Semi-autonomous robotic wheelchair controlled with low throughput human-machine interfaces

Dmitry Aleksandrovich Sinyukov
Worcester Polytechnic Institute

Follow this and additional works at: https://digitalcommons.wpi.edu/etd-dissertations

Repository Citation

This dissertation is brought to you for free and open access by Digital WPI. It has been accepted for inclusion in Doctoral Dissertations (All Dissertations, All Years) by an authorized administrator of Digital WPI. For more information, please contact wpi-etd@wpi.edu.
Semi-autonomous robotic wheelchair controlled with low throughput human-machine interfaces

by

Dmitry Aleksandrovich Sinyukov

A Dissertation
Submitted to the Faculty
of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the
Doctor of Philosophy in Robotics Engineering
April 2017

©Copyright by Dmitry A. Sinyukov 2017. All rights reserved
Semi-autonomous robotic wheelchair controlled with low throughput human-machine interfaces

by

Dmitry Aleksandrovich Sinyukov

Submitted to the Robotics Engineering
on April 21, 2017, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics Engineering

Abstract

For a wide range of people with limited upper- and lower-body mobility, interaction with robots remains a challenging problem. Due to various health conditions, they are often unable to use standard joystick interface, most of wheelchairs are equipped with. To accommodate this audience, a number of alternative human-machine interfaces have been designed, such as single switch, sip-and-puff, brain-computer interfaces. They are known as low throughput interfaces referring to the amount of information that an operator can pass into the machine. Using them to control a wheelchair poses a number of challenges.

This thesis makes several contributions towards the design of robotic wheelchairs controlled via low throughput human-machine interfaces: (1) To improve wheelchair motion control, an adaptive controller with online parameter estimation is developed for a differentially driven wheelchair. (2) Steering control scheme is designed that provides a unified framework integrating different types of low throughput human-machine interfaces with an obstacle avoidance mechanism. (3) A novel approach to the design of control systems with low throughput human-machine interfaces has been proposed. Based on the approach, position control scheme for a holonomic robot that aims to probabilistically minimize time to destination is developed and tested in simulation. The scheme is adopted for a real differentially driven wheelchair. In contrast to other methods, the proposed scheme allows to use prior information about the user habits, but does not restrict navigation to a set of pre-defined points, and parallelizes the inference and motion reducing the navigation time. (4) To enable the real time operation of the position control, a high-performance algorithm for single-source any-angle path planning on a grid has been developed. By abandoning the graph model and introducing discrete geometric primitives to represent the propagating wave front, we were able to design a planning algorithm that uses only integer addition and bit shifting. Experiments revealed a significant performance advantage. Several modifications, including optimal and multithreaded implementations, are also presented.

Dissertation Supervisor: Dr. Taşkın Padır
Title: Associate Professor
Electrical and Computer Engineering
Northeastern University
### 0.1 Acknowledgments

This dissertation is a result of almost six years of work at Worcester Polytechnic Institute. I would like to express my gratitude to all the people who made it possible:

- my parents Vera Dmitrievna Sinyukova (Синюкова Вера Дмитриевна) and Aleksandr Nikolayevich Sinyukov (Синюков Александр Николаевич) for always prioritizing their children over everything else in their life and making me whom I am;

- my high school teacher of mathematics Anna Vasilievna Sonina (Сонина Анна Васильевна) for her dedication to her profession, for instilling the high standards of mathematical thinking in young minds (the robotic wheelchair Anna discussed in this dissertation was named after her);

- my advisor Dr. Taşkin Padır for his financial and organizational support of this project, for providing me with an opportunity to work on it;

- An Yinglin (安颖琳) for her love and support;

- Inara Nigmatullina (Инара Нигматуллина) for her love and support;

- Olga Poppe for being a reliable friend, for reviewing parts of my work and helping with many small things;

- the English-speaking club *Speak Freely* in Chelyabinsk, Russia that I left on the shoulders of the amazing Daria Zelenina (Дарья Зеленина) for giving me the confidence in myself;

- the *Russian-speaking Students and Scholars Association* (RSSA) at WPI and all the wonderful people who contributed to the creation and the development of the organization that was almost like my family at WPI;

- WPI Student Events Office and particularly Amy L’Heureux for being very supportive in allocating the space for robot navigation experiments;

- Tracy Coetzee for being the best secretary I ever worked with;
• Fernando Quivira for his help when it was so urgently needed;

• all the talented undergraduate and master students whom I had the luck to work with and who contributed to this project: James Fleming, Ran Li, Ross Desmond, Nicholas Otero, Tenell Rhodes, Junqing Qiao, Jayam Patel, Runzi Gao, Rashida Nayeem, Tanishq Bhalla, Marisa Warner, Gilmar Da Vitoria;

• and several other great people who helped me in this work: Apoorv Walawalkar, Irina Poliakova (Ирина Полякова).

This material is based upon work supported by the National Science Foundation under Grant No. 1135854.
0.2 Acronyms

ADL - activities of daily living
ALS - Amyotrophic lateral sclerosis
AP - Action potentials
BCI - Brain-computer interface
BMI - Brain-machine interface (same as BCI)
BOLD - Blood-oxygen-level dependent
dHb - Deoxygenated hemoglobin
COG - Center of gravity
DNI - Direct neural interface (same as BCI)
DOF - Degree of freedom
ECoG - Electrocorticography
EEG - Electroencephalography
EMA - Electromagnetic articulography
EMG - Electromyography
EOG - Electrooculogram
ERD - Event-related desynchronization
ERS - Event-related synchronization
fMRI - Functional magnetic resonance imaging
fNIRS - Functional near-infrared spectroscopy
FP - Field potentials
Hb - Oxygenated hemoglobin
HMI - Human-machine interface
HMM - Hidden Markov model
HR - Haemodynamic response
ITR - Information transfer rate
LFP - Local field potentials
LIS - Locked-in syndrome
LTHMI - Low throughput human-machine interface
LTI - Low throughput interface (here it is the same as LTHMI)
MEA - Microelectrode arrays
MDP - Markov decision process
MUAP - Motor unit action potential
MUAPT - MUAP train
MEG - Magnetoencephalography
NAM - Non-audible murmur
NIR - Near-infrared
PET - Positron emission tomography
PDF - Probability distribution function (or probability density function)
POC - Proof-of-concept
POI - Point of interest
POMDP - Partially observable Markov decision process
RA - Reachability area
rtfMRI - Real-time functional magnetic resonance imaging
sEMG - Surface electromyography
SMR - Sensorimotor Rhythms
SNP - Sip-and-Puff system
SVR - Subvocal recognition
SVM - Support vector machine
SCI - Spinal cord injury
SSEP - Steady state evoked potentials OR Somatosensory evoked potential
SSI - Silent speech interface
VS - Vegetative state
VOG - Video-oculography
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Acknowledgments</td>
<td>5</td>
</tr>
<tr>
<td>0.2</td>
<td>Acronyms</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>Introduction</td>
<td>21</td>
</tr>
<tr>
<td>1.1</td>
<td>Disabilities and their prevalence</td>
<td>21</td>
</tr>
<tr>
<td>1.2</td>
<td>Nonconventional human-machine interfaces</td>
<td>24</td>
</tr>
<tr>
<td>1.3</td>
<td>Brain-computer interfaces</td>
<td>31</td>
</tr>
<tr>
<td>1.3.1</td>
<td>Methods of signal acquisition used for BCI</td>
<td>32</td>
</tr>
<tr>
<td>1.3.2</td>
<td>Types of signals used for BCI</td>
<td>37</td>
</tr>
<tr>
<td>1.3.3</td>
<td>BCI signal processing</td>
<td>41</td>
</tr>
<tr>
<td>1.3.4</td>
<td>BCI is practical</td>
<td>42</td>
</tr>
<tr>
<td>1.4</td>
<td>Classification of human-machine interfaces (HMIs)</td>
<td>43</td>
</tr>
<tr>
<td>1.5</td>
<td>Navigation and control with low throughput human-machine interfaces</td>
<td>45</td>
</tr>
<tr>
<td>1.5.1</td>
<td>Steering models</td>
<td>47</td>
</tr>
<tr>
<td>1.5.2</td>
<td>Direct steering control</td>
<td>49</td>
</tr>
<tr>
<td>1.5.3</td>
<td>Shared steering control</td>
<td>49</td>
</tr>
<tr>
<td>1.5.4</td>
<td>Shared position control</td>
<td>50</td>
</tr>
<tr>
<td>1.6</td>
<td>Structure of the dissertation</td>
<td>51</td>
</tr>
<tr>
<td>1.7</td>
<td>Contributions</td>
<td>51</td>
</tr>
<tr>
<td>2</td>
<td>Motion control with an adaptive controller</td>
<td>55</td>
</tr>
<tr>
<td>2.1</td>
<td>Background</td>
<td>56</td>
</tr>
<tr>
<td>2.2</td>
<td>Wheelchair Model</td>
<td>58</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Mathematical Model</td>
<td>59</td>
</tr>
</tbody>
</table>
2.2.2 Simulation ................................................. 61
2.2.3 Model Verification ...................................... 62
2.3 Adaptive Motion Control with Online Parameter Estimation .............. 62
  2.3.1 General formulation .................................... 62
  2.3.2 Application to the wheelchair model ........................ 65
2.4 Simulation Results ......................................... 67
  2.4.1 Simulation .................................................. 67
  2.4.2 Experiment 1: Constant parameters, unsaturated current ............ 68
  2.4.3 Experiment 2: Constant parameters, saturated current ............... 70
  2.4.4 Experiment 3: Parameter changed, unsaturated current ............. 71
  2.4.5 Experiment 4: Zero initial parameter vector estimate, unsaturated
current .......................................................... 71
2.5 Towards real robot experiments .................................. 71
2.6 Discussion ...................................................... 76

3 Shared steering control 79
  3.1 Background .................................................. 79
  3.2 Robot system overview ..................................... 82
  3.3 Navigation subsystem ...................................... 83
    3.3.1 Shared steering control with assistive navigation ............. 84
    3.3.2 Tuning assistive navigation ................................ 86
  3.4 Human-machine interface integration ................................ 91
    3.4.1 Voice control with CMU Sphinx ................................ 93
    3.4.2 Voice control with Google Glass ............................. 94
    3.4.3 Facial expression control with Emotiv EPOC .................... 95
  3.5 Discussion ...................................................... 97

4 CWave: high-performance single-source any-angle path planning on a grid 99
  4.1 Background .................................................. 100
  4.2 CWave on a grid without obstacles ................................ 103
    4.2.1 Overview of midpoint circle algorithm ....................... 104
4.2.2 Distance error of Bresenham circles .............................................. 107
4.2.3 Non-Bresenham points and their properties ................................. 109
4.2.4 Necessary and sufficient condition for an NBP .............................. 111
4.2.5 Distance error in Non-Bresenham points ..................................... 112
4.2.6 Possible vertex configurations .................................................... 113
4.2.7 Assigning integer distance values to vertices ............................. 118

4.3 CWave on a grid with simply-connected obstacles ........................ 119
   4.3.1 Coordinate frames and definitions ............................................ 120
   4.3.2 Visibility cone ............................................................................ 121
   4.3.3 Identifying corner vertices .......................................................... 123
   4.3.4 Iterating through boundary vertices ............................................ 124
   4.3.5 Determining arc endpoints from the boundary vertices .............. 126
   4.3.6 Distance accuracy on a simply-connected map ............................ 127
   4.3.7 Path extraction on simply-connected maps .................................. 128

4.4 CWave on a generic map ................................................................. 129
   4.4.1 Boundary between merging waves in continuous space ............. 129
   4.4.2 Integer criterion ......................................................................... 130
   4.4.3 Distance accuracy on a generic map ........................................... 132
   4.4.4 Wave merge with floating-point criterion ................................... 133

4.5 Performance tests .......................................................................... 134
   4.5.1 Verification and accuracy ............................................................. 134
   4.5.2 Speed comparison to other algorithms ....................................... 135
   4.5.3 Parallelization .............................................................................. 137
   4.5.4 Speed of CWave modifications ................................................... 139

4.6 Discussion ....................................................................................... 140
   4.6.1 Implementation ............................................................................ 140
   4.6.2 Vertex exact floating-point distance calculation .......................... 142
   4.6.3 Reduction of thread contention in the multithreaded implementation 142
   4.6.4 Improving wave merges ............................................................... 142
   4.6.5 Performance tests on various classes of maps ............................ 143
## Shared position control with NoVeLTI

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Background</td>
<td>145</td>
</tr>
<tr>
<td>5.2 Definition of a control system with a low throughput human-machine interface (LTHMI)</td>
<td>149</td>
</tr>
<tr>
<td>5.3 Problem Analysis</td>
<td>151</td>
</tr>
<tr>
<td>5.3.1 Modeling the system</td>
<td>152</td>
</tr>
<tr>
<td>5.3.2 Modeling human intent</td>
<td>154</td>
</tr>
<tr>
<td>5.3.3 Modeling human-machine interface</td>
<td>154</td>
</tr>
<tr>
<td>5.4 Synthesis</td>
<td>156</td>
</tr>
<tr>
<td>5.4.1 Intent estimate representation</td>
<td>156</td>
</tr>
<tr>
<td>5.4.2 Intermediate destination state selection</td>
<td>157</td>
</tr>
<tr>
<td>5.4.3 Intent estimate update (Inference)</td>
<td>160</td>
</tr>
<tr>
<td>5.4.4 State space segmentation</td>
<td>160</td>
</tr>
<tr>
<td>5.4.5 Map segmentation</td>
<td>162</td>
</tr>
<tr>
<td>5.5 Simulation Experiments</td>
<td>167</td>
</tr>
<tr>
<td>5.5.1 Automated test setup</td>
<td>167</td>
</tr>
<tr>
<td>5.5.2 Simulation experiment results</td>
<td>168</td>
</tr>
<tr>
<td>5.6 Robot Experiments</td>
<td>172</td>
</tr>
<tr>
<td>5.6.1 Experimental system setup</td>
<td>172</td>
</tr>
<tr>
<td>5.6.2 Parameters of the experiments</td>
<td>175</td>
</tr>
<tr>
<td>5.6.3 Results of robot experiments</td>
<td>176</td>
</tr>
<tr>
<td>5.7 Discussion</td>
<td>199</td>
</tr>
</tbody>
</table>

## Conclusions

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>207</td>
</tr>
</tbody>
</table>
List of Figures

1-1 Single-switch interface and a scanning interface .......................... 24
1-2 Sip-and-Puff system to drive a wheelchair ............................... 25
1-3 Tongue Drive System (TDS) .................................................. 25
1-4 Eye and gaze tracking systems ............................................. 26
1-5 Various techniques for Silent speech interfaces (SSIs) ................. 28
1-6 Electromagnetic articulograph AG500 ....................................... 29
1-7 Other alternative HMI ............................................................ 29
1-8 EEG headsets for gaming purposes .......................................... 31
1-9 Human-machine interfaces based on electromyography(EMG) ........ 31
1-10 Components of a brain-computer interface (BCI) ...................... 32
1-11 Motor and sensory regions of the cerebral cortex ...................... 33
1-12 Types of brain-computer interfaces based on their principle of operation. ... 33
1-13 Placement of EEG electrodes ................................................. 35
1-14 Comparison of temporal and spatial resolutions of brain imaging methods 37
1-15 Subject in front of SSVEP-based BCI display ............................ 40
1-16 Trajectories of the cursor controlled with an intracortical microelectrode array ................................................................. 44
1-17 Robot navigation layers, contributions, and outline of the dissertation. ... 46

2-1 In this chapter, we discuss the lowest level of the navigation system: motion control ................................................................. 55
2-2 Differential-drive robot (wheelchair) with XY-displacement of COG. ... 59
2-3 Wheelchair model test environment .......................................... 63
2-4 Complete system Simulink® diagram ........................................ 67
2-5 Experiment 1 (constant parameters, unasaturated motor current) simulation results. ................................................................. 69
2-6 Experiment 2 (constant parameters, saturated current) simulation results. .... 70
2-7 Experiment 3 (parameter changed, unsaturated current) simulation results. .... 72
2-8 Experiment (zero initial parameter vector estimate, unsaturated current) simulation results. ...................................................... 73
2-9 Wheelchair control system diagram ............................................ 74
3-1 In this chapter, we discuss shared steering control operating at the local planner layer ................................................................. 79
3-2 Robot sensing hardware diagram .............................................. 82
3-3 Wheelchair navigation software architecture diagram .................... 83
3-4 Test case diagrams for tuning assistive navigation .......................... 88
3-5 Robot footprint evolution .......................................................... 89
3-6 Wheelchair user interface architecture diagram ............................... 92
4-1 In this chapter, we discuss CWave, a high-performance path planning algorithm that was developed to enable shared position control method discussed in the next chapter ................................................ 100
4-2 Gradual expansion of a circular wave in a continuous space .............. 102
4-3 Iterating through vertices of a circular arc of radius $r = 9$ using midpoint circle (MPC) algorithm ...................................................... 105
4-4 Error function $\varepsilon$ as a function of distance error $\delta$. ................. 107
4-5 Circles of integer radii drawn with Bresenham’s midpoint algorithm in octant 1. N designates Non-Bresenham points (NBPs) ......................... 109
4-6 Formation of Non-Bresenham points (NBPs) ............................... 110
4-7 Possible configurations of two sequential vertices/pairs while iterating through a Bresenham circle .................................................. 114
4-8 An impossible configuration of two diagonal NBPs .......................... 114
4-9 Curve $d_2(d_1) = \sqrt{(d_1 + a)^2 + (10 - a^2)}$ and its asymptotes ......... 115
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-10</td>
<td>Proving an impossible diagonal configuration of NBPs</td>
</tr>
<tr>
<td>4-11</td>
<td>Visibility of various vertices from vertex S.</td>
</tr>
<tr>
<td>4-12</td>
<td>Classification of grid cells/pairs occupancy patterns (O represents a free cell, X represents an occupied cell)</td>
</tr>
<tr>
<td>4-13</td>
<td>An example visibility cone S</td>
</tr>
<tr>
<td>4-14</td>
<td>All 40 possible combinations of two sequential configurations in octant 1</td>
</tr>
<tr>
<td>4-15</td>
<td>Combination $OX \rightarrow XO$ allows a narrow visibility cone to pass between the occupied cells</td>
</tr>
<tr>
<td>4-16</td>
<td>Next vertex after $A$ ($r = 7$) should be $C$ ($r = 8$), not $B$ ($r = 9$)</td>
</tr>
<tr>
<td>4-17</td>
<td>A merge of wave-$A$ and wave-$B$</td>
</tr>
<tr>
<td>4-18</td>
<td>Triangle inequality used for the shake test: $</td>
</tr>
<tr>
<td>4-19</td>
<td>Test maps</td>
</tr>
<tr>
<td>4-20</td>
<td>Performance comparison of single-source path planning algorithms</td>
</tr>
<tr>
<td>4-21</td>
<td>Performance of 1- to 8-threaded implementations (CWave1 to CWave8) compared to the single threaded implementation (CWave)</td>
</tr>
<tr>
<td>4-22</td>
<td>Performance comparison of various modifications of CWave: measured in seconds (left), normalized by the average duration of CWaveInt (right)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-1</td>
<td>In this chapter, we develop a shared position control method at the top level of the robot navigation hierarchy</td>
</tr>
<tr>
<td>5-2</td>
<td>Local goal selection</td>
</tr>
<tr>
<td>5-3</td>
<td>Selecting semantic goals</td>
</tr>
<tr>
<td>5-4</td>
<td>Human-interface-object model</td>
</tr>
<tr>
<td>5-5</td>
<td>Not all low throughput human-machine interfaces are probabilistic</td>
</tr>
<tr>
<td>5-6</td>
<td>A simple discrete state space with 3 states</td>
</tr>
<tr>
<td>5-7</td>
<td>If the PDF is uniform over all accessible vertices, $\tilde{C}(.)$ will have 4 minima on this map: $A$, $B$, $C$, $D$, and $\tilde{C}(B) &lt; \tilde{C}(A)$.</td>
</tr>
<tr>
<td>5-8</td>
<td>Comparing two ways (I and II) of segmenting a space into two regions.</td>
</tr>
<tr>
<td>5-9</td>
<td>Horizontal tile map segmentation (htile)</td>
</tr>
<tr>
<td>5-10</td>
<td>Vertical tile map segmentation (vtile)</td>
</tr>
</tbody>
</table>
5-11 Equidistant map segmentation (equidist) ........................................ 166
5-12 Extremal map segmentation (extremal) ........................................... 166
5-13 System for automated testing of simulated robot navigation via LTHMI 167
5-14 Test framework time diagram .......................................................... 169
5-15 An example of the evolution of the intent probability distribution function (red/blue color) and corresponding simulated robot motion. The intended destination was inferred at 11s. ........................................ 170
5-16 Results of the automated simulation tests .......................................... 171
5-17 Semi-autonomous robotic wheelchair used for navigation tests. .......... 172
5-18 Test framework architecture .............................................................. 173
5-19 Robot experiment snapshot .............................................................. 174
5-20 An example of a single robot point-to-point navigation experiment. Here smoothening is enabled. The intended destination is inferred at 37s. .... 177
5-21 Effect of HMI accuracy when no POIs are defined ............................. 178
5-22 Effect of POIs, 94% HMI ................................................................. 180
5-23 Effect of POIs, 70% HMI ................................................................. 181
5-24 Effect of PDF smoothening, 94% HMI ............................................. 182
5-25 Effect of PDF smoothening, 70% HMI ............................................. 183
5-26 Effect of inference_unit matrix being less accurate than HMI matrix, without POIs ................................................................. 185
5-27 Effect of inference_unit matrix being less accurate than HMI matrix, with POIs ................................................................. 186
5-28 Effect of the goal marker, 94% HMI ..................................................... 187
5-29 Effect of the goal marker, 70% HMI ..................................................... 188
5-30 Navigating to Non-POI vertices ....................................................... 190
5-31 Changing intended destination on the way ....................................... 191
5-32 Performance effect of MAP SEGMENTATION policy when pose selection policy is no_move ......................................................... 193
5-33 Performance effect of MAP SEGMENTATION policy when pose selection policy is cog2lopt ......................................................... 194
5-34 Performance effect of MAP SEGMENTATION policy when pose selection policy is nearcog_obst ....................................................... 195
5-35 Performance effect of POSE SELECTION policy when division policy is altertile ................................................................. 196
5-36 Performance effect of POSE SELECTION policy when division policy is extredist ............................................................... 197
5-37 Performance effect of POSE SELECTION policy when division policy is nearcog_extremal .................................................... 198
5-38 Points of interest identified by clustering wheelchair navigation data and modeled with Gaussian distributions (ellipses on the map) .............. 206
List of Tables

1.1 A classification of disabilities and the prevalence in the US .................. 22
1.2 Comparison of brain imagining methods used for BCIs ....................... 34
1.3 Comparison of BCI signals ............................................. 38

2.1 Simulation parameter values ............................................. 68

3.1 Test values for “Free rotational motion” test ................................. 87
3.2 Test values for “Free linear motion” test .................................. 88
3.3 Test values for “Heading towards a wall” test ................................. 88
3.4 Test values for “Rotating at a wall” test ................................. 90
3.5 Test values for “Heading towards an obstacle” test ....................... 90
3.6 Test values for “Heading towards a doorway” test .................. 91
3.7 Commands accepted by Command Interpreter ........................ 94
3.8 Expressiv suite accuracy experiment results on 4 subjects ................. 96

4.1 List of variables used in this chapter ...................................... 104
4.2 Summary of CWave modifications ......................................... 140

5.1 Points of interest on the map are represented as a weighted sum of 2D
Gaussians. Gaussian parameters are listed in this table .................. 175
5.2 Relation between view size and smoothening radius r ..................... 179
Chapter 1

Introduction

The ultimate goal of robotics has always been to improve human life, be it a robotic vacuum cleaner, an industrial robot, an exoskeleton, or an autonomously driving car. For a wide range of people with limited upper- and lower-body mobility, however, interaction with robots remains a challenging problem. Due to various health conditions, these users are often deprived of basic mobility, because they are unable to manipulate standard joystick interface, most of assistive robot systems (wheelchairs, manipulators) are equipped with. In this chapter, we first identify the target audience (Section 1.1), then review various nonconventional human-machine interfaces developed to accommodate it (Sections 1.3 and 1.2), discuss properties of these interfaces (Section 1.4), outline existing approaches to robot navigation using such interfaces (Section 1.5). We present detailed literature reviews on these methods in corresponding chapters of the dissertation, as explained in Section 1.6. Finally, Section 1.7 summarizes the contributions of this work.

1.1 Disabilities and their prevalence

To understand what audience we are trying to help, and to look at the problem from users’ perspective, it is beneficial to examine various degrees of disabilities. Table 1.1 presents a matching of health conditions to the corresponding levels physical capabilities, and provides the prevalence data for the United States.

Paraplegia is a severe or complete loss of motor function in the lower extremities and
lower portions of the trunk. When four limbs are affected, the condition is known as Quadriplegia (Tertetraplegia in Europe). It can be caused by trauma (such as a traffic collision, diving into shallow water, a fall, a sports injury), disease (such as transverse myelitis, multiple sclerosis, or poliomyelitis), or congenital disorders (such as muscular dystrophy). According to (Quadriplegia and Paraplegia Information and Infographic 2017), at least in 98% of cases Paraplegia and Quadriplegia are caused by a spinal cord injury (SCI).

When Quadriplegia is combined with anarthria (loss of speech), but consciousness is preserved, the condition is known as Locked-in syndrome (LIS) (Smith and Delargy, 2005). LIS has been classified into three categories:

**Classic** Quadriplegia and anarthria (loss of speech) with preserved consciousness and vertical eye movement

**Incomplete** The same as classic but with remnants of voluntary movement other than vertical eye movement

### Table 1.1: A classification of disabilities and the prevalence in the US

<table>
<thead>
<tr>
<th>Body function</th>
<th>Healthy state</th>
<th>Paraplegia&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Quadriplegia&lt;sup&gt;b&lt;/sup&gt;</th>
<th>Locked-in syndrom (LIS)&lt;sup&gt;c&lt;/sup&gt;</th>
<th>Vegetative state&lt;sup&gt;d&lt;/sup&gt;</th>
<th>Coma&lt;sup&gt;e&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower body mobility</td>
<td>+</td>
<td>severe or complete loss</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Upper body mobility</td>
<td>+</td>
<td>+</td>
<td>loss remnants</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Neck movement</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Speech</td>
<td>+</td>
<td>+</td>
<td>vertical movement</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Facial expressions</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Eye movement</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Awareness</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Arousal</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Prevalence&lt;sup&gt;f&lt;/sup&gt; in the US</td>
<td>99,921&lt;sup&gt;g&lt;/sup&gt;</td>
<td>40.6&lt;sup&gt;g&lt;/sup&gt;</td>
<td>36.7&lt;sup&gt;g&lt;/sup&gt;</td>
<td>&lt; 0.1&lt;sup&gt;h&lt;/sup&gt; - 1.0&lt;sup&gt;i&lt;/sup&gt;</td>
<td>0.2-3.4&lt;sup&gt;i&lt;/sup&gt;</td>
<td>?&lt;sup&gt;k&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

<sup>a</sup>Based on definition in MeSH D010264  
<sup>b</sup>Based on definition in MeSH D011782  
<sup>c</sup>Classification from (Smith and Delargy, 2005)  
<sup>d</sup>Based on (Laureys, Owen, and Schiff, 2004)  
<sup>e</sup>Based on definition in MeSH D003128  
<sup>f</sup>Estimated number of people managing the condition at any given time per 100,000 of population  
<sup>g</sup>Based on (Spinal Cord Injury Facts & Statistics 2017) and US population estimated at 320 million  
<sup>h</sup>According to (Orphanet: Locked-in syndrome 2017).  
<sup>i</sup>Assuming “several thousand” people in the US as reported by (gocognitive, 2011)  
<sup>j</sup>(Pisa et al., 2014)  
<sup>k</sup>No data found
Many people with locked-in syndrome do not live beyond the early (acute) stage due to medical complications. However, others may live for another 10-20 years and report a good quality of life despite the severe disabilities caused by the syndrome (Locked-in syndrome | GARD 2017). The prevalence rate of LIS is unknown. In (Beaudoin and De Serres, 2010), it is estimated that less than 1% of all strokes result in LIS. Orphanet, a European website providing information about orphan drugs and rare diseases, estimates the prevalence as <1/1,000,000 (Orphanet: Locked-in syndrome 2017). Niels Birbaumer in (gocognitive, 2011) reports “several thousand” people in the US.

LIS can be caused by trauma, poisoning, brainstem stroke, multiple sclerosis, and several other conditions (Smith and Delargy, 2005). One of the causes of LIS which has recently received a lot of research attention in the world is Amyotrophic lateral sclerosis (ALS). ALS is a disease that causes the death of neurons which control voluntary muscles. ALS is a progressive disease, that eventually takes away the ability to walk, dress, write, speak, swallow, and breathe and shortens the life span. How fast and in what order this occurs is very different from person to person. While the average survival time is 3 years, about twenty percent of people with ALS live five years, 10 percent will survive ten years and five percent will live 20 years or more (The ALS Association 2017). According to (The ALS Association 2017), the prevalence of ALS in the US is estimated as 6.2 per 100,000, with approximately 6,000 people diagnosed with ALS each year.

