s_2)$, w.r.t an example command $h_{ex}$ on AP. The orange points are example paths on the Pareto front. Each of the blue point is an example of user’s intent made through AP. It can be seen that since $\theta_2$ is smaller than $\theta_1$, CosineSimilarity($h_{ex}, p(s_2)$) is larger than that of CosineSimilarity($h_{ex}, p(s_1)$). Thus, path $p(s_2)$ will serve as a better path for $h_{ex}$ than the path $p(s_1)$. We thus formulate the best path for $h$ as $s^*$ as:

$$s^* = \arg \max_{s_j \in S} \text{CosineSimilarity}(h, p(s_j))$$  \hspace{1cm} (2.20)

For a given intent vector, this best path is rendered on the map for both the sliders and palette interfaces.

![Figure 2.11: Path Comparison w.r.t example human intent vector $h_{ex}$.](image)

2.8 Subjective Evaluation

We have provided three interface designs. Each design can be extended to more than three objectives. This can be seen that from the fact that more basic color dabs can be added to the palette interface, as well as more control trackbars can be added to the sliders. Of course, blending more than three colors in the palette can cause ambiguities, so blending would need to be supplemented with something like textures.
While issuing a command through the palette interface the user can easily view the blend by observing the pie graph associated with the paint dab. Similarly, in sliders interface she can view the preference of each of the adverbs in the textboxes below the sliders. By mapping the intent vector to a path vector via the palette or sliders interface, every user action updates the map immediately thereby giving the idea to the user of the immediate consequences of her action/command.

Each interface has its benefits and limitations. On the palette it is possible to have multiple user created paint dabs, each representing a particular command. Such a history is not available with sliders, because the moment one of the sliders is moved, the map gets updated to show the recent path according to the current user’s action. On the other hand, while moving one of the adverb sliders it is possible to discover different paths associated with the different adverb values till certain adverb value is reached. The palette is devoid of this.

The user can also provide waypoints on the map to obtain a certain path. The resulting score for this waypoints path is calculated based on the cost of each path point given in subsections 2.4.3 and 2.5.1

### 2.9 Summary and Future Work

We have presented here three interfaces for exploring tradeoffs between robot paths with three objectives. In the future we plan to conduct a user study to measure which of the interfaces the user like, which is more easier to operate, and would capture time statistics with respect to each interface.

**Acknowledgments**

This work is supported by the Army RCTA program. All results and conclusions are the responsibility of the authors and do not necessarily reflect the opinions of the funding source.
Chapter 3

Design and Evaluation of Adverb Palette:
A GUI for Selecting Tradeoffs in Multi-objective Optimization Problems

Published in: 2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)

Abstract

An important part of expressing human intent is identifying acceptable tradeoffs among competing performance objectives. We present and evaluate a set of graphical user interfaces (GUIs), that are designed to allow a human to express intent by expressing desirable tradeoffs. The GUIs require an algorithm that identifies the set of Pareto optimal solutions to the multi-objective decision problem, which means that all the solutions are equally good in the sense that there are no other solutions better for every objective. Given the Pareto set, the GUIs provide different ways for a human to express intent by exploring tradeoffs between objectives; once a tradeoff is selected, the solution is chosen. The GUI designs are applied to interactive human-robot path-selection for a robot in an urban environment, but they can be applied to other tradeoff problems. A user study evaluates GUI designs by requiring users to select a tradeoff that satisfies a specified mission intent. Results of the user study suggest that GUIs designed to support an artist’s palette-metaphor can be used to express intent without incurring unacceptable levels of human workload.

Keywords: human-robot interaction, multi-objective decision making, user interface design, robot path-planning
3.1 Introduction

An important part of specifying human intent is identifying acceptable tradeoffs among competing performance objectives. In a multi-objective problem with conflicting performance objectives, the set of Pareto optimal solutions is precisely the set of all possible solutions that satisfactorily tradeoff between the different objectives. Recall that a solution is Pareto optimal if, roughly speaking, there is no other solution that is better for every objective. Since each Pareto solution represents a potentially acceptable tradeoff, specifying intent is roughly equivalent to selecting a desirable Pareto optimal solution.

When a human selects a single solution, making tradeoffs between the objectives creates the need for a robust and intuitive interface that allows a user to select a satisfactory tradeoff without imposing high workload. In a supervisory control problem, given a scenario of (a) what needs to be done — the strategic intent to be accomplished — and (b) a set of ways that a task can be performed, the human (c) determines “how” the task will be done using a GUI.

This paper presents three possible GUI designs that provide a medium to explicitly express human intent for how a task will be done given a set of objectives expressed as adverbs; the adverbs convey what could be important in how the task can be done. The GUI designs allow the human to evaluate different solutions and select one that best matches strategic objectives of the problem. The work in this paper builds from prior work on using Pareto Analysis for exploring tradeoffs [148], which defines the problem as follows:

The solution points [in] the Pareto [Set] are mathematically indifferent with respect to each other, and thus the selection phase . . . is subjectively driven by the human decision maker. This process involves exploration of the [Set], and eventually, the challenge in selecting a solution is to account for gains and losses while adhering to personal preferences.
We discuss different GUI designs and a user study that compares these designs. Generally speaking, the GUI designs are based on the metaphor of an artist’s palette, where an artist mixes different colors to produce a desired hue. The adverbs correspond to different objectives to be accomplished; each adverb is a different color, and the mixes of colors represent different tradeoffs between objectives. For example, in a robot path-planning application, consider a command for a robot to “Go quickly and safely from point A to point B.” The adverbs associated with these objectives are “quickly” and “safely.” Like an artist, the operator can mix the adverbs on the interface such that a single path is selected that is both quick and safe from a set of available paths. Results of a user study demonstrate that this metaphor can be very useful for helping a human find acceptable tradeoffs between competing objectives.

Figure 3.1 shows an example Pareto-optimal set for a two-objective problem. Each point in the curve represents a solution and its associated pay-offs for objective 1 and objective 2. The upper left dot represents a solution that has maximum pay-off for objective 2 at the expense of objective 1, and lower right dot represents a solution that has maximum pay-off for objective 1. Dots between the extremes represent tradeoffs between the objectives.

Figure 3.1: Path Planning with MORRF* for two objectives.

The paper presents GUI designs, palette, sliders, and prism, that are based on the color-blending metaphor and that can be generalized to many problems that require tradeoffs. A fourth GUI design, waypoints, is specific to path-planning. The three color-blending designs are described later in the paper.
We apply the GUIs to a robot path-planning problem where multiple performance objectives need to be satisfied. Although there exist many algorithms for multi-objective optimization (see, for example, [4, 25, 27, 87, 97, 172]), we use the MORRF* algorithm [172] because it efficiently generates Pareto optimal solutions specifically for path-planning.

Figure 3.2 shows an example of one of the designs in AP (the palette design) applied to robot path-planning. The left panel of the interface is problem-specific and shows the available alternatives/solutions for the problem. The right panel allows the human to express intent; it is the area where the human can explore many tradeoffs. Based on the human-actions on the right, the left panel updates to show the result/solution. For example, the left side of Figure 3.2 depicts a map that shows in gray Pareto optimal paths that a robot can take, and the right side of the interface provides an area that can be used by the human to find tradeoffs among the paths. Based on the command issued on right side panel, one of the gray paths gets highlighted on the left panel.

3.2 Related Work

Making tradeoffs in decision-making is also known as multiple criterion decision-making [166] and multiple attribute decision-making [74]. The goal is to select a decision over available alternatives in a way that balances or trades off between the objectives, i.e., to choose from among a finite set of discrete alternatives [56, 148]. This paper uses three objectives: minimizing distance from the robot’s start location to a goal location, avoiding exposure of the robot to one or more enemies, and avoiding collisions with obstacles.

The literature on designing user interfaces for human-machine interaction is vast (see [2, 10, 31, 67, 83] for some examples). There are indeed many graphical interfaces for managing robots in HRI [63, 102, 135, 155]. The interface in the paper differs from these other interfaces in that it is intended to enable interactive decision-making in selecting a solution that satisfies a decision tradeoff. The interfaces in this paper is more similar to decision-support systems than to traditional supervisory control interfaces.
The GUIs presented in this paper are perhaps best classified as ecological interfaces [15, 165] because they seek to enable decision-making easier and more intuitive for a human using a natural metaphor, in this case, a color palette. The metaphor is designed to help a human create a mental model of the tradeoffs and how changing from one solution to another alters how tradeoffs are balanced [156]. The three objectives that we consider are represented by the colors red, green and blue respectively, and the problem domain is supervisory control of a remote robot. Designing interfaces for supervisory control is an important part human-robot interaction (HRI), and designing intuitive and efficient interfaces has been a challenging issue in HRI [46, 63].

3.3 Adverb Palette

The Adverb Palette (AP) designs are mouse-based interactive GUIs designed to help a human express intent over Pareto optimal tradeoffs. AP interfaces help a user to trade off among objectives in a way a painter selects colors from a given set of colors. A blend/mixture of colors corresponds to a single tradeoff from the available Pareto optimal tradeoffs. The AP designs are general enough to work for many problems with tradeoffs, and the designs and evaluation are applied to robot path-planning.

The path planning problem is for a robot to go from a start location $x_{init}$ to a goal location $x_{goal}$ within a configuration space (in this paper, a 2-D world). Each GUI has two parts: the map in the left panel, which is a task-specific interface that aids visualization of paths, and the command interface (CI) in the right panel, which is a general-purpose interface that a user can use to balance different adverbs. The command area allows a user to express intent by balancing tradeoffs, and the map gets updated to show task-specific details by highlighting the corresponding path.

Consider three adverbs, Quickly, Stealthily, and Safely, symbolized by colors red, green and blue, respectively.

- Quickly: prefer shorter paths.
• Stealthily: avoid being viewed by enemies.
• Safely: stay away from obstacles.

Given the Pareto optimal solution set, the goal is to enable a user to find a tradeoff that best expresses his or her intent. Expressing intent has two subproblems to be solved:

1. express a desired tradeoff, and
2. map the tradeoff to a Pareto optimal solution.

We focus on the aspect of human intent that requires satisfactory tradeoffs between competing objectives.

All interfaces include a task-specific map that shows all the routes (paths) in gray. Before tradeoffs are explored, a highlighted path is displayed that gives equal preference to all the adverbs. This paper refines initial palette, sliders, and waypoints from prior work [143] and introduces the prism design. This paper also discusses the mapping from intent to solution, and evaluates the designs through a user study. We begin by discussing how to condition the objectives so that they can be expressed using the color-blending metaphor.

3.3.1 Conditioning Objectives

Objectives may be expressed in incommensurable units, and this causes problems for using the palette metaphor. We perform an affine transformation and normalize objectives so that the multiple objective criterion can be reasonably compared.

Let $S$ denote the set of Pareto optimal solutions, and let the costs associated with a particular solution $s_j \in S$ and objective $k \in \{1, \ldots, K\}$ be denoted by $c_k(s_j)$. (Please check our prior work [143] for cost functions computations applied to robotic path planning).

For each solution, we convert the solution costs to solution pay-offs by multiplying by $-1$ yielding pay-offs for each objective $p_k(s_j) = (-1)c_k(s_j)$ and then normalize them to $[0,1]$.

$$\hat{p}_k(s_j) = \frac{p_k(s_j) - \min_{s_l \in S}\{p_k(s_l)\}}{\max_{s_l \in S}\{p_k(s_l)\} - \min_{s_l \in S}\{p_k(s_l)\}}.$$
The corresponding normalized vector \( p(s_j) \in [0, 1]^K \) for a solution is thus given by

\[
p(s_j) = [\hat{p}_1(s_j), \hat{p}_2(s_j), \ldots, \hat{p}_K(s_j)]^T.
\] (3.1)

### 3.3.2 Palette

The *palette* displays three initial circles called the “primary dabs,” one for each adverb (objective). The user expresses intent by clicking on one of the primary dabs (e.g., take the shortest path) or creates tradeoffs by dragging and dropping adverbs color dabs into the white area of the *CI* to create smaller circles called “paint dab” that blend colors. By creating different paint dabs and then exploring how each dab corresponds to a different path, a user can visualize the consequences of different commands. Line segments connect either the primary dabs and paint dabs or paint dabs to other paint dabs, producing a tree structure that allows the human to see the proportions of each objective.

Figure 3.2 shows an example command where the user desires a path that is quick and safe but does not care about being seen by enemies, which is represented numerically as “50% quickly, 0% stealthily, 50% safely.” The pie graph on the lower left area in the *CI* shows the proportion of each objective in a particular paint dab. Blending in multiple adverbs (colors) is thus equivalent to making tradeoffs with multiple objectives. The default magenta paint dab in Figure 3.2 is an equal mixture “33.33% quickly, 33.33% stealthily, 33.33% safely”.

Let \( dab_d \) represent any paint dab on *CI*. Let \( n_i \) be the number of times the user has dragged adverb \( i \) on \( dab_d \), where \( 0 > i \leq K \). The total number of drags a user makes for \( dab_d \) is \( n = \sum_{i=1}^{K} n_i \). The *user’s intent* from the palette is the vector \( \vec{h}^{\text{pal}} \)

\[
\vec{h}^{\text{pal}} = \left[ \frac{n_1}{n}, \frac{n_2}{n}, \ldots, \frac{n_K}{n} \right]^T.
\] (3.2)
Figure 3.2: Quick and safe command in lowermost dab: map shows the corresponding highlighted path.

Figure 3.3: Adverb Palette: Slider interface.
3.3.3 Sliders

Figure 3.3 shows the sliders interface. The user adjusts the trackbars to get to a desired mixture, and the corresponding solution/path from the left panel is selected. The three sliders represent the three adverbs. The user can issue any of the three primary commands to the robot (e.g., take the shortest path) by sliding the corresponding slider (e.g., red) to the maximum units. The sum of the units on the sliders does not exceed 100 units, so if the red, green, or blue sliders are at say 33, 33, 34 units respectively, then moving the blue slider to 60 units will cause a change to the slider units to 20, 20, 60 units respectively. Unlike the palette, the user can explore different tradeoffs while moving a slider, and settle down to a certain position if she desires it. As the user moves one slider the other two sliders move automatically to maintain the sum to 100 units, and the corresponding solution/path gets shown on the map.

Let $s_i$ is the score specified by slider $i$. The maximum unit on a slider corresponds to the cheapest solution for that objective and the minimum unit corresponds to the most expensive solution. The human intent can be represented as a vector $\vec{h}_{\text{sl}}$ as:

$$\vec{h}_{\text{sl}} = \left[ \frac{s_1}{100}, \frac{s_2}{100}, \ldots, \frac{s_K}{100} \right]^T.$$  \hspace{1cm} (3.3)

3.3.4 Prism

Figure 3.4 shows the prism interface. Here the user can move the mouse over different areas of prism and discover its associated paths. Each point on the prism is a color corresponding to a certain proportion of adverbial objectives, expressed using a barycentric coordinate system.

As a review of barycentric coordinates, consider a triangle defined by three vertices, $R$, $G$, and $B$. Any point $P$ inside or on the triangle can be written as a unique convex combination of the three vertices. Figure 3.5 illustrates the concept. The dots on the edges and those inside the triangle are example points that $P$ may take. For a point $P$ there is a unique sequence of three numbers such that the sum of these three numbers is 1. The three numbers,
Figure 3.4: Adverb Palette: Prism interface.

Figure 3.5: Barycentric coordinates on an equilateral triangle.
denoted by $\alpha$, $\beta$, and $\gamma$ indicate the \textit{barycentric} coordinates of point $P$ with respect to the triangle. In the prism interface, $\alpha$, $\beta$, and $\gamma$ represent the proportion of \textit{quickly}, \textit{stealthily}, and \textit{safely}, respectively. Intent for the prism is represented as

$$\tilde{h}^{pri} = [\alpha, \beta, \gamma]^T.$$  \hfill (3.4)

The prism interface only works with three coordinates, and is therefore limited to three objectives.

\subsection*{3.3.5 Waypoints}

The \textit{waypoints} interface is path-planning specific while the other three interfaces are generic AP designs. The \textit{waypoints} interface assists a user to construct her own path on the map by allowing her to provide location guidelines that the robot should visit while taking a path. Unlike the other three interfaces, the user here does not make a tradeoff among the available paths from the algorithm but instead makes her own path on the map. She can however compare her path with the best or worst with respect to an adverb based on the Pareto optimal paths’ best and worst for that particular adverb. Figure 3.6 shows a path constructed using the waypoints interface. The graphs on the right panel show how well the user-created path compares to the best and worst objective scores from the solutions in the Pareto set.

\subsection*{3.4 Mapping between strategic intent and pay-offs}

The \textit{palette} and \textit{sliders} interfaces produce a human intent vector denoted by $\tilde{h}^{pal} = [h_1, h_2, ..., h_K]^T$ and $\tilde{h}^{sli} = [s_1, s_2, ..., s_K]^T$, respectively. The \textit{prism} interface produces the intent $\tilde{h}^{pri} = [\alpha, \beta, \gamma]^T$. We can interpret this tradeoff in a vector space by associating intent with an orientation/direction with respect to some reference frame. More precisely, we operationally define intent as a vector $\tilde{h}$ that represents an ideal tradeoff, that is the balance between competing objectives that the human wants to achieve.
The multi-objective optimization algorithm gives a pool of possible solutions. Since each Pareto optimal solution has pay-off values associated with it, it can also be represented as a payoff vector using Eq. (3.1). Thus we have two vector representations, a human intent vector and the Pareto-optimal solution expressed as a payoff vector.

Given the human intent $\vec{h}$ and the payoff vector for every Pareto optimal solution, we need a mechanism to match the intent to one of the solutions. Intuitively for $\vec{h}$, the solution that has each of the individual pay-off values most closely matching to the corresponding individual intent values would be the one that would be finally selected. The mapping between the tradeoff point and the solutions would then be defined as closeness of the intent to the Pareto optimal solution.

We subjectively evaluated four different mapping strategies. Two of these strategies, WPM and TOPSIS are detailed in [9]. The others that we considered are the popular methods euclidean distance and cosine similarity for finding similar or matching entities. TOPSIS, WPM, and cosine similarity all gave the same results for mapping. Cosine similarity is the most simple, and subjectively produced better results than euclidean distance.
The cosine similarity between a path vector, $p(s_j)$ and the human intent vector is $h$ is $\frac{h \cdot p(s_j)}{\|h\|\|p(s_j)\|}$. For the given $h$, if the solution $p(s_j)$ ends up with the same orientation, then they have the cosine similarity of 1, and if they are at, say, $90^\circ$ apart then they end up with the cosine similarity of 0 indicating that they have nothing in common.

Consider Figure 3.7. The triangle represents the set of possible tradeoffs. Each of the dots on the triangle represents a human intent, and each of the dots to the upper right of the triangle represents a Pareto optimal solution. The dark vectors represent the objective forming the space, stealth, safety, and quickness. The other vectors represent intent and solution vectors. $\theta_i$ represents the cosine similarities between the intent vector $h_{ex}$ and the solution vector $p(s_i)$.

3.5 Objective Functions

Solving multi-objective optimization problems require computing costs or pay-offs for the involved objectives. Since here we apply AP to robot path-planning, we briefly describe in the following paragraphs the costs computed for a particular robotic-path. Recall that for this application we consider three costs; quickly, stealthily and safely.

A safe path is a collision free path. Hence, the safety cost of a robot location in a configuration space is encoded as a function of inverse distance between the robot position and the nearest obstacle in that space. The cost can be computed for every possible point...
in the configuration space. Therefore, the safety cost of a particular robotic path is the accumulation of the safety cost of individual points that make the path; see Figure 3.8a for an example safety cost for the polygonal obstacles – darker colors are safer.

![Safety cost for every robot location.](image)

(a) Safety cost for every robot location.

![Stealth function with three enemies.](image)

(b) Stealth function with three enemies.

(c) Path **ALMNOB** is quicker than the path **AXYB**.

Figure 3.8: Objective functions.

A stealthy path is less likely to be detected or seen by enemies in the world. The stealth cost function is expressed in terms of the probability of the path being seen by the enemy, and is computed as a function of two factors: the distance of the robot from each of the enemies and the visibility of the robot from the enemies. The resulting effect yields detection likelihood of the robot from the enemies. The stealth cost can thus be determined for every possible point of the robot location in the configuration space. Therefore, the stealthy cost of a path (starting from the initial state to the goal state) can be determined as the sum of the stealthy costs of individual points constituting the path. Figure 3.8b illustrates a world with three enemies and its corresponding stealthily objective function. If the robot has to travel from the top left corner to the bottom right corner of the configuration space, a path that goes between the obstacles and lower part of the space is more stealthy than a path that goes through the left side of the left obstacle in the space.

A quick path minimizes path length (assuming constant robot speed). The ‘quickly’ cost is the euclidean distance between the start and the goal position such that the obstacles
do not intercept the path. Figure 3.8c shows two path options going from start location A to goal location B. The path distance of the path formed by points ALMNOB is less than the path distance for AXYB, hence the orange path is comparatively quicker than the blue path.

### 3.6 User Study

Following a pilot study among university students to refine the AP designs, we conducted an IRB-approved user study to evaluate the four GUI principal designs. The aim was to discover how successful would be the user in finding tradeoffs among the given solutions using the GUI designs.

Participants were invited for a one-hour study through an advertisement posted in various departments of the university. 24 people participated, 17 males and 7 females, with a mean age of 24.8. All but one participant was a university student. The participants belonged to diverse majors including food, film, elementary education, nursing, and computer science. Each participant received $15 as compensation. All participants completed all the tasks assigned for the study.