Awareness or sufficiently intact cognitive abilities is what distinguishes LIS from what is known as vegetative state (VS). Patients in a vegetative state (also known as unresponsive wakefulness syndrome) open their eyes spontaneously, but show only reflexive behavior (Erp et al., 2015). Finally, if the individual is in a profound state of unconsciousness associated with depressed cerebral activity from which he cannot be aroused, this is known as coma.

The primary group that experiences challenges using conventional human-machine interfaces (such as joystick, touchscreen, keyboard) are people suffering from quadriplegia and locked-in syndrome. A number of nonconventional human-machine interfaces have been designed to accommodate this audience. In the following two sections, various types
of these devices are discussed.

1.2 Nonconventional human-machine interfaces

A good overview of nonconventional communication systems for people with severe motor disabilities can be found in (Pinheiro et al., 2011). In this section, we will discuss methods that are not related to brain-computer interfaces (BCIs). The next section covers BCIs.

Users who have maintained some mobility can often operate a single switch interface. It is a simple mechanical button (Figure 1-1a) or a sensor panel which is usually accompanied with what is called a scanning interface (Figure 1-1b): various options are displayed on the screen, and the user has to push the button when the desired option is shown.

With the Sip-and-puff system (SNP) (Mougharbel et al., 2013) (Figure 1-2), the operator can send four distinctive commands to his wheelchair by blowing air in or out of a tube. SNP recognizes four different commands, hard sip, soft sip, hard puff, and soft puff. The system, however, typically has to be calibrated for each user individually.

A different and newer method is the Tongue Drive System (TDS) developed by a team at Georgia Tech (Kim et al., 2013) (Figure 1-3). This system uses two magnetic sensors placed on the side of the operator’s head and a magnetic tongue barbell. By moving the magnetic barbell to specific spots of their mouth, users are able to control the direction and speed of the wheelchair with better results than similar hands-free systems.
Figure 1-2: Sip-and-Puff system to drive a wheelchair (Mougharbel et al., 2013)

Figure 1-3: Tongue Drive System (TDS) (Kim et al., 2013): (A) TDS headset; (B) System-level diagram of the TDS; (C) Tongue positions; (D) Magnetic tongue barbell made of titanium, which contains a small disk-shaped magnetic tracer (4.8 mm diameter x 1.5 mm thick).
Eye and gaze tracking have also been utilized as a human-machine interface. Eye tracking is a research topic on its own (Duchowski, 2009; Bojko, 2013; Holmqvist et al., 2015). Eye trackers have been used in the research on the visual system, in psychology, in psycholinguistics, marketing. There are three main types of these devices (based on the principle of operation): (1) **Eye-attached tracking** measures the movement of an object attached to the eye, normally a special contact lens with an embedded mirror or a magnetic field sensor. This technique, however, being invasive, is not typically used as a human-machine interface (HMI). (2) Noninvasive **optical tracking** employs a video camera or other specially designed optical sensor to measure the eye motion. This approach is the most popular for HMI applications, and varies widely in the accuracy and complexity from a single consumer-level video camera, to a professional video-oculography (VOG) head-mounted mask (Furuya et al., 2016) (Figure 1-7a). (3) **Electrooculography** (EOG) utilizes the electric dipole properties of a human eye, in which the cornea and retina are the positive and negative poles, respectively. Several electrodes are placed around the eye, such that the movement of the eyes is translated into potential differences between the electrodes. The advantages of this technique is that it works in total darkness and even if the eyes are closed, and, since it requires low computational power, it can be implemented as an embedded wearable system (Vidal et al., 2012) (Figure 1-4b).

Another family of HMIs is speech recognition systems (L. Rabiner and Juang, 1993; Jurafsky and Martin, 2008; Yu and L. Deng, 2014). Speech recognition is, again, a sep-
arate interdisciplinary research area. It is the process of transcribing the sounds captured by a microphone into a written text. Speech recognition can be as simple as identifying one voice command from a set of predefined list of commands, or as complex as understanding the semantics of non-structured human conversation. Due to the advances in the development of hidden Markov acoustic models, n-gram statistical language models and artificial neural networks, particularly deep learning in recent years, speech recognition has gone from a single-speaker recognition system with a ten word vocabulary in 1950s to speaker-independent intelligent systems adopted in a wide variety of applications (Juang and L. R. Rabiner, 2005). A notable example of an intelligent use of a speech interface for wheelchair navigation is the MIT Intelligent Wheelchair (Hemachandra et al., 2014; Walter et al., 2014) where the wheelchair can learn the layout of its environment through a narrated, guided tour given by a human.

HMIs based on acoustic speech recognition have several inherent limitations (Denby, 2013): (1) their performance degrades in the presence of ambient noise, (2) audible speech may interfere with surroundings, (3) they do not preserve the privacy of the communication/control, (4) certain health conditions hinder audible communication. Various techniques have been developed to overcome these limitations and enable speech communication without producing audible sounds. Collectively they are known as Silent speech interfaces (SSI) and can be divided into several groups.

- **EMG-based SSIs** use several electrodes placed on articulatory muscles (Figure 1-5a) to acquire electromyographic (EMG) signal and convert it into speech or text. The recognition is possible when the subject utters the speech silently. This method is also known as subvocal recognition (SVR). Most of these recognizers use Hidden Markov Model with Gaussian Mixture Model as its base (Al_safi and Alhafadhi, 2015). In (Y. Deng, Heaton, and Meltzner, 2014), Word Error Rate with speaker-dependent model is reported at 10%. The main challenges are adaptation to different speaking modes and achieving session-independent systems.

- A newer approach is to record Non-audible murmur (NAM) which is defined as a “special speech that consists of articulated respiratory sound without vocal-fold vi-
Electromyography (EMG)-based SSI (Janke et al., 2012) 

Attachment of NAM microphone (Nakamura et al., 2012) 

Using video and ultrasound for SSI (Cai et al., 2011) 

Figure 1-5: Various techniques for Silent speech interfaces (SSIs). Abbreviations: EMG - Electromyography, NAM - Non-audible murmuration transmitted through the soft tissues of the head” (Nakamura et al., 2012). A special stethoscopic NAM microphone is utilized for this purpose. It is attached to the sternocleidomastoid muscle as shown in Figure 1-5b.

- A combination of ultrasound and video streams of the tongue and lips has also been successfully utilized for a continuous speech recognition. In (Cai et al., 2011), word-level recognition accuracy as high as 72% has been demonstrated on a single subject (syntactic information via domain-specific language has been used). They also suggest that nearly real-time recognition can be possible.

- Electromagnetic articulography (EMA) is yet another method for SSI. Precisely positioned induction coils around the subject’s head (Figure 1-6a) produce an electromagnetic field that induces a current in small sensor coils that are placed on the subject’s tongue and other articulating organs (Figure 1-6b). The monotonous relation between the induced current and the distance from the stationarity coils allows to determine positions of sensor coils, and thus to track the movement of the tongue and lips. In (J. Wang et al., 2012), a support vector machine (SVM) was trained using pre-segmented articulatory movement data and known sentences. The average accuracy was 94.89% with an average latency of 3.11 seconds was demonstrated.
Overall, the alternative modalities of silent speech interfaces, while solving the problems associated with audible speech recognition listed above, introduce new challenges. Reliable speaker- and session-independent real time solution is still a goal to reach. At the moment, the detected outputs of these systems are of probabilistic nature, often limited to a predefined list of commands or sentences, especially when speaker-independent silent speech recognition is attempted.

Electromyography (EMG) is yet another method of implementing alternative HMIs. When the brain commands a muscle to contract, the signals through motor neurons are sent to fibers. The depolarization of the membrane fiber generates an electrical impulse (potential) in the vicinity of the muscle fibers. The summation of the action potentials propagating through the fibers yields the motor unit action potential (MUAP). To maintain
the muscle force, the MUAPs are fired repeatedly at $7 - 20\, Hz$, forming the sequence known as MUAPT (MUAP train). The ensemble of MUAPTs from all motor units in the range, as measured by electrodes, is the EMG signal (Pinheiro et al., 2011). There are two methods of reading the signal: accurate highly localized invasive intramuscular EMG which is typically used in a clinical setting for detecting medical abnormalities, and noninvasive surface EMG (sEMG) that is more common for HMI applications. The state-of-the-art in surface EMG is covered in a recent book (Merletti, Farina, and Wiley-Blackwell Online Books, 2016).

A number of commercial products are available for EMG control. For example, a single-channel EMG bluetooth sensor Impulse™ from AbleNet Inc (Figure 1-9a) that can be attached to various parts or human body or a more recent hybrid Myo™ gesture control armband (Thalmic Labs, 2017a) from Thalmic Labs Inc(Thalmic Labs, 2017b) that integrates surface EMG sensors, gyroscopes and accelerometers to distinguish between hand gestures that can be used to control various devices (Figure 1-9b). As noted in (Pinheiro et al., 2011), however, if the operator is capable of physically moving any part of his or her body, it is often more practical to use a pressure device or a camera to track the motion instead of an EMG device, especially for switch-like control. Low-cost brain-computer interfaces, such as Emotiv EPOC (EMOTIV 2017) or MindWave™ headsets from NeuroSky (NeuroSky 2017), can measure EMG-signals to detect facial muscles activity and thus enable robot control with facial gestures (Sinyukov et al., 2014a).

There are three most common ways to use EMG-signal for HMI. (1) **Switch-like control** detects two muscle states: active and not active. The simplest version is the binary output from a single muscle that can be used with a scanning interface or a communication interface based on Morse or similar code (Park et al., 1999). Higher number of commands are possible when activation is measured from multiple muscles at the same time, such as done with Myo, Emotiv EPOC and MindWave devices discussed above. (2) **Proportional control** maps the muscle activation level to a certain controlled variable, such as the displacement of a cursor on the screen. For example, in (Huang, C.-H. Chen, and Chung, 2006) facial muscles Orbicular, Massetter and Mentalis were employed to control a 2D pointing device. In (Perez-Maldonado, Wexler, and Joshi, 2010) a 2D cursor control is
achieved by measuring power levels of two different frequency bands extracted from the EMG signal that was recorded from a single muscle. (3) Silent Speech Interfaces based on EMG were discussed above.

### 1.3 Brain-computer interfaces

A brain-computer interface (BCI), also known as a brain-machine interface (BMI), or a direct neural interface (DNI), is a hardware and software system that establishes a direct communication channel between a brain and a computer to control real or virtual external devices. It is a relatively recent technology, despite the fact that the first electroencephalogram (EEG) of a dog was measured in 1913 by a Russian physiologist Pravdich-Neminsky.
The method was then used to obtain a human EEG in 1929 by a German neurologist Berger (Berger, 1929). Later in 1938, Berger and Herbert Jasper suggested that EEG could also be used for human communication (X. Wang et al., 2016). By now, more than a hundred years since the first measurement of EEG, a number of BCI methods have been developed and proven to be working technologies that can be employed to detect human intent.

Most of BCIs can be conceptually divided into signal acquisition, pre-processing, feature extraction, and classification components (Bi, X.-a. Fan, and Liu, 2013a), as shown in Figure 1-10. Signal acquisition is discussed in the following two sections, where as the details of feature extraction and classification are out of scope of this work, and are just briefly discussed in Section 1.3.3.

1.3.1 Methods of signal acquisition used for BCI

Various regions of the cerebral cortex are responsible for different functions of the human organism as depicted in Figure 1-11. Figure 1-12 classifies methods of BCI signal acquisition based on their principle of operation. To analyze neural activity, any modern BCI directly or indirectly measures either: (a) the electric activity of neurons, or (b) the blood flow (haemodynamic response) associated with neuron activation. Now in case (a), the electric signals can be measured either: directly with electrodes (depending on the electrode placement in Figure 1-12: EEG, FP, ECoG, AP, LFP, discussed below), or indirectly by analyzing weak magnetic fields generated by the corresponding electric cur-
Figure 1-11: Motor and sensory regions of the cerebral cortex (Blausen.com staff, 2014)

Figure 1-12: Types of brain-computer interfaces based on their principle of operation (He et al., 2013; Krucoff et al., 2016). Abbreviations: AP - Action potentials, dHb - Deoxygenated hemoglobin, ECoG - Electrocorticography, EEG - Electroencephalography, EMG - Electromyography, fMRI - Functional magnetic resonance imaging, fNIRS - Functional near-infrared spectroscopy, FP - Field potentials, Hb - Oxygenated hemoglobin, LFP - Local field potentials, MEG - Magnetoencephalography, PET - Positron emission tomography.
Invasive methods provide the highest spatial and temporal resolution, since they can be placed closer to the area of interest, but suffer from the electrode deterioration and carry the risks associated with the surgical intervention.

- Intracortical recordings use microelectrode arrays (MEA) that are implanted into the...
cortex. They enable multichannel parallel recording of single-neuron activity (action potentials or APs) and local field potentials (LFP) (He et al., 2013). LFP is a signal resulting from the combination of electric currents flowing within a small volume of nervous tissue. Among the prominent results: a monkey that could control a robotic arm via this BCI to feed itself (Velliste et al., 2008), long term computer control in a patient with tetraplegia (Simeral et al., 2011a).

- Electrocorticography (ECoG) is less invasive, as the electrode arrays are placed over the cortex (but under the dura mater). Since most large cortical neurons are oriented perpendicular to the cortical surface, this method allows to measure the averaged activity of local neurons, thus it has a lower spatial resolution and lower signal-to-noise ratio than intracortical recordings. The method was mainly used to identify cortex zones that generate epileptic seizures.

Noninvasive methods do not require surgery and usually are based on the classification of different mental states, rather than decoding kinematic parameters as typically done in invasive BCIs (He et al., 2013, p. 103).

- Electroencephalography (EEG) is the most common method for BCI signal acquisi-
tion due to its practicality (noninvasive, portable, high temporal resolution). Up to 256 electrodes can be placed on the scalp to measure the voltage fluctuations on the scalp resulting from ionic current within the neurons of the brain. Compared to invasive methods, EEG signal is more stable, but has lower signal-to-noise ratio (He et al., 2013, p. 103). Figure 1-13 demonstrates the International 10-20 system (Nuwer et al., 1998) of arranging EEG electrodes. EEG has a low spatial resolution, but good temporal resolution.

- **Magnetoencephalography** (MEG) measures magnetic fields induced by electrical activity of neurons. While it is sensitive only to the tangential components of neural current sources, as opposed to EEG that can detect both tangential and radial components, it has a slightly better spatial resolution and approximately the same temporal resolution. Also, the magnetic signals are less distorted by the skull than electric signals measured by EEG. There are systems that combine EEG and MEG in one device. The main disadvantage of MEG is that it requires a shielded room to minimize external magnetic noises, and is, thus, not portable and more expensive than EEG.

- **Functional magnetic resonance imaging** (fMRI) and real-time fMRI (rtfMRI) are based on the haemodynamic response discussed above. They measure the dHb concentration based on the brightness of the voxels in the MRI images. The main advantage of this method is that it can build a full 3D image of the brain with a great spatial resolution. The fundamental limitation of fMRI is the intrinsic inertia of the metabolic process. It also requires a non-portable MRI machine.

- **Functional near-infrared spectroscopy** (fNIRS) projects infrared light into the skull, which penetrates the tissue as deep as 3-4cm. The reflected light is analyzed to measure the relative concentrations of Hb and dHb which have different absorption spectra. The main advantages of fNIRS are portability, affordability and flexibility of use. The main limitation is the low temporal resolution due to the inertia of haemodynamic response.

All of these methods of measuring brain activity have been successfully used for brain-computer interfaces. Figure 1-14 demonstrates a graphic comparison of temporal and spa-
Figure 1-14: Comparison of temporal and spatial resolutions of brain imaging methods based on Table 1.2. Red markers indicate invasive methods, blue markers indicate noninvasive methods.

Spatial resolution characteristics for various brain imaging techniques constructed using data from Table 1.2. As we can observe from Table 1.2 and as it is noted in (Ramadan and Vasilakos, 2017), among noninvasive techniques, EEG-based BCI is by far more practical for navigating a real object, unless it’s teleoperated. Indeed, MEG and fMRI are not portable, and fNIRS has a several second delay.

1.3.2 Types of signals used for BCI

Electric processes in the brain in an idling state have oscillatory properties. In the frequency domain, EEG signal is divided into several bands: delta band (0.5-3 Hz), theta band (4-7 Hz), alpha band (8-13 Hz), beta band (14-30 Hz), gamma band (>30 Hz). Oscillations over the sensorimotor cortex in the 8-13 Hz frequency range are also known as mu rhythms. These oscillations are considered to be a result of synchronized firing of neurons that form complex feedback loops (He et al., 2013). An increased synchronization of neuron firing, thus, results in a higher amplitude of the oscillations in a given frequency band, whereas a desynchronization will decrease the amplitude of oscillations. When such synchronization or desynchronization is caused by a specific event (endogenous or exogenous), they are called event-related synchronization (ERS) and event-related desynchronization (ERD).
<table>
<thead>
<tr>
<th>Signal type</th>
<th>Evoked signals</th>
<th>Spontaneous signals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Visual (SSVEP)</td>
<td>Auditory (AEP)</td>
</tr>
<tr>
<td></td>
<td>t-VEP</td>
<td>f-VEP</td>
</tr>
<tr>
<td>Brain area</td>
<td>occipital lobe (visual cortex)</td>
<td>parietal&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
<tr>
<td>Frequency band/</td>
<td>4 Hz&lt;sup&gt;c&lt;/sup&gt;</td>
<td>6-12 Hz&lt;sup&gt;d&lt;/sup&gt;</td>
</tr>
<tr>
<td>flickering rate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit rate,</td>
<td>30</td>
<td>100+&lt;sup&gt;d&lt;/sup&gt;</td>
</tr>
<tr>
<td>bits/min</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Training</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>required</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<sup>a</sup>(Dal Seno, Matteucci, and Mainardi, 2010)
<sup>b</sup>(Guo, Gao, and Hong, 2010)
<sup>c</sup>(He et al., 2013, Sec 6.4)
<sup>d</sup>(Gembler, Stawicki, and Volosyak, 2016)
<sup>e</sup>No data found
<sup>f</sup>(Penny et al., 2000)
<sup>g</sup>(Buttfield, Ferrez, and J. R. Millan, 2006)

Table 1.3: Comparison of BCI signals. Data without footnotes are taken from (Ramadan and Vasilakos, 2017).
Abbreviations:
AEP - Auditory evoked potential,
c-VEP - Code-modulated visual evoked potential,
f-VEP - Frequency-modulated visual evoked potential,
ErrP - Error potential,
MI - Motor imagery,
OC - Operant conditioning,
SCP - Slow cortical potentials,
SMR - Sensorimotor rhythms,
SSVEP - Steady state visually evoked potential,
t-VEP - Time-modulated visual evoked potential
Based on the necessity of an external stimulus, BCI signals are classified (Table 1.3) into: *evoked signals*, *spontaneous signals*, and *hybrid systems* which combine signals of both types.

**Evoked signals** are generated unconsciously when subject is presented external stimuli. While currently evoked signals provide the highest *information transfer rate* (ITR), usually they are more exhaustive for the user due to the constant presence of the stimulus. Several types of evoked signals have been successfully used in BCI-applications:

- *Steady state evoked potentials* (SSEP) are brain signals that are generated in response to a periodic stimulus. While the stimulus can be of different nature (flickering image, modulated sound, vibrations (Ramadan and Vasilakos, 2017)), visual stimuli were shown to be the most effective for BCI-applications. The signal in this case is referred to as *Visually evoked potential* (VEP) or, more specifically, *Steady state visually evoked potential* (SSVEP). In SSVEP-based BCIs, several flickering targets are presented to the user, such as buttons or other elements of user interface, or checkerboards. The flickering can be modulated by the frequency (f-VEP), time (t-VEP), pseudorandom code (c-VEP), or phase (p-VEP) (He et al., 2013, Sec 6.4). In a recent study (Gembler, Stawicki, and Volosyak, 2016), the maximum ITR of 130 bits/min has been reported for 15 simultaneously displayed targets, while one user could select one target out of 84 with 91.30% accuracy. They also confirm that in SSVEP-based BCI, speed and accuracy are affected by the number of targets.

- *P300* is described as a positive peak in an EEG recording approximately 300 ms after a certain stimulus is presented. P300 occurs in the context of the “oddball paradigm” (He et al., 2013): various stimuli are presented to the subject (visually or audibly). When a given stimulus is unexpected, rare, or particularly informative, P300 phenomenon occurs (Dal Seno, Matteucci, and Mainardi, 2010). This signal is associated with the subject reaction, rather than attributes of the stimulus. P300-based BCI is a rare case of a BCI that has been in long-term use by people with severe disabilities at their homes (Sellers, Vaughan, and Wolpaw, 2010). P300 is stronger
in parietal lobe (Dal Seno, Matteucci, and Mainardi, 2010).

- **Auditory evoked potential** (AEP) using a similar “oddball paradigm” has also been demonstrated to allow subjects with normal hearing to select one digit out of eight that were audibly spoken in sequences (Guo, Gao, and Hong, 2010). AEP is characterized by a negative shift around 200 ms and a late positive peak at around 500 ms in the parietal area. The advantage of AEP is that it can help people suffering **total LIS**, who are unable to use vision.

- **Error potential** (ErrP) is generated when subject makes a mistake or machine behaves differently from the user intent. ErrP is described as a negative shift in the electric potential over the fronto-central region occurring 50–100 ms after an erroneous response and a subsequent positive shift in the parietal region, whose maximum occurs between 200 and 500 ms after the error (Dal Seno, Matteucci, and Mainardi, 2010).

**Spontaneous signals** are generated by subject voluntarily without any external stimulation. There are several subtypes in this group:

- **Sensorimotor rhythms** (SMR) are the oscillations detected in the alpha, beta and gamma frequency bands as measured over the sensorimotor cortex (Figure 1-11). They change with motor and somatosensory function (He et al., 2013). The amplitude of these oscillations (rhythms) can be modulated by the subject using either **motor imagery or operant conditioning** (Ramadan and Vasilakos, 2017). It has been
repeatedly demonstrated that subject’s motor intention (such as an imagined movement of a hand or a foot) can increase or decrease the amplitude of oscillations in beta and mu bands over the motor cortex area. This method is known as Motor imagery (MI) and has been used in various BCI control applications. Training is not required for MI, however may improve the performance. Alternatively, through long training (weeks or months), the subjects can develop his/her own mental strategy to modulate alpha and beta rhythms. This method is known as operant conditioning. When learning to control SMRs, auditory feedback can be used instead of visual feedback, though less effectively (Nijboer et al., 2008).

- **Slow cortical potentials** (SCP) is a low frequency signal (<1Hz) caused by shifts in the depolarization levels of pyramidal neurons in cortex. It is detected in the frontal and central parts of the cortex (He et al., 2013) as a positive or negative change that may last from milliseconds to several seconds. After a long training using operant conditioning, users can learn to control SCP.

- **Non-motor cognitive tasks**, such as music imagination, mathematical computation, alphabet reciting, silent signing, are also used to drive BCI. Mental tasks are often combined with motor imagery (M. C. Dobrea and D. M. Dobrea, 2009; Hortal et al., 2013). In this case, the measurement can be taken from multiple areas of the brain: prefrontal (Power, Kushki, and Chau, 2012), central, parietal, occipital (M. C. Dobrea and D. M. Dobrea, 2009).

### 1.3.3 BCI signal processing

A detailed discussion on BCI signal processing is out of scope of this work. The reader may find more information in (Lotte et al., 2007; Lakshmi, Prasad, and Prakash, 2014; Ilyas, Saad, and Ahmad, 2015). For the sake of completeness, we note that for Signal pre-processing (Figure 1-10) the following approaches have been successfully employed: Common Spatial patterns (CSP) (Koles, Lazar, and Zhou, 1990), Principle Component Analysis (PCA), Common Average Referencing (CAR), adaptive filtering, Independent Component Analysis (ICA) and digital filters. To extract features from the preprocessed signal **Feature**
extraction unit in Figure 1-10) the following methods have been used: Principal Component Analysis (PCA), Independent Component Analysis (ICA), Wavelet Transformations (WT), Wavelet Packet Decomposition (WPD), Auto Regressive (AR), and Fast Fourier Transform (FFT) have been successfully utilized. Finally, once the features are extracted the extracted vectors need to be classified. The most popular classifiers used for BCIs are Artificial Neural Network (ANN), k-nearest Neighbour (k-NN), Linear Discriminant Analysis (LDA), and Support Vector Machines (SVM) (Hastie et al., 2009).

A number of BCI signal acquisitions systems, as well as open source and proprietary software packages for BCI signal processing are available. A comprehensive list of software and hardware solutions can be found in (Ramadan and Vasilakos, 2017, Sec. 6.2).

1.3.4 BCI is practical

Here are the main observations that we would like to make out of this section.

- BCIs are naturally probabilistic and severely constrain the information transfer rate (ITR) (no more than few bits per second).

- They can provide synchronous (constant frequency) and asynchronous (spontaneous) control signals to the application.

- The most practical noninvasive BCI techniques are usually designed as a selection of one out \( N \) options.

- BCIs are practical, they are not just pure research anymore, low-cost solutions are available, some BCIs (such as P300-based BCI) have been in long term use by severely disabled people in their daily life. There are more than 70 BCI companies in the world (Ramadan and Vasilakos, 2017).

- Despite the known limitations, for people suffering severe disabilities, such as LIS, BCI is the only way to interact with the world.
1.4 Classification of human-machine interfaces (HMIs)

As we can see, human-machine interfaces designed for people with severe disabilities share common properties. In this section, we will classify these systems.

A reasonable classification for noninvasive human-machine interfaces used for prosthetics, wheelchair control and other applications is proposed in (Lobo-Prat et al., 2014) (Table 1). The HMIs are categorized based on the utilized human system, physiological phenomena, type of signal and sensor, transduction principle, interface with the body, in other words based on the modality of the interaction.

We can classify human-machine interfaces by the properties of the output space into continuous and discrete. In fact, this classification applies to any non-hybrid HMIs.

**Continuous HMIs** output a continuous (or discrete, but with high resolution) value or values. They can be divided based on the dimensionality of the output space:

- **1D HMIs.** Examples: a car steering wheel, the classic 1D cursor control with an EEG-based BCI (Wolpaw, McFarland, et al., 1991) where mu-rhythms amplitudes were translated to the vertical movement of the cursor. It took several weeks to train the subjects to reliably control the cursor.

- **2D HMIs.** Examples: a wheelchair joystick that allows to control two variables independently by tilting the joystick along two perpendicular axes; 2D cursor control (Wolpaw and McFarland, 1994) by independently modulated mu-rhythms in right and left central sulci; more recently, 2D cursor control with intracortical microelectrode arrays (Simeral et al., 2011b).

- **3D HMIs.** Examples: a joystick with a rotating handle (used for manual robotiv arm control), 3D cursor control with EEG-based BCI using motor imagery (McFarland, Sarnacki, and Wolpaw, 2010).

- **HMIs of higher dimensions.** Example is (Velliste et al., 2008) where a monkey could control the 3D position of the robotic end effector and the 1D gripper to feed itself via a cortical (invasive) BCI.
Continuous HMIs, however, may have a major noise component especially when it comes to alternative modalities, such as BCI. As a demonstration of the noise, the 2D cursor trajectories from (Simeral et al., 2011b) are shown in Figure 1-16, where the task was to move the cursor from the center to one of the eight targets (grey circles) and in the reverse direction.

**Discrete HMIs** are designed to choose one option out of a finite number of possible options. Depending on whether the HMI provides probabilistic output or not, they can be classified into:

- *Probabilistic HMIs*. These output either a vector with probability values for all available commands/options, or a single index of the chosen command with a single probability value.

- *Nonprobabilistic HMIs* which simply output a single index of the chosen command.

HMIs can also be classified based on their temporal properties into:

- *synchronous HMIs* where the output is generated with constant period. All SSVEP-based BCI are of this type.

- *asynchronous HMIs* where the output can be generated at any moment at operator’s will. Non-continuous spontaneous BCIs belong to this group.
• *continuous HMIs* which are usually high frequency synchronous HMIs that constantly generate output values. The cursor control examples above belong to this type.

A number of other characteristics of HMIs play an important role in designing control systems. (Lobo-Prat et al., 2014) identify: intuitiveness, robustness, independence (should not require assistance of other people), customization (adaptability to the user). For discrete HMIs, the most critical quantitative characteristics are the number of commands, response time (latency) and accuracy of detection. Typically for nonconventional HMIs, the combination of these three characteristics is such, that the maximum information transfer rate (ITR) permitted by the given HMI is relatively low. As it can be seen from Table 1.3, among non-invasive BCI, the highest ITR is around 2 bits/sec. This is equivalent to choosing 1 option out of 4 once per second. A human-machine interface with a low ITR is known as a *low throughput human-machine intreface* (LTHMI or LTI). Discrete HMIs with low ITRs are the primary focus of this dissertation.