### 3.6.1 User Study Procedure

After completing the informed consent process, each participant completed a short demographic questionnaire that included questions on familiarity with using a mouse, exposure to video games, age, and education. The participant then watched a 19 minute video tutorial that described the four GUI designs and showed how to use them in response to a particular scenario or task. Participants were issued a command in written English for the robot to perform such as:

> It is critical to the commander that the robot takes a quick and safe path.

Enter the highlighted path number as per this command.
Following the training, participants executed four sets of practice tasks, one for each interface. The practice tasks and the world on which practice tasks were carried out were identical for each interface. Each practice task had an ideal path indicated by a path number, and the task was designed in a way that the user could easily figure out this path on the map in response to the command. The user was allowed three practice attempts to choose the correct path.

(a) Easy world/command: “Issue a command that makes the robot reach its goal as quickly as possible. Enter the highlighted path number below.”

(b) Hard world/command: “It is critical to the commander that the robot takes a path that hides the robot from enemy and that does not come close to buildings. The commander doesn’t care if the distance from the start to the goal is big. Enter the path number below.”

Figure 3.9: Difficulty level: (a) Easy and (b) Hard.

The experiment was a two-factorial experiment with factors being interface type (palette, sliders, prism, waypoints) and difficulty level (easy, hard). The difficulty level is a function of two components: The first component is fairly general, namely, choosing a tradeoff is harder if it has to deal with more objectives/adverbs. The second component is task specific, namely the number of obstacles in the worlds. Hard tasks demand tradeoffs that involve multiple objectives (more than one) and have more obstacles, and easy tasks demand tradeoffs on at the most only two objectives and have fewer obstacles. Four sets of easy tasks and four sets of hard tasks were designed (two tasks in each set), allowing unique pairings of interfaces and worlds. Figure 3.9(a) and (b) show an example of an easy and hard task, respectively.

**Interface Appeal.** After completing all tasks, participants ranked the interfaces that reflected their preference for three categories, ranked from most preferred to least preferred. The categories are how appealing the interface is, how easy the interface is to use, and how time-consuming the interface is to use.

**Objective Metrics.** In addition to subjective workload and user preferences, we evaluated the AP designs using three objective metrics. In each task, a command was given to the participant via the user interface; the command was constructed to describe an ideal path. The first metric evaluates how well participants could express tradeoffs, and the other metrics included both expressing tradeoffs and selecting paths.

- **Accuracy** quantifies the degree to which the tradeoff/solution selected by the user matches the intended tradeoff. Accuracy is measured as the cosine similarity between the intended tradeoff vector and the user-selected tradeoff vector.

- **Interface time** is the time spent performing all tasks required in a particular interface and world.

- **Answer time** is the time spent executing the tasks. For each individual task, answer time is captured from the first click or drag made on the GUI interface to the last click or drag made on the interface. It excludes the time spent to type in the answer for a task. The answer times for each individual tasks are then added to get the total answer times for all tasks on an interface for a particular world.

Participants were not given feedback on whether they executed the tasks correctly or not, and the order of interface/difficulty level was counterbalanced.

### 3.6.2 Hypothesis testing

We evaluated the following hypothesis:

1. **Hypothesis 1:** Each AP interface design can be used to successfully complete all assigned tasks.
2. **Hypothesis 2**: Hard tasks have longer completion times and higher subjective workload than easy tasks.

3. **Hypothesis 3**: The interfaces *palette* and *prism* would produce the lowest workload and shortest completion times.

### 3.7 Results

Hypotheses were tested using SAS with *Restricted Maximum Likelihood Estimation* for a mixed-design ANOVA using Tukey-Kramer adjustment on subjects.

#### 3.7.1 F statistics

Table 3.1 shows the effect of interface, difficulty level, and the combined effect of interface and difficulty on different measures/metrics of user’s interaction. The asterisk * denotes significant differences. There were significant differences in interface design and difficulty, but there were few differences for interface plus difficulty level.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Interface (I/F)</th>
<th>Difficulty (DL)</th>
<th>I/F × DL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F Value</td>
<td>Pr &gt; F</td>
<td>F Value</td>
</tr>
<tr>
<td>Accuracy</td>
<td>4.28</td>
<td>0.006*</td>
<td>0.02</td>
</tr>
<tr>
<td>Answer Time</td>
<td>44.24</td>
<td>&lt; 0.001*</td>
<td>2.32</td>
</tr>
<tr>
<td>Interface Time</td>
<td>98.20</td>
<td>&lt; 0.001*</td>
<td>6.39</td>
</tr>
<tr>
<td>Mental</td>
<td>27.02</td>
<td>&lt; 0.001*</td>
<td>2.33</td>
</tr>
<tr>
<td>Physical</td>
<td>8.77</td>
<td>&lt; 0.001*</td>
<td>0.07</td>
</tr>
<tr>
<td>Temporal</td>
<td>12.08</td>
<td>&lt; 0.001*</td>
<td>1.25</td>
</tr>
<tr>
<td>Performance</td>
<td>17.33</td>
<td>&lt; 0.001*</td>
<td>11.47</td>
</tr>
<tr>
<td>Effort</td>
<td>26.05</td>
<td>&lt; 0.001*</td>
<td>5.66</td>
</tr>
<tr>
<td>Frustation</td>
<td>13.39</td>
<td>&lt; 0.001*</td>
<td>0.87</td>
</tr>
</tbody>
</table>

Table 3.1: Effect of interface, difficulty level, and interaction of interface and difficulty level on various measures.
Accuracy

The waypoints interface was statistically less accurate than all other interfaces, and the other three interfaces had no statistically significant differences; see Figure 3.10. Difficulty level had no effect on accuracy.

Answer Time

Table 3.2 shows p values for pairwise differences between interfaces for answer time (negative t value indicates that the answer time on prism is higher than on palette). Palette and sliders are similar, prism is statistically slower than palette and sliders, and waypoints is statistically slower than them all. Figure 3.11 illustrates the differences for palette, prism, and sliders. Difficulty level has no impact on answer time.
### Interface Time

<table>
<thead>
<tr>
<th>Interface</th>
<th>Interface</th>
<th>t value</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Palette</td>
<td>Prism</td>
<td>-4.31</td>
<td>&lt; 0.001*</td>
</tr>
<tr>
<td>Palette</td>
<td>Sliders</td>
<td>-1.43</td>
<td>0.482</td>
</tr>
<tr>
<td>Palette</td>
<td>Waypoints</td>
<td>-10.65</td>
<td>&lt; 0.001*</td>
</tr>
<tr>
<td>Prism</td>
<td>Sliders</td>
<td>2.88</td>
<td>0.023*</td>
</tr>
<tr>
<td>Prism</td>
<td>Waypoints</td>
<td>-6.34</td>
<td>&lt; 0.001*</td>
</tr>
<tr>
<td>Sliders</td>
<td>Waypoints</td>
<td>-9.22</td>
<td>&lt; 0.001*</td>
</tr>
</tbody>
</table>

Table 3.2: Pairwise differences in Answer Time.

Except for waypoints, which was significantly slower, interface type did not affect interface time. Palette, sliders and prism were not significantly different (see Table 3.3).

Task difficulty did have an effect on interface time. Interface Time for hard tasks was higher than for easy tasks \((t = -2.55, p = 0.0117)\). Table 3.4 shows the statistics as a function of individual interfaces.

<table>
<thead>
<tr>
<th>I/F</th>
<th>Task</th>
<th>Interface time</th>
<th>t value</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Palette</td>
<td>E vs H</td>
<td>-3.03</td>
<td>0.006*</td>
<td></td>
</tr>
<tr>
<td>Prism</td>
<td>E vs H</td>
<td>-2.92</td>
<td>0.008*</td>
<td></td>
</tr>
<tr>
<td>Sliders</td>
<td>E vs H</td>
<td>-2.8</td>
<td>0.01*</td>
<td></td>
</tr>
<tr>
<td>Waypoints</td>
<td>0.689</td>
<td>-0.53</td>
<td>0.603</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.4: Interface Time on different interfaces. t value computed as E minus H.
Subjective Workload

We used a 20-point scale/score for each of the NASA-TLX question. Except for the performance scale, a value of one corresponds to least workload factor and the score of 20 suggests highest workload. For the performance NASA TLX factor, the highest value is the best. Table 3.5 shows the values obtained from the mixed-design ANOVA on the participant’s NASA TLX scores. It is seen that the interface type affects workload. The palette and sliders have similar workload profiles. Furthermore, waypoints deviates from every other interface, and prism deviates from palette and sliders. Summarizing, the workload increased roughly in the following order palette ≺ sliders ≺ prism ≺ waypoints.

<table>
<thead>
<tr>
<th>I/F</th>
<th>I/F</th>
<th>Mental</th>
<th>Physical</th>
<th>Temporal</th>
<th>Performance</th>
<th>Effort</th>
<th>Frustration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Palette</td>
<td>Prism</td>
<td>0.022*</td>
<td>0.02*</td>
<td>0.089</td>
<td>0.009*</td>
<td>0.009*</td>
<td>&lt; 0.001*</td>
</tr>
<tr>
<td>Palette</td>
<td>Sliders</td>
<td>1</td>
<td>1</td>
<td>0.93</td>
<td>1</td>
<td>0.97</td>
<td>0.98</td>
</tr>
<tr>
<td>Palette</td>
<td>Waypoints</td>
<td>&lt; 0.001*</td>
<td>&lt; 0.001*</td>
<td>&lt; 0.001*</td>
<td>&lt; 0.001*</td>
<td>&lt; 0.001*</td>
<td>&lt; 0.001*</td>
</tr>
<tr>
<td>Prism</td>
<td>Sliders</td>
<td>0.015*</td>
<td>0.02*</td>
<td>0.018*</td>
<td>0.021*</td>
<td>0.034*</td>
<td>0.002*</td>
</tr>
<tr>
<td>Prism</td>
<td>Waypoints</td>
<td>&lt; 0.001*</td>
<td>0.596</td>
<td>0.073</td>
<td>0.011*</td>
<td>&lt; 0.001*</td>
<td>0.648</td>
</tr>
<tr>
<td>Sliders</td>
<td>Waypoints</td>
<td>&lt; 0.001*</td>
<td>&lt; 0.001*</td>
<td>&lt; 0.001*</td>
<td>&lt; 0.001*</td>
<td>&lt; 0.001*</td>
<td>&lt; 0.001*</td>
</tr>
</tbody>
</table>

Table 3.5: Pairwise differences for subjective workload computed using NASA TLX. I/F = Interface.

Difficulty level impacted two NASA TLX scores. Performance is significantly worse on hard tasks \((t = 3.41, p < 0.001)\). Effort was also significantly worse on hard tasks \((t = 2.40, p = 0.018)\).

3.7.2 User Preference

<table>
<thead>
<tr>
<th>I/F</th>
<th>I/F</th>
<th>Appeal t value</th>
<th>p</th>
<th>Ease of Use t value</th>
<th>p</th>
<th>Time Cons t value</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Palette</td>
<td>Prism</td>
<td>-6.65</td>
<td>&lt; 0.001*</td>
<td>-11.4</td>
<td>&lt; 0.001*</td>
<td>-13.79</td>
<td>&lt; 0.001*</td>
</tr>
<tr>
<td>Palette</td>
<td>Sliders</td>
<td>-2.52</td>
<td>0.06</td>
<td>-5.87</td>
<td>&lt; 0.001*</td>
<td>-9.19</td>
<td>&lt; 0.001*</td>
</tr>
<tr>
<td>Palette</td>
<td>Waypoints</td>
<td>-10.08</td>
<td>&lt; 0.001*</td>
<td>-20.23</td>
<td>&lt; 0.001*</td>
<td>-25.66</td>
<td>&lt; 0.001*</td>
</tr>
<tr>
<td>Prism</td>
<td>Sliders</td>
<td>4.12</td>
<td>&lt; 0.001*</td>
<td>5.55</td>
<td>&lt; 0.001*</td>
<td>4.6</td>
<td>&lt; 0.001*</td>
</tr>
<tr>
<td>Prism</td>
<td>Waypoints</td>
<td>-3.44</td>
<td>0.004*</td>
<td>-8.81</td>
<td>&lt; 0.001*</td>
<td>-11.87</td>
<td>&lt; 0.001*</td>
</tr>
<tr>
<td>Sliders</td>
<td>Waypoints</td>
<td>-7.56</td>
<td>&lt; 0.001*</td>
<td>-14.36</td>
<td>&lt; 0.001*</td>
<td>-16.47</td>
<td>&lt; 0.001*</td>
</tr>
</tbody>
</table>

Table 3.6: Pairwise differences between interfaces for appeal variables.
Participants ranked the four interfaces with respect to appeal, ease of use, and time consuming on an integer scale of 1 to 4, where 1 is best 4 is worst. Results showed that all interfaces show significant differences.

Table 3.6 suggests that the most appealing interface to the users was the palette and the least appealing was the waypoints. The suggested order of appeal is palette \(\succ\) sliders \(\succ\) prism \(\succ\) waypoints.

Each interface differed significantly from the others in terms of ease of use, with palette being the easiest to use and waypoints being the hardest. Furthermore, sliders was easier than prism, with both lying between the two extremes of palette and waypoints.

The time-consuming variable for interfaces was very similar to ease of use. The p-values suggest that each of the interfaces differed significantly from each other with palette being the least time-consuming and waypoints being the most time-consuming. Furthermore, sliders took less time than prism, with both lying between the two extremes of palette and waypoints.

Thus, all the interfaces were different from each other for the appeal variables, where in each case palette was preferred to the interfaces with sliders second.

### 3.7.3 Discussion

Results indicate that waypoints interface is significantly worse than the other three. This is not surprising since the waypoints interface requires participants to both plan a path and explore tradeoffs. It takes more time, induces higher subjective workload, and produces paths that differ significantly from the path intended in the command. We ignore this interface and discuss the others.

**Hypothesis 1.** Results of accuracy showed that there were no significant differences between palette, sliders, and prism, meaning that the users were able to find an acceptable tradeoff using each interface. Each interface produced at least 90% accuracy, and difficulty
level had no impact on accuracy. We fail to reject hypothesis 1, which suggests that each user can use each of the interfaces successfully.

**Hypothesis 2.** Difficult tasks took more time and subjective workload than easy tasks. We therefore find support for hypothesis 2.

**Hypothesis 3.** Both palette and sliders produced similar interface times, but prism required more time to answer the tasks, thereby making it significantly less effective. Similarly, both palette and sliders produced similar subjective workload, and prism had significantly higher subjective workload. The results of users’ preferences demonstrated that users preferred palette to find tradeoffs among the Pareto optimal solutions. The interface sliders followed suit, and then prism. Hence, we reject hypothesis 3. Instead palette and sliders were similarly usable, and prism was significantly more challenging.

We hypothesize that participants found it hard to comprehend the mixing of adverbs through prism. Note that prism used the same optimal solutions that the palette and the sliders used, but participants found it hard to know where to click on the prism to get the solution for a task.

### 3.8 Summary and Future Work

We have presented four interactive GUI designs for selecting tradeoffs from among Pareto optimal solutions to a multi-objective optimization problem. The AP interface designs provide a novel way of blending objectives and enables users to find and express tradeoffs. The user study indicated that the palette and sliders designs were usable and relatively easy to use because of its metaphor of mixing colors in an artist’s way. A rough aggregation of all measures suggests a slight superiority for the palette over sliders, and both were superior to the prism design, presumably because participants had a hard time understanding this interface.

The results from the waypoints interface design suggest that providing an interface that explicitly enables a participant to express tradeoffs is useful. Since expressing tradeoffs
are an important part of expressing human intent, an interface that helps a user to understand and express tradeoffs may be useful for many problems.

Some of the participants explicitly disliked *prism*. Future work should be performed on *prism* such as naming or scaling the boundaries of the interface so that people can more easily understand how *prism* works.

Since the GUI designs presented here only consider three adverbs, future work should make GUI designs generic so that they can be applied to a variable number of objectives. It is possible that mixing more colors will make the interface less intuitive, so future work should explore the limitations on the interface as a function of more colors. Also, the current application was robotic-path planning. Future work should explore whether the GUI for other applications that require tradeoffs, such as a social robot that must find a path so that it balances proxemic concerns with energy or safety concerns. Finally, future work should explore palette-based designs that do not rely exclusively on color, adding redundant cues to aid easier human perception.

### 3.9 Acknowledgments

This work was funded by the United States Army Research Laboratory through contract W911NF-14-1-0633. All results, findings, and conclusions are the responsibility of the authors and do not necessarily reflect the opinions of the funding organization.
Chapter 4

When does a Human Replan? Exploring Intent-Based Replanning in Multi-Objective Path Planning

Abstract

In goal-based tasks such as navigating a robot from location A to location B in a dynamic environment, human intent can mean to choose a specific trade-off between multiple competing objectives. For example, intent can mean to find a path that balances between “Go quickly” and “Go stealthily”. Given human expectations about how a path balances such tradeoffs, the path should match the human’s intent throughout the entire execution of the path even if the environment changes. If the path drifts from the human’s intent because the environment changes, then a new robotic-path needs to be planned — referred to as path-replanning.

We discuss here three system-initiated triggers (prompts) for path-replanning. The objective is to create an interactive replanning system that yields paths that consistently match human intent. The triggers are to replan at (a) regular time intervals, (b) when the current robotic path deviates from the user intent, and (c) when a better path can be obtained from a different homotopy class. Further, we consider one user-generated replanning trigger that allows the user to stop the robot anytime to put the robot onto a new route. These four trigger variants seek to answer two fundamental critical questions: When is a re-planned path acceptable to a human? and How should a planner involve a human in replanning?
Keywords  intent, intention, plans, BDI, reasoning, commitment, multi-objective path planning, tradeoffs, replanning, graphical user interfaces, human-robot teams, human-robot interaction, human-robot collaboration

4.1 Introduction

The notion of intent has been conceptualized and defined by philosophers 6, 20, 22, 40, 44, 61, 111, 138, psychologists 5, 13, 26, 39, 62, 104, 105, 145, 159, 163, 169, neuroscientists 65, 66, 78, 115, and artificial intelligence researchers 35, 36, 59, 70, 81, 100, 130. Most of this literature expresses intent as a mental state that enables an agent to commit to achieve something in future. Many theories either consider or suggest that intentions are precursors to action or sequences of actions 6, 13, 20, 40, 44, 61, 62, 105, 117. Wikipedia uses a concise (albeit incomplete) summary of Bratman’s notion of intent 20:

"Intention is a mental state that represents a commitment to carrying out an action or actions in the future."\(^1\)

Much of the intent-based literature assumes a rational agent, which could be either human or a robot; this paper assumes that the human holds the intent and the robot executes intent. Task execution has two components: what it is to be achieved, that is, the desired outcome (the goal), and the means (trajectory) to achieve it. Accordingly, we assume intent includes (a) the agent’s capabilities, (b) the interaction environment where trajectories are executed, and (c) the agent’s commitment to a goal and trajectory over time. In this paper, a trajectory is a path taken to reach the goal. Paths are chosen based on constraints, objectives, and policies/strategies/plans that determine how intent is translated into action. This paper deals directly with the temporal aspect of intent, when a persistent commitment reaches its “expiration date” 34.

The primary contribution of this paper is a partial answer to the question: when does a human replan in dynamic environments such that the adverbial description of a task maintains intent while balancing multiple objectives.

\(^1\)https://en.wikipedia.org/wiki/Intention
4.2 Related Literature

The intent theory has been connected to many attitudes/pro-attitudes thereby bringing to
the table its multiple notions. In addition to intentions being seen as a product of desires
and beliefs that is strongly emphasized by Dennet 7, 41, 44 to name a few, intentions are
also shown to be related to commitment, reasoning and partial plans 20, 35. They are
associated with plans, actions, and time 20, 34, 111. Gibbs 60 portray intentions in the light
of interactions. He reports that people hardly ever act independently, hence, in addition to
intentions being private mental states, intentions are also emergent product of interactions.
Intentions are also viewed as a kind of persistent goals, where persistence involves an agent’s
internal commitment to a course of events over time 34. The vast literature on intent 5, 6, 20,
22, 34, 35, 39, 44, 59, 60, 65, 66, 70, 78, 81, 100, 104, 105, 115, 130, 138, 139, 145, 159, 163, 169
leads to following notions: commitment, persistent (that is, intentions are not to be abandoned
atleast for some time), beliefs, desires, rational processes, plans, partial plans, goals, action
and time.

We discuss intent here in light of human-robot teams or human-robot interaction
(HRI). In HRI, it is usually the human who owns the intent 12 and communicates his intent
either explicitly or implicitly to the robot. The robot assists the human to accomplish his
intent. In this work, what human wants is linked to decision making under multiple conflicting
objectives for robot navigation in a dynamic environment. Hence, intent is a tradeoff that
dictates ‘how’ the robot should navigate to the goal.

Success in human-robot tasks to a large extent depends on the success of communicating
the intent to the robot. The techniques for communicating the human intent can be classified
as (i) explicit and (ii) implicit. Explicit intent strategies include both verbal [112] such
as natural language/speech commands, and non-verbal communication [114] such as eye-
gaze, gesture and facial expression, as well as combination of these [132, 162]. Further,
conventional user interfaces that use devices such as keyboard, mouse, joystick, hap-tic and
touch interaction [43] had been around for more than half a century in order to commands
robots. Note that, human inputs such as from knobs or from sliders or from natural language can be transitioned into vectors to represent an intent. Implicit intent, also known as indirect intent by [37], include physiological signals such as ECG (Electrocardiogram), EMG (Electromyogram), EEG (Electroencephalogram), skin conductance, pupil-dilation [17, 29, 37] etc. Our path-replanning architecture uses the adverb palette (AP) discussed in Sec. 4.4.1 to communicate human intent to the robot.