A recent trend in designing robot systems with LTIs is to combining different modalities. This is known as hybrid BCIs or more generally hybrid LTIs. For example, in (H. Wang et al., 2014) the wheelchair is controlled using motor imagery, P300 and EOG (eye-blinking detection). This combination allows to send directional commands and regulate the speed of the wheelchair. A good review of hybrid systems can be found in (Amiri, Fazel-Rezai, and Asadpour, 2013).

### 1.5 Navigation and control with low throughput human-machine interfaces

Most of the research on robot control via low throughput interfaces is tied to a certain type interface, even though many challenges associated with the design of such systems are agnostic to the modality of human-machine interaction. One of the reasons for this could be that designing a specific LTI is a research challenge by itself. Scholars are primarily focused on improving the characteristics of the their LTI, and the control examples are
This dissertation addresses the problem of control (mainly navigation) via LTI from the robotics perspective. The classic robot navigation problem is often decomposed into several layers: low level motion control, medium level local planning, high level global planning (Figure 1-17). This hierarchical model is to a certain extent applicable to the problem of navigation via LTI as well.

In (Bi, X.-a. Fan, and Liu, 2013a), the control methods are divided into two groups: direct control and shared control. Direct control translates brain signals directly into motion commands, whereas in shared control systems, the user and an intelligent controller share the control over the robot. This definition of shared control was introduced in (Sheridan, 1992) and covers a large group of robot applications. With respect to the diagram in Figure 1-17, direct control is equivalent to communicating directly to the motion control layer, while the shared control can be anything above, where autonomous navigation components are involved. While (Bi, X.-a. Fan, and Liu, 2013a) focuses on mobile robots integrated with EEG-based BCI, these definitions are applicable to other types of LTIs.

For navigation via LTI tasks, we would like to expand this classification. More specifically, we would like to divide shared control methods into shared steering control and shared position control. Shared steering control maps HMI commands to robot velocities or local trajectories, but, as opposed to direct steering control, it also incorporates some robot intelligence (usually related to obstacle detection or direction prediction), whereas...
in position control, the robot is fully responsible for the navigation, while the user only needs to define/select his/her intended destination. Relative to the diagram in Figure 1-17, direct steering control interacts directly with the motion control level, shared steering control interacts with the local planning level, and position control interacts with the global planner. For direct and shared steering controls different steering models exist. We will briefly introduce them, and then discuss each of the control approaches.

1.5.1 Steering models

Steering models depend on the properties of the HMI. For 2D continuous HMIs, joystick-like control is the most natural choice. In this case, the HMI output values can be mapped proportionally to robot velocities or accelerations. This can work both for holonomic and non-holonomic objects. For example, in (McFarland, Sarnacki, and Wolpaw, 2010) an independent control of each of the 3 dimensions of movement of a cursor in 3D space was achieved using EEG-based BCI with motor imagery commands (left and right hands and a foot). The cursor velocities were proportional to the signal intensities. While this steering model is good for a demonstration of the BCI or other type LTI characteristics, it makes it challenging to maintain a static pose if the HMI output is noisy.

For a discrete HMI with a limited set of output states, the states are usually mapped to directional commands. For a differentially driven robot, these are typically “forward”, “turn left”, “turn right”, “go backward”, “stop”. The set of directional commands can vary. For example in (Tanaka, Matsunaga, and H. O. Wang, 2005) EEG-based BCI differentiated only between “left thinking” and “right thinking” with an average recognition rate of 80%. The wheelchair would move diagonally left or right. In other cases, the number of commands can be higher: in (Achic et al., 2016), the IMU in Emotiv EPOC headset was used to measure cervical movement and translate it into steering commands, including two forward velocities.

There are also some variations in the interpretation of the steering commands. One approach is to assign constant velocities to each command. Then the command can continue to be executed until a new one is received, or its execution can be limited by time, or
the absolute value of displacement can be limited. For example, in (H. Wang et al., 2014) each rotational command is limited to a 7° turn. They control a wheelchair with a hybrid LTI, where left/right hand motor imagery is mapped to forward/backward commands in the static state or turn left/right in the moving state state P300 allows to control acceleration, and eye-blinking allows to stop the wheelchair. Another approach is to interpret the steering commands as velocity increments (constant velocity increments steering model). This can result in smoother movement of the robot. One example is (Vanacker et al., 2007), where a simulated wheelchair was controlled with three commands “forward”, “left”, and “right”. Each command interpreted as an increment or decrement for the corresponding velocity. They however had just 2 non-zero values for forward velocities (0.5 and 1 m/s), and three non-zero rotational velocities in each direction (0.2, 0.4, 0.6 rad/s), and if no command was received for a certain period of time, the corresponding velocity would be reset to 0.

An interesting steering model was used in (Royer, Doud, et al., 2010; LaFleur et al., 2013). They were able to reduce the problem of 3D navigation problem to 2D control. In their steering model, the helicopter flies forward with constant velocity, while 2D continuous EEG-commands (modulated sensorimotor rhythms) are mapped to up/down and turn left/right velocities in (Royer, Doud, et al., 2010) and accelerations in (LaFleur et al., 2013). Several subjects successfully controlled virtual and real helicopters and were able to find and pass through rings.

It is worth to mention another interesting steering model that was used in remotely controlled toy cars. The 2D pose of a toy car was controlled with a single button remote control. Once turned on, the car would drive backwards in a circle. When the remote control button is pressed, the car would start moving forward with constant velocity. With enough free space around the car and good skills in timing the button presses, the car could have been driven to any pose (position and orientation) on the plane. There is also a variation for a differentially driven toy cars, where the car has three state: static, rotating in place with constant velocity and moving forward with constant velocity. The single remote control button simply switches between the states (Studio Avalanche, 2015).

48
1.5.2 Direct steering control

Direct steering control is the most straightforward way to connect an LTI to a mobile robot. While this method has clear practical disadvantages (tendency for collisions with obstacles, either slow or inaccurate navigation, early user fatigue), this approach is still popular among researchers, because it allows to demonstrate vividly the advantages of a given LTI, be it a BCI or an SSI. Since this dissertation is focused on the shared control, we will just briefly discuss a few relatively recent papers on the direct steering control. More examples can be found in (Bi, X.-a. Fan, and Liu, 2013a).

In (P. L. Lee et al., 2012), three SSVEP stimuli are mapped to forward, left, and right commands that are passed to a small differentially driven robot. Another example is the quadcopter control (LaFleur et al., 2013) discussed above. In (Herweg et al., 2016), 10 elderly subjects steered a wheelchair in a virtual environment using standard directional commands (“forward”, “left”, “right”, “backward”) detected with a P300-based BCI with tactile stimuli (four stimulators placed on left and right thighs, abdomen and lower neck). After training, reported accuracy was 92.56%, and ITR=4.98 bits/min. Such experiment with a real wheelchair would be impossible due to inevitable collisions (reported in the paper). In (Kucukyildiz et al., 2017), the wheelchair is steered with three commands: left, right and forward. First, with an EMG armband from Thalmic Labs that is used to differentiate between three gestures: making a fist, hand release and waving the hand right, and then with a consumer grade Emotiv EPOC EEG-based BCI where the commands are mapped to mental tasks: relaxing, math problem solving and text reading. While authors report obstacle detection with a depth camera, they don’t describe obstacle avoidance mechanism, and the experiments are only present for obstacle-free paths.

1.5.3 Shared steering control

Shared steering control is a more sophisticated approach to navigation. It combines robot intelligence with human intelligence. Similar to the direct steering control, it receives directional commands from the HMI, but the actual robot velocities that are passed to the motion control level can be altered by the shared control system based on the information
about surrounding obstacles (usually to avoid collisions) and predicted human behavior (such as predicted destination) (Lin and Kuo, 2016; Shi, H. Wang, and C. Zhang, 2015; Vanacker et al., 2007). The state-of-the-art shared steering control systems are discussed in Section 3.1.

One of the main challenges, associated with steering control through a discrete LTI is a fine control over the destination pose. Indeed, the execution of steering commands takes certain time, which means that to reach a pose on the map with a good accuracy, the moving steps should be rather small.

### 1.5.4 Shared position control

As opposed to shared steering control, position control is designed to allow the operator to specify his desired destination by some HMI, while the navigation is executed by the robot autonomously.

Simpler shared position control systems provide the operator with a list of predefined destinations. Given the limitations of discrete LTIs, the selection is performed in several steps, and once the destination is selected, the robot would autonomously navigate there (R. Zhang et al., 2016; X. a. Fan et al., 2015). More sophisticated systems, show local goals to enable step-by-step navigation (Iturrate et al., 2009). There are also methods that combine steering and position control (Demeester et al., 2007). These and other state-of-the-art systems are discussed in Section 5.1.

(Royer and He, 2009) is one of the rare works that compares position control (goal selection) and steering control (process control) with an LTI, though in a simplified form. They compare five control paradigms developed for a motor imagery BCI to control a cursor in 2D space without obstacles, but with targets. Three paradigms are based on goal selection, and two are based on process control. In their investigation, the goal selection outperformed process control in every measure studied.

One of the ideas that we would like to put forward in this work is that steering control is in many cases a necessity, rather than a preferred choice of the operator. While in certain situations, the user may indeed want to finely control the speed of a wheelchair, for many
real life needs (or activities of daily living), the main goal of the operator is to reach a
certain destination, be it close or far away. With the advances of robotics engineering,
decreasing cost of sensors and computers, robot autonomy becomes more affordable and
practical. The wheelchair navigation is likely to become more and more autonomous, just
like it is happening with cars. Especially when it comes to navigation via low throughput
HMIs.

1.6 Structure of the dissertation

The chapters in this dissertation are organized in the order of increasing autonomy of the
robot as shown in Figure 1-17. This also corresponds to the chronological order of the work
on this project. It started with the intent of improving the motion controller of the robotic
wheelchair by replacing it with an adaptive controller. Results from wheelchair navigation
have been achieved in simulation. The background and results of this work can be found
in Chapter 2. Then a shared steering control (assistive drive) was implemented on the real
robot and integrated with several types of low throughput human machine interfaces. This
work is covered in Chapter 3. Then, to design a shared position control that would allow for
probabilistic optimizations, an algorithm for high-performance single-source path planning
on 2D grids has been developed (Chapter 4). Finally, as the main result of this dissertation,
a novel approach to shared position control via low throughput human-machine interfaces
was developed and tested on a real robot. This work is covered in Chapter 5. Each chap-
ter has its own background section where the state-of-the-art systems are discussed and
compared to the proposed solutions. Finally, Chapter 6 concludes the document.

1.7 Contributions

While this work was initially motivated and primarily focused on the development of a
robotic wheelchair integrated with a brain-computer interface, the contributions made here
span across multiple relevant areas and may find their use in other applications, as discussed
below. This work makes contributions on several levels of the control system (Figure 1-17):
1. On the low level of motion control, an adaptive controller with online parameter estimation has been developed for a differentially driven wheelchair. The distinguishing characteristic of the proposed controller is that it has motor currents as system inputs, and an arbitrary position of the wheelchair center of gravity.

   (a) Mathematical and computer models of the closed-loop trajectory tracking system were developed.

   (b) Simulation results with different initial conditions and wheelchair parameters are shown and analyzed.

2. On the medium level of local planning, a navigation framework for electric wheelchairs which integrates various low throughput human-machine interfaces with assistive navigation was developed.

   (a) Assistive navigation was developed as a local planner providing an obstacle avoidance capability for a given desired wheelchair velocity.

   (b) A series of use and test cases were developed and used to configure the system for reliable local navigation tasks.

   (c) A universal framework that integrates various low throughput human-machine interfaces (facial expression control with Emotiv EPOC, voice control with Google Glass, voice control with CMU Sphinx, and Brain-computer interface) with this navigations system was developed.

3. To enable fast probabilistic reasoning on the level of global navigation, a novel high-performance algorithm for single-source any-angle path planning on 2D grids was developed:

   (a) A well-studied problem of path planning on 2D grids was addressed from a fresh perspective: The graph model of the grid was abandoned and discrete geometric primitives were introduced to represent the propagating wave front. This allowed to use efficient Bresenham algorithms (J. Bresenham, 1965; J.
Bresenham, 1977a) to iterate through grid vertices. As a result, a novel high-performance algorithm that requires only integer addition and bit shifting operations was proposed and implemented.

(b) The mathematical analysis was developed to calculate distance error bounds.

(c) Experimental results demonstrating significant performance advantages of the algorithm compared to alternatives on a set of maps are presented.

(d) To address various computational platforms, several modifications of the algorithm, including an optimal version and a multithreaded implementation, are presented and their performance is compared.

4. A novel approach to shared position control in a known indoor environment using a low throughput human-machine interface that attempts to probabilistically minimize time to destination is proposed and implemented. The distinguishing characteristics of the solution are: the belief desired state is maintained as a probability distribution over the whole discretized map, rather than a set of individual destinations, which allows to use prior information about the user habits (such as points of interest), but does not restrict navigation to a pre-defined set of points. The inference and motion are parallelized reducing the total navigation time. The inference is performed via a selection of colored map regions which combines local and global navigation in one interface.

(a) The general theoretical formalism is developed for designing control systems with low throughput human-machine interfaces that are aimed at minimizing the control time. and model the belief desired state as a probability distribution over the state space

(b) The formalism is used to design a proof-of-concept navigation system (POC-system) for a massless point-size holonomic robot in a known map

(c) An automated test framework was designed to search for the optimal configuration of the POC-system.

(d) The POC-system was adopted for a real differentially driven wheelchair, more
than 250 navigation experiments have been conducted to demonstrate the capabilities of the navigation system. The experimental data have also been collected and are one of the contributions of this work.
Chapter 2

Motion control with an adaptive controller

Following the control hierarchy discussed in Section 1.5 (Figure 1-17), we start with the lowest level of the navigation system: motion control (Figure 2-1). The motivation for using an adaptive controller was based on the fact a wheelchair is not an object with constant parameters. First, the pose of the human operator can change if he has to lean to any side, the weight of the operator can change over time, other additional items can be placed on the wheelchair, again affecting the distribution of the mass, finally the surface under the wheelchair may vary affecting the friction coefficients. These changing parameters affect the performance of the motion control. An adaptive control system, on the other hand, can guarantee satisfactory control characteristics even when parameters of the system change.

Figure 2-1: In this chapter, we discuss the lowest level of the navigation system: motion control.
While the adaptive controller developed in this chapter showed promising results as demonstrated in simulation, it was not eventually used on the real wheelchair. This is discussed in Sections 2.5 and 2.6.

2.1 Background

This affects the performance of PID controllers normally used for wheelchair velocity control. This paper presents an adaptive con- A robotic wheelchair can be treated as a two-wheel differentially driven mobile robot. There are several models describing such systems, thus, a number of approaches to implement an adaptive motion control have been reported. For example, (Jiang and Nijmeijer, 1997) and (Carelli and Oliveira Freire, 2003) consider only robot kinematics. With inertial robots, such as robotic wheelchairs, however, ignoring system dynamics leads to undesirable tracking errors. Other researchers have investigated various adaptive controllers based on robot dynamic models of different depth. In (Das and Kar, 2006), an adaptive fuzzy logic-based controller uses voltages as control signals of the system. Most of the researchers, however, when designing adaptive controllers for differential drive robots, choose motor torques as control signals (Fierro and Lewis, 1995; Fukao, Nakagawa, and Adachi, 2000; Soetanto, Lapierre, and Pascoal, 2003; Dong and Kuhnert, 2005; Petrov, 2010). Among this group one of the most popular controller design techniques is backstepping from kinematics to dynamics (Fierro and Lewis, 1998; Fukao, Nakagawa, and Adachi, 2000; Soetanto, Lapierre, and Pascoal, 2003; Dong and Kuhnert, 2005; Künhe, Gomes, and Fetter, 2005; Martins et al., 2008). This method, while providing certain structure to control systems and simpler design, is not proved to be the most efficient one. Few researchers consider the problem in a more general formulation (Künhe, Gomes, and Fetter, 2005; Petrov, 2010), however, in (Künhe, Gomes, and Fetter, 2005) the COG (center of gravity) position is considered lying on the symmetry line of the robot which doesn’t suit our application, and in (Petrov, 2010) friction forces are ignored and the same simplified model is used for COG. An interesting approach was chosen in (Martins et al., 2008), where the inputs to the control system are desired velocities, even though a dynamic model is considered.
A common disadvantage of these implementations is the assumption that the COG of the robot lies on the symmetry axis of the robot. With the wheelchair, however, the COG may be offset in both $x$ and $y$ directions. Moreover, the use case when the COG suddenly moves from left to right or any other direction is frequent in this application. It’s especially interesting how an adaptive controller would react to such a change. Thus in our model we incorporate an arbitrary position of the COG. Another difference in our approach is that we consider motor currents as the control signals for the system. The rationale for that is that most commercial motor controllers communicate over a relatively slow interface such as RS-232 with a significant overhead of a text-based protocol. This may not allow to run a relatively fast closed loop system with voltage being an input to the system.

The research of adaptive control in its application to robotic wheelchairs, has been fairly limited. While certain results towards parameter self-adjustment and adaptive control for wheelchairs have been reported (Katsura and Ohnishi, 2006; Tian and Xu, 2009), most of the commercially available wheelchairs still employ standard PID-controllers with fixed or manually configurable parameters.

A notable work is (De La Cruz, Bastos, and Carelli, 2011) where the COG is not required to be on the robot symmetry axis. In that work, the output of the adaptive controller is a vector of linear and angular velocities assuming an additional low-level PD velocity controller. The advantages of such approach when used on a wheelchair are somewhat questionable given that commercial wheelchairs are not typically equipped with velocity sensors, and thus do not implement velocity control. Moreover, the communication protocols for commercial wheelchair motor controllers are usually proprietary. These two factors usually require a replacement of the motor controller when any autonomy is desired. Once the controller is replaced, operating directly with motor voltages or currents seems a much more natural solution. Also, the mathematical model in (De La Cruz, Bastos, and Carelli, 2011) assumes a PD velocity controller on the low level, whereas the presented diagram of the low-level velocity controller used in the experiments clearly has features of a PI-controller. Thus the presented in (De La Cruz, Bastos, and Carelli, 2011) experimental results, while showing a satisfactory performance, do not seem to support the mathematical model presented in the paper.
In this chapter, we present the formulation and simulation details of adaptive control with online parameter estimation to a wheelchair using a dynamic model. We demonstrate mathematical and computer models of the closed-loop trajectory tracking system. Simulation results are obtained with MATLAB/Simulink® framework for modeling and visualization of a differential drive wheelchair. Simulation results for four experiments with different initial conditions and wheelchair parameters are shown and analyzed in this chapter.

2.2 Wheelchair Model

The wheelchair model presented here is based on a simplified model developed in (Johnson and Aylor, 1985). In the model we assume a flat ground, linear rolling friction model with a constant coefficient, caster wheels, air drag and frictional moments in the wheel/surface contact region are neglected, axle friction is proportional to the load applied to the wheel, motor torques are proportional to the armature current.

The model is built based on the following assumptions:

• Ground surface is flat.

• External forces and torques from caster wheels are neglected.

• Rolling friction force is constant (an increase of the rolling friction force on low speeds is neglected).

• Rolling resistance is proportional to load applied to the wheel.

• Axle friction is proportional to load applied to the wheel.

• Frictional moments which are present in the wheel/surface contact region are neglected.

• Air drag is neglected.

• Direct current motor torques are approximated as a linear function of the armature current.
• When voltage control is assumed instead of current control, the standard linear direct current motor model is employed.

2.2.1 Mathematical Model

With the above assumptions the wheelchair model can be written as:

\[ \frac{M_F R}{2} \left( \left( 1 + \frac{x}{r} \right) \dot{\omega}_1 + \left( 1 - \frac{x}{r} \right) \dot{\omega}_2 \right) = F_1 + F_2 - F_{r1} - F_{r2} \]  

(2.1)

\[ \frac{R}{2F} \left( I_0 + M_F \left( \bar{x}^2 + \bar{x}r + \bar{y}^2 \right) \right) \dot{\omega}_1 + \frac{R}{2r} \left( -I_0 + M_F \left( -\bar{x}^2 + \bar{x}r - \bar{y}^2 \right) \right) \dot{\omega}_2 = r \left( F_1 - F_{r1} - (F_2 - F_{r2}) \right) \]  

(2.2)

where \( F_1 \) and \( F_2 \) are wheelchair driving forces corresponding to right and left wheels, respectively (Figure 2-2), \( F_{r1} \) and \( F_{r2} \) are resistive forces corresponding to right and left wheels and generated by wheel rolling resistance and axle friction, \( \omega_1 \) and \( \omega_2 \) are right and left wheel angular velocities, \( M_F \) is the sum of frame and user mass, \( I_0 \) is the moment of inertia of the frame, \( \bar{x} \) and \( \bar{y} \) are the center of mass coordinates, \( R \) is the wheel radius, \( r \) is
half of the wheelbase width.

To account further for motor dynamics we can write for $F_k, (k = 1, 2)$:

$$F_k = \frac{G}{R}(K_T i_k - J_m \dot{\omega}_k - \beta_m \omega_k) \quad (2.3)$$

where $i_k$ is $k$-th motor currents, $G$ is the gear reduction factor, $K_T$ is motor torque constant, $J_m$ is motor shaft moment of inertia, $\beta_m$ is motor shaft damping.

When considering motor voltages as system inputs, the model is usually further refined with

$$\dot{i}_k = -\frac{R_a}{L_a}i_k - \frac{K_v G}{L_a} \omega_k + \frac{1}{L_a} E_k \quad (2.4)$$

where $R_a$ is motor armature resistance, $L_a$ is motor armature inductance, $K_v$ is motor velocity constant, $E_k$ is $k$-th motor armature voltage, however in this paper we take motor currents $i_k$ as system inputs.

The resistive forces $F_{rk}, (k = 1, 2)$, can be modelled with:

$$F_{rk} = \mu N_k \text{sign}(\omega_k) \quad (2.5)$$

where $\mu$ is friction coefficient, $N_1$ and $N_2$ are weights supported by right and left wheel, respectively. Another way to model the friction which better suites simulations with a variable step (Andersson, Söderberg, and Björklund, 2007) is:

$$F_{rk} = \mu N_k \tanh \left( \frac{\omega_k}{\omega_{th}} \right) \quad (2.6)$$

where $\omega_{th}$ is a constant velocity threshold.

In both eqs. (2.5) and (2.6):

$$\mu = c + R_{ax} C_{ax} \quad (2.7)$$

where $c$ is the coefficient of wheel rolling resistance, $C_{ax}$ is the coefficient of axle friction, $R_{ax}$ is wheel axle radius.

A wheelchair being a four-wheeled vehicle is a statically indeterminate system (Takezono, Minamoto, and Tao, 1999) which means that for accurate calculations, the deformations...
tion of the wheels and the ground need to be accounted for. Here we employ a simplified approach used in (Johnson and Aylor, 1985), where same elastic properties are assumed for each wheel, as well as linear dependence of normal forces on the position of COG:

\[ N_1 = \frac{M_F g}{2} \left(1 + \frac{x}{r}\right) \left(1 + \frac{y}{L}\right) \]  
\[ (2.8) \]

\[ N_2 = \frac{M_F g}{2} \left(1 - \frac{x}{r}\right) \left(1 + \frac{y}{L}\right) \]  
\[ (2.9) \]

where \( L \) is the distance between the front and the rear wheels (see Figure 2-2), \( g \) is the gravity acceleration.

Equations (2.1) to (2.5) and (2.7) to (2.9) comprise a complete mathematical model of the system. In this model, \( i_1 \) and \( i_2 \) are system input, \( \omega_1 \) and \( \omega_2 \) can be chosen as system state variables.

### 2.2.2 Simulation

By isolating the derivatives of angular velocities \( \dot{\omega}_1 \) and \( \dot{\omega}_2 \) and switching to vector notation, the defined model can be reduced to a format more suitable for simulation purposes:

\[ \dot{\omega} = S^{-1} \left(GK_T i - G^2 b_m \omega - F_{fr} \text{sign}(\omega)\right) \]  
\[ (2.10) \]

where \( \omega = \left[\omega_1 \ \omega_2\right]^T \), \( \text{sign}(\cdot) \) function is applied element-wise, \( i = \left[i_1 \ i_2\right]^T \),

\[ S = G^2 J_m \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}\right] \left(\frac{\mu}{2r}\right)^2 \left[I_0 \left[\begin{array}{cc} 1 & -1 \\ -1 & 1 \end{array}\right]\right] \]

\[ + \quad M_F \left[\begin{array}{cc} (r + \bar{x})^2 + \bar{y}^2 & r^2 - (\bar{x} + \bar{y})^2 \\ r^2 - (\bar{x} + \bar{y})^2 & (r - \bar{x})^2 + \bar{y}^2 \end{array}\right], \]  
\[ (2.11) \]

and

\[ F_{fr} = \left[\begin{array}{cc} \mu N_1 & 0 \\ 0 & \mu N_2 \end{array}\right] \]  
\[ (2.12) \]
Representation (2.10) allows to calculate the state variable derivative based on the current state and input signals.

Simulink® has been chosen as the end-user simulation environment and Simulink 3D Animation® toolbox was employed for 3D visualization.

2.2.3 Model Verification

Fig. 2-3 shows the wheelchair model test environment. Here motor currents can be set by the user during the simulation. The 3D visualization block on the bottom right allows the user to observe wheelchair motion in “real” time. A few experiments were conducted to verify the sanity of the model, such as: setting low current to one motor and then gradually increasing it to observe static friction threshold, setting motor currents to each wheel separately to verify correct rotation direction, providing motor currents and then setting it to zero to see the inertial motion, setting equal motor currents and then moving COG to observe the expected behavior, changing total mass of the wheelchair while keeping the same motor currents, changing friction coefficients on the fly, etc. The current formulation of the problem does not require an accurate knowledge of the actual wheelchair parameters, and thus there was no need at this stage to formally verify the accuracy of the model parameters as compared to the real wheelchair.

2.3 Adaptive Motion Control with Online Parameter Estimation

2.3.1 General formulation

Let us consider a dynamic system with unknown parameters defined by:

\[ M(q) \ddot{q} + C(q, \dot{q}) \dot{q} + F \dot{q} + G(q) = \tau \]  

(2.13)

where \( q \) is a vector of generalized coordinates of the system, \( \tau \) is the vector of forces and torques applied along the generalized coordinates, \( M(q) \) is inertia matrix (symmetric
Figure 2-3: Wheelchair model test environment. Top: Simulink diagram, bottom left: slider gains controlling motor currents, bottom right: 3D visualization
and positive definite for all \( q \) in \( \mathbb{R}^n \), \( C(q, \dot{q}) \) is matrix of centripetal and Coriolis torques (matrix \( \hat{M} - 2C \) is skew-symmetric), \( F\dot{q} \) contains friction terms, \( G(q) \) is a vector of gravitational forces. If this system model is linear in a set of physical parameters, such that:

\[
M(q)\ddot{q} + C(q, \dot{q})\dot{q} + F\dot{q} + G(q) = Y(q, \dot{q}, \ddot{q})\Theta
\]  

(2.14)

where \( Y(q, \dot{q}, \ddot{q}) \) is a matrix independent of system parameters (called a dynamic regressor matrix), \( \Theta \) is a parameter vector, then a control law in the form of

\[
\tau = u = K_D\sigma + Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\hat{\Theta}
\]  

(2.15)

can be designed (notation is explained further) to guarantee the stability of trajectory tracking even when the system parameters residing in \( M(q), C(q), F, G(q) \) are slowly changing (Kelly, Davila, and Loría, 2005).

The first term \( K_D\sigma \) in (2.15) is equivalent to a PD action. Indeed, let \( \tilde{q} \) be the current error of the system:

\[
\tilde{q} = q_d - q
\]  

(2.16)

where \( q_d \) is the desired state vector, and \( q \) is the current state vector as measured by the sensors. Then we defined \( \sigma \) as a linear combination of system error and its derivative:

\[
\sigma = \dot{\tilde{q}} + \Lambda\tilde{q}
\]  

(2.17)

where \( \Lambda \) is a user-configurable positive-definite matrix, thus when multiplied by another user-defined positive-definite matrix \( K_D \), \( K_D\sigma \) in (2.15) works as a PD-regulator.

The second term \( Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\hat{\Theta} \) in (2.15) attempts to compensate for the system dynamics. Here \( q \) and \( \dot{q} \) are assumed to be measured by sensors, whereas \( \dot{q}_r \) and \( \ddot{q}_r \) are the filtered tracking error vector and its derivative defined as:

\[
\dot{q}_r = \dot{q}_d + \Lambda(q_d - q)
\]  

(2.18)

\[
\ddot{q}_r = \ddot{q}_d + \Lambda(\ddot{q}_d - \ddot{q}).
\]  

(2.19)
The dynamic regressor matrix $Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)$ in (2.15) is computed as a linear with respect to system parameters representation of the system model in which the state vector is replaced with the filtered tracking error:

$$M(q)\ddot{q}_r + C(q, \dot{q})\dot{q}_r + F\dot{q}_r + G(q) = Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\Theta$$

(2.20)

and $\hat{\Theta}$ is the current estimate of the parameter vector which is initialized with $\hat{\Theta}_0$ calculated based on the available knowledge about the system and then updated online in a parameter estimation loop:

$$\dot{\hat{\Theta}} = K_\Theta^{-1}Y^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\sigma$$

(2.21)

where $K_\Theta$ is a user-defined positive-definite symmetric matrix.

The $K_D$ and $K_\Theta$ matrices define the performance of the controller, and the parameter estimation mechanism, respectively, whereas $\Lambda$ influences both because it defines the relation between the proportional and differential component in the system error. The desired dynamic performance of the controller can be tuned offline by adjusting the values of these three matrices.

### 2.3.2 Application to the wheelchair model

There are several ways to define generalized coordinates for the wheelchair system. In this work, we chose:

$$\dot{q} = v = \begin{bmatrix} V \\ w \end{bmatrix}^T$$

(2.22)

where $V$ and $w$ (not to be confused with $\omega$) are linear and angular velocity of the wheelchair, respectively, and thus

$$q = \begin{bmatrix} s \\ \phi \end{bmatrix}^T$$

(2.23)

where $s$ is the trajectory length, $\phi$ is the wheelchair angular displacement (see Figure 2-2).

Clearly,

$$\omega = T\dot{q},$$

(2.24)
where

\[
T = \frac{1}{R} \begin{bmatrix}
1 & r \\
1 & -r
\end{bmatrix}
\]  \hspace{1cm} (2.25)

and thus after substituting (2.24) into (2.10), we will have:

\[
\frac{1}{G_K T} S T \ddot{q} + \frac{G b_m}{K_T} T \dot{q} + \frac{1}{G_K T} F_{fr} \text{sign}(T \dot{q}) = i
\]  \hspace{1cm} (2.26)

In our case, we assume that the current control loop is fast enough, to use motor currents (rather than motor voltages) as the input of the system. It shall be noted, however, that in the original formulation of the problem the vector of controlling signals \( \tau \) was aligned with the generalized coordinates \( q \). To align the controlling vector with generalized coordinates we need to apply a certain transformation:

\[
\tau = X \dot{i}
\]  \hspace{1cm} (2.27)

Based on (2.26) and (2.27), we then can write:

\[
\frac{1}{G_K T} X ST \ddot{q} + \frac{G b_m}{K_T} X T \dot{q} + \frac{1}{G_K T} X F_{fr} \text{sign}(T \dot{q}) = \tau
\]  \hspace{1cm} (2.28)

One of the ways to define \( X \) is:

\[
X = G K_T T^T
\]  \hspace{1cm} (2.29)

In our case:

\[
Y = \begin{bmatrix}
\dot{q}_{r1} & 0 & \dot{q}_{r2} & \dot{q}_{r1} & 0 & \text{sign}(\omega_1) & \text{sign}(\omega_2) & 0 & 0 \\
0 & \dot{q}_{r2} & \dot{q}_{r1} & 0 & \dot{q}_{r2} & 0 & 0 & \text{sign}(\omega_1) & -\text{sign}(\omega_2)
\end{bmatrix}
\]  \hspace{1cm} (2.30)
\[
\Theta = \begin{bmatrix}
M_F + \frac{2G^2 J_m}{R^2} \\
I_0 + M_F (\bar{x}^2 + \bar{y}^2) + \frac{2G^2 J_m r^2}{R^2} \\
M_F \bar{x} \\
2\beta_m G^3 / R^2 \\
2\beta_m G^3 r^2 / R^2 \\
\mu N_1 / R \\
\mu N_2 / R \\
r \mu N_1 / R \\
r \mu N_2 / R 
\end{bmatrix} 
\] (2.31)

### 2.4 Simulation Results

#### 2.4.1 Simulation

Fig. 2-4 shows the complete wheelchair-controller Simulink\textsuperscript{®} diagram.

The model also includes a saturation block which is a natural representation of the motor controller/motor limits of currents.

Table 2.1 lists parameter values used in the simulation.
\[
\begin{align*}
C_{ax} &\quad 0.1 & M_F &\quad 150.0\text{kg} \\
c &\quad 0.02 & R &\quad 0.1\text{m} \\
G &\quad 13 & R_a &\quad 0.2\Omega \\
g &\quad 9.81\text{m/s}^2 & R_{ax} &\quad 0.01\text{m} \\
I_0 &\quad 12.0\text{kg m}^2 & r &\quad 0.3\text{m} \\
J_m &\quad 2.25 \cdot 10^{-4}\text{kg m}^2 & \bar{x} &\quad 0.0\text{m} \\
K_T &\quad 0.1\text{N m/A} & \bar{y} &\quad 0.0\text{m} \\
K_v &\quad 0.0573\text{V s/rad} & \beta_m &\quad 3.0 \cdot 10^{-4}\text{N m s/rad} \\
L_a &\quad 1.0 \cdot 10^{-3}\text{H} & &
\end{align*}
\]

Table 2.1: Simulation parameter values

### 2.4.2 Experiment 1: Constant parameters, unsaturated current

Initial values: \( x_0 = y_0 = \phi_0 = 0, \psi_0 = 0 \). Controller parameters: \( K_D = 500I_{2x2}, \Lambda = 20I_{2x2}, K_{\Theta}^{-1} = 200I_{9x9} \). Duration: 6sec. Motor current saturation limit: 20Amp.

In this experiment, we want the wheelchair to start moving along an arc with a constant acceleration for 2 seconds and then switch to a constant speed. As it can be observed from the plots presented on Fig. 2-5, the controller performs satisfactorily: through the whole duration of the experiment, there is no noticeable divergence of the desired \( s_d(t) \) and \( \phi_d(t) \) from the actual \( s(t) \) and \( \phi(t) \), as well as of the desired trajectory from the actual trajectory. It can be noted that the maximum motor current value is reached at \( t = 0 \) and is around 19Amp (below the saturation limit). Another interesting characteristic is how parameter vector \( \Theta \) is changing in the beginning of the simulation even though in this particular experiment the initial value of the parameter vector \( \Theta_0 \) are calculated based on the exact wheelchair parameters, and thus one might be expect them not to evolve during the simulation. The explanation for this is that the adaptive control does guarantee the convergence of the actual trajectory to the desired trajectory, but it does not guarantee that the value of of the parameter vector will converge to the actual parameters.
Figure 2-5: Experiment 1 (constant parameters, unasaturated motor current) simulation results.
2.4.3 Experiment 2: Constant parameters, saturated current

The purpose of this experiment is to identify how motor current saturation affects the controller performance, thus all the parameters and initial values in this experiment are equal to those of Experiment 1, except for the motor current saturation limit which is set to 15 Amp (below the maximum motor current expected in this conditions).

As it can be seen on Fig. 2-6, even if the motor current saturation limit is exceeded by a relatively small amount ($i_{1sat}$ and $i_{2sat}$ denote motor currents after the saturation block), this results in a rather sporadic controller behavior characterized by a large trajectory tracking errors, oscillating behavior of the parameter vector and motor current. The controller does not appear to be robust to the saturation of motor currents. One way to address this issue...
is to adjust the controller structure when the currents are saturated. Another method is to use a motion planner which will apply additional constraints to generated trajectories to prevent the saturation.

### 2.4.4 Experiment 3: Parameter changed, unsaturated current

The purpose of this experiment is to show how the controller behaves when one of the wheelchair parameters changes. Initial values: \( x = y = \phi = 0, v = 0_{2x2} \). Controller parameters: \( K_D = 200I_{2x2}, \Lambda = 20I_{2x2}, K_\phi = 20I_{9x9} \) Motor current saturation limit: 30Amp. Duration: 30sec. At \( t = 15sec \) the \( \bar{x} \) value (the center of mass y-position) was changed from default 0 to \(-0.2m\). This can be interpreted as if the human subject in the wheelchair suddenly leaned towards the left side of the wheelchair.

As it can be observed on Fig.2-7, the controller is able to ensure trajectory tracking performance with an unnoticeable tracking error. Even when at \( t = 15 \), the wheelchair model changes, it does not appear to be reflected in the tracking performance, whereas the evolution of the parameter vector reveals the identified change in the system.

### 2.4.5 Experiment 4: Zero initial parameter vector estimate, unsaturated current

The purpose of this experiment is to test how robust is the controller to the unknown wheelchair parameters. All the parameters and initial values in this experiment are equal to those of Experiment 1, except for the initial value of the parameter vector estimate which is set to a 9-dimensional zero-vector.

Fig. 2-8 shows that even with a zero initial estimate of the parameter vector controller exhibit stable and robust behavior with an unnoticeable trajectory tracking error.

### 2.5 Towards real robot experiments

The promising simulation results presented in the previous section have motivated an integration of the adaptive controller into the real robot system. Certain steps were made in
Figure 2-7: Experiment 3 (parameter changed, unsaturated current) simulation results.
Figure 2-8: Experiment (zero initial parameter vector estimate, unsaturated current) simulation results.
that direction, however, no stable robot control has yet been achieved on the real robot, due to several technical challenges discussed in this section.

Figure 2-9 shows a system diagram of the motion control scheme using the developed adaptive control. The motor controller already installed on the wheelchair is Roboteq SDC2130 v1.5, firmware: v1.2 RCB100 05/25/2013, the adaptive controller was translated from MATLAB into Python for a Linux machine. Initial experiments very quickly displayed an oscillatory behavior of the real robot. An analysis of the problem, revealed the following underlying technical issues.

**Low-latency communication with the motor controller** The first set of problems was in implementing a low-latency communication with the motor controller. The controller currently installed on the system provides RS-232 and USB interfaces, the first one being recommended for deployment on real systems. Various options for low-latency communication were explored:

- **USB:** Given that the Linux computer currently used on the robot is not equipped with a serial port, USB-communication was tested first. Using a simple tool written in C++, it was observed that computer-controller turnaround (request-response) communication duration was varying significantly, even when the low_latency flag was set for the Linux serial port device, sometimes reaching values as high as 50ms which is too slow for the given object dynamics. The source of this latency is likely to be two-fold: first the usbserial linux driver seems to operate using polling with a 3ms period, second, the serial-to-USB converter within the motor controller probably has a buffer adding to the latency.

- **RS-232 to USB/PCI:** While various serial-port adapter solutions are ubiquitous on
the market, the practical experience revealed that most of them have internal buffers which result in a noticeable latency of often exceeding 10ms one-way. This was observed not only with a lower cost FTDI-based solutions, but also with more professional solutions enabling serial ports through a PCI Express Mini bus, such as StarTech EC4S952 adapter.

- **Pure RS-232:** A direct communication with a stationary desktop computer equipped with a serial port (the low_latency flag set) allowed to reliably reduce the turnaround communication time to less than 3ms. Finally a portable single-board computer Pandaboard ES equipped with a serial port was tried. After finding an appropriate Linux image (Linux linaro-developer 3.18.0-1-linaro-omap) supporting low-latency serial port communication, a turnaround communication time was achieved reliably under 3ms

**Motor current closed-loop transient time** An extensive testing and parameter tuning was conducted to reduce the motor current transient time. And while, in most cases, it was possible to keep the closed loop system transient time (loop closed through the internal PID motor current controller) under 20ms, it was observed that in the open-loop system at the maximum input motor voltage (24V), the output motor currents were rising from 0 to 20Amp within 30ms. This means that the closed-loop system transient time in worst cases cannot be faster than 30ms.

**Parameter drift** It was observed in simulation that adding Gaussian noise to the measured velocities may lead to a known problem of parameter drift (Ioannou and Sun, 2012). Parameter drift is a monotonous increase or decrease of one or several values in the vector of estimated parameters $\hat{\Theta}$. When implemented on a real robot this may lead to the saturation of control signals.
2.6 Discussion

In this chapter, an adaptive controller with online parameter estimation for a differentially driven wheelchair was proposed and implemented in simulation. The results of simulation experiments with static and changing robot parameters demonstrate satisfactory performance.

Admittedly, due to the choice of the state vector \( q = [s, \phi]^T \) in this work, even the convergence of \( q(t) \) to \( q_d(t) \), does not guarantee the convergence of the robot pose \( (x, y, \phi) \) to the desired pose. This, however, should not be a problem if the controller used within a local planner which iteratively generates new trajectories.

Since the purpose of the work was to evaluate the applicability of the adaptive controller to the wheelchair platform, the parameter values used in this simulation were just rough estimates of the real values. However, to make the integration of the controller with the real wheelchair smoother, the controller should first be tested in an extended simulation model that will more accurately reflect nuances of the real system:

- The wheelchair model parameters should be calculated using real wheelchair parameter identification.
- The controller should account for possible difference in the characteristics of the left and right motors, and the variation of each motor characteristics depending on the direction of rotation.
- The simulation model should reflect the motor current dynamics and communication delays.
- Noise should be added to the measured values
- A gradient projection algorithm (Ioannou and Sun, 2012; De La Cruz, Bastos, and Carelli, 2011) or an alternative should be implemented to address the parameter drift problem.
- Optionally, the ground slope and caster wheels can be added into the wheelchair model.
Once a satisfactory controller performance is demonstrated in this extended simulation, a transition to the real system can take place. Within the scope of this project, however, it was decided to use a standard PID-controller for the real wheelchair motion control.
Chapter 3

Shared steering control

Moving from the motion control layer to the next level in the control hierarchy (Section 1.5, Figure 3-1), in this chapter we develop a shared steering control for a robotic wheelchair.

3.1 Background

Shared steering control combines robot intelligence with human intelligence. This method in literature is associated with *semi-autonomous navigation* and *assistive navigation*.

Similar to the direct steering control, it receives directional commands from the HMI, but the actual robot velocities that are passed to the motion control level may differ. The shared control unit calculates them by taking into account additional factors. These typically are the information about obstacles (environmental knowledge) around the robot, 

Figure 3-1: In this chapter, we discuss shared steering control operating at the local planner layer
and/or, less frequently, the human behavior model. Obstacle configuration is obtained with robot sensors, such as LIDARs, depth cameras, vision, ultrasonic and infrared sensors, while the behavioral model can be learned over time or manually configured in advance. By combining this data with the HMI commands, shared control can improve steering.

Due to the decreasing cost of the sensors, increasing performance of computing machines, and the advances in autonomous navigation algorithms, obstacle sensing and avoidance has become practical and affordable. There are two main approaches to using the obstacle information: non-probabilistic and probabilistic.

 Non-probabilistic approach seems to be more common, though less intelligent, because it does not utilize probabilistic information from the HMI, thus wasting some of the sparse information from the human. One of the simplest shared steering controls is implemented in (Achic et al., 2016) where obstacles are detected only in front and behind the wheelchair with 6 ultrasonic sensors. If obstacles are detected, in the direction of the human command it is not accepted.

 In (Carlson and J. d. R. Millan, 2013), the wheelchair normally moved forward with constant velocity while avoiding obstacles. They used an interesting obstacle avoidance mechanism. Ten zones were defined around the robot. Whenever obstacles were present in a given zone, corrective velocities would be calculated by modeling repulsive forces. Parameters for the calculation of corrective velocities for each zone were chosen empirically according to the wheelchair dynamics and sensor reliability. The operator could send turn left/right commands with a MI-based BCI. When such a command received, an attractor is placed in front of the wheelchair at an angle of 45° that remains there until robot rotate 45° or a new BCI command is received. Ten close-range sonars and two standard off-the-shelf are used to detect obstacles.

 In (Lin and Kuo, 2016), wheelchair steering control was implemented using standard four directional commands (“forward”, “turn left”, “turn right”, “stop”) that were were selected by the operator using SSVEP-based BCI, but obstacle artificial potential field has been added to repel the wheelchair from obstacles. The obstacles are detected with a laser-range scanner (LIDAR). This enabled safe collision-free navigation.

 Environment knowledge obtained by robot sensors enables another steering model,
namely opening selection which suits well Yes/No type of HMIs. For example, in (Shi, H. Wang, and C. Zhang, 2015) virtual and real UAVs are navigated in unknown 2D space (constant altitude) by 6 subjects using a Yes/No interfaces (mapped from right and left hand motor imagery) with the goal of finding a target. The UAV stops are every crossroad (intersection) and using range sensor determines feasible directions (openings). A scanning-like interface iterates through all openings by asking the operator if the given opening is the desired direction to go. The operator can answer with Yes or No. This system allows for manual control as well. When the operator discards all possible directions (openings), left and right hand MI commands are interpreted as turn left and turn right, while resting implies going forward. This model of selecting openings was initially proposed in (Perrin, Chavarriaga, et al., 2010).

When it comes to probabilistic reasoning in shared steering control, one of the first works on the topic is (Vanacker et al., 2007). They mapped three EEG-based BCI mental tasks (left hand motor imagery, relaxation, word task) to steering actions “forward”, “left” and “right”, but instead of simply taking the command with the highest probability, the whole vector $P_{EEG}$ of the three probabilities is kept. Then, another vector, $P_{env}$ is constructed based on the obstacle configuration around the robot, this vector again assigns a probability to each of the three actions. The vectors are then fused using a simple product operation, and the action with the highest probability was chosen. By using the velocity increment steering model they achieved a smooth context-aware wheelchair motion in a virtual environment. In (Galán et al., 2008) they report real wheelchair navigation using this shared steering control method.

In this chapter, a universal navigation framework incorporating voice control (CMU Sphinx and Google Glass) and facial expression control (Emotiv EPOC) with assistive navigation (obstacle avoidance) to enable shared steering control is presented. A set of use/test cases designed to ensure safe and reliable navigation is also discussed.
3.2 Robot system overview

The semi-autonomous robotic wheelchair was built on top of a commercially available electric wheelchair CTM HS-2800. It was equipped with the following sensor suite (Figure 3-2):

- 2D LIDAR module which is used for mapping, localization and obstacle avoidance
- Wheel-on-wheel encoder modules which provide odometry information and serve as feedback for the motor controller
- A set of infrared and ultrasonic range sensors mounted on and along the perimeter of the wheelchair to improve obstacle awareness
- Infrared cliff sensors mounted on the bottom of the wheelchair’s footplate.

On the software side, given that various types of user interfaces can be connected to the system (such as voice control, facial expression control, and others), we split the overall software architecture into two main blocks (subsystems): navigation subsystem, and user interfaces. Next two sections of the chapter address the two blocks in detail. Robot Operation System (ROS) Quigley et al., 2009 is used as the main software framework for both subsystems.
3.3 Navigation subsystem

The overall navigation subsystem architecture is shown on Figure 3-3. Following the steering control paradigm, this system accepts velocity commands from a user interface and ultimately outputs desired velocities for the low-level motor controller. In between, the Assistive Drive block “modifies” the given velocity commands to enable obstacle avoidance. The obstacle avoidance is made possible by the costmap which contains the latest information about all the detected obstacles and is built based on the range sensor measurements (LIDAR, infrared and ultrasonic sensors).

In addition, Cliff detector constantly monitors cliff sensor data to detect a possible increase in the detected range which would mean a cliff or a staircase being right under the robot footplate. If this occurs, it sends an emergency stop command to the motor controller, thus preventing the wheelchair from a tipping over.
Simultaneous localization and mapping is done by the standard ROS GMapping package\cite{Grisetti2007} based on odometry and LIDAR measurements, and localization on an existing map is realized with standard ROS amcl package.

The *Autonomous Navigation* block is shown “faded” on the diagram, because it was not used for steering control.

### 3.3.1 Shared steering control with assistive navigation

Assistive navigation is the core of the the *shared steering control*. It is designed to augment the user driving experience with obstacle avoidance. This functionality is especially important when dealing with low throughput input interfaces, such as brain-computer interfaces and voice control. With these interfaces, tasks such as passing through doorways or driving in narrow hallways is often challenging. Assistive navigation allows for safe and easy execution of these tasks without hitting walls and other objects.

A tentacle-based approach was utilized to provide the wheelchair with obstacle avoidance\cite{Hundelshausen2008}. The general paradigm of tentacle-based navigation can be summarized as follows. A number of possible trajectories are generated at every iteration of the control loop. A cost is assigned to every trajectory, then the trajectory with the lowest cost is chosen, and respective commands are given to the lower-level motion control system. Navigation methods based on tentacle navigation differ in how trajectories are generated and how costs are assigned to them.

In this work, trajectories are generated based on the following algorithm. First, a set of possible seeding velocities is identified based on the current velocity, acceleration and velocity limits. At this stage kinematic constrains of the nonholonomic differentially driven robot are also taken into account. Then for every seeding velocity, a trajectory is pre-calculated assuming that the velocity doesn’t change over the configurable simulation time period.

In this work, we chose to calculate the cost of $k$-th trajectory as follows:

\[
C[k] = k_{obst}C_{obst}[k] + C_{vel}[k]
\]
where \( C_{\text{obst}}[k] \) designates how close \( k \)-th trajectory is to detected obstacles, \( C_{\text{vel}}[k] \) shows how far the generated trajectory is from a user-desired trajectory, and \( k_{\text{obst}} \) allows to configure the algorithm giving more weight to obstacle cost or velocity cost. Again there is some freedom in how to calculate \( C_{\text{obst}} \) and \( C_{\text{vel}} \).

Currently, \( C_{\text{obst}}[k] \) is calculated with an implementation of costmap_2d and obstacle_cost_function of ROS-navigation stack. In this stack, a dynamic costmap is calculated as a grid in which each cell \((x, y)\) has a cost \( C(x, y) \) assigned based on how far it is from detected occupied cells (obstacles). This process is called “obstacle inflation” and is described in details in costmap_2d - ROS Wiki 2014. In order to calculate the cost of a particular pose using an up-to-date costmap, a user-defined two dimensional projection of the robot must be provided. This projection, known as a footprint, is then discretized and overlaid on the costmap in the position and orientation of the pose. Then, among all costmap cells that lay on the footprint boundary, the one with the highest cost is identified. The cost of this cell is then used as the pose cost. The obstacle cost \( C_{\text{obst}}[k] \) of each simulated trajectory is then calculated as a sum of costs of each pose on that trajectory. Mathematically:

\[
C_{\text{obst}}[k] = \sum_{i=1}^{n_k} C_{\text{pose}[i][k]} = \sum_{i=1}^{n_k} \max \{ C(x, y) : (x, y) \in F^k_i \} \quad (3.2)
\]

where \( n_k \) is the number of steps on the \( k \)-th trajectory, \( C_{\text{pose}[i][k]} \) is the cost of \( i \)-th pose on the \( k \)-th trajectory, \( F^k_i \) is the set of cells which belong to robot footprint is when it’s moved to \( i \)-th pose on the \( k \)-th trajectory. At this stage invalid trajectories (those which overlap with obstacles) are discarded.

For the current implementation, we proposed to calculate the velocity cost \( C_{\text{vel}} \) as:

\[
C_{\text{vel}}[k] = (V_d - V_{\text{seed}}[k])^2 + (\omega_d - \omega_{\text{seed}}[k])^2 \quad (3.3)
\]

where \( V_d \) and \( \omega_d \) are user-desired linear and angular velocities, respectively, \( V_{\text{seed}}[k] \) and \( \omega_{\text{seed}}[k] \) are seeding velocities of \( k \)-th trajectory, and \( k_{\text{ang}} \) is a configuration coefficient which allows to give more or less weight to angular velocity discrepancy as compared to linear velocity discrepancy. In other words, (3.3) defines velocity cost \( C_{\text{vel}} \) as a square of
the distance between the desired velocity vector and and $k$-th seeding velocity vector in $(V, k_{ang}\omega)$-space.

### 3.3.2 Tuning assistive navigation

One of the critical aspects of designing a shared steering control is ensuring its reliability. This aspect is often overlooked in research papers where experimental results are demonstrated only in several controlled environment configurations. The reality, however, reliable navigation with automatic obstacle avoidance is still challenging, especially for noncircular nonholonomic robots. The problems arise from range sensor blind spots, sensor noise, discrete representation of environment (costmap) and a high number of configurable parameters (see a complete list below). While it is not hard to tune the parameters for several specific navigation paths, achieving satisfactory performance in all possible use cases is not a trivial task. Often adjusting a parameter to improve the performance in one use case can lead to an inability of the robot to execute another use case.

The complete list of assistive navigation parameters for a nonholonomic robot consists of several groups (the value in parentheses is the final parameter value identified as a result of the experiments):

- controller frequency (5.0 Hz);
- costmap parameters: update frequency (6.0 Hz), publication frequency (4.0 Hz), width (4.0 m), height (5.0 m), cell size (resolution) (0.03 m), inflation radius (0.4 m), robot footprint polygon definition (in m: \[[0.47, 0.27], [0.4, 0.34], [0.0, 0.34], [-0.65, 0.30], [-0.65, -0.3], [0.0, -0.34], [0.4, -0.34], [0.47, -0.27], [0.6, 0.0]]);
- generator parameters: simulation time (2.5 s), simulation linear granularity (0.05 m), simulation angular granularity (0.05 rad), maximum (0.6 m/s) and minimum (0.1 m/s) translational velocities, maximum (0.5 m/s) and minimum (−0.25 m/s) linear velocities, maximum (0.6 rad/s) and minimum (0.15 rad/s) rotational velocities, maximum (1.5 m/s²) and angular (4.5 rad/s²) absolute accelerations, number of seeding velocity samples for linear (10) and angular (12) velocities;
It is intuitively clear that ultimately the quality of the assistive navigation should be estimated by the level of the wheelchair user satisfaction, and is, thus, highly subjective. From this perspective, the problem of tuning parameters may seem to lie in the human-robot interaction (HRI) domain. In reality, however, finding a set of parameters which would guarantee even minimal navigation requirements (such as not hitting obstacles in various environments, passing through doorways of certain width, etc.) is not a trivial task.

A set of use/test cases has been developed to address this issue. The choice of tests was a result of a trade-off between the criticality of a scenario for an indoor environment and an keeping the testing time within reasonable limits. All tests were run multiple times iteratively allowing to find problems with the configuration, properly address them to ensure that all tests are passed. The designed test suite is based on the following use cases.

**Free rotational motion** The purpose of this use case is to ensure that when obstacles are far enough from the robot, the assistive control does not introduce a significant disturbance into the robot rotational motion. The wheelchair is placed at a distance $d$ from a wall with empty space on all other sides (distances to obstacles on all other directions are $> d$). The criterion for passing tests of this use case is a relatively small difference between the input and output velocities of the assistive navigation through the whole duration $T$ of the test. The table 3.1 contains values for test cases executed for this use case ($\omega$ is the angular velocity passed to the assistive control as the desired input).

**Free linear motion** This use case is analogous to “Free linear motion”, but here the robot is placed at a distance $d$ from a wall, and given linear velocity commands ($V$) parallel to

<table>
<thead>
<tr>
<th>#</th>
<th>$d$, m</th>
<th>$\omega$, rad/s</th>
<th>$T$, s</th>
<th>#</th>
<th>$d$, m</th>
<th>$\omega$, rad/s</th>
<th>$T$, s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0</td>
<td>0.2</td>
<td>32.0</td>
<td>7</td>
<td>1.5</td>
<td>0.2</td>
<td>32.0</td>
</tr>
<tr>
<td>2</td>
<td>2.0</td>
<td>0.4</td>
<td>20.0</td>
<td>8</td>
<td>1.5</td>
<td>0.4</td>
<td>20.0</td>
</tr>
<tr>
<td>3</td>
<td>2.0</td>
<td>0.6</td>
<td>15.0</td>
<td>9</td>
<td>1.5</td>
<td>0.6</td>
<td>15.0</td>
</tr>
<tr>
<td>4</td>
<td>2.0</td>
<td>-0.2</td>
<td>32.0</td>
<td>10</td>
<td>1.5</td>
<td>-0.2</td>
<td>32.0</td>
</tr>
<tr>
<td>5</td>
<td>2.0</td>
<td>-0.4</td>
<td>20.0</td>
<td>11</td>
<td>1.5</td>
<td>-0.4</td>
<td>20.0</td>
</tr>
<tr>
<td>6</td>
<td>2.0</td>
<td>-0.6</td>
<td>15.0</td>
<td>12</td>
<td>1.5</td>
<td>-0.6</td>
<td>15.0</td>
</tr>
</tbody>
</table>

Table 3.1: Test values for “Free rotational motion” test

- scorer parameters: obstacle scale $k_{obst} (1.5 \times 10^{-4})$, angular scale $k_{ang} (0.7)$. 
the wall. Again, the criterion for passing tests of this use case is a small difference between the input and output velocities of the assistive navigation through the whole duration of the test. Test values for this use case are presented in Table 3.2.

**Heading towards a wall**  In this use case the robot is given a forward velocity command $V$ to move towards a wall at a particular angle as shown on Figure 3-4a. Test cases are considered to be passed if the robot does not hit the wall, and, at the end of the test, starts moving away from it. Test values for this use case can be found in Table 3.3.

This use case revealed a problem with using the “forward” command when the angle $\alpha$
between the robot and a wall was close to $90^\circ$. In that such, due to the kinematic constraints and the initial shape of the footprint (see Figure 3-5a), any rotation to move away from the wall would cause the trailing edge of the footprint to approach the wall. This resulted in a higher pose cost than simply driving parallel to the wall. Consequently, the robot was unable to turn away from the obstacle.