4.3 Operational Definition of Intent

Intent in this paper is based on Bratman’s 20 and Malle et al.’s 105 theories of intentions. Bratman approaches intentions by way of planning theory. Accordingly, intentions are partial plans brought about by deliberation and practical reasoning considering resources and coordination (both intrapersonal and interpersonal); plans which on a commitment get updated with time that eventually bring about the desired outcome. Note that partial plans does not mean plans are incomplete but rather that plans get updated in response to the interaction environment as the agent executes the initial partial plan. Updating and renewing a plan maintains intent.

This paper assumes that beliefs, desires, and intentions form the basis of an intentional action 20, 44, 105. Desires are what an agent wants or wishes for. When desires are combined with commitment, reasoning, and action, a subset of possible desired goals become intentions. A goal is a mental representation of a desired outcome that one wants to attain through action and desired means. Intention is therefore a commitment towards achieving the goal. While goals are outcomes that are measurable at the end of certain time, intentions also include the “journey” towards these goals satisfying desired attributes.

Belief is the knowledge that the agent carries about itself and the surrounding environment to bring about planning and action in order to attain the goal. For a path planning problem, the environment has information such as: ‘where is the enemy?’ ‘how far is the goal state?’, ‘what are the alternative trajectories available to consider?’ etc. Beliefs
also include all the constraints and objectives that dictate or specify how the goal needs to be attained; these attributes determine the desired means to be met on the trajectory or path of achieving the goal.

Capabilities are critical for goal attainment. For human, capabilities are associated to practiced or acquired human skills and for a robot, capabilities would be its functionality dictated by the algorithms endowed in it. We use capabilities, autonomy and algorithms interchangeably for robots.

Initially, before the action begins the agent will have a prior knowledge about itself and the environment. Based on the prior knowledge, the agent partially plans the solution. Later on, as action proceeds, the agent updates its knowledge in coordination with itself and with the environment.

Fig. 4.1 illustrates how the different components above relate to each other for a problem that includes one human and one robot. Our operational definition of intent is:

In an environment inhabited with agents of different capabilities, desires and beliefs, intent is a commitment of a rational agent to bring about a desired outcome in a reasonable time by shaping a sequence of environment states that (a) satisfies
both a set of *constraints* and a set of *objectives*, and (b) executes a *plan/policy* toward the desired outcome..

4.4 Replanning Architecture

An important element of intention is monitoring when and whether a particular intent is feasible or relevant, and when intent needs to be updated. For a specific human intent, if the real-time execution of the partial plan does not fulfill the human’s intent, then alternative plans or course of actions need to be built online that would satisfy the intent — referred to as *replanning*. Similarly, intent can change based on emerging behaviors and constraints in an environment, especially one with multiple agents [60]. We define *triggers* as events that signal the human the need to reconsider the current intent solution along with a new plan. This calls for (a) some sort of replanning framework for intent and (b) some sort of user interface that enables intent monitoring and path-replanning.

The design of replanning architecture here is inspired by the planning considerations presented in Chapter 8 of 151 which says: “...if a planning model is to generate planning behaviors that somehow mimic those of a human planner, the model must attempt to replicate the various stages of planning,...” Accordingly, our replanning framework includes elements for (a) information exchange, (b) situation assessment, (c) course-of-action development, and (d) monitoring and replanning.

Our replanning system architecture has three entities: a GUI that we call the *adverb palette* (AP), the robot, and path-planning/replanning algorithms.

4.4.1 Adverb Palette

The *adverb palette* (AP) is an interactive graphical interface that we described in 140, which was designed to express human intent; in this paper, we extend AP to support intent monitoring and replanning. The interface is tailored to a robot navigation task in a dynamic environment in which the robot needs to navigate from location A to location B under
conflicting objectives. The environment is depicted by a map that shows the robot’s current location, the goal where it has to reach, the enemy positions, and the obstacles. In this application, we have three objectives that are expressed as adverbs *quickly*, *stealthily*, and *efficiently*, each represented with a unique color on the user interface. The human expresses the intent using these adverbs that basically defines the objectives/constraints on how the robot should navigate to location B, which is the goal location. The adverb *quickly* is a command to the robot to take the shortest route based on Euclidean distance, the adverb *stealthily* is a command to the robot to take a route evading the enemy as much as possible, and the adverb *efficiently* is a command to the robot to take a route that minimizes fuel consumption. For a tradeoff, such as a path where both distance and stealth are intended, a mixture of *quickly* and *stealthily* needs to be created on the AP. The tradeoffs are thus represented as a mixture of colors. The tradeoff/color encodes the intent of the navigation task. We refer the reader to 140 for the basic working of the AP.

At the outset of the navigation task, AP serves as a means to visualize multiple solutions (plans/options/paths) generated by the robot. Figure 4.2 shows the *adverb palette* with 9 candidate paths shown on the map. Each path starts from an initial location, indicated by the black robot in the lower left of the map, and ends at the goal location in the upper
Figure 4.3: Example of a tradeoff that satisfied both quickly and stealthily.

right of the map. The path is a series of waypoints through the map, which determines a trajectory that either optimizes a single objective or balances a mixture of objectives. Each trajectory from start to goal is constructed using different mixtures of the adverbial objectives. That is, each solution is a tradeoff between different objectives. The human communicates intent to the robot by selecting one of the tradeoff solutions using the right panel of the interface. For example, if the user clicks on the big red circle on right panel of Figure 4.2, then a quick path is desired; a red path on the map in Figure 4.2. On the other hand, if the user clicks on the big green circle, a stealthy path is selected for travel. Figure 4.3 shows a tradeoff between quickly and stealthily formed by mixing the two adverbs on the right panel resulting into the brown mixture. Thus, the human intent here is a tradeoff.

4.4.2 Planning, Execution, and Replanning

The robot is equipped with a tree-based planner, online fast marching tree*($O\text{-}FMT^*$) \(30\) that is used to generate the initial set of plans. The planning and replanning problem is described in Section 4.5. The robot is an autonomous agent that has the capability to execute the path by following the path chosen by the user via $AP$.

During path execution, the robot’s planner is capable of updating the current path and proposing alternative paths that may better match intent. It can generate a new path in less than 2 secs for a given intent based on the weighted combination of speed, stealth,
Figure 4.4: Robot’s current and new path. Current path: dashed green, New path: solid green and energy-efficiency objectives. Updating a path theoretically allows a robot to adapt its execution so that it matches intent as the world changes. Proposing alternative paths theoretically allows a robot to present alternatives that better match the human’s intent or allow a human to change intent during execution. Figure 4.4 shows robot’s new path/plan in a solid green trajectory, and the current path in a dashed green trajectory for an example stealthily intent set at the beginning. The updated plan assists replanning. Sec. 4.5.4 details the concept of how the new plan differs from the current one.

The robot can prompt the human to consider alternate paths at specific events called triggers. Potential triggers include replanning at (a) regular time intervals, (b) when the current robotic path deviates from the user intent, and (c) when a better path can be obtained from a different homotopy class. The different triggers provide an opportunity to the human to either approve or disapprove the new robotic path. We discuss each of these triggers in detail in Sec. 4.6. The robot is said to have successfully navigated to the goal (location B)
if it maintains its intent throughout the entire navigation task or if the human is able to express a revised intent and the robot follows the revised intent.

At triggers, the human uses his judgment to either to remain on the current plan or change to the new plan, thereby collaborating towards a successful intended task execution. The new plan suggested by the robot may vary in four possible ways: (i) the old plan is no more according to intent, but the new plan is; (ii) the old and the new plan both follow intent but differ in trajectories; (iii) the old and the new plan differ only by a margin (dictated quantitatively), they both follow intent; (iv) the old plan and the new plan both do not follow intent. In addition to responding to the triggers generated by the robot, the human himself can pause the robot anytime and put the robot on a new plan which results into a user initiated trigger.

Human input is critical in automation. The AP serves as a platform by which the human monitors execution, becomes aware of triggers, and makes adjustments to paths or intent. The AP thus aids for intent exchange, situation assessment, course-of-action analysis and selection, and monitoring and replanning.

4.5 Path Planning

The path-planning problem and planner descriptions presented in this section are adapted from 30, 172.

4.5.1 Problem

For robotic path planning, an environment at any time is a topological space $X \subset \mathbb{R}^d$, with an obstacle space $X_{\text{obs}}$, an initial state $x_{\text{init}}$, and a goal region $x_{\text{goal}}$. The obstacle-free space is denoted by $X_{\text{free}} = X \setminus X_{\text{obs}}$. Consider the set of $J$ objectives determined by a vector cost function $c(\cdot) = [c_1(\cdot), \ldots, c_J(\cdot)]^T$ defined by $c : \mathbb{X} \to \mathbb{R}^J$. Note that $c$ is defined for all points in $X$ in free space. At a given time $t$, let $X_E$ be the set of the locations of $n$ enemies,
Let the location of the robot be denoted by $x_{\text{rob}}$, and $x_{\text{rob}}$ is not $x_{\text{goal}}$. The robot acts in this environment specification to create a trajectory towards the goal.

The term agent in this paper applies to the robot. The robot knows the goal location to attain and has the 'how' intent communicated to it from $AP$. The path-planning/path-replanning problem in this paper deals with a robot that navigates in a dynamic environment under multiple objectives, quickly, stealthily, and efficiently; thus, $J = 3$.

4.5.2 Terminology: What is a Path/Trajectory and What are its Costs?

A trajectory or a path is a continuous curve induced by an robot’s algorithm parameterized by $s$, denoted by $\sigma : [0, s] \rightarrow X$. Note that, the trajectory satisfies (a) $\forall \tau \in [0, s]$, $\sigma(\tau) \in X_{\text{free}}$; (b) $\sigma(0) = x_{\text{init}}$, $\sigma(s) = x_{\text{goal}}$; (c) causes/influences a sequence of environments $X_1, \ldots, X_g$ where $i \in \{1, \ldots, g\}$, $g$ is the number of elements in the sequence, the first element of the sequence, $X_1$, is adjacent to $x_{\text{init}}$, and finally the last element $X_g$ is adjacent to $x_{\text{goal}}$.

Given what a trajectory or a path is, $T$ is the set of all obstacle avoiding trajectories with an initial point as $x_{\text{init}}$ and end point as $x_{\text{goal}}$.

At the start, before the robot starts moving, given a set of three objective functions, let $\Sigma = \{\sigma_p\}$ denote the set of Pareto optimal paths. Since we are doing path-planning on a two-dimensional plane, a path is a parameterized curve that exists in $\mathbb{R}^2$. Thus, each path $\sigma_p$ is a mapping from a parameter space to $\mathbb{R}^2$. Without loss of generality, let the parameter space be the continuous interval $[0, 1]$. Thus,

$$\forall j \quad \sigma_p : [0, 1] \mapsto \mathbb{R}^2.$$  \hspace{1cm} (4.1)

For the kind of path-planning that we are doing, the path is constrained to begin at a starting location $(x_0, y_0)$ (that is, $x_{\text{init}}$) and end at the goal location $(x_f, y_f)$ (that is, $x_{\text{goal}}$), yielding
the constraints on the path as follows:

\[ \sigma(0) = (x_0, y_0) \]
\[ \sigma(1) = (x_f, y_f). \]

For a multiple objective problem expressed as cost functions, let \( J_i \) denote the \( i \)th cost function. Suppose that we have three cost functions, \( i \in \{1, 2, 3\} \). Each cost function assigns a real-valued cost to a path,

\[ \forall i \ J_i : \Sigma \mapsto \mathbb{R}. \quad (4.2) \]

Thus, a cost function takes a path, \( \sigma \), and assigns a real-value, \( a \in \mathbb{R} \), to it, \( J_i(\sigma_p) = a \).

The tree-based planner path is made up of \( m \) vertices and weighted, directed edges. The edges vary in costs. Each weighted, directed edge is a cost to traverse from a parent vertex, \( o_k \), to a child vertex, \( o_{k+1} \); let \( c(o_k, o_{k+1}) \) denote the cost of this edge. A path \( \sigma_p \) is a sequence of edges through the tree, with the first vertex located at \( (x_0, y_0) \) and the last vertex located at \( (x_f, y_f) \). Because of varying cost edges, we don’t have a uniform partition on the parameterization interval \([0, 1]\) so we will write the partition as \( m + 1 \) different points in \([0, 1]\), \( s_k \), where \( s_0 = 0, s_m = 1 \) and \( s_k < s_{k+1} \).

The cost of a path is the sum of the costs of the edges. Thus,

\[ J_i(\sigma) = \sum_{k=0}^{m-1} c[\sigma(s_k), \sigma(s_{k+1})] \quad (4.3) \]

where \( \sigma(s_k) \) equals the location of vertex \( k \).

4.5.3 Normalization and Scaling: What is the Color of a Path?

Our prior work found that the color palette of the AP was a useful way for a human to express intent to a multi-objective path-planner [140]. Consequently, we need to assign a color to each path. There are three objectives and, for design and usability purposes, we assign
colors from RGB space in a constrained way. The color is restricted to three components, one each from the RGB set of colors, but constrained such that the sum of the red component, green component, and blue component sum to one. Thus, we will create a color vector $h(\sigma_p) = [R(\sigma_p), G(\sigma_p), B(\sigma_p)]$ satisfying

\begin{align*}
R(\sigma_p) & \in [0, 1] \\
G(\sigma_p) & \in [0, 1] \\
B(\sigma_p) & \in [0, 1] \\
\sum_{\text{color} \in \{R,G,B\}} \text{color}(\sigma_p) &= 1.
\end{align*}

We need to translate the cost triple $(J_1(\sigma_p), J_2(\sigma_p), J_3(\sigma_p))$ into a color vector. We do this by associating each color to a different cost function; without loss of generality we assign cost functions such that $J_1$ corresponds to red, $J_2$ to green, and $J_3$ to blue. For reasonable correspondence to colors, we create a normalized objective $o_i(\sigma_p)$ from $J_i(\sigma_p)$ for each path $\sigma_p$ as follows:

$$o_i(\sigma_p) = \frac{J_i(\sigma_p) - \min_{\sigma' \in \Sigma} J_i(\sigma')} {\max_{\sigma' \in \Sigma} J_i(\sigma') - \min_{\sigma' \in \Sigma} J_i(\sigma')}.$$  \hfill (4.4)

The color of a path is defined as the vector, $h(\sigma_p)$, that maximizes the cosine similarity between the objective vector.

$$o(\sigma_p) = [o_1(\sigma_p), o_2(\sigma_p), o_3(\sigma_p)]^T$$

and the inverse color vector

$$h' = [R', G', B']^T$$

$$1 = R + G + B$$
where

\[
R' = 1.01 - R \\
G' = 1.01 - G \\
B' = 1.01 - B
\]
yielding

\[
h(\sigma_p) = \arg \max_{h'} \frac{h' \cdot o(\sigma_p)}{\|h'\| \|o(\sigma_p)\|} 
\] (4.5)

Since for costs (Equation 4.4), lower values are better, we subtract each of the RGB component of color from 1.01 in the equation above so that the higher preferred color values get converted to lower values and thus can be matched with the corresponding lower costs and vice versa. Further, the color element is subtracted from 1.01 instead of 1 so as not to nullify the effect of an objective with a corresponding color component of 1 in color ∈ R, G, B in the computation of cosine similarity in Equation 4.5.

Figure 4.5 illustrates the different spaces involved in assigning a path a color. The axes in the figure represent the objectives \(o_1, o_2, \text{ and } o_3\). Since \(o_i(\sigma_p) \in [0, 1]\), the unit cube shown in the figure bounds the ranges of the objectives for any possible path \(\sigma_p \in \Sigma\). Colors are normalized such that they must sum to one, so the triangular simplex represents the set of possible colors.
The square box represents the objective vector $o(\sigma_p)$ for a path $\sigma_p$, and the brown line segment indicates the vector emanating from the origin to the vector. The color of the path is given by the coordinates at which the red line segment intersects the triangular simplex, which occurs at the brown ellipse obtained by using Equation 4.5.

### 4.5.4 Replanning Trigger

Suppose that (a) a human has specified a desired path color and (b) path $\sigma^h$ is the path from $\Sigma$ that most closely matches that color. Suppose further that the robot has been following path $\sigma^h$ for some period of time and has reached location $\sigma(s)$, where $s \in (0, 1)$; the open interval $(0, 1)$ indicates that the robot has been traveling for some positive time, meaning that $s \neq 0$, but hasn’t reached the end of the path, meaning that $s \neq 1$. Even though $s$ isn’t technically a time, we can treat it as if it is a time unit, so suppose at time $s^*$ something happens and the costs change. For simplicity, suppose that cost function $J_i$ has changed. For example, suppose that objective $i$ generates the edge cost in Equation 4.3 with high edge costs if the edge is close to an enemy, but the enemies move at time $s$. Should the robot change paths?

Figure 4.6a replicates Figure 4.5, but for only two objectives. The unit square represents the set of possible objective vectors; the small brown square is the end point of the objective vector, $o$ for a particular path, $\sigma_p$; the brown line segment emanating from the origin is the objective vector $o(\sigma)$; the diagonal blue line is the set of possible normalized colors; and the small brown ellipse represents the color for the path $h(\sigma_p)$. The small brown square indicates the cost with respective to all objectives at time $s = 0$.

Figure 4.6b illustrates what happens when one of the cost functions changes at time $s^*$. Note that the path path $\sigma^h$ hasn’t changed, but the costs have changed in response to the changes in the environment. For example, say that the enemy has approached closer to this path at time $s^*$ resulting into a higher stealthily cost. As a result the objective vector that includes the change in cost function has shifted down and to the right. Note the shift in costs.
indicated by a dashed curve in Figure 4.6b. Now, because the objective vector $o^*(\sigma_p)$ has changed, the color associated with the path $\sigma_p$ has changed from brown to light blue. Since we assumed that the human intent was indicated by the brown vector, the original path $\sigma_p$ no longer matches the human intent. Should this be a trigger for replanning?

Thus, the costs of a path are associated with color and the human expresses intent by selecting a color. Once the robot starts moving the change in environment may lead to changes in objectives, which may correspond to a different color thereby indicating a deviation from intent. Sometimes the change in path costs may not induce a big color change, but other times the path costs may cause a large color change.

Note that when we replan, we don’t care to compute the path from the start point, $(x_0, y_0)$, to the end point, $(x_f, y_f)$, anymore. Rather, we only care to compute a new path such that the replanned path is identical to $\sigma^h$ up to time $s^*$; after time $s^*$ the replanned path may differ from $\sigma^h$. In other words, the replanned path should shift to a better path from time $s^*$ onwards. The problem is illustrated in Figure 4.7. The robot has followed the original orange path up until time $s^*$. At time $s^*$, it needs to decide whether to continue along the original orange path or switch to a new path that builds from the original orange path. The green path in the figure represents a new replanned path.
4.6 Replanning Taxonomy

Given the intent, path-planning, and replanning formalism, we can describe four different replanning triggers. For the remainder of the paper, we consider only two paths: the robot’s current path and a path that is automatically replanned as the robot and enemies move in the world. The replanned path is a proposed change or suggestion to the human. Replanning triggers are possible ways that a human might use the current path and replanned path together to maintain intent.

On a trigger, the AP displays (a) the robot’s current location with an “I’m here” status denoted with a robot with a red top icon, (b) the current path in a dashed pattern, (c) and the replanned path in a solid pattern. The path already travelled is shown as tiny circular footsteps on the map. On a trigger, the interface pops up buttons that allow the user to either ‘Stay with the current path’ or ‘Switch to the new path’. The AP also displays path color history to help the human understand original intent and measure drift in intent in the lower left section of the left panel; the first triangular arrow of the history indicates the initial intent with which navigation had started. See Figure 4.4.
4.6.1 Time Trigger

Replanning at regular intervals is the simplest replan strategy. A time trigger occurs at deterministic time intervals to make the human aware of the current environment and the two paths: the current one and the replanned one. The two paths may or may not vary in different aspects such as the intent, homotopy (discussed later), and/or a combination of these. Figure 4.8 and Figure 4.9 show two occurrences of time trigger in an example stealthily navigation (the intent chosen by the user was stealthily on AP and is represented by green color). In our planned experiments, the robot generates a time trigger every \( n \) seconds unless a different trigger occurs.

![Figure 4.8: Time trigger at point \( s_{t1} \).](image1)

![Figure 4.9: Time trigger at point \( s_{t2} \).](image2)

4.6.2 Change-in-Intent trigger

Section 4.5.4 showed how the costs of a path after time \( s^* \) may change due to changes in the environment during navigation. Let \( c_{\text{threshold}} \) be the cosine similarity value given by Equation 4.5 for \( h \) associated with path \( o(\sigma_h) \), the original path. If the new costs of the path \( o(\sigma_h) \) change such that Equation 4.5 produces a value equal to or greater than \( c_{\text{threshold}} \), then we say that the path \( (\sigma_h) \) maintains the original intent. However, if the new costs after time \( s^* \) change as illustrated in Figure 4.6b such that the Equation 4.5 yields a value below \( c_{\text{threshold}} \) then the path \( (\sigma_h) \) is far enough away from \( h \) that the current path no longer matches the original intent.
Figure 4.10 shows an example of change in intent for a stealthily navigation task. The robot had started with a stealthily intent, green. At some time \( s \), the current path color changes to red — the path became expensive because of the approaching enemy. The automatically replanned path better matches the original intent, so the human may want to switch paths.

### 4.6.3 Homotopy Trigger

Quoting from [173], path \( \sigma_1 \) is said to be *homotopic* to path \( \sigma_2 \) if \( \sigma_1 \) can be mapped to \( \sigma_2 \) without encroaching on any obstacle 16, 80. Otherwise, the two paths are said to belong to different homotopy class or said to be *non-homotopic*. We denote homotopic paths by \( \sigma_1 \simeq \sigma_2 \).

A *homotopy trigger* occurs when the robot’s replanned path and original path are non-homotopic. In our planned experiments, the robot uses the algorithm 173 to check homotopy. The idea behind this trigger is that if the current and replanned paths go around obstacles in different ways, then the human may need to consider whether the path matches intent even if the path colors stay the same. Figure 4.11 shows an example of a homotopy trigger that occurs for a navigation task meant for the robot to go *stealthily* as well as *quickly*, a brown color path.

![Figure 4.10: Change in intent trigger example.](image1)

![Figure 4.11: Homotopy trigger example.](image2)
In order to not irk a human from very frequent homotopy or change-in-intent triggers, in the planned experiments we will restrict the frequency of these triggers such that a time trigger would essentially separate any two homotopy or change-in-intent triggers. That said, the time trigger may show up a non-homotopic or intent-violated path if any as shown in Figure 4.8.

4.6.4 User-Initiated Trigger

In addition to the above three triggers generated by the robot, in the planned experiments the human can pause the robot anytime and put it on a replanned path. This is possible because during the walk the AP displays the robotic replanned path. The AP facilitates user initiated trigger with a ‘Pause and Replan’ button on the left panel.

4.7 Summary and Future Work

In this paper, we applied an operational definition of intent to human-robot teams where both the human and the robot work in collaboration towards a common goal. Using a replanning architecture, we applied the definition to robot navigation in dynamic environment and under multiple objectives. We proposed a replanning taxonomy to answer a critical question: when does a human replan such that the human-intent is preserved.

Our in-house simulations of replanning triggers show promise to have potential in maintaining human intent in HRI by engaging the best of the capabilities of the human and the robot thereby improving the chances towards successful goal attainment. Our hypothesis is that the triggers would help a human to judge and decide the paths at critical times that would eventually support the intended travel. In the very near future, we plan to conduct a user study to answer the question; does the adverbial description of task represent intent. Further, the user study will let us know about the critical triggers that would help maintain intent. To realize this goal, we would look for the subjective scores as to which triggers
appealed to the users and which were found useful. We plan to record the navigation trigger sequence that would reflect the statistics of change in intent during travel.

Acknowledgments

This unnumbered section is used to identify those who have aided the authors in understanding or accomplishing the work presented and to acknowledge sources of funding.
Chapter 5

Intent-Based Robotic Path-Replanning: When to Adapt New Paths in Dynamic Environments

Published in: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC).

Abstract

For goal-based robot navigation in a dynamic environment, human intent includes expectations about what performance objectives are satisfied by a planned path in terms of objectives to be met. If the planned path drifts from the human’s intent as a result of environment changes, the path needs to be replanned. This paper presents a replanning framework with three elements: (a) the integration of fast online path-planning algorithms that generate trajectories conforming to the given intent; (b) a mathematical model that says when replanning must happen; and (c) an evaluation of events that trigger replanning. An interactive graphical user interface enables a human to accept or reject replanned paths when a trigger happens. A study of 50 MTurk participants is used to assess what replanning triggers best enable a human-robot collaboration to persistently satisfy intent?

5.1 Introduction

A human’s intent for a robot includes the robot’s activity — what the robot should do — as well as the objectives associated with the robot’s activity — how the robot should do it. This paper applies this definition of intent to the problem of multi-objective robot path-replanning in dynamic environments. The human’s intent is represented by a planned trajectory that
reaches a desired end state while appropriately balancing tradeoffs between objectives. While the robot executes the trajectory, the environment may change causing the objective functions to change over time.

When objectives change over time, the initial chosen trajectory may fail to meet the human’s intent while the robot moves to the goal. For example, suppose that the selected trajectory was to evade enemies in the environment but during execution the enemy moves really close to the initially planned trajectory. Under such conditions, an alternative path needs to identified. This is referred to as ‘replanning’. Importantly, the revised plan should align with human intent. The design question is therefore, **under what circumstances should the robot switch from it’s current trajectory to a replanned trajectory?**

*Triggers* are events that signal the human to consider replanning. They provide an opportunity to correct a planned path to keep it aligned with intent.

This work complements our previous works that define triggers [141] and graphical user interface (GUI) for robot path planning [140]. Accordingly, the replanning system architecture has an interactive GUI, the robot, and path-planning/replanning algorithms. In this work, we extend the GUI to enable a human to manage replanning. On a trigger, the GUI communicates (a) the robot’s *current location*, (b) the *current path*, (c) the *replanned path*, and (d) interface elements to switch to preferred path choices, such as pop up buttons that allow the user to either ‘Stay with the current path’ or ‘Switch to the new path’.

We provide a mathematical model for when a robot should replan while navigating in changing environments. The model helps quantify the intent-mismatch associated with a path. The mismatch is monitored through three classes of triggers: (a) *time-based*: replanning at regular time intervals, (b) *intent-based*: replanning when the executing path no longer matches intent, and (c) *region-based*: replanning when there is reason to believe that a better path can be obtained from a different homotopy class. We evaluate these three replanning triggers using MTurk participants.
5.2 Problem Formalism

5.2.1 Path-Planning Task

The path-planning task is to find a solution, $\sigma$ starting from an initial state (or robot configuration) and terminating at a specified goal state (or robot configuration) bounded by time. Let $X_{init}$ be the initial state, and let $X_{goal}$ be the goal state. A solution or a trajectory, $\sigma$, is a sequence of states, $\langle x_0, x_1, \ldots, x_{n-1}, x_n \rangle$ such that $x_0 = X_{init}$ and $x_n = X_{goal}$. This paper uses the term ‘trajectory’ and ‘path’ interchangeably.

5.2.2 Multi-Objective Path-Planning

Denote the set of finite possible trajectories from $X_{init}$ to $X_{goal}$ as $\Sigma = \{\sigma_i\}$. Each trajectory, $\sigma_i$, is represented as a sequence of directed edges made of $n$ vertices. Thus, the sequence of configurations replaces $\langle x_0, x_1, \ldots, x_{n-1}, x_n \rangle$ with $\langle v_0, v_1, \ldots, v_{n-1}, v_n \rangle$ where $v$ denotes a vertex in the path. Assuming that the problem has $J$ objectives to deal with, each $\sigma_i$ is associated with a cost vector defined as $c(\sigma_i) = [c_1(\sigma_i), \ldots, c_J(\sigma_i)]^T$.

Let $c_j(v_k, v_{k+1})$ denote the cost for objective $J$ to traverse from a parent vertex, $v_k$, to a child vertex, $v_{k+1}$. The $j^{th}$ objective cost of $\sigma_i$ is the sum of the costs of the edges. Thus,

$$\forall j \quad c_j(\sigma_i) = \sum_{k=0}^{n-1} c_j[v_k, v_{k+1}] \quad (5.1)$$

where $v_k$ equals the location of vertex $k$.

The multi-objective path-planning problem is to find a trajectory $\sigma$ such that the resulting cost vector $c(\sigma)$ satisfies some trajectory predicate. For example, a trajectory predicate could be to find the path that minimizes the cost for objective $j$, in which case the solution to the multi-objective path-planning problem would be $\sigma^* = \arg\min_{\sigma \in \Sigma} c_j(\sigma)$. Similarly, a trajectory predicate could be to find the path that uses a weighting vector $w = [w_1, \ldots, w_J]^T$ to find a tradeoff among objectives that satisfies $\sigma^* = \arg\min_{\sigma \in \Sigma} w^T c(\sigma)$. 

96
5.2.3 Collaborative Human-Robot Path-Planning/Replanning

Given the robot’s initial configuration, a collaborative human-robot path-planning problem requires (a) a human to specify the goal state, $X_{\text{goal}}$, and the trajectory predicate encoded as intent, $h$, and (b) a robot to generate and follow trajectory solutions, $\sigma$, that reaches a goal state and satisfies the trajectory predicate. For predicate, the human will use natural language-like descriptors like “find a safe path and a stealthy path”.

For the solution, the robot generates $\Sigma = \{\sigma_i\}$. We consider trajectories that are on the Pareto front, as in other multi-objective problems [42], and the Pareto front is assumed to be convex. The human evaluates the trajectories in $\Sigma$ and selects a trajectory that matches his or her intent. Selecting a trajectory $\sigma_h \in \Sigma$ is equivalent to selecting a desired cost vector tradeoff, $c(\cdot)$, associated with the trajectory that dictates preference among the $J$ objectives.

In a dynamic environment, the cost vector varies over time. While following the selected trajectory, the robot simultaneously computes a new trajectory, one that matches intent as the world changes. The Replanned trajectory, $\sigma_R$, is the result of the robot’s ongoing perception. The design question is, under what circumstances should the robot switch from it’s current trajectory $\sigma_h$ to $\sigma_R$? A trigger is an event that gives the human opportunity to switch to the replanned path, $\sigma_R$.

5.3 Related Literature

Following are four major areas of related work:

Planning/Replanning Algorithms Researchers have created several path-planning algorithms to move a robot from a start configuration to a goal configuration, both for static and dynamic environments; see, for example, [68, 85–88, 97, 152]. Most existing replanning algorithms find shortest paths. Others are triggered by environment changes such as the emergence of obstacles [49, 57, 120, 171, 174]. In Hyperion robot navigation [161], progress-based replanning was triggered if the rover did not reach the expected navigation waypoint.
at the scheduled arrival time. In contrast, Cummings et al. [38] studied how time-based replanning triggers and replanning rates affected operator performance and workload when supervising a decentralized network of heterogeneous unmanned vehicles. Yoshida et.al [174] explored replanning using two threads, one for execution and the other for planning. When a collision is expected along the current path, the execution thread queries the planning thread for a better plan. In this work, we use time-based, region-based and intent-based replanning for robot navigation.

**Multi-Objective Planning** In real-time navigation, multiple objectives include path length, energy consumed, coverage, smoothness, traversal risk, safety, stealth, etc. [55, 164]. Multi-objective path-planning is typically applied to static environments [42, 79, 106]. Research on combining multi-objectives and replanning is rare [161, 164]. Work in [161] produced plans that are optimal with respect to weighted combinations of minimum plan length and energy cost. The authors of [164] view the cost of a trajectory as a function of time for traversal, traversal risk, stealth, and visibility.

We explored two path-planners; MORRF* algorithm [172] and online fast marching tree* (O-FMT*) [30]. MORRF* blends two concepts: optimal rapidly exploring random tree (RRT*) [84] for efficient path finding, and a decomposition-based approach to multi-objective optimization [176]. MORRF* can be slow and is therefore not appropriate when considering replanning. To achieve faster replans, O-FMT* is evaluated. However, any fast replanning algorithm could replace O-FMT*.

**Human Intervention** This paper combines algorithmic (re)planning and human supervision thereby placing it in the category of human supervisory control [146]. His work emphasizes monitoring the automatic action to detect failures followed by corrections. The trigger mechanism discussed here is analogous to the term intervention from Scholtz [137], which means identifying when the expected actions of the robot are not appropriate given
the current situation. Thus, our notion of a trigger is closely associated with prior uses of ‘intervention’ and ‘correction’.

**Intent** In human-robot interaction (HRI) applications, intent is generally “owned” by the human and expressed through a command and/or correction. Commands dictate (a) what the robot should do and (b) how to do it. This intent is explicitly or implicitly communicated to the robot [8, 12, 24, 93]. Commands to the robot can be in the form of plans, images, sketches, etc. that are convenient when the robot is remotely working in difficult, dangerous, and unstructured environments [147].

![Figure 5.1: Life cycle of human-robot collaboration task.](image)

5.4 **Intent-based Multi-objective Path Planning**

Fig. 5.1 illustrates the execution phases for intent-based planning and replanning. The process starts when the human formulates and expresses intent and ends when the goals associated with the intent are accomplished. In between, the robot follows the planned or replanned trajectory.

During execution, the robot may follow an *initial plan*, an *adaptable plan*, and a *closing plan*. In the initial plan phase, the robot follows the original planned trajectory, $\sigma_h$, transitioning from one configuration to another until replanning is triggered. In the adaptable plan phase, the robot adapts plans. The phase is metaphorically wider than the initial phase to emphasize that the robot may have to replan multiple times. When the robot is close by the goal state, the human and the robot may decide to ignore intent and choose instead a closing plan that effectually disregards nuances in intent in favor of “just reaching the goal.”
The remainder of this section is a review of our previous published work on multi-objective path planning [140].

5.4.1 Creating Meaningful Cost Functions

This work is motivated by path-planning in adversarial environments, hence, we consider three costs, quickly, stealthily and safely. The “quickly” cost is the sum of the Euclidean distances of each edge in the trajectory. The “stealthily” cost function is loosely modeled as the probability of the robot being seen by the enemy. It is the sum of costs for each point on the trajectory, computed as a function of two factors: the distance of the robot from each enemy and the visibility of the robot from all enemies. The safety of a collision-free path is the sum of the inverse distance between the robot position and the nearest obstacle in the environment. This type of “safely” objective is also referred to as “clearance”, defined as the maximum possible distance from obstacles [42].

5.4.2 Normalization

The objectives used by the path planner are expressed as cost functions, $c(\sigma_i) = [c_1(\sigma_i), \ldots, c_J(\sigma_i)]^T$, and these may be in incommensurate units. To consider trajectories as commensurate payoffs, each of the cost term $c_j$ is converted to payoff term as $p_j(\sigma_i) = -c_j(\sigma_i)$ and subsequently normalized. The normalized payoff objective for trajectory $\sigma_i$ is then:

$$O_j(\sigma_i) = \frac{p_j(\sigma_i) - \min_{\sigma \in \Sigma_P} p_j(\sigma)}{\max_{\sigma \in \Sigma_P} p_j(\sigma) - \min_{\sigma \in \Sigma_P} p_j(\sigma)}$$

with a corresponding payoff vector $O(\sigma_i)$.

5.4.3 Intent on the Pareto Front

We are interested in Pareto optimal trajectories. Consider two non-trival objectives, $O_1$ and $O_2$ for path planning. Objectives are non-trivial if it is not possible to get the most of
objective $O_1$ without sacrificing $O_2$ and vice versa. In Figure 5.2, objectives are encoded as payoffs, meaning higher values are preferred to lower values. Each of the red and blue circles in the figure denote a trajectory represented by its payoff vector. The extreme right blue circle in Figure 5.2 corresponds to a trajectory that has highest payoff for objective $O_1$, and similarly, the extreme left blue circle corresponds to a trajectory that maximizes $O_2$. All other blue circles on the blue curve indicate the best trajectories for different tradeoffs between $O_1$ and $O_2$. Notice that each of the red trajectories are “dominated”, meaning, there is another trajectory in which all payoffs are higher. The blue Pareto front curve is made up of non-dominated trajectories.

![Figure 5.2: Two objectives Pareto front of trajectories and intent/trajectory mapping.](image)

For the three adverbs or objectives; ‘quickly’, ‘stealthily’, and ‘safely’, the intent predicate is represented as a three-element vector, $\mathbf{h} = [h_1, h_2, h_3]^T$ [140], where $j \in \{1, 2, 3\}$ and $h_j = [0, 1]$. A value of 1 indicates utmost preference of the corresponding objective, and a value of $h_j = 0$ means ignore objective $O_j$. For example, if $\mathbf{h} = [1, 0, 0]^T$ then the human wants trajectories that pay attention to only the first objective, and $\mathbf{h} = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]^T$ means that the human wants each objective weighted equally.

Figure 5.2 illustrates how the normalized objective vectors, $\mathbf{O}(\sigma_i)$ and the human intent vector, $\mathbf{h}$, are represented in the same payoff space. For simplicity, this is illustrated when there are two objectives. The vectors emanating from the origin represent possible human intent vectors.
5.4.4 Matching Intent to Robot Paths

The intent predicate, \( h \) and the objective vectors, \( O(\sigma_i) \), are scaled so that a) each element \( h_i \) and \( O_j(\sigma_i) \) fall between 0 and 1 and (b) \( h_1 + h_2 + h_3 = 1 \). Each intent component \( h_i \) is mapped uniquely to one of the RGB colors, which is equivalent to using a color palette to \( R + G + B = 1 \).

For a given intent, the trajectory that best matches human intent is given by

\[
\sigma_h = \arg \max_{\sigma_i \in T} \text{CS}(h, O(\sigma_i))
\]

where \( \text{CS}(h, O(\sigma_i)) \) is the cosine similarity between \( O(\sigma_i) \) and \( h \). In other words, the trajectory vector (in payoff) that aligns closely to the intent vector is the trajectory that gets associated with the intent. The following section aid in visualizing this mapping of intent and the robot path.

![Figure 5.3: Path vector \( O(\sigma_h) \), intent vector \( h \), and cost parts.](image)

5.5 Replanning in Dynamic Environments

Suppose the human intended for the robot to follow a stealthy route — a plan that evades enemies. Since as the robot moves the enemy may also move, the objective costs associated with the robot’s trajectory may change such that it may fail to satisfy the intent. The trajectory therefore needs to be adapted or replanned.

5.5.1 Replanning trigger

Suppose that the robot has been following the human-selected trajectory \( \sigma^h \) with intent \( h \) for some period of time and has reached vertex \( v_s \in \{1, 2, \ldots, n - 1\} \), where \( n \) is the number
of vertices in the original path \( v = \langle v_0, v_1, \ldots, v_{n-1}, v_n \rangle \). Even though \( s \) parameterizes the trajectory and isn’t technically a time, we can treat it as if it is a time unit. So suppose at time \( s^* \) something happens and the costs change. For simplicity, suppose that cost function \( O_j \) has changed. Should the robot change paths?

Let the cost function after the change be denoted by \( O_j^* \) corresponding to new edge costs of \( c_j^*(v_s, v_{s+1}) \). The cost function of the path \( \sigma^h \) is adapted from Equation 5.1 by changing from

\[
O_j(\sigma^h) = \sum_{k=0}^{n-1} c_j[v_k, v_{k+1}]
\]  \hspace{1cm} (5.2)

to

\[
O_j^*(\sigma^h) = \sum_{k=0}^{s-1} c_j[v_k, v_{k+1}] + \sum_{k=s}^{n-1} c_j^*[v_k, v_{k+1}].
\]  \hspace{1cm} (5.3)

The difference between Eq. 5.2 and Eq. 5.3 isn’t the path; both use the same path \( \sigma^h = \langle v_0, v_1, \ldots, v_n \rangle \). The difference is that Eq. 5.2 uses the original edge costs for all time and Eq. 5.3 uses the original edge costs up until the cost function changes, which occurs at time \( s \), and then switches to the new cost function.

We’ll use a series of figures to illustrate Equation 5.3. Figure 5.3 left shows the mapping of a trajectory with intent vector \( h \), but for only two objectives. The unit square represents the set of possible objective vectors; the small square is the end point of the objective vector, \( O \) for a the path, \( \sigma^h \) the red line segment emanating from the origin shows the alignment of \( \sigma^h \) with \( h \) as a result of the mapping discussed in Section 5.4.4; the diagonal line is the set of possible normalized intents; and the small circle represents the intent for the path \( \sigma^h \). The squiggly black line below and to the right of the objective vector indicates the total costs that would accrue when the robot walks along the path \( \sigma^h \) if there is no change in the environment. The origin represents the beginning of the problem, before any movement is made, corresponding to \( k = 0 \); no costs have yet accrued. The curve terminates at the small square, indicating the cumulative cost for following the entire path, corresponding to the accumulated cost at time \( k = n \).
What happens if the robot starts moving along $\sigma_h$, and environment changes resulting into costs changes? Figure 5.3 right illustrates the two parts of Equation (5.3). The total cost of the path turns into the sum of the cost of the path segment up to time $s$ and the cost of the path segment after time $s$. Note that for this figure, the cost function didn’t actually change so the the squiggly line stays the same. The next figure illustrates what happens if the cost function changes.

Figure 5.4 (a) depicts what happens when the cost functions changes at time $s$. The squiggly line before time $s$ is precisely what it was in Figure 5.3 right, but the squiggly line changes after time $s$ because objective costs have changed. As a result, the objective vector has shifted down and to the right.

Figure 5.4 (b) illustrates that, because the objective vector $O^*(\sigma_h)$ has changed, the intent associated with the path $\sigma_h$ has changed. Since we assumed that the human’s intent was indicated by the small circle on the diagonal line intersecting the red line, the original path $\sigma_h$ no longer matches this human intent. Instead, the objective vector now matches another intent indicated by the small circle at the intersection of now a blue line and the new path/objective vector. Should this be a trigger for replanning? Yes, here is the situation when we need to replan a new path.
Note that, we only compute a new path such that the current path and the new path, also called as the replanned path, are identical to \( \sigma^h \) up to time \( s \); after time \( s \) the replanned path may differ from \( \sigma_h \).

The problem is illustrated in Figure 5.4 (c). The robot has followed the original blue path up until time \( s \). At time \( s \), a new path needs to be computed that would match the original intent — the intent at the intersection of the red line and the diagonal in Figure 5.4 (b). Thus, at time \( s \), the human needs to decide whether it wants the robot to continue along the original blue path or switch to a new brown path that builds from the original blue path.

### 5.5.2 When to Replan

Previous work identified multiple triggers when a human may replan [141]. This paper evaluates three triggers: time-trigger, intent-mismatch trigger, and homotopy trigger. At each of the trigger, the human is presented with a replanned trajectory and allowed to choose between the replanned trajectory and the original trajectory.

The time trigger signals the human to check if something is wrong at regular time intervals. Most of these checks may result in the human concluding that the path still matches intent, with an occasional need to replan detected.

The intent-mismatch trigger is analogous to system alerts on human-machine systems. These alerts seek the human’s attention if something goes wrong with respect to expectations, and uses the cosine similarity distance between the path objective vector and the intent vector (red vector in Figure 5.4 (b)). The intent-mismatch trigger indicates that the current path no longer satisfies human intent.