To address this issue, the footprint of the robot was modified as shown on Figure 3-5 (update I). With this footprint, rotation off the wall are assigned lower costs which allows the wheelchair to slightly digress from it.

**Rotating at a wall** The purpose of the use case is to ensure that the robot, when placed close and parallel to wall (see Figure 3-4b) can follow a safe trajectory without hitting the wall even if given only a rotational command. A test case is considered to be passed if the robot did not hit the wall. Test values for this use case can be found in Table 3.3.

This use case revealed that it’s necessary to have another range sensor in addition to the LIDAR shown on Figure 3-2 which has a blind area in the left back corner of the robot. When a wall was on the left and the robot was given a “turn right” command, being unable to see the wall behind itself, the robot would turn on the spot and thus hit the wall. Two
smaller LIDARs were installed under the seat to ensure full visibility of obstacles.

**Heading towards an obstacle**  The purpose of this use case was to ensure that the robot could safely avoid an obstacles located somewhere in front of the robot (see Figure 3-4b). Test cases are considered to be passed if the robot did not hit the obstacle. Test values for this use case can be found in Table 3.3.

No collisions occurred in any of the tests for this use case.

**Heading towards a doorway**  This is one of the most important use cases. It’s designed to ensure a collision-free doorway passage given various possible robot orientations and velocities (see Figure 3-4d). The criterion of successful execution of the tests in this use case is to pass through the doorway without hitting the walls. Test values for this use case can be found in Table 3.3.

This use case helped to make another improvement to the robot footprint (see update II on Figure 3-5). It appeared that when the robot faced a flat obstacle with one of the front facets of the former footprint it could start to behave indecisively and make sudden moves.

<table>
<thead>
<tr>
<th>#</th>
<th>Side the wall is on</th>
<th>d, m</th>
<th>ω, rad/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>right</td>
<td>0.75</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>right</td>
<td>0.75</td>
<td>0.6</td>
</tr>
<tr>
<td>3</td>
<td>right</td>
<td>0.75</td>
<td>-0.3</td>
</tr>
<tr>
<td>4</td>
<td>right</td>
<td>0.75</td>
<td>-0.6</td>
</tr>
<tr>
<td>5</td>
<td>left</td>
<td>0.75</td>
<td>0.3</td>
</tr>
<tr>
<td>6</td>
<td>left</td>
<td>0.75</td>
<td>0.6</td>
</tr>
<tr>
<td>7</td>
<td>left</td>
<td>0.75</td>
<td>-0.3</td>
</tr>
<tr>
<td>8</td>
<td>left</td>
<td>0.75</td>
<td>-0.6</td>
</tr>
</tbody>
</table>

Table 3.4: Test values for “Rotating at a wall” test

<table>
<thead>
<tr>
<th>#</th>
<th>d, m</th>
<th>h, m</th>
<th>V, m/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0</td>
<td>0.0</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>2.0</td>
<td>0.0</td>
<td>0.6</td>
</tr>
<tr>
<td>3</td>
<td>2.0</td>
<td>0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>4</td>
<td>2.0</td>
<td>0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>5</td>
<td>2.0</td>
<td>-0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>6</td>
<td>2.0</td>
<td>-0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>7</td>
<td>1.0</td>
<td>0.0</td>
<td>0.3</td>
</tr>
<tr>
<td>8</td>
<td>1.0</td>
<td>0.0</td>
<td>0.6</td>
</tr>
<tr>
<td>9</td>
<td>1.0</td>
<td>0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>10</td>
<td>1.0</td>
<td>0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>11</td>
<td>1.0</td>
<td>-0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>12</td>
<td>1.0</td>
<td>-0.5</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Table 3.5: Test values for “Heading towards an obstacle” test
<table>
<thead>
<tr>
<th>#</th>
<th>$d$, m</th>
<th>$h$, m</th>
<th>$\alpha$, °</th>
<th>$V$, m/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>1.0</td>
<td>45.0</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>1.0</td>
<td>45.0</td>
<td>0.6</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>0.5</td>
<td>14.0</td>
<td>0.3</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
<td>0.5</td>
<td>14.0</td>
<td>0.6</td>
</tr>
<tr>
<td>5</td>
<td>1.0</td>
<td>0.0</td>
<td>26.6</td>
<td>0.3</td>
</tr>
<tr>
<td>6</td>
<td>1.0</td>
<td>0.0</td>
<td>14.0</td>
<td>0.3</td>
</tr>
<tr>
<td>7</td>
<td>1.0</td>
<td>0.0</td>
<td>14.0</td>
<td>0.6</td>
</tr>
<tr>
<td>8</td>
<td>1.0</td>
<td>0.0</td>
<td>-14.0</td>
<td>0.3</td>
</tr>
<tr>
<td>9</td>
<td>1.0</td>
<td>0.0</td>
<td>-14.0</td>
<td>0.6</td>
</tr>
<tr>
<td>10</td>
<td>1.0</td>
<td>0.0</td>
<td>-26.6</td>
<td>0.3</td>
</tr>
<tr>
<td>11</td>
<td>1.0</td>
<td>-0.5</td>
<td>-14.0</td>
<td>0.3</td>
</tr>
<tr>
<td>12</td>
<td>1.0</td>
<td>-0.5</td>
<td>-14.0</td>
<td>0.6</td>
</tr>
<tr>
<td>13</td>
<td>1.0</td>
<td>-1.0</td>
<td>-45.0</td>
<td>0.3</td>
</tr>
<tr>
<td>14</td>
<td>1.0</td>
<td>-1.0</td>
<td>-45.0</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Table 3.6: Test values for “Heading towards a doorway” test

Rounding the two front facets helped to resolve the problem and made robot motion smooth in this scenario.

### 3.4 Human-machine interface integration

Since the primary focus group of the project are people with limited mobility, incorporating HMIs which do not require limb motion is necessary. Voice control and facial expression control HMIs have been integrated with the robot navigation system. We also provide a conventional joystick interface whose implementation is rather trivial and is not discussed in this chapter. As previously noted, for the purpose of modularity all HMI units have the same interface with the navigation subsystem (see Figure 3-6).

Currently the system is not designed to be controlled *simultaneously* through several interfaces, thus normally only one interface is enabled at a particular moment. When several interfaces are enabled, the fusion of commands is based on the first-come, first-served principle, thus if sent simultaneously, commands will compete with each other on the lower level of velocity control causing indecisive robot behaviour.
Figure 3-6: Wheelchair user interface architecture diagram
### 3.4.1 Voice control with CMU Sphinx

Speech recognition technologies have come a long way in the past few years. There are a number of open source speech recognition systems, such as CMU Sphinx, Julius, Kaldi, iATROS, and others. In this project, we have utilized CMU Sphinx. CMU Sphinx is a large vocabulary continuous speech recognizer developed by Carnegie Mellon University and released under BSD-license. It recognizes speech by taking the waveform, splitting it to utterance and recognizing each of them. The recognition takes place by matching all the possible combinations with the audio Lamere et al., 2003.

PocketSphinx is a Cross-platform version of Sphinx that works on: Linux, Windows, Mac OS X, iOS. It was chosen for our system which is running on Ubuntu and ROS.

To report certain status information back to the user, Festival System, a speech synthesis framework developed by Carnegie Mellon UniversityBlack et al., 2001 is employed. It comes with full text to speech functionalities, easy to use APIs, and a variety of built-in voices to choose from.

The normal workflow is as follows (Figure 3-6). Once a sequence of words is recognized by the CMU Sphinx, it is sent as a String Command to String Command Interpreter. If the sequence is known to the system, the interpreter executes a relevant action and reports to the Speech Synthesizer, to confirm the action to the user, otherwise the Speech Synthesizer will ask the user to repeat his/her command. This design allows to ensure that commands are interpreted correctly and the user knows what the robot is doing.

The set of commands currently accepted by the String Command Interpreter (Figure 3-6) can be divided into several categories (Table 3.7).

All Velocity commands ultimately result in a desired velocity vector being sent to the navigation subsystem. Rotational commands are actuated by setting the angular component of the desired velocity vector without changing the linear component. This allows for smoother turns if the robot is already moving. By default the robot will be executing the previous velocity command, until a new command is received, however this is configurable allowing for a timeout to be set up. Once the timeout expires, a zero desired velocity is sent to the navigation subsystem, ensuring a safer operation.
Table 3.7: Commands accepted by Command Interpreter

<table>
<thead>
<tr>
<th>Category</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity</td>
<td>Linear: move=go=come forward=straight/ backward=back</td>
</tr>
<tr>
<td></td>
<td>Rotational: move=go=come=turn=rotate left/right</td>
</tr>
<tr>
<td></td>
<td>Stop: stop=stop now=abort=freeze=shut=halt=panic</td>
</tr>
<tr>
<td>Velocity magnitude</td>
<td>Absolute: quarter/half/full speed</td>
</tr>
<tr>
<td></td>
<td>Relative: faster=speed up/slower=slow down</td>
</tr>
<tr>
<td>Destination</td>
<td>go to the kitchen/the lab</td>
</tr>
<tr>
<td>Misc.</td>
<td>pause/continue speech: turns off/on speech synthesizer</td>
</tr>
<tr>
<td></td>
<td>whats the time: reports the current date/time</td>
</tr>
<tr>
<td></td>
<td>whats your name: says “Anna”</td>
</tr>
<tr>
<td></td>
<td>hi/hello: replies “Hi, how are you?”</td>
</tr>
</tbody>
</table>

Velocity magnitude commands adjust desired velocity magnitude, either setting it to a percentage of the maximum velocity (absolute), or increasing/decreasing it by a configurable increment (relative).

3.4.2 Voice control with Google Glass

Google Glass is a wearable computer with an optical head-mounted display (OHMD). Developed by Google, the glass provides a number of ways for the user to interact with the device and the world: a touchpad, a camera, a partially transparent display, and a microphone.

With the focus on patients with limited mobility, our first goal was to implement voice control through Google Glass. We utilized Google Glass Development Kit Sneak Peek to acquire the output of the speech recognition engine. Through Bluetooth, a list of acceptable commands is sent to Glass. Glass then uses a nearest match algorithm to recognize the commands. A command recognized from the vocabulary is then passed over bluetooth to the main computer where a ROS-package serving as a Google Glass server converts it into a String command. This String command is passed to the String Command Interpreter (Figure 3-6), preserving all the functionality which was available through CMU Sphinx voice control.

From a few not very formal experiments, we observed that Google Glass speech recognition generally works better, than CMU Sphinx engine. Whether the subjectively observed
performance difference is due to hardware differences between the two systems, or it is a result of a low-level sound processing (such as noise cancelling), or it is explained by a more effective algorithm used in Google Glass is not yet clear.

In addition to a better speech recognition performance, Glass’s lightweight semi-transparent display can be used to visualize various robot data which the user may be interested in. Such data could include a map, obstacles around the robot, and status messages.

3.4.3 Facial expression control with Emotiv EPOC

Electromyography (EMG) is a technique widely used in biomedical research. The principle of its operation is based on measuring electrical potential between two electrodes placed on the skin. The signal is measured for activation level and interpreted as certain behavior patterns of the muscles, such as contraction state or relaxation state. The signal is usually at millivolt level, thus differential amplifiers are used to make the voltage differences more observable for analysis.

There are multiple products on the market providing EMG-based brain-computer interfaces, such as Mindwave mobile, Ibrain, and EPOC Emotiv. In our project, we utilized Emotiv EPOC. Its headset is equipped with 14 signal electrodes and is supplied with documentation and Linux API.

For the steering control, we utilize facial expressions (Expressiv suite) to convey user intent to the computer. Unlike in Cognitiv and Affectiv suite modes, in Expressiv suite, the device does not attempt to read mental activity, but picks up signals of facial muscles. This allows for relatively faster (∼10 ms) and more accurate detection. Expressiv suite allows for detection of individual eyelid and eyebrow positions, eye position in the horizontal plane, smiling, laughing, clenching, and smirking.

As depicted on Figure 3-6, the Emotiv control is implemented in two stages: Emotiv ROS Driver outputs a raw state vector containing the most recent state of each expression detected. This vector is then analyzed by Emotiv Interpreter which sets a desired velocity vector for the navigation subsystem and reports the status to Speech Synthesizer. As with voice control, Speech Synthesizer helps the user to understand what the robot is
Table 3.8: Expressiv suite accuracy experiment results on 4 subjects

<table>
<thead>
<tr>
<th>Expression</th>
<th>Subj.1</th>
<th>Subj.2</th>
<th>Subj.3</th>
<th>Subj.4</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>laugh</td>
<td>0.0%</td>
<td>20.0%</td>
<td>0.0%</td>
<td>60.0%</td>
<td>20.0%</td>
</tr>
<tr>
<td>left wink</td>
<td>70.0%</td>
<td>0.0%</td>
<td>50.0%</td>
<td>30.0%</td>
<td>37.5%</td>
</tr>
<tr>
<td>smile</td>
<td>70.0%</td>
<td>10.0%</td>
<td>40.0%</td>
<td>40.0%</td>
<td>40.0%</td>
</tr>
<tr>
<td>look right</td>
<td>100.0%</td>
<td>0.0%</td>
<td>70.0%</td>
<td>40.0%</td>
<td>52.5%</td>
</tr>
<tr>
<td>blink</td>
<td>0.0%</td>
<td>50.0%</td>
<td>100.0%</td>
<td>80.0%</td>
<td>57.5%</td>
</tr>
<tr>
<td>look left</td>
<td>100.0%</td>
<td>0.0%</td>
<td>100.0%</td>
<td>30.0%</td>
<td>57.5%</td>
</tr>
<tr>
<td>right wink</td>
<td>100.0%</td>
<td>0.0%</td>
<td>100.0%</td>
<td>40.0%</td>
<td>60.0%</td>
</tr>
<tr>
<td>clench</td>
<td>100.0%</td>
<td>100.0%</td>
<td>30.0%</td>
<td>40.0%</td>
<td>67.5%</td>
</tr>
<tr>
<td>furrow brow</td>
<td>90.0%</td>
<td>70.0%</td>
<td>40.0%</td>
<td>80.0%</td>
<td>70.0%</td>
</tr>
<tr>
<td>raise brow</td>
<td>100.0%</td>
<td>40.0%</td>
<td>100.0%</td>
<td>50.0%</td>
<td>72.5%</td>
</tr>
<tr>
<td>right smirk</td>
<td>100.0%</td>
<td>40.0%</td>
<td>70.0%</td>
<td>90.0%</td>
<td>75.0%</td>
</tr>
<tr>
<td>left smirk</td>
<td>100.0%</td>
<td>80.0%</td>
<td>40.0%</td>
<td>100.0%</td>
<td>80.0%</td>
</tr>
</tbody>
</table>

To properly design an interpreter, it was necessary to measure the detection accuracy for each expression. For this purpose, we have developed a simple tool which runs 10 tests on every expression. The experiment has been conducted on four subjects and the results are present in Table 3.8:

From Table 3.8, it can be observed that there are 9 expressions whose accuracy is more than 50%: look right, blink, look left, right wink, clench, furrow brow, raise brow, right smirk, left smirk. Right smirk, left smirk and blink, however, had a high false detection rate, thus only four expressions were found reliable for the first implementation: raise brow, look left, look right, clench.

The initial version of the Emotiv Interpreter (Figure 3-6) allowed to send the following velocity commands (similar to the voice control velocity commands): clench (go forward), raise brow (stop), look left (turn left), look right (turn right).

When it was realized that “go backwards” command is necessary for navigation of a noncircular differentially driven robot, the prolonged clench command (longer than 2 seconds) was assigned the semantics of the “go backwards” command. This command takes longer to issue, but since the reverse motion is required only in rare cases of being stuck in a nook, it’s generally acceptable.

After several experiments, it was found that on arriving to a desired destination, the
user may want to turn off the control interface to prevent random commands being received and interpreted by the system. For this need, a passive mode was added in which only one command is picked up by the system: switch back to active mode, and all others are ignored. For switching between the modes the user has to send stop command (raise eyebrow) three times in a row within 5 sec.

3.5 Discussion

The unified HMI framework developed in this chapter along with the obstacle avoidance mechanism implemented and tested on a robotic wheelchair provide a possibility for people with limited lower- and upper-body mobility to achieve independent locomotion in an indoor environment. The further progress towards a real world solution for steering control with low throughput HMIs is also dependent on defining formal use and test cases for such systems. In addition to the cases proposed in this chapter, more need to be added to cover backwards motion, dynamic obstacles, entering doors with non-zero initial linear and angular velocities. At the same time, based on the observations made in this work, the number of tests for each case can probably be reduced. The trajectory generator can be improved to account for additional constraints, such as limitations on the speed of acceleration change (an abrupt change of acceleration, even of low magnitude, is perceived by the human in the wheelchair as sudden push).

Admittedly, the implemented approach does not use probabilistic reasoning as opposed to (Vanacker et al., 2007), where the context (obstacle) information and BCI input commands are fused probabilistically. On the other hand, the probabilistic fusion for steering navigation may not always behave as expected. As reported, (Vanacker et al., 2007), it may cause timing problems that may result in the robot hitting obstacles. From this perspective, our method that was tested in multiple environments to cover multiple navigation use cases is more practical.

Another limitation of the developed shared steering control is that it does not model user habits (popular locations). Such information can be helpful for complex maneuvers as reported in (Demeester et al., 2007). The integration of user predicted behaviour (based
on regularly visited location) is addressed in Chapter 5 where a shared position control is developed.
Chapter 4

CWave: high-performance single-source any-angle path planning on a grid

As it was discussed in Section 1.5, for activities of daily living (ADL), it is often more important to be able to reach a desired goal pose, rather then control a wheelchair by steering. Moreover, steering often makes it challenging for the operator to reach arbitrary destinations with good accuracy. To enable effective shared position control, we need to address another challenge associated with low throughput human-machine interfaces. LTIs are usually probabilistic. In terms of position control that implies that the user-intended destination may not be immediately and deterministically known, but rather it is gradually and probabilistically inferred. This is discussed in detail in Chapter 5 where a novel shared position control is developed. One way to work effectively with partial probabilistic knowledge about the destination is to pre-compute possible navigation scenarios (trajectories) and then, based on the probabilistic knowledge of the user intent, execute an optimal movement. Path planning problems that ask for distances from a given source point to all other points are known as single-source planning problems. To make real time decisions such algorithm should be fast. The development of such algorithm for a high-performance single-source any-angle path planning on a 2D grid that was named CWave is the focus of this chapter. In terms of the control hierarchy introduced in Section 1.5, this chapter is one level up, at the global planning layer (Figure 4-1).
Figure 4-1: In this chapter, we discuss CWave, a high-performance path planning algorithm that was developed to enable shared position control method discussed in the next chapter

4.1 Background

2D grid is a popular, but not the only method to represent a 2D environment. Other representations include Navigation Meshes, Circle-Based Waypoint Graphs (Nash, 2012), Probabilistic Quadtrees (Kraetzschmar et al., 2004), but these require an additional preprocessing of the map. A 2D grid is simple, easy to use, and is often a direct output of Simultaneous Localization and Mapping (SLAM) algorithms (Grisetti, Stachniss, and Burgard, 2007). A grid representation of the environment is also internally used in many autonomous and semi-autonomous robot navigation systems (Marder-Eppstein et al., 2010), (Sinyukov et al., 2014b) and video games (Rabin, 2000).

The common approach to finding the shortest path on a grid is to represent the grid as a 4- or 8-connected graph. Various graph search algorithms can then be employed to find the shortest paths. The problem, however, is that these paths will be suboptimal, because the graph representation of the grid restricts path segment angles to $90^\circ$ (4-connected graph) or $45^\circ$ (8-connected graph). Increasing the connectivity of the graph by including 2nd order neighbors, can find paths closer to the optimal, but it cannot solve the fundamental problem arising from the graph representation of the grid. In addition to that, since only $45^\circ$ turns are allowed, these suboptimal paths are usually unnatural. Path planning algorithms which do not restrict path segment angles are called any-angle path planning algorithms. They are used on various mobile robots including such advanced systems as the Mars rovers Spirit, Opportunity and Curiosity (Nash and Koenig, 2013), (Carsten et al., 2009).
Most of the graph search algorithms designed for point-to-point path planning can be modified to solve single-source path planning problems. Indeed, if the search is not stopped when the destination is reached, it will eventually find distances to all vertices. Thus it makes sense to consider existing any-angle path planning algorithms. It is clear, however, that algorithms that first find a suboptimal path and then smooth it (for example, A* on a graph with post smoothing (Thorpe and Matthies, 1984)) are not suitable for high-performance tasks, because they process each destination separately.

Algorithms which can be easily adapted for single-source path planning include Theta* (Nash, Daniel, et al., 2007) and its modifications (Nash, Koenig, and Tovey, 2010), as well as Field A* (Koenig and Likhachev, 2005). Theta* developers (Nash and Koenig, 2013) consider these as interleaving the A* search and the smoothing. These algorithms assume a graph representation of the grid.

Another approach is to first identify special points on the map (usually associated with corners of obstacles), and then do a search in the graph constructed from these points. Visibility graph (Lozano-Pérez and Wesley, 1979) is a method of this type, but search in this case is typically slow, since the number of edges can grow quadratically in the number of vertices (Daniel et al., 2010). Another method which can be considered an adaptation of visibility graphs for a grid (Uras and Koenig, 2015), is Subgoal Graphs (Uras, Koenig, and Hernández, 2013). These are shown to be very fast for point-to-point path planning (Uras and Koenig, 2015), but they have to pre-process the map, requiring the knowledge of the whole map in advance. Contrarily, CWave has the potential to be applicable for partially known maps evolving as the environment is explored. Initial preprocessing required to construct Subgoal Graphs is reported (Uras, 2010) to be in the order of 100ms which is about 50 times slower than a single run of CWave on the tested maps. Moreover, Subgoal Graphs, as opposed to CWave, are not optimal and have not yet been adapted for single-source planning.

Conceptually CWave is a wave-propagation algorithm and, in this sense, is a special case of the Fast Marching Method (Sethian, 1996) where the interface velocity is constant. It is also similar to a well-known wave (Lee) algorithm (C. Lee, 1961) that deals with octagonal or square waves propagating over an 8- or 4-connected graph, respectively. In
Figure 4-2: The gradual expansion of a circular wave from the start source point $A$ (blue wave) allows to assign distances to all points directly visible from $A$. Then at every point where the wave meets an obstacle (corner points $B$, $C$, $D$, $E$), a new source point can be placed. Simultaneous expansion of circular waves from new sources (green waves expanding from points $B$, $C$, $D$, $E$) allows to further assign distances to points in the bounded area. At a certain moment, some of the waves may merge (for example, waves expanding from sources $H$ and $G$).

Case of CWave, however, the wave front has a circular shape (that’s what “C” stands for in “CWave”) to the extent permitted by the grid. The main idea of CWave is to abandon the graph model and operate directly on the grid geometry using discrete geometric primitives (discrete circular arcs and lines), instead of individual vertices, to represent the wave front.

Let us first consider a continuous space (Fig. 4-2) where we want to calculate distances from a given point $A$ to all other points of the bounded area. The gradual expansion of a circular wave from the start source point $A$ (blue wave) allows to assign distances to all points directly visible from $A$. Then at every point where the wave meets an obstacle (corner points $B$, $C$, $D$, $E$), a new source point can be placed. Simultaneous expansion of circular waves from new sources (green waves expanding from points $B$, $C$, $D$, $E$) allows to further assign distances to points in the bounded area. At a certain moment, some of the waves may merge (for example, waves expanding from sources $H$ and $G$). This procedure, in the same way as the Lee algorithm, allows to find paths from a given source point to all other reachable points on a 2D map. The circular shape of the wave front, however, in our case ensures that the paths are the shortest.
Note that there are two key geometric primitives making this whole construction possible: circular arcs which limit the wave on the front, and straight lines which limit the wave on the sides.

CWave allows to iterate through vertices on a grid while gradually incrementing the distance from a given source vertex. This can be used for 2D-space segmentation, for example, a map can be segmented into several regions with boundaries that are equidistant curves or extremals (lines perpendicular to equidistant curves) for a given source vertex. That is how CWave is used in Chapter 5.

In order to demonstrate a high-performance adaptation of these concepts for 2D grid, we decompose the problem. In Section 4.2 we consider wave propagation on an obstacle-free grid, in Section 4.3 we introduce simply-connected obstacles, and finally Section 4.4 discusses CWave on a generic map. Performance tests are presented in Section 4.5. An analysis of the algorithm and future work are discussed in Section 4.6.

### 4.2 CWave on a grid without obstacles

In general, 2D grid is a set of uniformly arranged square *cells*, where each cell can be either occupied or free, and grid *vertices* coincide with the vertices of the square cells (Fig. 4-3). In this section, we will discuss the most basic case of CWave propagation on a grid without occupied cells. It should be noted that, similar to Theta*, CWave calculates paths between grid vertices not between cell centers. This is not a completely free “design choice”, it has a strong theoretical motivation. From the calculus of variations (Elsgolc, 2007), it is known that the shortest path between any two points in a constrained 2D space consists of straight line segments and parts of obstacle boundaries, but the grid model implies that any obstacle is a set of occupied square cells, and thus obstacle boundaries are also straight line segments. Therefore the shortest path is a polygonal curve with turning points at corners of the occupied cells (at vertices). Placing start and end points of the paths at vertices allows to keep those points in the same set with turning points.

Our goal is to assign each vertex a distance value in such order that would emulate a propagation of a circular wave. The midpoint circle (MPC) algorithm (J. Bresenham,
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>𝛿</td>
<td>distance error defined as 𝛿 = d − r</td>
<td>real</td>
</tr>
<tr>
<td>𝜌</td>
<td>exact distance of the shortest path from the start source to a given vertex</td>
<td>real</td>
</tr>
<tr>
<td>𝜌̅</td>
<td>integer distance value assigned by CWave to a given vertex, approximately</td>
<td>integer</td>
</tr>
<tr>
<td>ε</td>
<td>error function used to replace 𝛿 to avoid real number calculations</td>
<td>integer</td>
</tr>
<tr>
<td>d</td>
<td>exact distance from a given vertex (x, y) to its nearest source vertex</td>
<td>real</td>
</tr>
<tr>
<td>d̅</td>
<td>integer distance value assigned by CWave that is approximately equal to 2d</td>
<td>integer</td>
</tr>
<tr>
<td>r</td>
<td>radius of a circle iterated by the midpoint algorithm</td>
<td>integer</td>
</tr>
<tr>
<td>s</td>
<td>exact distance from the start source to a given source</td>
<td>real</td>
</tr>
<tr>
<td>̅s</td>
<td>rounded 2s</td>
<td>integer</td>
</tr>
</tbody>
</table>

Table 4.1: List of variables used in this chapter.

1977b), also known as Bresenham’s circle algorithm, can be utilized for this purpose. It was developed to paint out pixels (cells) on a digital display in a shape of circular arcs. It is highly efficient, because it requires only integer arithmetics and multiplication by 2 for calculations. In this work, we will employ it to iterate through grid vertices, rather than cells.

### 4.2.1 Overview of midpoint circle algorithm

Without loss of generality, we will discuss only octant 1 on XY-plane with the origin at the start vertex. Octant 1 is defined as x > 0, y ≥ 0, and x ≥ y. In other octants the algorithm is similar.

The main idea of MPC algorithm is that in octant 1 when drawing a circular arc of an integer radius r, y-coordinate of each new vertex is always incremented by 1, whereas the x-coordinate is either decremented by one (vertices B → C in Fig. 4-3) or stays the same (vertices A → B in Fig. 4-3). If we define the distance error\(^1\) at any vertex D(x, y) as a difference between the real distance d to D and the circle of integer radius r:

\[
\delta_D = \delta_D' = \delta(x, y, r) = d - r = \sqrt{x^2 + y^2} - r
\]

\(^1\)Table 4.1 lists the key variables used in this chapter.
then between two potential candidates (E and F in Fig. 4-3), vertex E can be chosen over F if

\[ |\delta_E| < |\delta_F|. \]  

To simplify the calculations, Bresenham introduced another error function:

\[ \varepsilon_D = \varepsilon_D = \varepsilon(x_D, y_D, r) = (x_D^2 + y_D^2) - r^2 \]  

By definition, \( \varepsilon \) is always an integer, does not require a square root for calculation, and in the domain of interest is monotonously related to \( \delta \). Thus, condition (4.2) can be replaced with:

\[ |\varepsilon_E| < |\varepsilon_F|. \]  

105
If we take $D = (x, y)$, then

$$
\varepsilon_D = \varepsilon(x, y, r) = x^2 + y^2 - r^2 \quad (4.5)
$$

$$
\varepsilon_F = \varepsilon(x, y + 1, r) = x^2 + (y + 1)^2 - r^2
= \varepsilon(x, y, r) + 2y + 1 = \varepsilon_D + 2y + 1 \quad (4.6)
$$

$$
\varepsilon_E = \varepsilon(x - 1, y + 1, r) = (x - 1)^2 + (y + 1)^2 - r^2
= \varepsilon(x, y + 1, r) - 2x + 1 = \varepsilon_F - 2x + 1 \quad (4.7)
$$

To eliminate the absolute value function, we can write (4.4) as

$$
\varepsilon_E^2 - \varepsilon_F^2 < 0 \quad (4.8)
$$

And then, in view of (4.6) and (4.7):

$$
(\varepsilon_F - 2x + 1)^2 - \varepsilon_F^2 < 0
$$

$$
(\varepsilon_F - 2x + 1 - \varepsilon_F)(\varepsilon_F - 2x + 1 + \varepsilon_F) < 0
$$

$$
(-2x + 1)(\varepsilon_F - x + 0.5) < 0 \quad (4.9)
$$

For a positive integer $x > 0$, $(-2x + 1) < 0$, thus the inequality is further simplified:

$$
\varepsilon_F > x - 0.5
$$

Finally, given that $x = x_D = x_F$ and $\varepsilon_F$ are integers, (4.9) becomes:

$$
\varepsilon_F \geq x_F \quad (4.10)
$$

Condition (4.10) defines whether $x$-coordinate needs to be decreased by 1 or not, whereas equations (4.6) and (4.7) are used to incrementally calculate $\varepsilon$. Note that only integer addition, subtraction and bit shifting (multiplication by 2) are used. For circles, the initial value of $\varepsilon$ is $\varepsilon_0 = \varepsilon(r, 0, r) = 0$, for arcs, it has to be determined by another method (see Section 4.3).
4.2.2 Distance error of Bresenham circles

Based on (4.1) and (4.3), we can express error function $\varepsilon$ as a function of distance error $\delta$:

$$\varepsilon(\delta) = \delta^2 + 2r\delta$$  \hspace{1cm} (4.11)

Fig. 4-4 depicts the continuous mapping defined by (4.11), where point $M$ is the minimum of $\varepsilon(\delta)$, and $L$, $R$, and $N$ correspond to $\delta$ values of $-0.5$, $0.5$ and $1$, respectively. It should be noted, that in reality $\varepsilon$ can take only integer values, hence $\delta$ (the domain of the function) is restricted to the corresponding isolated individual points.

In order to analyze the distance accuracy of the method, it will be necessary to find bounds for the distance error.

**Lemma 1.** In all vertices iterated by MPC algorithm

$$|\delta| < 0.5$$  \hspace{1cm} (4.12)

**Proof.** Given that the initial value of distance error $\delta_0 = \delta(r, 0, r) = 0$, to prove by induction, we need to show that if (4.12) is true for the distance error at $k$-th step $\delta_k$, then it’s also true for $\delta_{k+1}$. From (4.11), assuming that $|\delta_k| < 0.5$ and taking into account that $\varepsilon$ is
an integer, we will have

\[-r + 1 \leq \varepsilon_k \leq r \quad (4.13)\]

**Case 1: Condition (4.10) is false.** Based on the update rule (4.6):

\[\varepsilon_{k+1} = \varepsilon_k + 2y_k + 1 < x_k \quad (4.14)\]

This, in combination with (4.13), implies:

\[-r + 2y_k + 2 \leq \varepsilon_{k+1} < x_k \quad (4.15)\]

Now, given that in octant 1, \(x_k \leq r\) and \(y_k \geq 0\), (4.15) implies

\[-r + 0.25 < \varepsilon_{k+1} < r + 0.25 \quad (4.16)\]

which based on Fig. 4-4, entails \(|\delta_{k+1}| < 0.5\).

**Case 2: Condition (4.10) is true.** Based on the update rules (4.6) and (4.7):

\[\varepsilon_{k+1} = \varepsilon_k + 2(y_k - x_k + 1)\]

\[= \varepsilon_k + 2(y_{k+1} - x_{k+1} - 1) \geq x_k \quad (4.17)\]

This, in combination with (4.13), implies:

\[x_k \leq \varepsilon_{k+1} < r + 2(y_{k+1} - x_{k+1} - 1) \quad (4.18)\]

Now, given that in octant 1, \(0 \leq y_{k+1} \leq x_{k+1}\), and thus \((y_{k+1} - x_{k+1} - 1) \leq -1\), then

\[0 \leq \varepsilon_{k+1} < r - 2 \quad (4.19)\]

Based on Fig. 4-4, the last system of inequalities implies that \(0 \leq \delta_{k+1} < 0.5\).
4.2.3 Non-Bresenham points and their properties

It is noted that not all vertices on the grid are accessible by MPC algorithm. Indeed, in Fig. 4-5, we can see Bresenham circles of integer radii $r \leq 7$. Numbers at the vertices designate the radius, vertices that haven’t been visited are marked with N. In this chapter, these vertices are referred to as Non-Bresenham Points (NBPs). For path planning, it is important to iterate through all vertices to check their reachability, thus for further analysis, we need to investigate how NBPs are formed.

Let us consider a configuration shown in Fig. 4-6a. Here vertices A and B corresponding to circles of radii $r$ and $(r+1)$, respectively, are adjacent, but then on the next iteration
Figure 4-6: Formation of Non-Bresenham points (NBPs). NBPs are marked with red. (a) Next vertex of \((r + 1)\)-circle after \(E\) is always \(F\) (b) If next vertex after \(C\) is \(I\), then no new NBPs (c) If next vertex after \(C\) is \(H\), then \(I\) is a new NBP, and next vertex after \(F\) is always \(J\)

\(r\)-circle moves diagonally to vertex \(C\), whereas \((r+1)\)-circle goes up to vertex \(E\). This leaves vertex \(D(x, y)\) unvisited, thus making it an NBP. We will now show that in this configuration, the next vertex for \((r+1)\)-circle is always \(F\).

According to condition (4.10) of MPC algorithm, we just need to prove that \(\varepsilon^r_{G} + 1 \geq x\).

\[
\varepsilon^r_{G} = \varepsilon(x+1, y+1, r+1) = \varepsilon^r_{D} + 2(x + y - r) + 1
\]

(4.20)

Based on condition (4.10):

\[
\varepsilon^r_{D} \geq x > 0
\]

(4.21)

which means that \(\delta^r_{D} > 0\) and \(r < \sqrt{x^2 + y^2}\). At the same time, from the triangle inequality, \(\sqrt{x^2 + y^2} \leq (x + y)\), and thus

\[
(x + y - r) > 0
\]

(4.22)

After substituting (4.22) and (4.21) into (4.20), we will have

\[
\varepsilon^r_{G} + 1 > x + 1 > x
\]

(4.23)

Now we can explore the genesis of NBPs a bit more. If \(\varepsilon^r_{I} < x\) (Fig. 4-6b), then
$r$-circle will proceed to vertex $I$, and no new NBP will be formed. If, however, $\varepsilon_I^r \geq x_I$ (Fig. 4-6c), $r$-circle will proceed to vertex $H$, and $I$ will become an NBP. In this case, we can apply the same proof that was used for points $D$ and $F$, to show that the next vertex of $(r+1)$-circle will always be $J$.

### 4.2.4 Necessary and sufficient condition for an NBP

**Proposition 1.** When condition (4.10) is checked for some vertex $F$ while iterating through an $r$-circle, $F$ will be an NBP of the $r$-circle iff (4.10) is true and

$$
\varepsilon_F^r \leq 2r - x_F \tag{4.24}
$$

**Proof.** Based on condition (4.10), in scenario shown in Fig. 4-6a, where initial points $A$ and $B$ of $r$- and $(r+1)$-circles are adjacent, the necessary and sufficient condition for vertex $D(x, y)$ to be an NBP is:

$$
\begin{cases}
\varepsilon_D^r \geq x_D \\
\varepsilon_E^{r+1} < x_E
\end{cases} \tag{4.25}
$$

Given that $\varepsilon_E^{r+1} = \varepsilon(x + 1, y, r + 1) = \varepsilon_D^r + 2(x - r)$, the second condition in (4.25) can be written as $\varepsilon_D^r - 2r + x_D < 1$, and since all variables are integers:

$$
\varepsilon_D^r - 2r + x_D \leq 0 \tag{4.26}
$$

And while for the second scenario shown in Fig. 4-6c (vertex $I$), where initial vertices $C$ and $D$ of $r$- and $(r+1)$-circles are separated by an NBP vertex $D$, the first condition is
sufficient, we can show that (4.26) still holds true. Indeed,

\[ \varepsilon^r_I = \varepsilon(x_D - 1, y_D + 1, r) = \varepsilon^r_D + 2(y_D - x_D + 1) \]  
(4.27)

\[ (\varepsilon^r_I - 2r + x_I) = \varepsilon^r_D - 2r + x_I + 2(y_D - x_D + 1) \]  
(4.28)

\[ = (\varepsilon^r_D - 2r + x_D) + 2(y_D - x_D) + 1 \]  
(4.29)

\[ \leq 2(y_D - x_D) + 1 \]  
(4.30)

In octant 1 \((y_I - x_I) \leq 1\), and then \((y_D - x_D) \leq -1\) and \(2(y_D - x_D) + 1 < 0\). We thus showed that \(\varepsilon^r_I - 2r + x_I < 0\).

\[ \square \]

4.2.5 Distance error in Non-Bresenham points

Based on conditions (4.10) and (4.24), we can determine bounds for distance error at NBPs. Indeed, given that in octant 1, \(r \leq \sqrt{2}x\), conditions (4.10) and (4.24) imply that

\[ (1/\sqrt{2})r \leq \varepsilon_{NB\!P} \leq (2 - 1/\sqrt{2})r \]  
(4.31)

or, roughly,

\[ 0.7r < \varepsilon_{NB\!P} < 1.3r \]  
(4.32)

Based on (4.11) (Fig. 4-4), this guarantees that

\[ 0 < \delta_{NB\!P} < 1 \]  
(4.33)

We can determine tighter bounds for distance error in NBPs. It’s easy to verify that if \(r \geq 3\), then \((1/\sqrt{2})r > 0.33^2 + (2r)(0.33)\). Thus

\[ 0.33^2 + (2r)(0.33) < \varepsilon_{NB\!P} < 0.65^2 + (2r)(0.65) \]  
(4.34)
Based on the relation (4.11), we then conclude that

\[ 0.33 < \delta_{NBP} < 0.65 \]  

(4.35)

The only NBP with \( r < 3 \) is \((1, 1)\), where \( \delta_{NBP} \approx 0.414 \), thus bounds (4.35) are correct for all NBPs.

### 4.2.6 Possible vertex configurations

In order to visit all vertices of the grid, it is necessary to develop a method of visiting NBPs while iterating through regular Bresenham vertices. The properties of the NBP formation process described above determine that NBPs cannot be horizontally or vertically adjacent to each other, and, thus, every NBP can be paired with a regular vertex to the left from it (in octant 1) as shown in Fig. 4-5. And thus to visit all vertices of the grid, whenever there is a diagonal move (such as \( D \rightarrow E \) on Fig. 4-3), an additional condition (4.24) has to be checked. If it’s true, then vertex \( F \) is an NBP that also belongs to the Bresenham \( r \)-circle and has to be visited and assigned the distance.

We can also observe that among all imaginable configurations of two sequential vertices/pairs (Fig. 4-7), only the following five are valid: (a), (b), (e), (g) and (h). Indeed, (c) and (d) are impossible because NBPs can only be formed when there’s a diagonal move while iterating a Bresenham circle (for example, \( A \rightarrow C \) in Fig. 4-6a). Configuration (f) is impossible because the diagonal move would result in an NBP to the right from the top vertex (Fig. 4-6c).

It can also be shown that another configuration of diagonal NBP vertices is impossible. Even though the current implementation of CWave does not rely on the following proposition, we present it and its proof here as it might be used in the future (can be skipped at first reading).

**Proposition 2.** The diagonal configuration of NBPs in octant 1 shown in Fig. 4-8 is impossible.

**Proof.** We will employ a proof by contradiction. Based on the law of cosines for \( \triangle SAB \)
Figure 4-7: Possible configurations of two sequential vertices/pairs while iterating through a Bresenham circle.

Figure 4-8: An impossible configuration of two diagonal NBPs.
Figure 4-9: Curve \( d_2(d_1) = \sqrt{(d_1 + a)^2 + (10 - a^2)} \) and its asymptotes

(refer to Fig. 4-8):

\[
d_2 = \sqrt{d_1^2 + |AB|^2 - 2d_1|AB|\cos \alpha}
\]

\[
d_2 = \sqrt{d_1^2 + 10 + 2\sqrt{10}d_1 \cos (\beta - \arctan \frac{1}{3})}
\]

\[
d_2 = \sqrt{d_1^2 + 2d_1a + 10} = \sqrt{(d_1 + a)^2 + (10 - a^2)}
\]

where

\[
a = \sqrt{10} \cos \left( \beta - \arctan \frac{1}{3} \right)
\]

Fig. 4-9 depicts the plot of \( d_2(d_1) \) curve. Given the asymptotic properties of the curve, for all \( d_1 \):

\[
d_2 - d_1 > a \tag{4.40}
\]

\[
r + 2 + \delta_2 - (r + \delta_1) > a \tag{4.41}
\]

\[
\delta_2 - \delta_1 > a - 2 \tag{4.42}
\]
Figure 4-10: Proving an impossible diagonal configuration of NBPs. (a) The previous vertex for \( B \) is \( D_1 \), not \( B_1 \); (b) If the previous vertex for \( B \) is \( B_1 \), then the diagonal NBP configuration has to repeat itself at vertices \( C_1 \) and \( D_1 \), and it can be continued indefinitely.

We can see from (4.39) that for \( \beta \leq 2 \arctan \frac{1}{3} \):

\[
a \geq 3 \tag{4.43}
\]

and thus from (4.42)

\[
\delta_2 - \delta_1 > 1 \tag{4.44}
\]

which contradicts the distance error boundaries in (4.12), thus we only need to prove the statement for

\[
\beta > 2 \arctan \frac{1}{3} \tag{4.45}
\]

First, let us show that the previous vertex for \( B \) must have been \( D_1 \), not \( B_1 \) (see Fig. 4-10a). Indeed, if the previous vertex is \( B_1 \) (refer to Fig. 4-10b), then \( D_1 \) is an NBP as well, and then the previous vertex for \( E \) is \( E_1 \), and then \( C_1 \) is an NBP. We can see that now we have a new pair of diagonal NBPs \( C_1 \) and \( D_1 \), which is located below \( C \) and \( D \). We can continue this procedure further until we reach \( A_N \) for which \( \beta \leq 2 \arctan \frac{1}{3} \), but we showed that for such values of \( \beta \) the given configuration is impossible. Thus, based on
condition (4.10) for vertex $B$, we can write:

$$\varepsilon_B < x_B = x + 3 \quad (4.46)$$

On the other hand, given (4.45),

$$x < r \cos \beta = \frac{1 - \tan^2(\arctan \frac{1}{3})}{1 + \tan^2(\arctan \frac{1}{3})} r = \frac{1 - \frac{1}{9}}{1 + \frac{1}{9}} r = 0.8r \quad (4.47)$$

and thus

$$\varepsilon_B < 0.8r + 3 \quad (4.48)$$

We can bring inequality (4.48) into a form suitable for estimating distance error $\delta_1$ at vertex $A$ (see (4.11)). Indeed, for $r \geq 142$, $0.8r + 3 < 0.41^2 + (2r)(0.41)$, and thus for such values of $r$

$$\varepsilon_B < 0.8r + 3 < 0.41^2 + (2r)(0.41) \quad (4.49)$$

The last inequality gives us an upper boundary for $\delta_2$:

$$\delta_2 < 0.41 \quad (4.50)$$

Now similarly we can find a lower boundary for $\delta_1$. From condition (4.10) for vertex $C$:

$$\varepsilon_C \geq x_C = x + 1 \quad (4.51)$$

and thus

$$\varepsilon_A = \varepsilon_C - 2x_C + 1 = \varepsilon_C - 2x - 1 \geq -x > -0.8r \quad (4.52)$$
Again, for \( r \geq 9 \), \(-0.8r > (-0.41)^2 + (2r)(-0.41)\), and thus

\[
\delta_1 > -0.41 \quad (4.53)
\]

On the other hand, even though the upper boundary for \( \beta \) in octant 1 can be slightly bigger than 45° (if NBPs are located on 45° diagonal), still for \( r > 150 \), \(|SC| > 150 \Rightarrow \tan \beta < \frac{150}{149} \Rightarrow \)

\[
a > \sqrt{10} \cos \left( \arctan \frac{150}{149} - \arctan \frac{1}{3} \right) > 2.82 \quad (4.54)
\]

and, thus, from (4.42):

\[
\delta_2 - \delta_1 > 0.82 \quad (4.55)
\]

Inequalities (4.55), (4.53) and (4.50) contradict each other, therefore the proposition is proven for \( r > 150 \). For \( r \leq 150 \) the proposition can be verified by a computer program or manually.

4.2.7 Assigning integer distance values to vertices

Now that we are able to iterate through all vertices, we need to assign each vertex \( P \) an integer distance value \( d_P \). One way of doing it is to define \( d_P = r \). In this case, however, given the distance error bounds (4.12) and (4.33) the true distance \( d_P \) will stay in a rather large interval

\[
\overline{d}_P - 0.5 < d_P < \overline{d}_P + 1 \quad (4.56)
\]

We developed a distance assignment method that allows to shrink the length of the interval to 0.5. It achieves three goals: 1) allows fractional numbers in the integer variable \( \overline{d}_P \) by using \( 2r \) instead of \( r \), 2) adds an additional 0.5-offset if \( d_P > r \), 3) adds an additional
0.5-offset to NBP vertices:

\[
\overline{d}_P = \begin{cases} 
2r + 1 & \varepsilon_P > 0 \quad \text{if } P \text{ is a regular vertex} \\
2r & \varepsilon_P \leq 0 \\
2r + 2 & \varepsilon_P > r \\
2r + 1 & \varepsilon_P \leq r 
\end{cases}
\] (4.57)

Overall, this assignment guarantees that

\[
\frac{\overline{d}_P}{2} - 0.5 < d_P \leq \frac{\overline{d}_P}{2}
\] (4.58)

Indeed, from (4.11) (Fig. 4-4), given that \( \varepsilon_P \) and \( r \) are integers, the following relations follow:

\[
\varepsilon_P > 0 \Rightarrow \delta_P > 0 \quad \varepsilon_P \leq 0 \Rightarrow \delta_P \leq 0 \quad (4.59)
\]

\[
\varepsilon_P > r \Rightarrow \delta_P > 0.5 \quad \varepsilon_P \leq r \Rightarrow \delta_P < 0.5 \quad (4.60)
\]

When these inequalities are combined with (4.12) and (4.33), the inequality (4.58) becomes evident.

### 4.3 CWave on a grid with simply-connected obstacles

By this point, we have established how CWave assigns distances in the special case of a free map. In this section, we will consider maps with *simply-connected obstacles*, that is maps where any two occupied cells can be connected by a path that never passes through a free cell. This topology guarantees that during CWave expansion no vertex is visited more than once. We can, thus, defer the problem of wave merging to Section 4.4. First, however, we need to introduce some definitions.
Figure 4-11: Visibility of various vertices from vertex $S$. Visible vertices: $B$, $D$. Not-visible vertices: $A$ ($SA$ passes through inner points of cell [8, 4]), $C$, $E$, $F$ (crossing line segment connecting two adjacent occupied cells)

4.3.1 Coordinate frames and definitions

Similarly to the Theta* approach, we add a single-occupied-cell wall around the perimeter of every map, as shown on Fig. 4-11, to eliminate the need for out-of-boundary checks. The bottom left cell has absolute coordinates [0, 0]. Vertex located at the left bottom corner of cell $[x, y]$ has same coordinates $[x, y]$. Square brackets $[.]$ are used for absolute coordinates, and parentheses $(.)$ are used for relative coordinates (w.r.t. the source vertex). For example, vertex $A$ in Fig. 4-11 has absolute coordinates [9, 5] and relative coordinates (with reference to source $S$) (7, 3).

We introduce the following definition of visibility of vertex $Y$ from vertex $X$ (Fig. 4-11).

**Definition 1.** Vertex $Y$ is NOT visible from vertex $X$ if there is at least one point of the open line segment $XY$ that 1) is also an inner point of any occupied cell OR 2) belongs to a line segment connecting centers of any two adjacent occupied cells. Otherwise, it is visible.

Fig. 4-11 demonstrates several examples. Here vertex $A$ is not visible from $S$, because
the line segment $AS$ crosses an occupied cell $[8, 4]$, and, thus, there are inner points of $AS$ that are also inner points of that cell. Vertex $C$ is not visible, because vertex $[8, 5]$ (an inner point of $SC$), belongs to the line segment connecting centers of two adjacent occupied cells: $[7, 5]$ and $[8, 4]$. Similarly, vertex $E$ is not visible, because vertex $D$ is the inner point pf $SE$ and belongs to the line segment connecting centers of two adjacent occupied cells: $[7, 2]$ and $[8, 1]$ Note, however, that $D$ itself is visible because $D$ is not an inner point of line segment $SD$. Vertex $F$ is not visible, because $SF$ crosses line segment connecting adjacent occupied cells $[1, 4]$ and $[2, 4]$. Finally vertex $B$ is visible because inner points of $SB$ do not overlap with any occupied cell or any line segment connecting two adjacent occupied cells.

We will say that cell $A$ corresponds to vertex $B$ if $B$ is the furthermost corner of cell $A$ (as measured from the source). In octant 1, thus, vertex $A \ (y > 0)$ will not be visible if its corresponding cell is occupied. Overall, six occupancy patterns are possible. In Fig. 4-12 they are classified into free, non-free, semi-free, and occupied. If all corresponding cells are free, we call the pattern free, otherwise the pattern is non-free. The non-free patterns are further divided into occupied (all corresponding cells are occupied), and semi-free (some corresponding cells are free, some are occupied). Note the pattern names comprised of symbols O (designating a free cell) and X (designating an occupied cell).

### 4.3.2 Visibility cone

Consider a configuration shown on Fig. 4-13. Here a circular wave is expanding from source $S$, as occupied cells, represented by the shaded squares, block visibility of certain
Figure 4-13: An example visibility cone $S$, where $S$ is the start source vertex, $A$ and $B$ are secondary sources, $\alpha$ and $\beta$ are continuous boundaries of the cone $S$, Solid blue squares are discrete closed boundaries of the cone $S$, Circular green outlines are discrete open boundaries of cones $A$ and $B$, Cells marked with question marks can block parts of the visibility cone $S$, even though their corresponding vertices (blue square outlines) are not visible from $S$. 

\[ \alpha : y = \frac{y_A}{x_A} x \]

\[ \beta : y = \frac{y_B}{x_B} x \]
4.3.3 Identifying corner vertices

In section 4.2.6 we showed that there are only five valid configurations of vertices on two sequential iterations (Fig. 4-7). Given the six possible occupancy patterns in Fig. 4-12, we can identify 40 possible combinations (Fig. 4-14).

For all combinations, except for a special case of $OX \rightarrow XO$, a corner vertex appears only when a non-free pattern follows a free pattern or vice versa. In the first case, the new source will have a boundary of type $A$, whereas in the second case the new source will have a boundary of type $B$. Combination $OX \rightarrow XO$ is classified as a special case (Fig. 4-15). Even though a non-free pattern $XO$ follows another non-free pattern $OX$, a narrow visibility cone can still pass between the occupied cells. And in this case two new sources
Figure 4-15: Combination $OX \rightarrow XO$ allows a narrow visibility cone to pass between the occupied cells are placed at $A$ and $B$.

### 4.3.4 Iterating through boundary vertices

To iterate through boundary vertices (solid blue squares on Fig. 4-13), we developed a modified version of the Bresenham’s line algorithm (J. E. Bresenham, 1965). It needs only integer addition to operate. First, we will consider boundary of type $A$. At every step, the $x$-coordinate is incremented, but the $y$-coordinate is only incremented if

$$y + 1 \leq \frac{y_A}{x_A} (x + 1)$$

(4.61)

It can be noted that this expression has multiplication and division in it. If, however, we introduce function

$$G(x, y) = (x+1)y_A - (y+1)x_A$$

(4.62)
then the condition can be transformed into:

\[ G(x, y) \geq 0 \]  

(4.63)

It is easy to observe, that \( G(x, y) \) can be calculated iteratively using only integer addition. Indeed, the initial value for \( G \):

\[ G(x_A, y_A) = y_A - x_A \]  

(4.64)

and the update rules (depending on whether \( y \) is incremented or not) are

\[ G(x+1, y) = G(x, y) + y_A \]  

(4.65)

\[ G(x+1, y+1) = G(x, y) + y_A - x_A \]  

(4.66)

For type-\( B \) boundaries, \( y \)-coordinate is incremented if

\[ y < \frac{y_B}{x_B} (x + 1) \]  

(4.67)

or, if we introduce

\[ G(x, y) = (x+1)y_B - yx_B - 1 \]  

(4.68)

then the condition can be transformed into:

\[ G(x, y) \geq 0 \]  

(4.69)

The initial value for \( G \):

\[ G(x_A, y_A) = y_B - 1 \]  

(4.70)
and the update rules are

\[
G(x+1, y) = G(x, y) + y_B
\]
\[
G(x+1, y+1) = G(x, y) + y_B - x_B
\]

(4.71)

(4.72)

The boundaries, thus, can be iterated using only integer addition operation. Types \(A\) and \(B\) differ only in the initial value for \(G\). Note that the value of the error function \(\varepsilon\), is propagated along the boundary vertices, again using only integer addition and bit-shifting:

\[
\varepsilon(x+1, y, r+1) = \varepsilon(x, y, r) + 2(x - r)
\]
\[
\varepsilon(x+1, y+1, r+1) = \varepsilon(x, y, r) + 2(x + y - r) + 1
\]

(4.73)

(4.74)

These equations determine the initial value of \(\varepsilon\) for the MPC-algorithm.

### 4.3.5 Determining arc endpoints from the boundary vertices

CWave iterates through MPC arcs and increments the radius at every step. In some cases, however, when moving from one boundary vertex to the next one, the radius increases by 2, not by 1. For example, in Fig. 4-16, when moving from vertex \(A\) to vertex \(B\) the radius changes from 7 to 9. In fact the correct starting vertex after \(A\) \((r = 7)\) is \(C\) \((r = 8)\), not \(B\) \((r = 9)\). Such configurations are easy to detect, because for vertex \(B\) condition (4.10) will hold true. The arc endpoint vertex may also have an NBP vertex paired with it. That has to be checked with condition (4.24).

Another caveat is that even though the vertices marked with blue square outlines (not solid squares) in Fig. 4-13, are located outside of the visibility cone \(S\), their corresponding cells (marked with question marks) can actually block visibility of some of the vertices inside the cone, thus the occupancy of those cells has to be checked as well when calculating the arc endpoints.
4.3.6 Distance accuracy on a simply-connected map

When any new source is added (see Section 4.3.3), its initial distance value $\bar{s}$ is taken from the integer distance previously assigned to the vertex by CWave. Given the distance error boundaries (4.58), each new source may, thus, introduce an accumulative distance error $|\delta| < 0.5$. That is the price of the purely integer arithmetic (addition and bit-shifting operations only) solution for the path planning problem. We will refer to such integer implementation of CWave as CWaveInt.

To eliminate the accumulative error, we developed the following modification of CWave (CWaveFpuSrc, where Fpu stands for Floating-Point Unit, Src stands for source). In addition to the integer source distance $\bar{s}$, for every source, we maintain a floating-point source distance $s$. It is initialized as

$$ s = \sqrt{x_S^2 + y_S^2 + s_{parent}} $$

but then is immediately rounded as

$$ \bar{s} = \text{round}(2s) $$
The floating-point value $s$ is used only when the source creates new sources at corner vertices (see Section 4.3.3). For basic distance assignment only integer value $\bar{s}$ is employed. This approach allows every source to maintain the exact length of the shortest path to the start vertex (to the extent permitted by the machine epsilon), and, thus, eliminates the accumulative distance error.

We can calculate the total error of integer distances assigned by CWave in this case. First, from 4.76:

$$\frac{\bar{s}}{2} - 0.25 \leq s < \frac{\bar{s}}{2} + 0.25$$  \hspace{1cm} (4.77)$$

Now, combining this with (4.58), we will have:

$$\frac{s + \bar{d}}{2} - 0.75 < s + d < \frac{s + \bar{d}}{2} + 0.25$$  \hspace{1cm} (4.78)$$

$$\frac{\bar{\rho}}{2} - 0.75 < \rho < \frac{\bar{\rho}}{2} + 0.25$$  \hspace{1cm} (4.79)$$

where $\rho = s + d$ is the exact length of the shortest path from the start source to a given vertex, and $\bar{\rho} = \bar{s} + \bar{d}$ is the corresponding CWave integer distance value.

Given the minimal computational overhead of this modification (4.75 is calculated only when a new source is created), it is used in the further analysis.

### 4.3.7 Path extraction on simply-connected maps

Path extraction is the process of determining the shortest path (not just the length of the path) from the initial source vertex to any other vertex on the map. When there is a need for path extraction, two additional data structures are maintained: a 1D ordered \textit{sources\_array} that accumulates coordinates of all source vertices and a 2D \textit{track\_map} that for every vertex on the map, stores the index of the source that assigned the vertex its distance value. Once the distance map is calculated, a path from any given vertex $P(x_P, y_P)$ can be recursively extracted: read $P$’s nearest source id $S_{1_{id}} = \text{track\_map}[P.x, P.y]$,
read $S_1$’s coordinates: $S_1 = sources\_array[S1\_id]$, read $S_1$’s nearest source id $S2\_id = track\_map[S1.x, S1.y]$, and so on until the start source vertex is reached. The modification of CWave that maintains $sources\_array$ and $track\_map$ is referred to as CWaveTrack.