Given that a path replanner is always running in the background, the homotopy trigger signals the human when the replanned path is in a different homotopy class compared to the current path, giving the human the opportunity to switch to the new path that resembles a detour.
We hypothesize that the intent-mismatch and homotopy triggers help the operator to replan at critical times/events that should improve the performance of human-robot collaborative tasks. A natural limitation of this assumption is whether the replanned trajectory matches human intent.

5.6 Evaluation of Triggers

To determine the usefulness of these triggers, we recruited Amazon Mechanical Turk (MTurk) workers to answer questions in a survey format on each of the three trigger mechanisms. When a trigger occurs, the robot stops to seek advice from the human.

![Example survey question, obstacles, enemies, original dashed path, and replanned solid path.](image)

Figure 5.5: Example survey question, obstacles, enemies, original dashed path, and replanned solid path.

For evaluating each trigger type, we used three time lapse images of trigger occurrences of a simulated robot moving from start to goal using the interface illustrated in Figure 5.5. Each trigger image shows the environment with (a) the robot’s current location, (b) the
current path — the path which the robot is currently following shown by a dashed pattern, (c) the new path that the robot has recalculated and thinks is better — shown by a solid pattern, and (d) the enemy location (orange entities).

For each image, participants were asked two types of questions. The first question type, Q1 category, asked for a participant’s opinion on whether s/he was satisfied on being asked for advice by the robot at that particular walk juncture. The second question type, Q2 category, asked for a participant’s opinion of whether s/he would recommend the robot to change the path. Fig. 5.5 shows an example of the two questions related to a time trigger image.

The response to each of the question is evaluated on a 5-point Likert scale. For the first question, the scale went from “Extremely satisfied” indicated with 1 to “Extremely dissatisfied” with 5. Responses collected closer to 1 for this question would mean that the trigger under investigation captured critical juncture when replanning was desired. For the second question, response scale goes from “Extremely likely” as 1 to “Extremely unlikely” as 5. Responses closer to 1 for this question would mean that the robot could be guided to a better path than the one it is following using that particular trigger mechanism.

5.6.1 Data

Given the three trigger types, three images in each type, and two questions on each image, we had 18 questions for each participant to answer. The trigger types were presented in pseudo random order to avoid bias towards any trigger type. 50 MTurk workers participated, P=50. After completing an IRB-approved consent form and reading through training, each of participant provided 18 responses. We report results for all 50 participants.

5.6.2 Hypothesis

Time Trigger  Since, time triggers may or may not capture intent-mismatch, we hypothesize that participants will not favorably view time triggers.
**Intent-Mismatch Trigger**  Based on intent-mismatch theory, we hypothesize that participants will express satisfaction regarding the occurrence of the trigger. Further, we hypothesize that most participants will recommend the robot to switch to the new path.

**Homotopy Trigger**  Since homotopy triggers offers an intent-based path from a different homotopy class, we hypothesize that the participants may want to consider alternate route, a detour, if available shown by this trigger.

**Replanned path**  Since the replanned path is derived from an algorithm that is seeking to most closely match human intent, we hypothesize that the new path will almost always better match intent than the current path.

**Correlation between Q1 and Q2 responses**  We expect a correlation to be evident between Q1 category and Q2 category questions responses. For example, if the participant was extremely satisfied that the robot stopped at an alert because the current path was violating intent, then s/he would most likely recommend the robot to change its path, unless the new path is equally bad or worse.

### 5.7 Results

#### 5.7.1 Summary Statistics

Table 5.1 shows the summary response statistics with 50 participants for the three evaluated trigger types. The ‘Mean’ column conveys the importance of each trigger type. For mean, we were expecting that an appreciated trigger would have response values between 1 (Extremely satisfied/likely) and 2 (Somewhat satisfied/likely) both for both Q1 and Q2 categories. The intent-mismatch trigger and the homotopy trigger have average means of 1.82 and 1.81 respectively. These means indicate that the responses lie between ‘Extremely satisfied and Somewhat satisfied’ and ‘Extremely likely and Somewhat likely’ for the Q1 and Q2 category
Table 5.1: Different trigger type response statistics.

<table>
<thead>
<tr>
<th>Image Sequence</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Std Error</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time_1.Q1</td>
<td>2.04</td>
<td>1.07</td>
<td>0.15</td>
</tr>
<tr>
<td>Time_1.Q2</td>
<td>1.92</td>
<td>1.43</td>
<td>0.2</td>
</tr>
<tr>
<td>Time_2.Q1</td>
<td>3.42</td>
<td>1.2</td>
<td>0.17</td>
</tr>
<tr>
<td>Time_2.Q2</td>
<td>2.92</td>
<td>1.29</td>
<td>0.18</td>
</tr>
<tr>
<td>Time_3.Q1</td>
<td>3.32</td>
<td>1.33</td>
<td>0.19</td>
</tr>
<tr>
<td>Time_3.Q2</td>
<td>3.62</td>
<td>1.03</td>
<td>0.15</td>
</tr>
</tbody>
</table>

(a) Time trigger image sequence. Average Mean: 2.87

<table>
<thead>
<tr>
<th>Image Sequence</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Std Error</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alert_1.Q1</td>
<td>2.3</td>
<td>1.07</td>
<td>0.15</td>
</tr>
<tr>
<td>Alert_1.Q2</td>
<td>2.54</td>
<td>1.43</td>
<td>0.2</td>
</tr>
<tr>
<td>Alert_2.Q1</td>
<td>2.32</td>
<td>1.35</td>
<td>0.19</td>
</tr>
<tr>
<td>Alert_2.Q2</td>
<td>1.96</td>
<td>1.19</td>
<td>0.17</td>
</tr>
<tr>
<td>Alert_3.Q1</td>
<td>1.58</td>
<td>1.09</td>
<td>0.15</td>
</tr>
<tr>
<td>Alert_3.Q2</td>
<td>1.34</td>
<td>0.87</td>
<td>0.12</td>
</tr>
</tbody>
</table>

(b) Alert trigger image sequence. Average Mean: 1.82

<table>
<thead>
<tr>
<th>Image Sequence</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Std Error</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detour</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detour_1.Q1</td>
<td>1.74</td>
<td>1.12</td>
<td>0.16</td>
</tr>
<tr>
<td>Detour_1.Q2</td>
<td>1.62</td>
<td>1.26</td>
<td>0.18</td>
</tr>
<tr>
<td>Detour_2.Q1</td>
<td>1.64</td>
<td>1.17</td>
<td>0.17</td>
</tr>
<tr>
<td>Detour_2.Q2</td>
<td>1.44</td>
<td>1.07</td>
<td>0.15</td>
</tr>
<tr>
<td>Detour_3.Q1</td>
<td>2.2</td>
<td>1.21</td>
<td>0.17</td>
</tr>
<tr>
<td>Detour_3.Q2</td>
<td>2.24</td>
<td>1.45</td>
<td>0.21</td>
</tr>
</tbody>
</table>

(c) Detour trigger image sequence. Average Mean: 1.81
questions, respectively. The average means provide evidence that support the hypothesis that robot seeking human advice at these triggers was appreciated by the participants.

By contrast, the average mean of 2.87 of the time trigger indicate that the participants were less appreciative of regular checks of robotic paths. A value of 3 for a response indicate neutral feedback for a trigger occurrence. These results provide evidence that although monitoring the navigation regularly is important it may not be an important reason to ask a human about whether the robot should change paths.

### 5.7.2 Comparing Trigger Types

Table 5.2 shows the significant differences between the means of the three triggers for Q1 category. Significance was computed using the Least Squares Means method using Tukey adjustments on participants. The asterisk * denotes significant differences. There were significant differences between time and intent-mismatch trigger. Similarly, time trigger differed significantly from homotopy trigger. However, there was no significant difference between the means of intent-mismatch and the homotopy trigger. Similar significance pattern was observed for Q2 category responses (separate table not shown).

<table>
<thead>
<tr>
<th>Trigger</th>
<th>Trigger</th>
<th>Std Error</th>
<th>t value</th>
<th>Adj P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert</td>
<td>Detour</td>
<td>0.155</td>
<td>1.33</td>
<td>0.38</td>
</tr>
<tr>
<td>Alert</td>
<td>Time</td>
<td>0.155</td>
<td>-5.54</td>
<td>&lt;.0001*</td>
</tr>
<tr>
<td>Detour</td>
<td>Time</td>
<td>0.155</td>
<td>-6.87</td>
<td>&lt;.0001*</td>
</tr>
</tbody>
</table>

Table 5.2: Triggers comparison.

### 5.7.3 Change Path Recommendation

Table 5.3 shows the mean recommendation values obtained from 50 participants for Q2 category at each trigger juncture. That is, the statistics about the preference of participants recommending the robot to switch to the new/replanned path. Based on the ‘Mean’ column in the table, participants recommended changing path for alerts and detours more compared to the time trigger junctures.
<table>
<thead>
<tr>
<th>Image Sequence</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Std Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time_Image1_Q2</td>
<td>1.92</td>
<td>1.43</td>
<td>0.2</td>
</tr>
<tr>
<td>Time_Image2_Q2</td>
<td>2.92</td>
<td>1.29</td>
<td>0.18</td>
</tr>
<tr>
<td>Time_Image3_Q2</td>
<td>3.62</td>
<td>1.03</td>
<td>0.15</td>
</tr>
<tr>
<td>Alert_Image1_Q2</td>
<td>2.54</td>
<td>1.43</td>
<td>0.2</td>
</tr>
<tr>
<td>Alert_Image2_Q2</td>
<td>1.96</td>
<td>1.19</td>
<td>0.17</td>
</tr>
<tr>
<td>Alert_Image3_Q2</td>
<td>1.34</td>
<td>0.87</td>
<td>0.12</td>
</tr>
<tr>
<td>Detour_Image1_Q2</td>
<td>1.62</td>
<td>1.26</td>
<td>0.18</td>
</tr>
<tr>
<td>Detour_Image2_Q2</td>
<td>1.44</td>
<td>1.07</td>
<td>0.15</td>
</tr>
<tr>
<td>Detour_Image3_Q2</td>
<td>2.24</td>
<td>1.45</td>
<td>0.21</td>
</tr>
</tbody>
</table>

Table 5.3: Mean recommendation at different triggers.

5.7.4 Correlation between Q1 and Q2 responses

Table 5.4 shows the correlation between Q1 and Q2 responses. Pearson correlation coefficient given with System Analysis System tool was adopted to determine if there existed any linear relationship between Q1 and Q2 responses. A high correlation is evident between the two response categories for all trigger images except for ‘Image 1’ of time trigger. This is indicated by the significant p-values in the table. Hence, we can conclude that if one is extremely satisfied with the robot pausing at a trigger and seeking advice, s/he is more likely to recommend the robot to switch paths and vice-versa. The significance pattern shows that Q2 responses closely follow the response pattern of Q1 type except for a few deviations in the time trigger.

<table>
<thead>
<tr>
<th>Trigger Type</th>
<th>Metrics</th>
<th>Image 1</th>
<th>Image 2</th>
<th>Image 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Trigger</td>
<td>Correlation</td>
<td>0.27</td>
<td>0.66</td>
<td>0.55</td>
</tr>
<tr>
<td></td>
<td>p-value</td>
<td>0.058</td>
<td>&lt;.0001*</td>
<td>&lt;.0001*</td>
</tr>
<tr>
<td>Alert Trigger</td>
<td>Correlation</td>
<td>0.34</td>
<td>0.54</td>
<td>0.63</td>
</tr>
<tr>
<td></td>
<td>p-value</td>
<td>0.015*</td>
<td>&lt;.0001*</td>
<td>&lt;.0001*</td>
</tr>
<tr>
<td>Detour Trigger</td>
<td>Correlation</td>
<td>0.3</td>
<td>0.36</td>
<td>0.63</td>
</tr>
<tr>
<td></td>
<td>p-value</td>
<td>0.032*</td>
<td>0.011*</td>
<td>&lt;.0001*</td>
</tr>
</tbody>
</table>

Table 5.4: Q1/Q2 responses: Pearson Correlation.
Acknowledgment

This work was partially supported by the U.S. Army RCTA program and by the U.S. Office of Naval Research under grant N000141613025. All results and conclusions are the responsibility of the authors and do not necessarily reflect the opinions of the funding agencies.
Chapter 6

A Measure to Match Robot Plans to Human Intent:
A Case Study in Multi-Objective Human-Robot Path-Planning*

*Under review

Abstract

*Measuring how well* a potential solution to a problem matches the problem-holder’s intent and *detecting when* a current solution no longer matches intent is important when designing resilient human-robot teams. This paper addresses intent-matching for a robot path-planning problem that includes multiple objectives and where human intent is represented as a vector in the multi-objective payoff space. The paper introduces a new metric called the *intent threshold margin* and shows that it can be used to rank paths by how close they match a specified intent. The rankings induced by the metric correlate with average human rankings (obtained in an MTurk study) of how closely different paths match a specified intent. The intuition of the intent threshold margin is that it represents how much the human’s intent must be “relaxed” to match the payoffs for a specified path.

6.1 Introduction

At the heart of multi-objective decision-making is the selection of a solution from a set of alternatives, where each alternative represents a different tradeoff among the objectives. When a human is managing the multi-objective decision problem, the selected solution should match the human’s intent. Intent has been studied in many forms (see the review of related literature), and this paper focuses on how intent can be used in a problem where a ground
robot must plan a path from start to goal while balancing multiple objectives. Thus, the paper operationally uses a notion of intent that can be represented as a numerical vector in a multi-objective payoff space.

Consider, for example, the path planning problem illustrated in Figure 6.1. In the figure, the robot is in the upper left of the map (partially obscured by possible paths) and the goal is in the lower right (also partially obscured by possible paths). Four possible paths are presented which tradeoff between three objectives: safety (stay far from objects), stealth (don’t be seen by the orange “enemies”), and speed (reach the goal as quickly as possible). The bottom path maximizes distance from the gray obstacles, so it satisfies the safety intent. The middle gray path minimizes path length so it satisfies the speed intent. The top path never passes through the sensor range of the enemies, so it satisfies the stealth intent. Each intent can be assigned a numerical value (cumulative proximity to obstacles, path length, portion of path length where the robot can be seen by an enemy, respectively). For mixed intents such as “reach the goal safely and quickly”, a tradeoff between the numerical scores of
safety and speed must be found, resulting in the purple path that stays away from obstacles but is still relatively short.

There are many ways to measure how closely a given vector matches another vector in a multi-objective payoff space including Euclidean distance, cosine similarity, TOPSIS, WPM, etc. [23, 113, 177]. Prior work by the authors [140, 143] demonstrated that many existing measures are not useful in determining how closely a planned 2D path matches a human’s intent, when a verbal intent is expressed as a numerical vector in the multi-objective payoff. This prior work also showed that the cosine similarity metric provides a useful mapping between payoff vector for different possible paths and the payoff vector for the desired tradeoff in objectives. However, as shown in this paper, the cosine similarity metric has a known limitation when a robot is following a path while objectives change. Specifically, in dynamic worlds it is desirable to be able to use an intent-mismatch metric detect when the current path no longer satisfies the human’s intent. This paper presents an example that illustrates how a favorable change in the world can correspond to a large but undesirable change in the cosine similarity metric. A large change indicates the need to replan even though the current path is objectively better than it was when the path was originally planned.

This paper proposes the intent threshold margin (INTHRESH) metric that overcomes the limitation of the cosine similarity metric. The metric is applied to a three-objective path-planning in a known 2D environment. Like cosine similarity, the metric operates by comparing the payoffs of different potential paths to a vector representation of human intent. When no solution perfectly matches human intent, distance information between the numerical intent vector and achievable payoffs is obtained by relaxing the intent criteria by an $\epsilon$ margin until solution(s) are found that match the relaxed intent.

There are a number of limitations of the paper. First, the paper does not address how a human can express intent in numerical form (though see prior work in [140, 143]). Second, the metric is applied only to 2D robot planning, and future work is needed to understand how and whether it can be used for planning of a manipulator in higher dimensions. Finally,
the paper does not address what happens when the intent threshold margin indicates that the planned path no longer matches the human’s intent; future work should explore real-time replanning of a path that will match the intent.

6.2 Related Literature

This paper deals with robot path-planning in 2D worlds with multiple objectives that can be satisfied. There are too many robot path-planning algorithms for a full review, but example approaches include sampling-based approaches, graph-based approaches, field-based approaches, and parametric curve-based approaches both for static and dynamic environments [69, 85, 87, 98, 107, 152, 172]. Many such algorithms cater to multiple objectives such as path length, energy consumed, smoothness, stealth, etc. [55, 164]. For the examples in this paper, paths can be planned before the experiment so the speed of the planner isn’t a large constraint. Because future research will include real-time replanning, the work adopted an algorithm that is a modified version of the FMT* algorithm [77], which is a fast sampling-based planner. The modified version, known as O-FMT*, can take advantage of resampling to replan paths in dynamic environments [30].

As with path-planning, the literature on intent and intentionality is vast. Just in the context of human-robot interaction, intent has been interpreted as activity recognition [133], action prediction [73], and goal identification [175]. It could be argued that intention is closely related to so-called legiblity [45], though the term legibility was originally used to express how a robot’s plan conveyed its intention to a human partner. Moreover, human intent can be expressed and communicated from human-robot robot explicitly as verbal [112, 118] and non-verbal commands [12, 52]. This paper uses an operational notion of intent that can be expressed as a payoff vector in a multi-objective planning space. The authors subjectively mapped verbal descriptions of intent (which are used in the MTurk study) to their numerical representation using tools from prior work [140].
6.3 Representing Path Tradeoffs, Human Intent, and Tradeoffs

A path-planning problem is (a) to find a path that goes from an initial location to a goal location (b) given a map of the environment that (c) satisfies a set of user-defined objectives. In general, there may be tradeoffs among the objectives, meaning that increasing performance on one objective may decrease performance of another.

6.3.1 Representing Tradeoffs as Vectors

Consider a path-planning problem with \( K \) objectives \( \{ o_1, o_2, \ldots, o_K \} \). Suppose that some path-planning algorithm has generated a set of \( N \) solutions to the path-planning problem, \( S = \{ S_1, S_2, \ldots, S_N \} \). Each solution \( S_i \) weighs the \( K \) objectives differently, yielding a numerical objective vector for each path, \( o(S_i) = [o_1(S_i), \ldots, o_K(S_i)] \).

Given the \( N \) paths, it is possible to normalize the objective vector so that each path is represented by a normalized payoff vector \( p(S_i) = [p_1(S_i), \ldots, p_K(S_i)] \) where \( p_k(S_i) \in [0, 1] \). A value of \( p_k(S_i) = 1 \) indicates the highest payoff for the specified objective from the set of possible paths (corresponding to the best path for that objective); similarly, \( p_k(S_i) = 0 \) represents minimum payoff, corresponding to the worst path for that objective.

Figure 6.2 illustrates three possible paths for a two objective problem. Objective 1 and objective 2 are notional, but can be thought of as stealth and speed respectively; for this problem, safety is not important so that objective is not shown. There are three paths shown in the figure: \( s1, s2, \) and \( sx \); the \( sx \) notation is meant to indicate that this is an objective that will be important to a later section. For path \( s1 \), the payoff value for objective 1 (stealth) is about 0.25 and the payoff value for objective 2 (speed) is approximately 0.92. Thus, this path is shown in the payoff space as the vector \( p(s1) = [0.25, 0.92] \) that begins at the origin and terminates at the dot. Similarly, the payoff vector for path \( s2 \) is \( p(s2) = [0.3, 0.91] \) and for path \( sx \) is \( p(sx) = [0.7, 0.9] \). The path \( S_i \) can be thought of as a decision variable with \( K \) features.
6.3.2 Representing Human Intent as a Vector

Let \( \mathbf{I}_H \) be the human intent variable with \( K \) features. This paper assumes that intent is already specified and focuses on matching that intent. In the experiments, the authors subjectively chose the intent vector to match a verbal description of intent using tools from prior work [140]. The intent vector \( \mathbf{I}_H \) specifies the desired human tradeoff between multiple objectives. The intent vector communicates a preference over the \( K \) objectives as \( \mathbf{I}_H = [I_1, \ldots, I_K] \) where \( I_k \in [0, 1] \). An intent value of \( I_k = 1 \) indicates that the objective is maximally important and an intent value of \( I_k = 0 \) indicates that the objective is not important at all.

As mentioned in the example in the introduction, this paper restricts attention (a) to 2D paths from a start location to a goal location and (b) to three pure intentions and multiple possible mixed intentions:

- **quickly**: preferred paths minimize path length.
- **stealthily**: preferred paths minimize path length in enemy sensor range.
- **safety**: preferred paths maximize cumulative distance from obstacles and world boundaries.
- **mixed**: preferred paths blend objectives such as ‘go stealthily and quickly’.

Given these three objectives, \( \mathbf{I}_H = [I_1, I_2, I_3] \): 1 indicates stealth, 2 indicates speed, and 3 indicates safety.

Consider again the example in Figure 6.1, which shows an example map with four possible paths running from the top left corner to the bottom right corner. Suppose that a human indicates that the stealth is very important, speed is only slightly important, and safety is somewhat important. This verbal expression of intent can be encoded as the intent vector \( \mathbf{I}_H = [0.9, 0.1, 0.2] \). In the example, the brown path labeled 1 best matches the given intent from among the four paths because the path keeps away from the enemy. Recall that this paper does not address the way that verbal intentions are translated into the numerical intent vector; this mapping is done by the authors prior to the experiment.