In future, gradient methods can be explored for path extraction directly from the distance map. Such approach would render $sources\_array$ and $track\_map$ unnecessary, but, given the integer nature of CWave distance map, it may encounter certain challenges.

4.4 CWave on a generic map

On a generic map, obstacles are not necessarily simply-connected, and thus waves from two or more sources may eventually “meet” each other. This section discusses the case of merging waves for a realistic 2D map in which we release the assumption of simply-connected obstacles.

4.4.1 Boundary between merging waves in continuous space

Fig. 4-17 illustrates the scenario in which two waves are merging. Here $S$ is a parent source for secondary sources $A$ and $B$ which generate wave-$A$ (red vertices) and wave-$B$ (blue vertices), respectively. These two waves merge along a certain boundary $\alpha$. In a continuous 2D space, we can define $\alpha$ as a set of points $P$ for which the lengths of the polygonal paths $SBP$ and $SAP$ are equal.

**Proposition 3.** $\alpha$ is a hyperbola.

**Proof.** By definition of $\alpha$:

\[
|SB| + |BP| = |SA| + |AP| \tag{4.80}
\]

\[
|AP| - |BP| = |SB| - |SA| \tag{4.81}
\]

\[
|AP| - |BP| = const \tag{4.82}
\]

The last equation is one of the standard definitions of a hyperbola. \qed
Figure 4-17: A merge of wave-\(A\) and wave-\(B\). Red and blue numbers designate \(\text{CW}{}\) wave distance from the source \(S\) to the given vertex for paths passing through \(A\) and \(B\), respectively. For pink and light blue vertices, the integer criterion (4.83) is sufficient to determine which path from \(S\) is shorter: through \(A\) (pink) or through \(B\) (light blue). For yellow vertices the criterion is not sufficient, they form an overlapping area.

For vertices located on one side of the hyperbola, the shortest path to \(S\) passes through \(A\), and for vertices on the other side, the shortest path passes through \(B\).

### 4.4.2 Integer criterion

Let us imagine, what will happen if we run the current implementation of \(\text{CW}{}\) wave on a map with non-simply-connected obstacles as shown on Fig. 4-17. wave-\(A\) and wave-\(B\) will eventually overlap, that is some of the vertices on the map will be visited by both of the waves. Moreover each wave will independently place a new source at the top right corner of the obstacle (vertex \([3, 5]\)) and then at the next corner, and so on. The waves will “wrap”
around the obstacle infinitely.

Visiting the same vertex multiple times, while somewhat degrading the performance, will not affect the accuracy, if the new distance value assigned to the vertex is smaller than the previous value, but it’s critical to prevent infinite loops.

The following proposition serves as an integer criterion to determine when wave-\( B \) should stop penetrating wave-\( A \):

**Proposition 4. If for a given point \( P \)**

\[
\rho^A \leq \rho^B - 2 \tag{4.83}
\]

**then**

\[
\rho^A < \rho^B \tag{4.84}
\]

where \( \rho^A \) and \( \rho^B \) are integer distance values assigned to \( P \) by sources \( A \) and \( B \), respectively, and \( \rho^A \) and \( \rho^B \) are exact lengths of the paths \( SAP \) and \( SBP \).

**Proof.** From (4.79),

\[
\rho^A - 0.75 < \frac{\rho^A}{2}; \quad \frac{\rho^B}{2} < \rho^B + 0.25 \tag{4.85}
\]

Substitution of (4.85) into (4.83) proves the proposition.

The practical interpretation of this proposition is that if source \( B \) tries to assign value \( \rho^B \) to a certain vertex \( P \), whereas for the current value \( \rho^A \) of \( P \), inequality (4.83) holds true, then we can be certain that \( SAP \) is shorter than \( SBP \). That means that we can treat vertex \( P \) as if we reached an occupied cell and limit the visibility cone angular range of the source \( A \) to \( P \). That also implies that \( A \) should not place a new source at vertex \( P \).

For yellow vertices (Fig. 4-17), however, condition (4.83) is false, which means that the integer values \( \rho^A \) and \( \rho^B \), are not sufficient to determine which path is shorter for a given vertex \( P \): \( SAP \) or \( SBP \). The two waves, thus, **overlap** at the yellow vertices. If there is an occupied cell in that area, both waves may place a new source at the same corner.
4.4.3 Distance accuracy on a generic map

Proposition 5. Let \( \bar{p}^A \) be the integer distance value assigned by CWave to vertex \( P \) by source \( A \), and \( \bar{p}^B \) be the integer distance value that is later attempted to be assigned by CWave to vertex \( P \) by source \( B \). If we choose to update the distance value at \( P \) with \( \bar{p}^B \) only when

\[
\bar{p}^B < \bar{p}^A
\] (4.86)

then

\[
|\rho - \bar{p}| < 0.75
\] (4.87)

where \( \rho \) is the length of the true shortest path to \( P \), and \( \bar{p} \) is the integer distance value assigned by CWave.

Proof. Given the previous definitions of \( \bar{p}^A \) and \( \bar{p}^B \), the following cases are possible:

1. \( \bar{p}^B \leq \bar{p}^A - 2 \) (based on condition 4.86, path via \( B \) is chosen)

Proposition 4 implies that, in this case, \( \rho^B < \rho^A \), and thus path through vertex \( B \) is chosen correctly. This means that the error boundaries for \( \bar{p} = \bar{p}^B \) can be calculated using 4.79. The interval defined by 4.79 is contained in the interval defined by 4.87.

2. \( \bar{p}^B = \bar{p}^A - 1 \) (based on condition 4.86, path via \( B \) is chosen)

In this case, two sub-cases are possible:

(a) \( \rho^B \leq \rho^A \): path via \( B \) is chosen correctly, thus, again, error boundaries 4.79 are true;

(b) \( \rho^B > \rho^A \): path via \( B \) is chosen incorrectly, the length of the shortest path to \( P \) is \( \rho = \rho^A \), but, for \( \rho^A \), error boundaries 4.79 are still true:

\[
\frac{\bar{p}^A}{2} - 0.75 < \rho^A < \frac{\bar{p}^A}{2} + 0.25
\] (4.88)
And, thus, after substituting $\bar{\rho}^B = \bar{\rho}^A - 1$, we will have:

$$\frac{\bar{\rho}^B}{2} - 0.25 < \rho^A < \frac{\bar{\rho}^B}{2} + 0.75$$  \hspace{1cm} (4.89)

$$\bar{\rho} - 0.25 < \rho < \bar{\rho} + 0.75$$  \hspace{1cm} (4.90)

This interval is also contained in 4.87.

3. $\bar{\rho}^B = \bar{\rho}^A$: (based on condition 4.86, path via $A$ is chosen):

Since $\bar{\rho}^B = \bar{\rho}^A$, error boundaries 4.79 will be correct no matter which real distance $\rho^A$ or $\rho^B$ is shorter.

4. $\bar{\rho}^B = \bar{\rho}^A + 1$ (based on condition 4.86, path via $A$ is chosen):

This case is symmetric to case (2), that is, it can be rewritten as $\bar{\rho}^A = \bar{\rho}^B - 1$.

5. $\bar{\rho}^B \geq \bar{\rho}^A + 2$ (based on condition 4.86, path via $A$ is chosen):

This case is symmetric to case (1), that is, it can be rewritten as $\bar{\rho}^A \leq \bar{\rho}^B - 2$.

Proposition 5 provides distance error boundaries for a generic map.

### 4.4.4 Wave merge with floating-point criterion

If path tracking (see Section 4.3.7) is enabled and floating-point operations on a given platform are cheap, then another modification of CWave ($CWaveFpuMerge$) can be more preferable on certain types of maps. Indeed, when a wave-merge is detected at point $P$, but Proposition 4 does not hold, we can utilize path tracking to calculate exact floating-point distances $\rho^A$ and $\rho^B$:

$$\rho^A = s_A + \sqrt{(x_P - x_A)^2 + (y_P - y_A)^2}$$  \hspace{1cm} (4.91)

$$\rho^B = s_B + \sqrt{(x_P - x_B)^2 + (y_P - y_B)^2}$$  \hspace{1cm} (4.92)
Now, if $\rho^B \geq \rho^A$, then wave $B$ penetrated wave $A$ too much, and vertex $P$ value should not be reassigned.

Even though this criterion requires additional floating-point calculations (including two square roots), it eliminates the overlapping area, and thus, on certain types of maps where overlapping creates a significant overhead, it might be faster than CWaveFpuSrc or even CWaveInt. Additionally, by making an exact distance comparison (permitted by the machine epsilon) at all merge vertices CWaveFpuMerge is able to find optimal paths. The exact length of the optimal path can be calculated as discussed is Section 4.6.2.

### 4.5 Performance tests

The key performance characteristics of a path planning algorithm are accuracy and speed. They are discussed in the following sections 4.5.1 and 4.5.2, respectively.

#### 4.5.1 Verification and accuracy

The implementation of the CWave algorithm discussed above is not a trivial task, due to numerous special cases that are easy to overlook (vertices on 45°-diagonals, cones that are narrower than a single cell, NBPs at non-marked vertices, vertices with $y = 0$). To achieve a desired level of reliability, several verification methods were developed: Let us first introduce a definition.

**Definition 2.** If a vertex is surrounded by four occupied cells, we will call it a blocked vertex. Otherwise, it is an non-blocked vertex.

In the order of decreasing test speed and increasing test accuracy, the developed tests are:

1. *Fill test*, for a set of simply-connected maps, runs CWave from every non-blocked vertex on the map and verifies on the fly that, during each run, none of the vertices is visited more than once. At the end of each run, it checks that all non-blocked vertices on the map are assigned a distance value. This test is fast, but doesn’t check for distance errors.
2. **Shake test**, for a set of generic maps, indirectly verifies the distance accuracy by checking the triangle inequality. Indeed, for the lengths $\rho_{SA}$ and $\rho_{TA}$ (Fig. 4-18) of the two shortest paths calculated from adjacent vertices $S$ and $T$ to any vertex $A$, inequality $|\rho_{SA} - \rho_{TA}| \leq 1$ should hold true. Given the error bounds (4.87), this inequality implies

$$|\bar{\rho}_{SA} - \bar{\rho}_{TA}| < 5 \quad (4.93)$$

Shake test verifies that (4.93) is true for all non-blocked starting vertices $S$ on the map and all non-blocked $A$. This self-test is several times slower than **Fill test**, but it catches accumulative distance errors.

3. **Accuracy test**, checks that error bounds (4.87), are correct by comparing the CWaves distances to the those calculated by ANYA (D. D. Harabor and Grastien, 2013), (D. Harabor et al., 2016), an optimal any-angle path planning algorithm. Given that ANYA is designed for point-to-point path planning, a complete all-to-all distance check is very slow (for map in Fig. 4-19c it takes about 10h).

### 4.5.2 Speed comparison to other algorithms

In (Uras and Koenig, 2015), a comprehensive comparative analysis is presented for most popular any-angle path planning algorithms (Theta*, Lazy Theta*, Block A*, Field A*, ANYA, and Any-Angle Subgoal Graphs). In particular, their speed and accuracy when solving point-to-point path planning problems are compared. We expanded on their test framework: adapted Theta*, Lazy Theta*, Field A* for single-source problems (by stopping the searches only when all vertices are visited and setting heuristic to 0), and incorpo-
rated CWave algorithm for speed testing.

![Figure 4-19: Test maps: (a) map arena2 (resolution: 281x209, number of planning problems: 822), (b) map aklabs (resolution: 500x370, number of planning problems: 800), (c) map cwave (resolution: 150x100, number of planning problems: 2000). The blue curves are equidistant from the given source vertex as calculated by CWave. They visualize a solution of one path planning problem for each map.](image)

The framework measures the time required by each algorithm to solve a set of path planning problems on a given map. The tested maps are shown in Fig. 4-19a, 4-19b, and 4-19c, where the blue “curves” designate equidistant lines as calculated by CWave algorithm (distance between the curves is equal to 5 cell widths).

The tests were executed on Intel(R) Core(TM) i7-3610QM CPU @ 2.30GHz (20 runs for each map). The results (Fig. 4-20), demonstrate that on all maps, CWave performed faster than other algorithms. Remarkably, the highest performance advantage (10.5 times faster than Theta*) is achieved on map aklabs that was the initial target map for robot navigation. On the other hand, on map cwave Filed A* was just 43% slower than CWave.
Figure 4-20: Performance comparison of single-source path planning algorithms: measured in seconds (left), normalized by the average duration of CWave (right)

We believe that the main CWave performance advantage comes from representing the wave front using discrete arcs and lines (rather than individual vertices), but if the number cones grows too high (for example, in map cwave), the wavefront starts to lose its circular shape, and that could be the reason that our representation performs almost as fast as a set of individual vertices.

4.5.3 Parallelization

Most modern computers have several cores (CPUs) and some are even equipped with GPUs. A parallelized algorithm can distribute the computational load between multiple cores effectively decreasing the computational time.

One way to parallelize CWave is to distribute the wave sources available for expansion at a every step between multiple threads running simultaneously. The effectiveness of such parallelization is highly dependent on the map. Indeed, a map without obstacles will never have more than one source, and, thus, only one thread will be doing useful work.

An important aspect of this parallelization scheme is the distance equivalence between the threads. In our current implementation, we chose to synchronize threads at every distance step \((d, d + 2 \ldots d + 2k)\). This allows to keep frontier at the same distance and expand the wave at constant velocity. The practical implementation of such parallelization revealed a high degree of contention between the threads, rendering the usage of traditional thread synchronization means (semaphores, mutexes, and even spinlocks), ineffective: performance tests showed that all multithreaded implementations using these synchronization tools were slower than a single-threaded implementation. That can be explained by the fact...
that it is not only the set of sources that is shared between the threads, but also map vertices as well. A significant performance improvement, however, has been achieved by resorting to C++11 atomic variables and atomic CAS (compare and swap operation)-operation on map vertices (Williams, 2012).

The results of the performance tests comparing N-threaded implementation $CWaveN$ with the single-threaded implementation $CWave$ are presented on Fig. 4-21. $CWaveFpuSrc$ was utilized as the basis for the test.

![Performance graph](image)

Figure 4-21: Performance of 1- to 8-threaded implementations ($CWave_1$ to $CWave_8$) compared to the single threaded implementation ($CWave$): measured in seconds (top), normalized by the average duration of the single-threaded implementation of $CWave$ (bottom).

Among the expected results, we can observe that $CWave_1$ (a multithreaded implementation running a single thread) is slightly slower than $CWave$ (the single-threaded implementation), due to the overhead of atomic variables. When the number of threads reaches the number of simultaneous threads supported by the given CPU ($CWave_8$), the performance noticeably decreases. We can also observe that more intricate maps (resulting in more source vertices) show a better improvement from parallelization: in the best case on $cwave.map$ 8-threaded implementation $CWave_8$ is 2.4 times faster than the single-threaded implementation $CWave$, whereas the best multithreaded implementation $CWave_5$ on a topologically simpler map $aklabs.map$ is just 50% faster than the single-threaded implementation.

As we can see, this approach to parallelization, while noticeably improving the per-
formance, does not make it N-times faster (where N is the number of threads) than the single-threaded implementation. When the wave frontier is not required to stay at the same distance, a parallelized implementation with less frequent synchronization between the threads might yield a better performance increase.

From a practical point of view, for the tasks where distance maps need to be calculated from multiple sources, it might be more effective (time-wise) to run simultaneously N independent algorithms, one for each start source, rather than parallelize each run. This will almost completely eliminate the contention between the threads, but, of course, would require N-times more RAM.

### 4.5.4 Speed of CWave modifications

In this section, we compare the speed of the four modifications of CWave discussed so far: CWaveInt, CWaveFpuSrc, CWaveTrack, CWaveFpuMerge. The properties of these modifications are summarized in Table 4.2. Their speed has been measured using the same test setup as discussed in Section 4.5.2. The results (Figure 4-22) expectedly show that CWaveFpuSrc is just a little bit slower than CWaveInt, because the additional usage of floating point operations in CWaveFpuSrc is minimal. Predictably, maintaining data for path extraction, increases the runtime of CWaveTrack (as compared to CWaveFpuSrc) by 4-7%. What is interesting to observe is that on a more intricate map cwave, where more wave merges occur, CWaveFpuMerge is faster than CWaveFpuSrc. This supports our hypothesis that, on some maps, eliminating wave overlapping can increase the speed of the algorithm, despite the need to maintain additional tracking data and execute floating-point calculations at merge vertices.

![Figure 4-22: Performance comparison of various modifications of CWave: measured in seconds (left), normalized by the average duration of CWaveInt (right)](image-url)
<table>
<thead>
<tr>
<th>Modification</th>
<th>Description</th>
<th>FPU operations</th>
<th>Distance accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWaveInt</td>
<td>Pure integer implementation using only addition and bit shifting operations</td>
<td>None</td>
<td>Accumulative error (</td>
</tr>
<tr>
<td>CWaveFpuSrc</td>
<td>Calculates floating-point distance value at corner vertices</td>
<td>At corner vertices</td>
<td>Non-accumulative error (</td>
</tr>
<tr>
<td>CWaveTrack</td>
<td>Same as CWaveFpuSrc, but maintains additional data for path extraction</td>
<td>At corner vertices</td>
<td>Non-accumulative error (</td>
</tr>
<tr>
<td>CWaveFpuMerge</td>
<td>Same as CWaveTrack, but uses a floating-point criterion at wave merge vertices</td>
<td>At corner &amp; merge vertices</td>
<td>No distance error (optimal)</td>
</tr>
</tbody>
</table>

Table 4.2: Summary of CWave modifications.

4.6 Discussion

In this section, we will discuss certain advantages and drawbacks of CWave, as well as the potential directions for future work.

4.6.1 Implementation

The main drawback of CWave is its implementation complexity which resulted in about 1500 lines of C++ code. It required very rigorous and time consuming test process to debug. It is very likely that certain aspects of the algorithm/implementation can be generalized and simplified.

However, CWave has the main advantage of having superior performance over its predecessors. Given that CWave, in its most basic form CWaveInt, requires only integer addition and bit shifting (multiplication by two), it can be ported to low-cost embedded platforms that lack support for floating-point operations, for example, those used in swarm robotics.

The high-level pseudo-code of CWave is presented in Algorithm 1, where \( GetNextVx(src, cur_vx) \) iterates through vertices of a Bresenham arc as discussed in Section 4.2.1, \( ExtendBdry(boundary) \) iterates through vertices along boundary lines as discussed in Section 4.3.4, \( CheckOccupancyPattern(vx) \) checks for the occupancy of the cells corresponding to vertex \( vx \) and its NBP if it is present,
assigns distance values to the vertices in \( \text{dist}_\text{map} \) (Section 4.2.7), returns one of the patterns from Figure 4-12, \( \text{ProcessConfiguration}(\text{prev}_\text{pttrn}, \text{new}_\text{pttrn}) \) identifies one of the 40 configurations shown in Figure 4-14, and, if corners are detected, updates boundary lines (may create new cones), places new sources at the corners (adds them to \( \text{new}_\text{sources} \) array), \( \text{ProcessStartVertex}() \) and \( \text{ProcessEndVertex}() \) can result in the same actions as \( \text{ProcessConfiguration}() \).

**Input:** \( \text{map} \) (2D grid modeling the map), \( \text{init}_\text{vx} \) (initial source vertex)

**Output:** \( \text{dist}_\text{map} \) (2D array with distance value for every vertex)

**Initialization**

create source object from \( \text{init}_\text{vx} \), place it into array \( \text{sources} \)

\[
\text{front}_\text{dist} \leftarrow 2
\]

**while** \( \text{sources} \) is not empty **do**

**for** \( \text{src} \) in \( \text{sources} \) **do**

\[
\text{src}\.\text{dist} \leftarrow \text{src}\.\text{dist} + 2
\]

\[
\text{src}\.\text{radius} \leftarrow \text{src}\.\text{radius} + 1
\]

// Source maintains a set of cones

**for** \( \text{cone} \) in \( \text{src}\.\text{cones} \) **do**

\[
\text{start}_\text{vx} \leftarrow \text{ExtendBdry}(\text{cone}\.\text{start} \_ \text{boundary})
\]

\[
\text{end}_\text{vx} \leftarrow \text{ExtendBdry}(\text{cone}\.\text{end} \_ \text{boundary})
\]

\[
\text{prev}_\text{pttrn} \leftarrow \text{ProcessStartVertex}(\text{start}_\text{vx})
\]

// Iterate Bresenham arc vertices between start & end boundaries:

\[
\text{cur}_\text{vx} \leftarrow \text{start}_\text{vx}
\]

**while** \( \text{cur}_\text{vx} \neq \text{end}_\text{vx} \) **do**

\[
\text{cur}_\text{vx} \leftarrow \text{GetNextVx}(\text{src}, \text{cur}_\text{vx})
\]

\[
\text{pttrn} \leftarrow \text{CheckOccupancyPattern}(\text{cur}_\text{vx})
\]

\[
\text{ProcessConfiguration}(\text{prev}_\text{pttrn}, \text{pttrn})
\]

\[
\text{prev}_\text{pttrn} \leftarrow \text{pttrn}
\]

**end**

\[
\text{ProcessEndVertex}(\text{end}_\text{vx})
\]

**end**

merge \( \text{new}_\text{sources} \) into \( \text{sources} \)

from \( \text{sources} \) delete sources without cones

\[
\text{front}_\text{dist} \leftarrow \text{front}_\text{dist} + 2
\]

**end**

**Algorithm 1:** High-level CWave pseudo-code
4.6.2 Vertex exact floating-point distance calculation

Note that after the CWave integer distance values are calculated, the true floating-point distance can be extracted for any vertex $P$ if needed. First, $P$’s parent source needs to be identified (see section 4.3.7). The source exact floating-point distance to the start source is already known to be $s$. Then using relative coordinates of $P$ the exact floating-point distance $d$ from $P$ to parent source is calculated. Finally, $s + d$ gives the exact length of the shortest path from the start source vertex to $P$.

4.6.3 Reduction of thread contention in the multithreaded implementation

As it was mentioned above, the performance improvement of the current multithreaded implementation of CWave is pretty far from increasing proportionally to the number of threads. Certain changes to the design, however, are likely to make the multithreaded implementation more efficient.

For example, instead of distributing sources between the threads, individual cones can be distributed. Synchronizing threads less frequently would reduce thread contention, the key factor of performance degradation. Currently the threads are synchronized such as to keep the wave front at approximately the same distance, which is good to minimize the wave overlaps. If we allow threads to synchronize less frequently, some redundant occasional wave overlaps will be more likely, but at the same time there is a good chance that the reduced thread contention will result in an overall significant performance increase. These are still open directions for research.

4.6.4 Improving wave merges

In section 4.4, we showed that the current implementation of CWave can generate multiple waves that overlap (that is they visit the same vertices multiple times). Even duplicate source vertices with slightly different distances can be created. This drawback can potentially be mitigated by not allowing source duplicates. Indeed, if among all sources there
are two or more sources with the same coordinates, one of them will have its floating-point
distance $s$ not greater than that of all others. Only that source can thus be kept, and others
can be deleted.

An alternative solution to the problem of overlapping is to add a hyperbolic boundary to
the overlapping waves, but that might not be as easy to construct using simple arithmetics.
However, there is a certain chance that in many cases that boundary can be very well
approximated by a straight line.

4.6.5 Performance tests on various classes of maps

An important direction for further research is a comprehensive comparison of CWave per-
formance on different types of maps. Several classes of maps can be defined based on
their resolution, obstacle density, obstacle size and connectivity. Comparing CWave per-
formance on these types of maps versus the performance of other single-source any-angle
path planning algorithms can reveal other stronger and weaker sides of CWave.
Chapter 5

Shared position control with NoVeLTI

Having developed the fast path planning algorithm, in this chapter, we use it on the top layer of the robot control hierarchy (Section 1.5, Figure 5-1) to design a shared position control method. As opposed to shared steering control, position control is designed to allow the operator to specify his desired destination rather than to send steering commands. And navigation, in this case, is performed by the robot autonomously.

5.1 Background

Shared position control is somewhat underrepresented in research on mobile robots with LTI. One of the reasons for this might be that scholars often focus on improving the characteristics (accuracy, latency, bit rate) of the HMI itself, rather than designing a practical
navigation platform, and steering control, in this sense, may serve as a better demonstration. Moreover, reliable autonomous navigation in cluttered environments is still a not a solved problem, therefore combining a state-of-the-art LTI with a state-of-the-art navigation solution can be particularly challenging.

Very often position control is simply a choice of one out of multiple predefined destinations. When the number of predefined destinations is not very high, destinations can be mapped directly to the HMI outputs. For example, multiple SSVEP stimuli can be presented to the user, each corresponding to an individual destination, such as it was done in (Hooman Nezamfar et al., 2013).

When the number of destinations is higher than the number of options supported by HMI, the destinations are repeatedly grouped into subsets. Among recent works (R. Zhang et al., 2016) is an example of such approach. They implement both shared steering and position control for a wheelchair control with a BCI (motor imagery + P300). The latter is implemented for both Yes/No type of motor imagery interfaces, where one of the predefined destinations is selected using dichotomy (no probabilistic reasoning), and for P300, where the destinations IDs are arranged in a $10 \times 4$ matrix, and the selection is similar to P300 typing. Similar systems for destination selections are reported in (Rebsamen et al., 2010; Bi, X. A. Fan, et al., 2013; X. a. Fan et al., 2015).

More intelligent systems model user habits by assigning prior probabilities to each destination based on how often it is visited by the operator. This can allow for more efficient division into subsets that maximizes the amount of information received from human through the LTI. Overall, modeling user habits can improve the shared control experience (Perrin, Colas, et al., 2011).

At the border between steering and position control there’s a notable work (Demeester et al., 2007). While using steering commands at the user level, it aims at brining the wheelchair to one of the predefined destination poses (goals). The systems is designed to control a mobile robot with a continuous or discrete HMI and provides a shared control that is able to recognize the user plan based on the steering commands to assist with the navigation. The navigation and inference process was modeled with a partially observable Markov decision process (POMDP) in which user intended plan (trajectory) was
taken as the POMDP state, user commands (nine of them) were POMDP observations, and wheelchair (nine of them) actions were POMDP actions the POMDP state was a user plan (a set of wheelchair states). The system was demonstrated to be able to infer and execute complex maneuvers, such as parking next to a wall. One of the strong sides of that approach is an ability to model and account for user capabilities in using the HMI. The main problem with this approach is that it limits user navigation to a specific set of goals. The demonstrated results are shown only in relatively simple environment with the maximum of only 21 user plans (hypothesis) maintained. The solution may not scale when the number of destination is significantly higher.

A method for topological navigation is demonstrated in (Perrin, 2009), where the robot calculated the most probable action for wheelchair, and the operator would accept it or not using ErrP-based BCI. The environment was analyzed to determine places of interest (such as crossings). Overall, the systems with pre-defined lists of destinations limit the user in his freedom to move anywhere he wants.

One more reason for steering navigation being popular in navigation with LTIs is that it naturally parallelizes the inference and navigation processes, thus often making it faster to navigate to the desired goal pose, as compared to first choosing the destination and then autonomously reaching it. Such parallelization also gives the operator the feeling that he has a constant control over the moving robot.

Some shared position control methods, however, also provide such functionality. For example, in (Iturrate et al., 2009), a 3D model of the local environment is constructed based on the sensor data and presented two the user with a set of possible local goals placed in the vertices of a polar grid centered at the current robot pose (Figure 5-2). The operator can then select one of the goals with a P300-based BCI, and the wheelchair would autonomously drive to the selected pose. Additionally the interface allows to select 90-degree left and right turns to be able to choose local goals that are on the sides. A similar approach is developed in (Wei, W. Chen, and J. Wang, 2012; Wei, W. Chen, J. Wang, et al., 2013), but in addition to local goals placed at fixed position relative to the wheelchair, potential points of interest (such as tables) are identified based on 3D sensor (Kinect) data and added to possible goals (Figure 5-3). This enabled easy docking. Similarly, in (Goil, Derry, and
Argall, 2013) door ways are automatically are detected to be selected as destinations.

In this chapter, we look at the problem of navigation and overall control of dynamic systems with low throughput human-machine interfaces from a fresh perspective. As a result a novel method for shared position control is developed. Our solution has a number of advantages compared to the discussed position control systems. Namely, it:

- allows to navigate to any point (limited to the map resolution) in a known environment, not just predefined states, as opposed to all other methods known to the author;

- parallelizes the inference and navigation (as opposed destination selection);
• uses probabilistic inference and maximizes the information throughput;

• allows for mostly smooth navigation;

• has configurable policies for waypoint selection, for example to probabilistically minimize time to destination;

• accounts for map topology;

• allows to model user habits (popular areas, not just destinations, and overall is highly extensible)

• supports any number of discrete commands in an HMI

The method was named “NoVeLTI” as an acronym for “Navigation Via Low Throughput Interfaces”. The high-performance single-source path planning algorithm developed in Chapter 4 is a critical component of NoVeLTI.