In addition to the example in Figure 6.1, it is useful to demonstrate how the human intent vector can be represented in the multi-objective space. Consider again Figure 6.2 and recall that there are two objectives for the problem. Suppose that the human’s intent can be expressed verbally as “speed is very important, stealth is somewhat important, and safety is not important at all”. Since safety is not important, the figure ignores that dimension. Objective 1 corresponds to stealth and objective 2 corresponds to speed, so the human’s intent vector is subjectively represented as \( \mathbf{I}_H \)\][0.1, 0.9]. The vector begins at the origin and terminates at the location indicated by \( I_H \) in the figure; the circle is larger than for the paths to help the reader differentiate between paths and intent.

### 6.3.3 Intent-Matching Metrics

As mentioned in the introduction, prior work evaluated the TOPSIS and WPM multi-objective blending criteria [9], as well as Euclidean distance and cosine similarity for finding path vectors that matched the human intent vector [143]. Cosine similarity either produced better intent
matches or produced equivalent matches with greater computational efficiency than the other metrics.

Cosine similarity is the angle between the intent vector and the path payoff vector. If the path vector aligned perfectly with the intent vector, that is, the angle is 0, then $cos \ 0$ yields a maximum similarity of 1. This method of checking similarity between the intent vector and the payoff vector works well if all the elements in the intent vector are close in value to all the corresponding values in the payoff vector. The example in Figure 6.2 shows that the angle between $p(s1)$ and $I_H$ is smaller than both the angle between $p(s2)$ and $I_H$ as well as the angle between $p(sx)$ and $I_H$. Thus, the cosine similarity metric would select path s1 as the path that most closely matches intent.

Now, rather than interpreting the vectors in the example from Figure 6.2 as payoffs for different paths, consider a problem where objectives vary over time. Time-varying objectives are important because they mean that a path that once matched intent may not always match intent. For example, suppose that enemies can move, changing the value of the stealthy objectives. Suppose that a path $S^*$ was planned that perfectly satisfied the intent. The vector for this path coincides with the intent vector $I_H$. Now suppose that while the robot is following its path the enemies gradually move away from the path so that the planned path decreasingly intersects with the enemies sensor region. After following the path until time $t = 1$, the payoff objective for the planned path $S^*$ is represented by $s1$; objective 2 (speed) increased a bit because there is some noise in its estimate, but objective 1 (stealth) has increased quite a bit because the enemies are moving away from the planned path. The robot continues to follow its path and enemies continue to move away. The vector $s2$ represents the payoff vector at time $t = 2$ and indicates that the payoffs for the path are becoming more favorable for the agent. Finally, at some time in the future, time $t = x$, the planned path has a payoff vector indicated by $sx$, which is still a very fast path but has also become a very stealthy path.
This example calls attention to the problem with cosine similarity. Because the angle between intent and the payoff vector is increasing (because the path is becoming more stealthy as enemies move away), at some point the angle becomes so high that the cosine similarity metric indicates that the path no longer matches what the human intends. At that point, the robot begins to replan even though there is no need to do so.

Prior work did not identify this limitation of the cosine similarity metric because the problems were constructed such that there were always paths with payoff vectors that were distributed across the Pareto front. Thus, there was always a path that had a small angle between its payoff vector and the intent vector. This limitation of the cosine similarity metric calls for another approach to measuring similarity between the intent vector and a path’s payoff vector.

### 6.4 Intent Threshold Margin

Essentially, the problem with cosine similarity is that it evaluated paths by how similar they were to the intent vector. The intent threshold margin doesn’t seek to find how similar a path is to intent, but rather how much of the intent has to be sacrificed before a path becomes satisficing. This section first presents the definition of the intent threshold margin and then shows how it can be used to rank paths relative to the human intent vector.

#### 6.4.1 Definition

Consider a subset of paths for which each payoff variable is within an $\varepsilon$-threshold of the corresponding variable in $I_H$:

$$
\mathcal{T} = \{ S_i \in \mathcal{S} \mid \forall_k (p_k(S_i) \geq (I_k - \varepsilon_k)) \}
$$

(6.1)

The epsilon threshold $\varepsilon_k$ represents a margin, similar to Simon’s satisficing aspiration levels [149], by which an intent criterion may be relaxed for an objective $k$. We call this
relaxed margin the *intent threshold margin* (INThresh), which is represented by the vector \( \mathcal{E} = [\varepsilon_1, \ldots, \varepsilon_K] \). Thus, the region specified by \( \mathbb{T} \) is a function of the threshold vector, \( \mathcal{E} \), so we can make this dependence explicit by writing \( \mathbb{T}(\mathcal{E}) \).

![Figure 6.3: Paths that make to intent-satisfying-region.](image)

The *degree to which a solution satisfies* the given intent \( I_H \) depends on the smallest values in the epsilon vector \( \mathcal{E} = [\varepsilon_1, \ldots, \varepsilon_K] \) for which the solution is satisficing. Consider Figure 6.3, which, for illustration purposes, assumes \( K = 2 \) and two notional objectives \( o_1 \) and \( o_2 \) (e.g., stealth and speed). Each small (blue or orange) circle in the 2D plane represents the payoff vector for a possible path, placed in the figure according to the normalized payoff vector. Suppose that the human intent vector \( I_H \) is given by \([0.4, 0.7]\); 0.4 for objective \( o_1 \) and 0.7 for objective \( o_2 \) respectively, expressing intent as preference for paths that favor objective \( o_2 \) more than objective \( o_1 \). This example intent is shown as the (larger) solid violet circle at the intersection of the blue dashed lines.
The solid circle at this intersection of the two dashed lines indicates that the payoffs of a desired solution should lie in the region to the right and above the intersection. Thus, the darker gray region to the right and above the intersection corresponds to $T(0)$ since $E = 0$. In other words, the dark gray *intent-satisfying-region* satisfies Equation 6.1 with $\varepsilon_k = 0$ for both objectives $o_1$ and $o_2$. In the figure, the topmost three small blue circles that lie in the dark shaded region all correspond to paths that satisfy intent. That is, the system is able to find a set of three paths with no epsilon relaxed because each of these solutions exceeds the thresholds.

Now suppose that the three paths in the dark shaded region do not exist, indicating that there are no solutions that satisfy the given objectives specified by $I_H$. $\text{INTHRESH}$ relaxes the values in the intent vector by factor of $\varepsilon_k = \varepsilon$ for both objectives. The value of $\varepsilon_k$ is gradually increased over time. Referring to Figure 6.3 again, the relaxation by $\varepsilon_k$ causes the intent-satisfying-region to grow a little towards the left and the down, yielding the light gray region. Let $E' = [\varepsilon, \varepsilon]$. The lighter gray region is $\overline{T(E')}$. Given the new values of $\varepsilon_k > 0$, two solutions, indicated by the small orange circles make an entry to the set $\overline{T(E')}$.  

### 6.4.2 Ranking Solutions

The intent threshold margin can be used to rank paths by how much must be given up before a path becomes satisficing. Formally, we say that a solution $S$ is satisficing given a threshold vector $E$ if $S \in \overline{T(E)}$. For an intent $I_H$, a solution $S_x \in S$ is *superior to* a solution $S_y \in S$ if each of the elements of $E_x = [\varepsilon_{x_1}, \ldots, \varepsilon_{x_K}]$ needed to make $S_x$ satisficing is less than the corresponding counterparts in $E_y = [\varepsilon_{y_1}, \ldots, \varepsilon_{y_K}]$ needed to make $S_y$ satisficing. That is, if $\forall k \in K, \varepsilon_{x_k} < \varepsilon_{y_k}$ then solutions in $\overline{T(E_x)}$ are ranked higher than solutions in $\overline{T(E_y)} \setminus \overline{T(E_x)}$.

For example, each of the solutions in the darker shaded region, $\overline{T(E_x)}$, in Figure 6.3 is superior to the ones lying in the lighter shaded region, which is the set difference between the paths above the first set of dashed lines and the lower set of dashed lines, $\overline{T(E_y)} \setminus \overline{T(E_x)}$. 

123
Given an intent vector, $\mathbf{I}_H$, solutions can be ranked by how much has to be sacrificed, that is, how big $\varepsilon$ must become, before a solution becomes satisficing. Algorithm 1 produces a set $\mathcal{E} = \{\mathcal{E}_0, \mathcal{E}_1, \ldots, \mathcal{E}_m\}$, which is an indexed set of $m$ intent threshold margins.

**Init:** $\mathcal{E}_0 = [0, 0, \ldots, 0]$, a vector of all zeroes;
$\mathbb{R} = \mathbb{S}$, residual set;
$\varepsilon_k = 0$;
$r = 1$;
while $\mathbb{R} \neq \emptyset$ do
  $\mathcal{E}_r = [\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_K]$;
  compute $\mathbb{T}(\mathcal{E}_r)$;
  if $\mathbb{T}(\mathcal{E}_r) - \mathbb{T}(\mathcal{E}_{r-1}) \neq \emptyset$ then
    $\mathbb{R} = \mathbb{S} - \mathbb{T}(\mathcal{E}_r)$, update residual set;
    $r \leftarrow r + 1$;
  end
  $\varepsilon_k \leftarrow \varepsilon_k + \delta_k$, for all $k$ objectives;
end

**Algorithm 1:** Partitioning Solutions.

Algorithm 1 iteratively increases the value of the threshold variable $\varepsilon$ beginning at zero, and uses this value to construct a vector of thresholds $\mathcal{E}_r$. The amount that $\varepsilon$ changes is given by $\delta \ll 1$, which is a small value that slowly lowers the threshold at which solutions become satisficing. For each threshold vector, the set of solutions that are satisficing is computed, $\mathbb{T}(\mathcal{E}_r)$. When the threshold is lowered enough (that is, when $\varepsilon$ is high enough) so that a new solution becomes satisficing, the threshold vector $\mathcal{E}_r$ is stored, the iterator value $r$ is increased, and the residual set $\mathbb{R}$ is computed. The residual set consists of all those solutions that are not yet within the satisficing region for the given value of $\varepsilon$, so when the residual set is empty then all solutions have been partitioned into a satisficing set.

The index of the intent threshold margin set determines the extent to which its associated solution(s) satisfies intent. The index gives a rank to a solution for a given human command represented as an intent vector. The solution(s) at index 0 all exceed the human’s intent. Solution(s) of rank 1 is/are the top ranked solution(s) among those for which at least one part of the human’s intent must be relaxed. And the solution(s) at index $m$ has the last rank. Every element in $\mathbb{S}$ is associated with a single index in $\mathcal{E}$, so every path is ranked.
In the algorithm shown, the $\delta_k$ parameter is a small percentage of the intent parameter values specified in $I_H$. In other words, thresholds are relaxed in proportion to the magnitude of their weight in the human intent vector. For the experiments in this paper, $\delta_k$ was defined as $\delta_k = (p/100) \times I_k$ and $p = 15$ was used.

### 6.5 Evaluations

We conducted an Amazon Mechanical Turk© (MTurk) study with 50 participants in order to assess the intent threshold margin metric. The goal of the study was to evaluate whether the rankings induced from the intent threshold margin correlated with rankings from MTurk participants. Similarly, paths ranked low by participants should have higher indices in $E$.

Prior to the study, an indexed set of 14 configuration maps, $C = \{C_1, \ldots, C_{14}\}$ was produced, showing the robot’s start location and goal location, the obstacles, and enemy positions. Five of the configurations were used for training, and the remaining nine were used for evaluation, presented to participants in a counterbalanced way. For each configuration, nine paths were planned using an Online-FMT* algorithm presented in [30] using different weights that were uniformly selected from among the three objectives. The different weights created paths that were all in the Pareto set, meaning that no path in was payoff dominated by any other path.

An intent was specified as an English sentence for each configuration. The intent was either a single objective such as ‘go quickly’, or a multi-objective intent such as ‘go stealthily or safely’. The English sentences were designed to either put all weight on a single objective or to give preference for two objectives. Thus, each command resulted in a unique intent vector $I_H$. The intent threshold algorithm was then applied to the nine paths and intent vector to produce the set $E$.

From the set $E$, four paths that corresponded to different indices within $E$ were selected. These four paths included one path from the lowest index (“best” path), one from the highest index (“worst” path), one from roughly the second quartile, and one roughly from the third
quartile. Please refer back to Figure 6.1 that shows four paths for an example configuration. Thus, each configuration had a set of four paths that were designed to match a specified intent to varying degrees. An IRB-approved pilot study was conducted to determine whether there was sufficient differences in the paths to justify a complete study.

The full study preceded with participants giving informed consent as approved by the university’s IRB office. Each participant received $3 as compensation. Next, participants were trained using (a) illustrations of the configuration, (b) illustrations of the four paths per configuration, and (c) definitions of the path-planning objectives.

MTurk was set up to show participants the path-planning configuration (e.g., robot, goals, enemies, obstacles) and the four paths selected for that configuration; Figure 6.1 is an image from one of the configurations. Each participant was asked to rank 4 paths from 1 to 4, with 1 indicating the path that best matched the specified intent. 5 configurations were used for training, and the data collected from remaining 9 configurations was used for analysis.

Responses from 47 participants out of the 50 who participated were included in the data analysis. We discarded the results of two participants because they took fewer than 5 minutes required to respond to the survey, indicating that they did not seriously consider each configuration; the median completion time was 14.4 minutes. Further, a technical glitch caused MTurk data to be lost for one participant.

Before proceeding, please note that the term rank is used to indicate the ordinal value assigned to a particular path in a particular configuration; ranks are values in the set \{1, 2, 3, 4\}. Further, the term ranking is used to indicate the ordering of the set, that is, to indicate the vector of ranks. For example, the hypothetical ranking is always the vector \[1, 2, 3, 4\] for path 1, path 2, etc; but participants might not rank each path the same, so a ranking for a particular participant might be the vector \[1, 3, 2, 4\] indicating that the participant swapped the ranks of the second and third paths.
6.5.1 Hypotheses

We hypothesize the following:\(^1\):

- **Hypothesis 1**: For each path in each configuration, there will be no significant difference between the ranks from participants and the rank induced by the intent threshold distance.

- **Hypothesis 2**: For each configuration, the ranking of the four paths from the participants will be strongly and positively correlated with the ranking of the four paths induced by the intent threshold distance.

- **Hypothesis 3**: The smallest value of \( \varepsilon \) for which the path is part of \( E \), which is the minimum intent threshold distance, will positively correlate with ranks from participants; small values of epsilon correspond to top ranks (e.g, rank 1), and high values correspond to poor ranks (e.g., rank 4).

6.5.2 Data

The following data were gathered for each configuration path:

- **Hypothetical rank, \( R_i \)**: a rank between 1 to 4 (inclusive) obtained from Algorithm 1 and ordered from the set \( E \), with 1 indicating the path with the smallest intent threshold distance.

- **User rank, \( R_{us} \)**: a rank between 1 and 4 (inclusive) as selected by the participant.

- **Intent threshold distance**: the smallest value of \( \varepsilon \) for which the path is part of \( E \).

The data used for the *intent threshold distance* needs to be explained. Recall from Algorithm 1 that each path produces a vector of epsilon values, one for each objective (stealth, safety, speed). This vector was denoted as \( E = [\varepsilon_1, \ldots, \varepsilon_K] \), where \( K \) indicated the number of objectives. If a particular objective is not part of an intent, there is no need to find a value of \( \varepsilon \) for that objective.

\(^1\)Unfortunately, we did not register our hypotheses before the experiment via the Center for Open Science, cos.io/prereg/. We learned about registering hypotheses after the data was gathered.
This paper considers pure and mixed intents. For pure intents, there is only one $\varepsilon$ in $\mathcal{E}$, that is, $\mathcal{E}^{\text{pure}} = \varepsilon_i$ where $i \in \{1, 2, 3\}$ depending on which objective is chosen. Mixed intents considered only two of the three, so $\mathcal{E}^{\text{mix}} = [\varepsilon_i, \varepsilon_j]$ where $i, j \in \{1, 2, 3\}$ and $i \neq j$. For mixed intents, the epsilon value is defined as the maximum of the $\varepsilon$’s needed to make a path satisficing. Thus $\text{epsilon value} = \max[\varepsilon_i, \varepsilon_j]$. In other words, the worst case threshold is used to measure “how far” a path is from the intent.

### 6.6 Results and Discussion

#### 6.6.1 Comparing InThresh to Cosine Similarity

In static problems with solutions somewhat uniformly sampled from the Pareto front, cosine similarity was shown to be a useful metric [140]. Thus, for problems that satisfy these conditions, compatibility of rankings from InThresh and cosine similarity would provide evidence in support of the utility of InThresh. The four paths used in each experimental configuration satisfied the conditions. For all but one configuration, the ranks of InThresh and cosine similarity were identical, and in the other configuration the two rankings switched second and third ranked paths.

#### 6.6.2 Difference Between Hypothesized and Participant Ranks

Consider hypothesis 1. Recall that there are nine configurations $\mathcal{C} = \{C_1, \ldots, C_9\}$ and that for each configuration there are ranks produced by each participant and the ranks induced by the epsilon threshold distance. Table 6.1 shows the hypothetical ranks and the mean (median) ranks across participants for each configuration.

It is obvious from Table 6.1 to see that there is a strong relationship between average user rank and the hypothetical ranks. The mean for the hypothetically best path is higher than one because some participants did not rank this path as the best path; similarly the mean rank for the hypothetically worst path is less than four.
Table 6.1: Mean (and median, in parentheses) user ranks for the 9 configurations compared to the hypothetical ranks. The † superscript by the configuration indicates a mixture intent of ‘quick and stealthy’, and the ‡ superscript indicates a mixture of ‘stealthy and safe’; intent for all other configurations are for single attributes.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Hypothetical Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$C_1^†$</td>
<td>1.3 (1)</td>
</tr>
<tr>
<td>$C_2^†$</td>
<td>1.23 (1)</td>
</tr>
<tr>
<td>$C_3$</td>
<td>1.19 (1)</td>
</tr>
<tr>
<td>$C_4$</td>
<td>1.28 (1)</td>
</tr>
<tr>
<td>$C_5^‡$</td>
<td>1.49 (1)</td>
</tr>
<tr>
<td>$C_6^‡$</td>
<td>1.23 (1)</td>
</tr>
<tr>
<td>$C_7$</td>
<td>1.15 (1)</td>
</tr>
<tr>
<td>$C_8$</td>
<td>1.0 (1)</td>
</tr>
<tr>
<td>$C_9$</td>
<td>1.4 (1)</td>
</tr>
</tbody>
</table>

Note that the median rank for all participants for the hypothetically best path is always 1. By contrast, the median rank across participants and the hypothetical ranks does not always agree. One possible explanation is that it might be easier for participants to determine when a path best matches intent than to determine to what degree a path differs from an intent; future work should explore this explanation.

Figure 6.4 gives a box-and-whiskers plot of the hypothetical ranks and average user ranks. The basis for this plot is the average user ranks across configurations. The mean is indicated by the diamond in the box, the median by the horizontal line in the box, the shaded region of the box indicates the range between the first and fourth quartile, the whiskers indicate the span of the 90th percentile, and the circles indicate outliers. For example, the circle above hypothetical rank 1 comes from configuration $C_5$ whose mean rank is 1.49, the circle below hypothetical rank 1 comes from configuration $C_8$ whose mean rank is 1.0, and the circle below hypothetical rank 4 is from configuration $C_1$ whose mean rank is 2.55.

Hypothesis 1 can be evaluated from two perspectives: first, that each participant’s ranks should match the hypothetical ranks, and second, that the average ranks across participants match the hypothetical ranks.
Do Individual Ranks Match Hypothetical Ranks?

We performed a double sided t-test (n=47) in SAS with pseudo Bonferroni correction on the difference between user rankings and hypothetical rankings to test whether the differences were statistically different at a level of $p=0.001$.

Table 6.2 shows the outcomes with significant differences marked with an *. Results are that 22 out of 36 average responses are not significantly different than the hypothetical ranking. This provides evidence in support of hypothesis 1, but there are some individual participant ranks which are obviously different from the hypothetical ranks. Thus, considering only individual participants provides marginal support in favor of hypothesis 1.

Do Average Ranks Match Hypothetical Ranks?

For each path in a configuration, we computed the average rank across the 47 participants. This gives $4 \times 9 = 36$ different average ranks. Using a Pearson Correlation, we computed the
Table 6.2: 22 out of 36 responses indicate that the user ranks were not significantly different from the hypothetical ranks.

<table>
<thead>
<tr>
<th>C</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>0.0032</td>
<td>0.0011</td>
<td>&lt;.001*</td>
<td>&lt;.001*</td>
</tr>
<tr>
<td>$C_2$</td>
<td>0.0147</td>
<td>0.9</td>
<td>0.0375</td>
<td>0.0375</td>
</tr>
<tr>
<td>$C_3$</td>
<td>0.05</td>
<td>0.103</td>
<td>0.005</td>
<td>&lt;.001*</td>
</tr>
<tr>
<td>$C_4$</td>
<td>0.022</td>
<td>0.57</td>
<td>0.29</td>
<td>0.044</td>
</tr>
<tr>
<td>$C_5$</td>
<td>&lt;.001*</td>
<td>&lt;.001*</td>
<td>0.011</td>
<td>&lt;.001*</td>
</tr>
<tr>
<td>$C_6$</td>
<td>0.02</td>
<td>&lt;.001*</td>
<td>0.71</td>
<td>&lt;.001*</td>
</tr>
<tr>
<td>$C_7$</td>
<td>0.11</td>
<td>&lt;.001*</td>
<td>0.027</td>
<td>&lt;.001*</td>
</tr>
<tr>
<td>$C_8$</td>
<td>.</td>
<td>&lt;.001*</td>
<td>&lt;.001*</td>
<td>0.0011</td>
</tr>
<tr>
<td>$C_9$</td>
<td>&lt;.001*</td>
<td>0.55</td>
<td>0.002</td>
<td>&lt;.901*</td>
</tr>
</tbody>
</table>

correlation between the average ranks and the hypothetical ranks. The correlation value is $\rho = 0.9118$ (n=36) with a $p$-value of less than 0.001.

The high correlation value indicates that the hypothetical ranks and average ranks across participants are strongly and positively correlated. This provides strong evidence for Hypothesis 1. In other words, if ranks are obtained by a group and then averaged, the results correlate strongly with the ranks induced by the epsilon threshold margin even though some participants differ from the hypothetical ranks.

### 6.6.3 Spearman’s Rho for User Ranking

Hypothesis 2 says that the hypothetical rankings are highly, positively correlated with participant rankings. Recall that the term *ranking* means the vector of ranks. Hypothesis 2 is evaluated using Spearman’s rank correlation coefficient.

Spearman’s rank correlation coefficient provides information on the strength and direction of relationship between two ranked variables. Recall that for each configuration we have the individual user rankings and the ranking induced by $\text{InThresh}$. For each of the nine configurations and for every participant’s ranking, we computed the Spearman’s rank correlation coefficient. Thus we computed 423, that is, $47 \times 9$ coefficients.

Figure 6.5 shows the distribution analysis of rhos for all the rankings of all the 47 participants. Notice that the mean is of 0.707 indicating a strong positive association between
the hypothetical ranking and the user ranking. Notice further that the majority of the rank correlation coefficients exceeded 0.5, showing a positive correlation between individual user ranking and the metric ranking. Negative values show a negative association, and such values rarely occurred in our analysis. Although evaluating the distribution of the coefficients uses only descriptive statistics, the distribution provides good support for Hypothesis 2.

6.6.4 Correlation Between Intent Threshold Margin and User Ranks

Hypothesis 3 asserts a positive relationship between the intent threshold margin and the ranks from the participants. Recall that the epsilon value associated with a path is obtained from the $L_1$ norm of the minimum values of $\varepsilon$ required to make a path satisficing.
Figure 6.6: Correlation between epsilon distances and average user ranks. Pearson’s Correlation coefficient 0.823, p-value < 0.001.

Figure 6.6 shows a strong positive correlation between the epsilon values and the user ranks. The Pearson coefficient is 0.823 (n=36, p < 0.001, $R^2 = 0.68$). The trendline gives an idea of the fit to the data. The $R^2$ value gives some confidence that the relationship between rank and epsilon values are linear, and the p-value gives strong confidence that there is a positive correlation between epsilon value and ranks. Thus, we conclude that there is evidence in support of Hypothesis 3.

6.7 Summary and Future Work

This paper proposes the intent threshold margin as a measure of how well a solution to a multi-objective problem matches human intent. The need for a new measure was motivated by a limitation in the cosine similarity metric that had proven very useful in prior work. Cosine similarity gave preference to paths that produced small angles between an intent vector and a payoff vector. By contrast, the intent threshold margin gave preference to
paths that required less sacrifice from the ideal intent to become satisficing. Results from a Mechanical Turk study indicate that the rankings induced by the intent threshold margin are strongly correlated with human rankings in a three-objective problem; the problem was planning a path from a starting location to a goal location in a 2D world.

Future work should address four important problems. First, although the intent threshold margin measure was motivated by a special need in dynamic worlds, the results in this paper are restricted to static worlds. The measure should be used in dynamic world to detect when a path ceases to match intent as the world changes, and a user study should be performed to see if these detections match human expectations. Second, the measure was only applied to a 2D path-planning problem. Work should be done to explore the generalizability of the approach to, for example, planning manipulator trajectories or ranking heterogeneous problem with respect to a multi-objective robot assignment problem. Third, the analyses in this paper compared the way the rankings induced by the measure compared to average user ranking, where averages were either across users or across multiple configurations. Future work should explore how individual differences might affect representation and perception of rank in a multi-objective problem. Finally, analyses only considered mixed intents with two objectives. Future work should evaluate how humans rank solutions when there are four or more objectives, and explore whether the intent threshold margin would correlate to human rankings on these more complicated problems.
Chapter 7

Path Replanning in a Dynamic Environment using Intent Threshold Margin

*Under review

Abstract

This chapter makes the assumption that intent is invariant across the lifespan of a goal, and then uses this assumption to evaluate robot path replanning. Specifically, the chapter applies the cosine similarity and intent threshold margin metrics towards path replanning in dynamic environments. An important use of these metrics is determining when the situation in the world has changed so much that the original plan no longer satisfies the problem holder’s intent. Both CS and ITM are used to detect intent mismatches when the robot starts executing a path but costs change while the robot is moving along the path. Experiment results conducted through Amazon Mechanical Turk indicates that ITM has the potential to detect intent mismatches of a current path so that the current path must be replanned. Furthermore, the experiment results provide evidence that supports a theoretical limitation with using CS to trigger replanning. Data from the study include no replanning as a control condition.

7.1 Introduction

Through Chapter 5 of this dissertation, intent was represented as a tradeoff in a multi-objective planning problem. For a three-objective path planning problem, intent was expressed as a three-dimensional vector, \( \mathbf{I} = [I_1, I_2, I_3]^T \) on a 2D probability simplex embedded in 3D-space; each \( I_k \in [0, 1] \) and \( \sum_{k=1}^{3} I_k = 1 \). For example, an intent vector of \( \mathbf{I} = [.2, .5, .3]^T \) indicates
that the first objective has weight of 20%, the second objective has weight of 50%, and the third objective has weight of 30%. Experiments with human participants indicated that it is possible for humans to interpret balancing tradeoffs as percentages allocated to each objective.

Chapter 6 removed the constraint that objectives must be interpreted as percentages; each $I_k \in [0, 1]$ but $\sum_{i=1}^{3} I_i \neq 1$. The previous chapter also proposed a new intent-matching metric, the *intent threshold margin* (ITM), and showed that it can be used to rank paths that correlate with how humans rank paths.

This chapter evaluates whether the ITM and *cosine similarity* (CS) metrics can be used to identify when a robot needs to replan its path given the intent representation from Chapter 6. Replanning might need to occur if the path costs change while the robot is following the path. More specifically, replanning may need to occur if enemies move in the environment in a way that makes a path either more or less stealthy. The CS metric has a theoretical limitation when used in replanning, which is discussed in the next section.

Using either the CS or ITM metric to trigger replanning requires reevaluating the costs of following a plan. Reevaluating costs, in turn, presents a challenge for normalizing the costs in such a way that they can be compared to human intent. A key insight presented in this chapter is that re-normalizing can be done using only paths that satisfy *pure intentions*, meaning weights of the form $[1, 0, 0], [0, 1, 0]$, and $[0, 0, 1]$. These pure intentions guarantee that there are three paths $\sigma_{i}^{*} \in \Sigma, i \in \{1, 2, 3\}$ that are optimal for each objective; path $\sigma_{i}^{*} \in \Sigma$ is the optimal path with respect to objective $o_i$. The so-called pure intentions effectually create the corners of the Pareto space, and finding these corners is the key to re-normalization.
7.2 CS Metric Limitations

This section identifies situations where the CS metric (a) can fail to detect when the current path no longer satisfies intent or (b) can erroneously indicate that the current path fails to satisfy intent when the path actually does satisfy intent.

7.2.1 Pure Intent

Consider a scenario where an agent cares only about a pure intent. Suppose without loss of generality that the pure intent places all weight on the stealthy objective. Let the corresponding intent vector be given by $\mathbf{I} = [0, 1, 0]^T$, which means that the stealth objective is the second element in vector. Suppose further that a set of path planners have been run that produce four paths on the Pareto front, denoted path $w$, $x$, $y$ and $z$, each of which has high payoffs for objective $o_2$. The payoff vector for each path and the cosine similarity value for each vector is shown in Table 7.1.

<table>
<thead>
<tr>
<th>Path</th>
<th>Payoffs</th>
<th>CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w$</td>
<td>$o_1$ 0.2 $o_2$ 0.95 $o_3$ 0.5</td>
<td>0.87</td>
</tr>
<tr>
<td>$x$</td>
<td>$o_1$ 0.26 $o_2$ 0.94 $o_3$ 0.32</td>
<td><strong>0.92</strong></td>
</tr>
<tr>
<td>$y$</td>
<td>$o_1$ 0.76 $o_2$ 0.94 $o_3$ 0.31</td>
<td>0.75</td>
</tr>
<tr>
<td>$z$</td>
<td>$o_1$ 0.24 $o_2$ 1 $o_3$ 0.7</td>
<td>0.80</td>
</tr>
</tbody>
</table>

Table 7.1: Cosine similarity values for various paths relative to the pure intent $\mathbf{I} = [0, 1, 0]^T$.

Path $x$ has the highest CS value. Note that having high payoffs of objective $o_1$ and objective $o_3$ can negatively impact the cosine similarity value for this intent vector. For example, compare the payoffs for path $x$ and $y$. Both paths have the same payoff for objective $o_2$ and very similar payoffs for objective $o_3$. Path $y$ has a payoff of 0.76 for $o_1$, which is higher than path $x$’s payoff of 0.26. Even though path $y$ produces nearly identical payoffs for two objectives and much higher payoff for the third objective, the cosine
similarity for \( y \) is much lower than the value for path \( x \). The table entries for paths \( w \) and \( z \) illustrate similar problems.

### 7.2.2 Mixed Intent

The CS limitation can also be illustrated for a mixed intent preference. Suppose that the intent vector is \( \mathbf{I} = [0.01, 0.68, 0.73]^T \), which indicates a preference for a safe (the third vector element) and stealthy (the second vector element) path for the robot to navigate to the goal. Table 7.2 shows the CS values for a path \( t_1 \), the CS value for the same path at time \( t_2 \) where \( t_2 > t_1 \) when obstacles and enemies have moved in the world, and the CS value for the same path at the same time but obstacles and enemies have moved in the world in a different way; the latter situation is indicated by time \( t'_2 \), where \( t'_2 > t_1 \).

<table>
<thead>
<tr>
<th>Time</th>
<th>Payoffs</th>
<th>CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>( a_1 ) 0.25, ( a_2 ) 0.95, ( a_3 ) 0.99</td>
<td><strong>0.99</strong></td>
</tr>
<tr>
<td>( t_2 )</td>
<td>( a_1 ) 0.75, ( a_2 ) 0.95, ( a_3 ) 0.99</td>
<td>0.88</td>
</tr>
<tr>
<td>( t'_2 )</td>
<td>( a_1 ) 0, ( a_2 ) 0.02, ( a_3 ) 0.32</td>
<td>0.96</td>
</tr>
</tbody>
</table>

Table 7.2: Cosine similarity values for various paths relative to the mixed intent \( \mathbf{I} = [0.01, 0.68, 0.73]^T \).

Recall that replanning can be triggered if CS metric shows a difference between the path’s payoff and the intent vector that is is less than a threshold value, \( c_{\text{threshold}} \); see Section 4.6.2, and recall that high values (near one) of the CS metric are better low values (near zero) because high values indicate alignment of the intent and payoff vectors. Suppose that \( c_{\text{threshold}} = 0.9 \). The following paragraphs discuss two cases when the CS metric would fail to detect intent failures correctly. When the path is originally planned, which occurred at time \( t_1 \), the CS metric is 0.99, indicating that the path aligns with the intent.

**False Positive** Suppose that the robot moved into a situation at time \( t_2 \) where the path payoffs produce an objective vector of \([0.75, 0.82, 0.85]^T\), which is shown in the middle row
of Table 7.2. The CS metric is 0.88 indicating that the current path no longer matches the
given intent $I = [0.01, 0.68, 0.73]^T$. This triggers a replanning event even though the payoffs
for the second and third objective stay the same as at time $t_1$, but the payoff for the first
objective has increased. In other words, the situation in the second row is at least as good
as the situation in the first row. A false positive occurs when the CS metric erroneously
indicates that the path does not align with intent.

Replanning triggered using CS is vulnerable to such false positives. In fact, the reason
the ITM metric was created was because of a situation frequently encountered near the end
of the time the robot was following the path. In some situations, the enemies would move
away from the goal as the robot approached the goal. Even if the robot was seeking to satisfy
only a safety and stealth mixed intent, when the enemies moved from the goal the payoff for
speed went up and triggered replanning.

**False Negative** Instead of the situation changing in a way that increases the payoff of one
objective as in the second row of Table 7.2, suppose that obstacles and enemies move in a
way to make the path unsafe and not stealthy. Suppose further that the robot discovers that
the path leads through a swamp that dramatically decreases the speed at which the goal
is reached. The third row in Table 7.2 illustrates the very low payoffs that result for this
situation.

The CS measure results in a cosine similarity of 0.96 indicating that the current path
satisfies the given intent $I = [0.01, 0.68, 0.73]^T$. The CS measure is above the threshold level of
0.9, indicating that the robot should continue on its path. However, the payoffs for both the
second and third objective are really low. From a subjective point of view, replanning should
be considered in this situation even though the CS values have not changed. Replanning
triggers missed because of such false negatives may cause the robot to continue on a path
that does not satisfy the problem holder’s intent.
Cosine similarity triggers replanning if similarity is less than a threshold, indicating that the path no longer aligns with intent. By contrast, the ITM metric is designed to trigger replanning if the “distance” between the intent vector and the payoffs for the path exceeds some threshold vector, which indicates that the intent and payoff vector no longer align.

To understand how ITM triggers replanning, we need some new notation. Suppose that each path, $\sigma$, is parameterized by a “time” variable $s \in [0, 1]$. Let $\sigma_s$ denote the location along the path when “time” is $s$. The value of $s = 0$ occurs when the agent is at the starting position on the path, $\sigma_0$, and the value of $s = 1$ occurs when the agent is at the goal location on the path, $\sigma_1$. As the agent follows the path, the value of $s$ changes continuously to indicate progress along the path. The path segment for an interval of time, say from $s'$ to $s''$, is denoted by $\sigma_{[s',s'']}$. 

Suppose that a path $\sigma$ is selected to be stealthy with intent vector $I = [0, 1, 0]^T$ and a payoff vector of $[0, 1, 0]^T$. As the agent moves along the path, $0 < s < 1$, the enemies might also move in the world. Suppose that at time $s = \frac{1}{2}$ one of the enemies moves to intersect the portion of the path yet to be traversed. This means that the remaining path, the segment $\sigma_{[\frac{1}{2},1]}$ from time $s = \frac{1}{2}$ to $s = 1$, is no longer stealthy. At time $s = \frac{1}{2}$, the world switched from a configuration where the path was stealthy to a configuration where the path was no longer stealthy.

Figure 7.1: Decision scenario to replan.
Figure 7.1 illustrates the notation. The dotted line from the green flag in the upper left to the robot avatar is the path segment that has already been traversed, $\sigma_{[0:s]}$. The dashed line from the robot avatar to the goal represents the portion of the planned path that has not been traversed, $\sigma_{[s:1]}$. The solid line represents a replanned path from the robot’s location $\sigma_s$ to the goal that was triggered when the ITM metric indicated that the costs of $\sigma_{[s:1]}$ no longer aligned with intent because two enemies had moved to intersect the original path.

The portion of the path that has already been traversed should not affect the decision about whether the remaining path satisfies intent. To illustrate this point, suppose that the agent was beginning the planning process precisely in the configuration where the robot is at location $\sigma_s$ but the enemies are all lined up along path segment $\sigma_{[s:1]}$. If the robot was beginning the planning process in this configuration, surely it would choose a path different from the original path $\sigma_{[s:1]}$. This thought exercise tells us that the portion of the original path from $s$ to $1$, which is the path segment denoted by $\sigma_{[s:1]}$, does not match intent regardless of what happened while the robot traversed the path segment $\sigma_{[0:s]}$.

Suppose that the robot is at location $\sigma_s$, and suppose that the robot has time to plan ten paths from the current location $\sigma(s)$ to the goal. We can compute the unnormalized payoff vector for the path segment $\sigma_{[s:1]}$ as well as the unnormalized payoffs for each of the other of the ten paths. The normalized payoff vector, $\mathbf{o}(\cdot)$ for the path segment $\sigma_{[s:1]}$ can then be computed using the payoffs from the other ten paths. The robot will switch from the original planned path to one of the ten new paths if

$$d(I, \mathbf{o}(\sigma_{[s':1]})) \geq \theta,$$  \hspace{1cm} (7.1)

where $\theta$ is a threshold, where $d(\cdot, \cdot)$ is the ITM distance. Simply put, if the normalized payoffs for the remainder of the path deviate too far from the intended path, the robot needs to replan and switch to an appropriate new path.
The threshold vector $\theta$ used by the ITM metric is set subjectively and represents the human’s tolerance for deviations from his or her intent for each objective. Thus, $\theta$ has the form $[\varepsilon_1, \varepsilon_2, \varepsilon_3]$.

Example 1. Suppose the threshold $\theta = [0.01, 0.1, 0.01]$, and suppose that we are considering an intent vector of $I = [0, 1, 0]^T$ as in Table 7.3. The table shows the payoff vector and the elements of the vector $I - o(\sigma)$ for various points of time while the robot follows the path. Observe that each path segment in the table satisfies intent because each objective for each path is no greater than $\varepsilon_k$ from the intent vector. Indeed, unless the payoffs for objective 2 fall below 0.9 (that is, $I_2 - 0.1$), replanning will never be triggered. Observe that in this example no false positive occurs.

<table>
<thead>
<tr>
<th>Time</th>
<th>Payoffs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s = \frac{1}{5}$</td>
<td>$o_1$ $o_2$ $o_3$</td>
</tr>
<tr>
<td></td>
<td>0.2 0.95 0.5</td>
</tr>
<tr>
<td></td>
<td>$I - o(\sigma_{[\frac{1}{5}]}$)</td>
</tr>
<tr>
<td></td>
<td>$&lt; 0$ 0.05 $&lt; 0$</td>
</tr>
<tr>
<td>$s = \frac{2}{5}$</td>
<td>$o_1$ $o_2$ $o_3$</td>
</tr>
<tr>
<td></td>
<td>0.26 0.94 0.32</td>
</tr>
<tr>
<td></td>
<td>$I - o(\sigma_{[\frac{2}{5}]}$)</td>
</tr>
<tr>
<td></td>
<td>$&lt; 0$ 0.06 $&lt; 0$</td>
</tr>
<tr>
<td>$s = \frac{3}{5}$</td>
<td>$o_1$ $o_2$ $o_3$</td>
</tr>
<tr>
<td></td>
<td>0.76 0.94 0.31</td>
</tr>
<tr>
<td></td>
<td>$I - o(\sigma_{[\frac{3}{5}]}$)</td>
</tr>
<tr>
<td></td>
<td>$&lt; 0$ 0.06 $&lt; 0$</td>
</tr>
<tr>
<td>$s = \frac{4}{5}$</td>
<td>$o_1$ $o_2$ $o_3$</td>
</tr>
<tr>
<td></td>
<td>0.24 1 0.7</td>
</tr>
<tr>
<td></td>
<td>$I - o(\sigma_{[\frac{4}{5}]}$)</td>
</tr>
<tr>
<td></td>
<td>$&lt; 0$ 0 $&lt; 0$</td>
</tr>
</tbody>
</table>

Table 7.3: ITM differences for the payoff vector of a path at various times for the intent vector $I = [0, 1, 0]^T$. When $\theta = [0.01, 0.1, 0.01]$ no replanning is triggered.
Example 2. Similarly, suppose that the distance $I - o(\sigma)$ was computed for the various points of time in the situations in Table 7.2; no false positive occurs. For the second row or the table, the row identified by $t_2$, no replan is triggered for $t_2$ because each element of the difference vector is less than zero (and therefore less than the positive values in the threshold vector). However, a replan is triggered for $t'_2$ because the difference for objective $o_2$ and $o_3$ exceed the ITM threshold; no false negative occurs.

7.4 Normalizing When a Path is Being Followed

Recall from Chapter 5 that the normalized payoff for objective $o_j$ for path $\sigma$ is obtained using

$$o_j(\sigma) = \frac{p_j(\sigma) - \min_{\sigma' \in \Sigma} p_j(\sigma')}{\max_{\sigma' \in \Sigma} p_j(\sigma') - \min_{\sigma'' \in \Sigma} p_j(\sigma'')}$$

with a corresponding normalized payoff vector $o(\sigma) = [o_1(\sigma), o_2(\sigma), o_3(\sigma)]$. The normalization is determined by a set of possible paths $\Sigma$. In all previous chapters, $\Sigma$ was obtained using an optimizing planner over a range of objective weights. By changing the weight values, solutions are computed that correspond to multiple points on the Pareto front. Thus, normalization occurs relative the set of payoffs afforded for paths on the Pareto front.

The biggest challenge in computing the normalized payoff for a path segment $\sigma_{[s:1]}$ is that it is not possible to obtain samples from the Pareto front by planning ten new paths from point $\sigma_s$ to the goal in real-time for each point along the path. This challenge can be overcome by noting that the maximum and minimum values used to normalize $o_j(\sigma)$ occur at the “corners” of the Pareto front, and noting that the corners of the Pareto front are determined by the three pure intents quickly, stealthily, and safely. Thus, the normalized objective vector for the path segment $o(\sigma_{[s:1]})$ can be determined using only three paths.

The quickest path can be planned from any point $\sigma_s$ to the goal can be computed in near real-time in two simple steps. First, the visibility graph is created for the map. The visibility graph is static so it can be computed before the robot begins moving. Second,
the robot’s current location is added to the visibility graph and connected to the corners of obstacles visible to it. Dijkstra’s algorithm can then be run to find the cost of the shortest path from location $\sigma_s$ to the goal.

The safest path from any point $\sigma_s$ to the goal can also be approximated in near real-time in two simple steps. First, the Voronoi graph is computed for the obstacles in the world. Since the world is static, the Voronoi graph is also static and can be formed before the robot begins moving. Second, the robot’s current location is added as a node to the Voronoi graph and an edge is added from this node to the the nearest unobstructed path-segment of the Voronoi graph. Dijkstra’s algorithm can then be run to find the cost of the safest path from location $\sigma_s$ to the goal.

The most stealthy path is more difficult to compute in near real-time because it changes each time step. The naive approach requires doing an RRT* planner from the current point $\sigma_s$ to the goal using edge costs computed as the intersection of the enemy sensor regions with the path segments. This is slow. We approximated this process using OFMT* [30], which is an online, sampling-based planner that operates in real-time to adjust the path to dynamic path costs. OFMT* is continuously run in the background while the robot is moving along the path, and the cost of the new path provided by OFMT* is a close approximation to the cost of the most stealthy path.

These three paths are the “corners” of the Pareto front, so they are used to normalize the payoffs, yielding $o(\sigma_{s,1})$. This value can then be used by either CS and the ITM to trigger replanning.

Note that the algorithms above ran in near real-time but were still slow enough that they weren’t compatible with the time limits needed for the Mechanical Turk study presented in the next section. Rather than try to optimize each algorithm to get real time performance, we observed that the OFMT* algorithm ran faster than the visibility-graph and Voronoi-graph algorithms used to compute the quickest and safest paths, respectively. We therefore approximated the quickest and safest path using a second and third OFMT*
algorithm, which means that there were three versions of the OFMT* algorithm running, one for each objective. The execution speed allowed us to create videos that were based on real-time triggering and replanning.

![Figure 7.2: The 3 best paths used for normalization during robot walk.](image)

Figure 7.2 illustrates the three paths that are computed at some time step along the robot’s path using the OFMT* algorithm with a safety objective, speed objective, and stealth objective. The dotted line from the green flag in the upper left to the robot avatar represents the portion of the path that the robot has traversed, $\sigma_{[0:s]}$. The black path is the untraversed portion of the path, $\sigma_{[s:1]}$. The path in red is the quickest path from the robot’s current location $\sigma_s$ to the goal, obtained using the OFMT* algorithm with minimizing path length as the objective. The path in green is the stealthiest path from $\sigma_s$ to the goal, obtained using the OFMT* algorithm with minimizing exposure to enemy sensors as the objective. The path in blue is the safest path from $\sigma_s$ to the goal, obtained using the OFMT* algorithm with maximizing proximity to obstacles as the objective.

### 7.5 Human Factors Study

A Mechanical Turk study was performed to evaluate whether replanning triggers based on the CS and ITM metrics aligned with when humans believed that replanning should occur. A control condition was included, namely where the robot followed its path without replanning;
the control condition was called the No Metric condition and identified by NOM in the results.

7.5.1 Hypotheses

We propose the following hypotheses. Note that all hypotheses are made with respect to robot navigation in a dynamic, multi-objective environment with three objectives: speed, stealth, and safety.

- **Hypothesis 1:** ITM-based replanning triggers will be rated excellent by participants.

- **Hypothesis 2:** CS-based replanning triggers will be rated excellent by participants when no false positives or false negatives occur, but will be rated poorly when either a false positive or false negative occurs.

- **Hypothesis 3:** Participants will prefer ITM-based replanning triggers over CS-based triggers and CS-based triggers over no replanning triggers.

7.5.2 Methodology

We conducted an Amazon Mechanical Turk© (MTurk) study with 60 participants in order to assess CS-based, ITM-based, and NOM-based replanning triggers for robot navigation tasks. The purpose of the study was to evaluate which metrics helped the robot to replan when a path stopped aligning with human intent because enemies moved.

The study included a training session and evaluation sessions. The study was designed to take no more than 40 minutes per each participant. Each participant received $6 as compensation. After obtaining consent, participants were trained using written instructions, images, and videos about the tasks they would be performing. The study was limited to MTurk workers in the United States to mitigate language issues.

In the evaluation sessions, participants were told which command the robot should obey. Participants were then shown videos where the robot follows a path from the start to
the goal location and replanning was controlled by the replanning triggers. After watching each video, participants answered a question about how well the robot complied with the command. The participant’s answer was given using a five point Likert scale going from *Extremely good* to *Extremely bad*. Figure 7.3 shows a task example. The only data from the experiment were the scores from each participant for each video; *Extremely good* was set to a score of 1, and *Extremely bad* was set to a score of 5.

Each video represented either a pure intent travel or a mixed intent that involved only two objectives. The pure intent conditions used the stealth objective, and the mixed intent conditions used the stealth objective and the safety objective. The quick objective was not included in the experiment conditions. The stealth objective was always included because that was the only objective that varied over time; enemies moved but obstacles did not.

The study was a two-factorial study with factors being metric type (CS, ITM, or NOM) and intent type (pure or mixed).

To avoid results that were specific world configurations and patterns of enemy movement, a set of different “worlds” were created. In the *simple worlds*, the enemy moved in a deterministic way within a small region of the map. The movements of the enemies followed precomputed paths designed by the experimenter. The paths were designed to intersect with the robot’s initial planned path at specific instants of time. In the *complex worlds*, additional enemies were added and those enemies followed random paths that covered large regions of the world. There were a sufficient number of enemy agents that each complex world required at least one replan. We were not able to create scenarios where false positives or false negatives occurred for the ITM metric, but that doesn’t mean that such scenarios cannot be constructed.

The density of obstacles, the location of obstacles, the starting position of the robot, and the position of the flag were also varied. Four worlds were subjectively selected for use in the experiment: two simple worlds and two complex worlds.
When a trigger occurred, the OFMT* algorithm was used to compute a new path using a just the stealth cost for pure intents and using a weighted cost function for mixed intents.

Figure 7.3: Task example for participant evaluation.

Twenty-four videos were created by combining each world (four) with the six experiment conditions (three triggers by two intent types). Each participant performed twelve tasks made up of three different conditions in each of the four worlds. Both the worlds and the tasks were counterbalanced. Since, the participant solved twelve of the twenty-four tasks, distributed evenly in randomized order, the study resulted in a between-subjects study. Sixty participants were recruited to give multiple samples across each of the six experiment conditions.
7.6 Results

Responses were obtained for 61 participants, one extra participant was able to participate because of technical glitch with the MTurk software. Responses of two participants were discarded because they took less than 3 minutes to complete the study, which was not possible given the length of the 12 videos that they needed to watch. Since their were 59 remaining data points, for each task/video pairing, we considered the first 29 responses yielding 58 data points. Participants took an average of 20.2 minutes to complete the study.

7.6.1 Summary Statistics

Table 7.4 and Figure 7.4 show the summary statistics of the ratings given as a function of trigger type and intent type. Results for when no metric is used, which means the robot never replans, are marked as ‘NOM’. Each row in the table represents the statistics of 112 observations (28 responses collected for each of the four worlds). Recall that a value of 1 corresponded to the rating Extremely good, whereas 5 represented Extremely bad behavior of the robot in terms of obeying intent.

<table>
<thead>
<tr>
<th>Trigger Type</th>
<th>Intent</th>
<th>Mean</th>
<th>Median</th>
<th>Std Dev</th>
<th>Std Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>Pure</td>
<td>2.41</td>
<td>2</td>
<td>1.5</td>
<td>0.14</td>
</tr>
<tr>
<td>ITM</td>
<td>Pure</td>
<td>1.46</td>
<td>1</td>
<td>0.83</td>
<td>0.08</td>
</tr>
<tr>
<td>NOM</td>
<td>Pure</td>
<td>4.44</td>
<td>5</td>
<td>0.99</td>
<td>0.09</td>
</tr>
<tr>
<td>CS</td>
<td>Mixed</td>
<td>3.25</td>
<td>4</td>
<td>1.63</td>
<td>0.15</td>
</tr>
<tr>
<td>ITM</td>
<td>Mixed</td>
<td>2.03</td>
<td>2</td>
<td>1.23</td>
<td>0.12</td>
</tr>
<tr>
<td>NOM</td>
<td>Mixed</td>
<td>3.81</td>
<td>4</td>
<td>1.25</td>
<td>0.12</td>
</tr>
</tbody>
</table>

Table 7.4: Summary statistics of CS, ITM, and NOM enabled tasks.

**Pure Intent.** Table 7.4 reflects both the pure and mixed intent statistics for each trigger type, as do Figures 7.4(a) and 7.4(b), respectively. For pure intent, the ITM trigger generated ratings with a mean and median of 1.46 and 1, respectively, suggesting that the trigger generated replanned paths when participants felt that the existing path violated intent; these results support hypothesis 1. For pure intent, the CS trigger generated ratings
with a mean and median of 2.41 and 2, respectively, suggesting that trigger was good but not quite as good as the ITM trigger, but the distribution shown in Figure 7.4(a) indicate that an ANOVA needs to be run to see if the differences are significant. This result provides partial support for hypothesis 2. For pure intent, the control condition when no replanning occurred had a mean and median of 4.44 and 5, respectively. The relative ratings of ITM, CS, and NOM conditions provide support for hypothesis 3.

**Mixed Intent.** For mixed intent, the ratings for both the ITM and CS triggers are lower than they were for pure intent. This needs to be studied further, but is likely caused by a difficulty for participants to understand how to evaluate mixed intent commands. The means and medians in Table 7.4 and the distributions in Figure 7.4(b) provide support for hypothesis 1, partial support for hypothesis 2, and mixed support for hypothesis 3.

![Figure 7.4: Box-and-whiskers plots for the distribution of ratings across trigger type. Median is indicated by the horizontal line, the mean is represented by the diamond, the 25% – 75% quartile range is represented by the shaded region, the 90% range is enclosed in the whiskers, and outliers are indicated by black circles.](image)

Further statistical analysis needs to be performed to determine whether the low average ratings for the CS trigger in the mixed intent condition were caused by false positives in the worlds.
Differences across trigger types. A one-way ANOVA with tukey adjustments for multiple comparisons was conducted to determine whether there were statistically significant differences across trigger types under both the pure intent and mixed intent condition. Tables 7.5 and 7.6 present the results. Under both pure and mixed intent conditions, there is statistically significant support for hypothesis 3. However, as shown in Section 7.2, there is one world where it appears that the ITM condition is indistinguishable from the NOM condition (and both superior to the CS condition), so a larger ANOVA needs to be conducted to find multi-way interactions between trigger type, intent type, and world.

<table>
<thead>
<tr>
<th></th>
<th>CS</th>
<th>ITM</th>
<th>NOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td></td>
<td>&lt; .0001*</td>
<td>&lt; .0001*</td>
</tr>
<tr>
<td>ITM</td>
<td>&lt; .0001*</td>
<td></td>
<td>&lt; .0001*</td>
</tr>
<tr>
<td>NOM</td>
<td>&lt; .0001*</td>
<td>&lt; .0001*</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.5: One-way ANOVA for the pure intent: The means of CS, ITM, and NOM differed significantly

<table>
<thead>
<tr>
<th></th>
<th>CS</th>
<th>ITM</th>
<th>NOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td></td>
<td></td>
<td>0.0074*</td>
</tr>
<tr>
<td>ITM</td>
<td>&lt; .0001*</td>
<td></td>
<td>&lt; .0001*</td>
</tr>
<tr>
<td>NOM</td>
<td>0.0074*</td>
<td>&lt; .0001*</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.6: One-way ANOVA for the mixed intent: The means of CS, ITM, and NOM differed significantly

7.6.2 Summary Statistics across Worlds

Figures 7.5 through Figure 7.7 show the summary statistics with respect to the different worlds. The blue and orange data on the left side of the box and whiskers plots (worlds S1 and S2) are for the simple worlds, and the grey and yellow data on the right side of the box and whiskers plots (worlds X1 and X2) are for the complex worlds.

**CS Trigger.** Figures 7.5(a) and (b) show summary statistics for the CS trigger for pure and mixed intent, respectively. The S1 and S2 prefixes in the world names indicate simple worlds, and the prefixes X1 and X2 indicate complex worlds. The C in these names
indicates the CS trigger, and the $P$ and $M$ suffixes in the name indicate pure and mixed intent, respectively. The boxplots in Figure 7.5 (a), suggest that when the enemy does not move all over the map and when the intent is pure, the CS trigger and the replanned path it is easy for the robot to update to good stealthy paths. However, in the case of complex worlds where the enemy moves randomly all over the map, either the CS trigger or the replanned path are not satisfactory to the participants.

**ITM Trigger.** Figure 7.6(a) and Figure 7.6(b) show the summar statistics for the ITM trigger for the pure and mixed intent conditions, respectively. Note that the ratings for all the four worlds are better than the ratings for the CS trigger. Figure 7.6(a) shows that for pure intent, the ITM trigger and replanner work well whether the enemies move in a confined space (simple worlds) or randomly across a large space (complex worlds). Comparing the results for the pure intent with the ITM trigger to the results for the CS trigger suggest that the problem is with the trigger rather than the replanning algorithm. For mixed intent, Figure 7.6(b) shows that for both the simple worlds and one of the complex worlds the ITM trigger worked satisfactorily. For complex world X2, either the trigger or replanner did not work well.

One of the differences between the simple worlds and the complex world is how well participants can predict the movements of the enemies. In the simple worlds, predicting movements is subjectively easy, but in complex worlds, predicting movements is subjectively
difficult because the paths were randomly generated. This means that in the complex worlds it is possible to have a path replanned because an enemy intercepts the planned path only to have the enemy quickly move from the path negating the need to replan. Future work should explore the relationship between the predictability of changes in the world and when replanning should be triggered.

Figure 7.6: User rating statistics for ITM-based replanning. x is mean, horizontal line is median.

No Trigger. Figure 7.7(a) and Figure 7.7(b) show the results for the control condition, where no replanning is performed. Comparing the high numbers across all worlds for the NOM trigger condition to the CS trigger and ITM trigger conditions suggests that the replanning was needed in every world. The result for world X2 under the mixed intent condition is interesting because it suggests that for this world the need to replan is ambiguous. Indeed, the distribution of responses for world X2 under the mixed intent condition and no replanning is visually indistinguishable from the distribution for the ITM trigger.

7.7 Summary and Future Work

This chapter examined a problem where a robot must find a path from a starting location to a goal, but when the world can change while the robot is following its path. The robot’s task was to plan a path that conformed to human intent from a start configuration to a goal configuration, detect when the original plan no longer aligns with human intent because the world has changed in an unfavorable way, and replan a new path that satisfies intent
when a misalignment is found. We investigated two metrics, cosine similarity and the intent
threshold margin, that can be used to identify when the world has changed in such a way
that the planned path no longer matches human intent. Algorithms were described that can
be used in near real-time to plan paths for the three pure intents. Since these three paths
correspond to the three corners of the Pareto front, the payoffs for these three paths can be
used to normalize the payoff vector for the path segment that the robot has not yet traversed.

Results from a Mturk study indicated that human participants considered the ITM-
based replanning trigger successful. Moreover, the results showed that the ITM-based trigger
was superior to the CS-based trigger and to the control condition with no replanning. The
CS-based trigger was usually superior to the control condition. This chapter identified
conditions under which the CS trigger would theoretically not work, and designed experiment
conditions that satisfied those theoretical conditions. Results from the MTurk study suggest
that the CS trigger did indeed fail under those theoretical conditions, as judged by the
experiment participants. Future work should provide a precise statistical analysis of the
relationship between the conditions under which the CS trigger failed (e.g., false negatives or
false positives) and the ratings from participants.

In one of the worlds, the control condition performed as well as the ITM condition,
though neither performed very well according to participant ratings, and both the ITM and
control condition performed better than the CS trigger. We speculated that the changes

Figure 7.7: User rating statistics for NOM-based tasks. x is mean, horizontal line is median.
in the world that triggered the replanning were quickly undone, i.e., an enemy crossed a path and then moved away from the path, meaning that it would have been satisfactory if no replanning had ever occurred. Future work should explore the relationship between the predictability and duration of changes in the world (like where an enemy is going and whether its intended path will last long enough to cause concern) and when replanning should be triggered.

Finally, future work should include designing an interface that allows a human to specify intent and then interactively manage various replanning triggers. Usability and human workload should be measured to understand more about how human-robot interaction in dynamic worlds can use human intent to make interaction easier.
Chapter 8

Summary and Future Work

8.1 Summary

This dissertation dealt with path-planning and replanning for a robot navigating in both static and dynamic environments. It described and evaluated three novel user interfaces: palette, sliders, and prism. Each interface can be used to make intent-based tradeoffs among solutions that cater to multiple objectives. Two distinct metrics, cosine similarity and the novel intent threshold margin, for mapping from human intent to a reasonable path were explored. Critical events that triggered replanning were described and evaluated. A set of user studies gave evidence that the interfaces, metrics, and triggers were effective for planning paths for a simulated ground robot in a world with three objectives.

8.2 Future Work

8.2.1 Spider Web Interface

This dissertation evaluated the color-based interfaces: palette, sliders, and the prism. The research also included preliminary work on an interface that did not depend on color and that did not require the components of the intent vector to sum to one. This interface is described in Appendix A.

The Spider Web interface is a region-based interface that was labeled the ‘Spider Web’ interface because of the way regions had some similarity to the structures in a spider web. In
fact, as part of the Mechanical Turk study reported in Chapter 6, intent was expressed by the experiment designer using the interface. However, it was not evaluated by this dissertation.

Future work should design and evaluate an interface based on the Spider Web metaphor, and compare the usability of the interface to the palette, sliders and prism interfaces. The future work should include evaluating how the Spider Web interface could be used (a) to express intent when there are more than three objectives and (b) to evaluate performance when color is not available such as with color-blind users.

8.2.2 Path Planning Algorithms

This dissertation has extensively used sampling-based planning algorithms. Other algorithms for path planning, such as planners for aerial vehicles, should be explored and evaluated using the described interfaces and metrics.

8.2.3 Multi-objective Applications

In this dissertation, the working assumption was that, given a trajectory, the robot navigates at a constant speed across a flat world in navigable terrain. Many robot applications include difficult terrain (e.g. search and rescue, disaster sites, and agriculture); the described framework should be extended to model traversibility of the region. Paths may be adapted considering whether the terrain is static or dynamic. In a dynamic terrain scenario, on an earthquake disaster site for example, while a robot is searching the onset of rain can change the present plan’s feasibility or payoffs.

Another navigation objective this dissertation did not address is energy consumption. Energy optimization is an important criterion in many robot navigation tasks. This objective often dictates the capabilities of a robot, including the distance the robot can travel before it is powered up again. The path-planning framework described here should be extended to include the energy objective. This objective might conflict with the traversibility objective outlined above, as difficult settings may consume more energy.
8.2.4 Other Replanning Domains

Intent-based replanning should be applied to other multi-objective problems and to other goals. For example, consider a treatment for attention-deficit/hyperactivity disorder. Relevant features include benefits and side-effects of medicine, nutrition, exercise, and behavior-training where each can be modelled. Plans that consider various tradeoffs of these features can be consolidated as solutions, and the effectiveness of a treatment plan might change due to patient response or external influences (e.g., changing school or changing family status).

8.2.5 Clustering

The intent threshold margin gives a region that gives a set of solutions that satisfy a set of multi-objective criterion. This idea of of intent threshold margin can be investigated further where one may cluster solutions belonging to certain domain into different groups. Each group may reflect some similarities, whereas different groups would exhibit differences. Hence, the proposed intent threshold margin could potentially be used as a clustering algorithm.

One application of clustering includes multi-robot tasks. For example, in problems where information is sought about different areas of the environment, multiple robots can be used, each navigating on different paths within a cluster either to ensure that the mission is accomplished (defined by one or many of the robots reaching the destination), or to include a larger area of information to be covered in given scenario, relative to speed.
Appendix A

Spider Web Interface

As part of the dissertation research, a fourth interface was designed for allowing a human to express intent. The interface had some novel features, but was not evaluated in a full user study. This appendix describes the features of this interface.

Figure A.1 and Figure A.2 are examples of the basic concept of this interface with three objectives.

![Figure A.1: Example of a Covert Intent Set on Spider Web.](image)

Figure A.1 shows the spider web interface where the covert intent is set. The top-right panel shows the command interface with a triangle, where the vertices depict the maximum
intent desired for a specific objective, shown here by a distinct color dab at the vertex. Any less-desirable intent lies on the line connecting the apex and the center of the triangle. Each of these lines have a dab that can be dragged and dropped on the line thereby specifying intent. In order to make distinction between the three objectives, the same color apex dabs as in earlier interfaces, red, green and blue, are used here in this top-right panel. The intent set here in this example is covert made by dragging the light green dab close to the green dab of the triangle vertex. The other two objectives are placed near to the center indicating the user is not intent towards these objectives.

Referring to the same Figure A.1, the spider web picture in the lower-right depicts solutions with payoffs superimposed on a triangle similar to that above. Here, the vertex shows maximum payoffs for an objective. A triangle or a frame on this spider web represents one solution and its three associated payoffs. The center of the web is again the zero payoffs area. For the example covert intent, two solutions, one purple and one orange, resulted in the intent satisfying region. These two solutions/paths are shown on the map in brown color on the left panel. The gray paths are example paths that do not satisfy intent for the given intent. The solutions/paths are mapped in this example using intent threshold margin metric.

Similarly, Figure A.2 shows an example of a mixed intent of a safe and covert path set by the user on the top-right panel of the spider web interface. Here the brown paths shown on the map satisfy the intent: ‘go stealthily and safe’. Again, the gray paths are the ones that do not satisfy the intent.

The above description is very basic and evokes the need for a much finer representation of the spider web interface. Future work on refining this work can be carried out using the C# Unity platform, chosen for its appeal and the readily-available drawing libraries.
Figure A.2: Mixed Objective Example of a Covert and safe Intent.
References


172


176