The chapter is organized as follows. In section 5.2, we propose a definition of a control system with a LTI from the controls perspective, section 5.3 dissects such systems into individual blocks and introduces models for each of them. Based on these models, in section 5.4 we generate principles of optimal design of Shared Control and exemplify them by developing Shared Control for a massless holonomic robot navigating in a known 2D environment. Section 5.5, presents results of comparative simulation experiments. In section 5.6, the designed Shared Control is adopted for a robotic wheelchair, and results of real robot experiments are presented. Section 5.7 discusses the overall results and potential extensions of the NoVeLTI approach.

5.2 Definition of a control system with a low throughput human-machine interface (LTHMI)

Currently, in the research community, the adjective “low” in the term low throughput HMI is usually understood relative to other HMI information throughputs (Perrin, 2009). In this section, we propose a slightly different view.
From the controls or engineering perspective, the critical factor is not the information transfer rate (ITR) itself, but how it relates to the complexity of system/robot that is controlled. Indeed, imagine a very slow system with only two states, switching between which takes hours. In this case, even if a BCI with $ITR=5\, bit/min$ is used to control such system, the side affects associated with the latency and accuracy of it will be negligible. Alternatively, imagine a joystick interface that is used to control a humanoid robot with tens of DOFs that is expected to execute high-speed manipulations. We can say that only when the ITR of HMI becomes an impeding factor in the control system, then we can call it an LTHMI.

We can give a more formal definition.

**Definition 3.** Let $T_{task}$ be the average time of executing a task by a given robot/control system, $H_{task}$ be the amount of information needed to define the task. $T_{infer} = H_{task}/ITR_{HMI}$ will thus be the average time to pass the task definition into the control system. Now, this control system can be called a control system with a low throughput HMI (CSLTI) iff

$$T_{infer} \sim T_{task}$$

(5.1)

where $\sim$ means “is of the same order of magnitude as”. The task can be defined using state space terminology. $T_{task}$, in this case, is the average time of moving the system between two states in the state space $\mathcal{X}$ or the output space $\mathcal{Y}$. $H_{task}$ is the average amount of information to encode a state vector with the desired accuracy.

When $T_{infer} \ll T_{task}$, the peculiarities of HMI do not affect the performance, and it becomes a standard control problem. On the other hand, when $T_{infer} \gg T_{task}$, the dynamics of the system does not affect the performance, and we deal with information transfer optimization problem. And only when condition 5.1 is true, it makes sense to parallelize the inference process and system motion which opens door for a variety of optimization questions.
5.3 Problem Analysis

Figure 5-4 demonstrates a general model of human-robot interaction via an HMI. The three main blocks of this system are:

- **Human** and his **Intent** which is modulated by external world observations and internal human habits;

- actual **human-machine interface** which includes a **Input Device** and **Presentation Device**;

- **Shared Control** block which combines inputs and outputs of the robot with human input to generate certain propositions the user can later choose from.  

For a standard joystick-controlled wheelchair, **Input Device** is the joystick itself, **Shared Control** block is a motor controller translating the joystick position into the wheelchair velocities, **Presentation Device** is a static image on the joystick panel instructing the user how the stick positions correspond to the wheelchair velocities, on the joystick static assignment of the joystick position to the wheelchair velocities which the user has to learn. In a more sophisticated example of a BCI-controlled wheelchair, such as in (Perrin, 2009), **Presentation Device** modulates propositions to the users with visual, audio or tactile feedback, the user can then reply through an ErrP-based BCI which constitutes the **Input Device**, and the **Shared Control** block maintains a POMDP model generates propositions based on the available knowledge of the human behavior and robot sensor data, and sends a velocity

---

1 One may consider the **Shared Control** block to be part of HMI itself, but here we explicitly differentiate them to avoid confusion.
commands to the wheelchair. In this section, we will discuss our assumptions about the controlled system, human intent, and HMI.

More advanced Shared Control systems take into account certain models of the human intent, controlled object, and the human-machine interface. In the following sections, we discuss existing models and describe what model we adopted in here.

**Asymmetry of human interaction.** An important observation to make from Figure 5-4 is that the human command arrow is drawn much thinner than the modulated proposition arrow. This demonstrates the fact that the capacity of the channel transferring information from human into the system is much smaller than the capacity of the reverse direction channel. This asymmetry is especially evident in case of a brain-computer interface. The highest information transfer rates achieved by BCI systems are of the order of 2bits/sec (Nicolas-Alonso and Gomez-Gil, 2012b), whereas the amount of information that can be fed to the human through various senses is enormously larger. Indeed, (Koch et al., 2006) suggests that the theoretical maximum of the information rate between human eyes and the brain is about 9Mbit/sec which is 6-7 orders of magnitude larger than what the current state-of-the-art BCIs can offer. Even for a joystick interface, the human output capacity seems to be significantly smaller than the input capacity, again due to the high-throughput of various senses (vision, audio, tactile, etc) \(^2\). This asymmetry and how it can be utilized to improve HMIs is discussed more in the following sections.

### 5.3.1 Modeling the system

In general, a dynamic system can be modeled by:

\[
\dot{x} = f(x, u), \quad x \in X \subseteq \mathbb{R}^n, \quad u \in U \subseteq \mathbb{R}^m
\]  \hspace{1cm} (5.2)

where \(x\) is the *state vector*, \(u\) is the is the vector of control signals, \(x \in X\) and \(u \in U\) represent phase (state space) constraints and control signal constraints, respectively.

\(^2\)In fact, this asymmetry seems to be an innate characteristic of a human being, as opposed to, let’s say, a computer. The amount of information we can perceive is by several orders of magnitude larger than the amount of information we can produce.
Even though many concepts discussed in this chapter are applicable to the general system 5.2, the implementation presented in this work as a proof-of-concept for the proposed methodology assumes a simple massless holonomic circular robot navigating in a known 2D environment. Mathematically: \( x = [x, y] \), \( u = [V_x, V_y] \), \( f(x, u) = u \), control constraints are given by \( ||u|| = \sqrt{V_x^2 + V_y^2} \leq V_{\text{robot}} \), and phase constraints are dictated by the navigation environment map. The physical size of the circular robot can be taken into account by simply inflating obstacles on the map by the radius of the robot. We model the map with a uniform 2D grid where each cell can be free or occupied/impassable. This system is further referred to as POC-system.

To enable the methodology discussed below, we have to make another important assumption: we assume that, for the given system, an optimal control problem as defined below is solved.

For system (5.2), optimal control problems are usually formalized as follows: Devise a control signal \( u(t) \) such that it moves the system from its initial state \( x(t_0) = x^0 \) to a desired state \( x(t_1) = x_1 \) and minimizes a functional:

\[
J[x, u, x^0, x^1] = \int_{t_0}^{t_1} F(x, u)dt
\]  

For example, \( F \equiv 1 \) minimizes the time of moving from \( x^0 \) to \( x^1 \).

This well-known control theory problem is usually solved with classic methods of Lagrange multipliers, Pontryagin’s minimum principle or Bellman’s dynamic programming. A solution to the problem is an optimal control signal \( u^*(t) = u^*(t, x^0, x^1) \), and optimal trajectory \( x^*(t) = x^*(t, x^0, x^1) \).

Considering all permitted values for \( x^0 \) and \( x^1 \), one can further construct a complete cost function which will assign an optimal value of the functional to every pair \( (x^0, x^1) \):

\[
C(x^0, x^1) = J[u^*(t, x^0, x^1), x^*(t, x^0, x^1), x^0, x^1]
\]  

Clearly, for most real systems, finding \( C(x^0, x^1) \) is not a trivial problem, however, in some real scenarios an approximation of the function can be utilized.

For the POC-system, we assume \( F \equiv 1 \). In other words, we minimize the time to
destination or, equivalently, the length of the path. In this case, \( C(x^0, x^1) \equiv d_{obst}(A, B) \), where \( d_{obst}(A, B) \) is the length of the shortest path between vertices \( A \) and \( B \) referred to as \textit{obstacle-free distance} (as opposed to euclidean distance). The high-performance algorithm for single-source any-angle path planning developed in Chapter 4 is utilized to calculate \( d_{obst}(A, B) \) in real time.

### 5.3.2 Modeling human intent

With a certain level of generality, we take the following assumptions about the human intent: 1) At every moment the user has an intended state (destination) \( x_d \in X \subset \mathbb{R}^n \) in mind, 2) In general, \( x_d \) can change over time, 3) This state is unknown to the Shared Control (has to be inferred), 4) Given other constraints, human wants the intended state to be reached as soon as possible.

For the POC-example, we have additional assumptions: 1) On the given known map grid, only vertices are permitted as desired locations 2) The robot velocity at the desired location is 0.

### 5.3.3 Modeling human-machine interface

HMIs can be classified by \textbf{temporal properties} (Long et al., 2012; Bi, X.-a. Fan, and Liu, 2013b) into \textit{synchronous} (commands are selected at a certain constant frequency) and \textit{asynchronous} (commands can be initiated by operator at any moment). Roughly any asynchronous interface can be converted into synchronous by enforcing periodic inquiries to the user. The scanning interface (see Section 1.2) is a classic example of turning a normally asynchronous button interface into a synchronous interface. Similarly SSVEP-based BCI often operate with a constant period (for example, for gTec it’s about 1sec).

Any HMI has a limited \textbf{resolution}, allowing the user to choose one option (command/letter) out of a finite number \( r \) of possible options (commands/letters). By selecting a specific option the user conveys his intent to the machine. The set of all \( r \) permitted options is known as \textit{alphabet}. For a regular joystick interface, \( r \) is so large that the notion of alphabet become less practical, but it becomes more relevant for low throughput HMIs. Indeed,
SSVEP BCIs are usually designed for a selection of one pattern out of 2-4 stimuli displayed on the screen. Similarly, with scanning interfaces, only a limited number of options $r$ can be presented to user. The semantics of the commands is up to the designer of the *Shared Control*.

HMIs can also be divided into *deterministic* and *probabilistic* based on the accuracy of intent detection. For example, joystick interface in a healthy user hand or a single-switch interfaces can usually be considered deterministic (user commands are always detected correctly), whereas BCIs are naturally probabilistic (user-selected commands are detected correctly with a certain probability). In the latter case, the relation between the detected ($D$) and intended commands ($I$) can be described with a conditional probability matrix $P(D|I)$ (known as the *interface matrix*) of the following structure:

<table>
<thead>
<tr>
<th>Intended</th>
<th>Detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_1$</td>
<td>$P(D_1</td>
</tr>
<tr>
<td>$L_2$</td>
<td>$P(D_1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$L_r$</td>
<td>$P(D_1</td>
</tr>
</tbody>
</table>

where $L_1$ ... $L_r$ are permitted commands (letters of the alphabet), $P(D_i|I_j)$ shows what is the probability that commands $L_i$ is detected given that command $L_j$ was intended.

\[
\begin{array}{cccc}
D_1 & D_2 & \cdots & D_m \\
I_1 & 0.8 & 0.1 & \cdots & 0.01 \\
I_2 & 0.1 & 0.7 & \cdots & 0.04 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
I_m & 0.03 & 0.04 & \cdots & 0.85 \\
\end{array}
\]

(5.5)

where $I_j$ is the intended $j$-th command, and $D_i$ is the detected $i$-th command. Each row in the matrix sums up to 1.

The *interface matrix* can be obtained by repetitive experiments. In general, it may evolve over time due to the user fatigue, weariness of electrodes and other factors. Mathematically this dependency can be represented as $P(D|I,O)$ where $O$ represents other
We can observe that deterministic HMIs are a special case of probabilistic HMIs where \( P(D|I) \) is an identity matrix. Not all low throughput HMIs are probabilistic. Indeed, if a certain HMI is deterministic, but operates at low frequency, it is still a low throughput HMI. (Figure 5-5). One can observe, that resolution, accuracy and frequency are conflicting factors (higher value of \( r \) can lead to lower accuracy or frequency, and so on). In any given HMI, these parameters can be configured to maximize the information throughput.

In this work, we assume: 1) HMI operates at a known constant \( f_{HMI} = 1/T_{HMI} \) Hz. 2) The alphabet size is \( r \) 3) The interface matrix is known at every moment.

5.4 Synthesis

5.4.1 Intent estimate representation

Given the human intent model defined in Section 5.3.2, a natural representation of the estimate of the intent in continuous state space is a probability density function \( p(x) \) which allows to calculate for any region in the state space the probability that the user intended destination is located in that region. For POC-system, where only grid vertices are permitted to be destinations, the intent estimate is a probability distribution function (PDF) that assigns each vertex a probability value. The exact process of learning the PDF is discussed in Section 5.4.3. At this point we can assume that PDF is known and updated at the rate of \( f_{HMI} \).
5.4.2 Intermediate destination state selection

To parallelize the inference and robot motion, Shared Control unit can utilize the information which comes with every update of the intent PDF by moving the system into an intermediate destination state. Clearly, the next intermediate state must belong to the $T_{HMI}$-reachability area (the set of states that the robot from its current state can reach within $T_{HMI}$ sec). For the POC-system, the reachability area (RA) is the set of vertices $P$ for which $d_{obs}(C, P) \leq V_{robot} \cdot T_{HMI}$, where $C$ is the current robot position. It is calculated using CWave developed in Chapter 4.

The actual choice of the next intermediate state opens a wide range of possibilities. In the simplest case (no_move-policy), the robot should not move unless a small region in the PDF reaches a probability threshold ($x_d$ inferred). This, however, implies no parallelization of inference and motion, making the total control time $T_{control} = T_{infer} + T_{task}$. Another approach (opt-policy) proposed in this work is to move to the state that will probabilistically minimize $C(x_{cur}, x_d)$. Consider the following function:

$$\tilde{C}(x) = \int \cdots \int X C(x, \overline{x})p(\overline{x})d\overline{x}$$  

(5.6)

It measures the probabilistic cost of reaching the intended destination from a given state $x$, only based off the intent PDF $p(x)$. Thus, with opt-policy, the intermediate state can be found with:

$$x_{opt} = \arg\min_{x \in RA} (C(x_{cur}, x) + \tilde{C}(x))$$  

(5.7)

For the minimum time problem, the first term is $T_{HMI} = const$, thus can be omitted. For the POC-system, the probabilistic cost of moving from vertex A to the destination is

$$\tilde{C}(A) = \sum_{i=1}^{n} d_{obs}(A, B_i)p(B_i)$$  

(5.8)

And the time-optimal intermediate vertex is

$$P_{opt} = \arg\min_{A \in RA} \tilde{C}(A)$$  

(5.9)
Illustrative example for a simple discrete system

Consider a simple discrete state space with 3 states (see fig. 5-6)

\[ P_d(x) : \begin{array}{ccc} 1/3 & 1/3 & 1/3 \\ x^1 & x^2 & x^3 \end{array} \]

Figure 5-6: A simple discrete state space with 3 states

The average cost of choosing \( x^1 \) as the desired state is:

\[
\tilde{C}(x^1) = \frac{1}{3} \cdot 0 + \frac{1}{3} \cdot 1 + \frac{1}{3} \cdot (1 + 1) = 1
\]  \hspace{1cm} (5.10)

The average cost of choosing \( x^2 \) as the desired state is:

\[
\tilde{C}(x^2) = \frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 0 + \frac{1}{3} \cdot 1 = \frac{2}{3}
\]  \hspace{1cm} (5.11)

It’s easy to see that \( \tilde{C}(x^2) < \tilde{C}(x^1) \)

Finding optimal robot pose

Equation (5.7) defines an optimization problem where a) calculation of \( \tilde{C}(.) \) is slow, b) the gradient of \( \tilde{C}(.) \) cannot be easily calculated, c) the constraints are defined by the grid map, thus can be of any complexity. These properties render the direct use of brute force search and gradient methods impractical.

For the POC-system, we implemented the following algorithm of finding local minima.

Start with a certain vertex \( P_{\text{cur}} := P_{\text{start}} \), calculate \( \tilde{C}(.) \) in the visible neighbor vertices, and find the neighbor \( P_{\text{min}} \) with the lowest value of \( \tilde{C} \). If \( \tilde{C}(P_{\text{min}}) < \tilde{C}(P_{\text{cur}}) \), then move to \( P_{\text{min}} \): \( P_{\text{cur}} := P_{\text{min}} \), otherwise \( P_{\text{cur}} \) is the local minimum. The algorithm we developed in Chapter 4 was utilized for fast calculation of \( \tilde{C}(.) \)

It can be shown that, in general, the \( \tilde{C}(.) \) function can have multiple minima with different values. Indeed, consider a sample map shown in Figure 5-7 with a uniform probability distribution over all accessible vertices. Due to the symmetry, \( \tilde{C}(A) = \tilde{C}(D) \), \( \tilde{C}(B) = \tilde{C}(C) \), \( \tilde{C}(E) = \tilde{C}(F) = \tilde{C}(G) = \tilde{C}(H) \). A brute force verification shows that on this map \( A, B, C, D \) are all local minima and \( \tilde{C}(B) < \tilde{C}(A) \).
If the PDF is uniform over all accessible vertices, \( \tilde{C}(.) \) will have 4 minima on this map: \( A, B, C, D \), and \( \tilde{C}(B) < \tilde{C}(A) \).

This implies that the algorithm described above may not necessarily find the global minimum. At this stage, however, we will consider the local minimum satisfactory.

One can observe that when probability of a single vertex in the intent PDF is converging to 1, these vertices:

- \( P_{\text{maxprob}} \) vertex with maximum probability in intent PDF
- \( P_{\text{cog}} \) vertex closest to the PDF center of gravity (COG),
- \( P_{\text{near}_cog} \) accessible vertex that is closest to \( P_{\text{cog}} \) in terms of euclidean distance;

all converge to the intended destination vertex \( P_d \).

Given that, in addition to \textit{no\_move}, we implemented the following intermediate pose selection policies:

- \textit{cog2lopt}: finds local minimum in RA as described above, with \( P_{\text{start}} = P_{\text{near}_cog} \)
- \textit{maxprob\_obst}: vertex in RA closest to \( P_{\text{maxprob}} \) in terms of obstacle-free distance;
- \textit{nearcog\_obst}: vertex in RA closest to \( P_{\text{near}_cog} \) in terms of obstacle-free distance;
ra_maxprob: vertex in RA with maximum probability

In section 5.5, we compare the performance of these policies.

### 5.4.3 Intent estimate update (Inference)

There are various methods to learn an intent PDF. Here we propose to do that by simply asking the user what part of the state space (or the output space), his intended goal is in. In other words, we repetitively divide the space into \( r \) regions, where each region corresponds to an HMI command and let the user choose the region with his intended destination state.

We can use Bayesian inference to update the intent PDF. Let

\[
P(I_k^i|O^k) = \int_{X_k^i} p^k(x) dx
\]

be an a priori probability of the user choosing \( i \)-th region at \( k \)-th iteration given all external factors \( O^k \), and let \( P(I^k|O^k) \) be a vector of such probabilities. Here \( X_k^i \) is the \( i \)-th region at the \( k \)-th iteration. Now using an extended Bayes’s theorem we can write the intent update rule:

\[
P(I_{k+1}^i|O^k) = P(I_k^i|D_k^i, O^k) = \frac{P(D_k^i|I_k^i, O^k)P(I_k^i|O^k)}{P(D_k^i|O^k)} = \alpha P(D_k^i|I_k^i, O^k)P(I_k^i|O^k)
\]

(5.13)

where \( P(D_k^i|I_k^i, O^k) \) is the interface matrix.

Note that the initial value \( p^0(x) \) can be calculated based on the prediction model or can be initialized with a uniform distribution. Clearly, the shape and size of regions \( X_k^i \) affect the performance of the control system. We will call the process dividing the state space into regions state space segmentation.

### 5.4.4 State space segmentation

State space segmentation is the problem of formulating the “right questions” for the operator. It affects both the inference time and the robot behavior. Here, we will discuss some
of the considerations that can be taken into account when designing a state segmentation policy.

**Criterion I: Maximization of the entropy decrease** is the first optimization that we may consider. It will allow for the optimal use of the limited HMI throughput. Omitting $O^k$ symbol, the information entropy (Shannon entropy) at k-th step is:

$$h^k = - \sum_{i=1}^{m} P(I_i^k) \log_2(P(I_i^k))$$ (5.14)

The information entropy change at k-th step is then:

$$\Delta h^k = h^{k+1} - h^k = - \sum_{i=1}^{m} P(I_i^{k+1}) \log_2(P(I_i^{k+1})) + \sum_{i=1}^{m} P(I_i^k) \log_2(P(I_i^k))$$ (5.15)

Depending on the interface matrix and on how the $X$ is divided into areas, $\Delta h^k$ differs. To maximize the entropy decrease, the cumulative a priori probabilities of the regions should be as close to the *optimal a priori vector* as possible. The latter is dictated by the HMI matrix (for example, for a matrix with equal diagonal elements, the elements in the optimal a priori vector should be equal). We identify this as criterion I for state space segmentation.

Clearly, there are infinitely many ways to divide a continuous state space into $r$ regions such that this criterion is satisfied. In fact, it is possible to devise a segmentation method in a continuous space such, that despite maximizing the entropy decrease at every iteration and overall entropy tending to zero, the robot using the optimal state selection policy will not move. We hypothesize that the following criterion II can serve as an optimal strategy for state space segmentation.

**Criterion II: Maximization of the probabilistic cost decrease.** Consider the following example (Figure 5-8), here we compare two ways, namely I and II, to divide the same space into two regions. In each case, we know a priori probabilities of the user choosing options A and B: $P(^1A)$, $P(^1B)$ for method I, and $P(^{1I}A)$, $P(^{1I}B)$ for method II. Then we can also find the next optimal state in each case: $^1x^*_A$, $^1x^*_B$, $^1Ix^*_A$, $^1Ix^*_B$. And then we can calculate the the probabilistic cost in each state: $\tilde{C}(^1x^*_A)$, $\tilde{C}(^1x^*_B)$, $\tilde{C}(^{1I}x^*_A)$, $\tilde{C}(^{1I}x^*_B)$.
The probabilistic cost of segmentation I is, thus,

$$\tilde{C}_I = P(I A) \cdot \tilde{C}(I x_A^*) + P(I B) \cdot \tilde{C}(I x_B^*)$$  \hspace{1cm} (5.16)$$

while the probabilistic cost of segmentation II is

$$\tilde{C}_{II} = P(II A) \cdot \tilde{C}(II x_A^*) + P(II B) \cdot \tilde{C}(II x_B^*)$$  \hspace{1cm} (5.17)$$

If $$\tilde{C}_I < \tilde{C}_{II}$$, then the segmentation I results in a higher probabilistic cost decrease than segmentation II and is, thus, more preferable. Criterion II as hypothesized here does not yet provide a practical algorithm for state space segmentation. We keep it here for completeness. For the POC-system, where the state space segmentation translates into a grid map segmentation problem, we developed several segmentation methods and compared their performance in simulation.

### 5.4.5 Map segmentation

Map segmentation in case of a 2D grid is the process of assigning each accessible vertex an integer index between 1 and \( r \). Each index corresponds to an HMI command and can be represented with a color. At every iteration of the inference process, the colored map divided into regions can be presented to the operator. He or she can then select the HMI command.
command that corresponds to the region where his/her desired (intended) destination is located.

As it was discussed in section 5.4.3, the optimal way of dividing a map is yet to be discovered, certain requirements, however, are already known:

1. To maximize the information throughput, the cumulative a priori probabilities of the regions should be as close to the optimal a priori vector as possible (Section 5.4.3).

2. If human commands are detected correctly, the map segmentation process shall guarantee that a probability of a single vertex converges to 1.0

We developed a simple one-pass map segmentation Algorithm 2. By iterating through accessible grid map vertices in any given order, the algorithm can generate a segmented map that satisfies the above requirements. The order of iteration, defines the geometric shapes of the regions. Based on Algorithm 2, 4 one-pass map segmentation have been implemented:

1. Horizontal tile \((h_{tile})\) (Figure 5-9): vertices are iterated sequentially based on the scheme shown in Figure 5-9a. Unaccessible vertices are skipped.

2. Vertical tile \((v_{tile})\) (Figure 5-10): vertices are iterated sequentially based on the scheme shown in Figure 5-10a. Unaccessible vertices are skipped.

3. Equidistant segmentation \((e_{quidist})\) (Figure 5-11): the order of iteration is defined by the CWave algorithm developed in Chapter 4. The boundaries between regions in this case are close to curves equidistant from the center vertex \(C\).

4. Extremal segmentation \((e_{xtremal})\) (Figure 5-12): is in a sense perpendicular to the equidistant segmentation. The boundaries between the regions are externals of the central field with center at vertex \(C\). To achieve this segmentation, all vertices were divided into individual visibility cones using CWave algorithm (Chapter 4), then vertices in each cone were sorted using two-criteria sorting: first, based on the angle, second, based on the distance as shown in Figure 5-12a).
Input: \( \text{optimal}\_\text{apriori}\_\text{probs} \) (vector of optimal apriori probabilities), \( \text{pdf}[k] \) (intent PDF estimate array)

Output: \( \text{segmented}\_\text{map} \) (array of region indexes)

\[
\text{scaled}\_\text{probs} \leftarrow \text{optimal}\_\text{apriori}\_\text{probs}
\]

\[
\text{actual}\_\text{probs} \leftarrow \text{zerovector}
\]

\[
\text{cur}\_\text{region} \leftarrow 0
\]

\[
\text{prob} \leftarrow 0
\]

\[
k \leftarrow 0
\]

\[
\text{for } k < \text{length(\text{accessible}\_\text{vertices}) do}
\]

\[
\text{prob} \leftarrow \text{prob} + \text{pdf}[k]
\]

\[
\text{if } \text{prob} > \text{scaled}\_\text{probs}[\text{cur}\_\text{region}] \text{ then}
\]

\[
\text{actual}\_\text{probs}[\text{cur}\_\text{region}] := \text{prob} - \text{pdf}[k]
\]

\[
\text{updateScaledProbs()}
\]

\[
\text{prob} := \text{pdf}[k]
\]

\[
\text{cur}\_\text{region} := \text{cur}\_\text{region} + 1
\]

\[
\text{end}
\]

\[
\text{segmented}\_\text{map}[k] \leftarrow \text{cur}\_\text{region}
\]

\[
k \leftarrow k + 1
\]

\[
\text{end}
\]

Function \text{updateScaledProbs()}

\[
\text{sum}\_\text{optimal} \leftarrow 0.0
\]

\[
\text{sum}\_\text{actual} \leftarrow 0.0
\]

\[
i \leftarrow 0
\]

\[
\text{while } i \leq \text{cur}\_\text{region} \text{ do}
\]

\[
\text{sum}\_\text{optimal} \leftarrow \text{optimal}\_\text{apriori}\_\text{probs}[i]
\]

\[
\text{sum}\_\text{actual} \leftarrow \text{actual}\_\text{probs}[i]
\]

\[
i \leftarrow i + 1
\]

\[
\text{end}
\]

\[
\text{correction} := \frac{1.0 - \text{sum}\_\text{actual}}{1.0 - \text{sum}\_\text{optimal}}
\]

\[
i \leftarrow \text{cur}\_\text{region} + 1
\]

\[
\text{while } i < \text{num}\_\text{commands} \text{ do}
\]

\[
\text{scaled}\_\text{probs}[i] \leftarrow \text{correction} \ast \text{optimal}\_\text{apriori}\_\text{probs}[i]
\]

\[
i \leftarrow i + 1
\]

\[
\text{end}
\]

Algorithm 2: One-pass map segmentation
Figure 5-9: Horizontal tile map segmentation ($htile$): a) scheme demonstrating the order of vertices iteration b) real map example

Figure 5-10: Vertical tile map segmentation ($vtile$): a) scheme demonstrating the order of vertices iteration b) real map example
Figure 5-11: Equidistant map segmentation (equidist): a) scheme demonstrating the order of vertices iteration b) real map example

Figure 5-12: Extremal map segmentation (extremal). Vertices in each cone are sorted using two-criteria sorting. First, based on the angle, if the angle is the same, then the vertices are sorted based on their distance from the local source. The number in each vertex designates the index after sorting. a) scheme demonstrating the order of vertices iteration b) real map example
In addition to the one-pass segmentations we implemented two alternating segmentation methods: *altertile* (alternates *vtile* and *htile*), and *extredist* (alternates *extremal* and *equidist*). In the next section, we compare how different map segmentation policies affect navigation performance.

### 5.5 Simulation Experiments

To compare various methods of pose selection and map segmentation, for the *Shared Controlled* developed for the POC-system, we designed an automated test framework. For seamless integration with the real navigation system, the framework was implemented in Robotic Operating System (ROS).

#### 5.5.1 Automated test setup

The system consists of the following ROS nodes (Figure 5-13):

- *inference_unit, best_pose_finder* and *map_divider*, are the blocks that are discussed in section 5.4.

- *experimentator* publishes the navigation map, start and goal poses that were randomly generated in advance.

- *human_model* emulates accepts the map divided into four colored regions, finds what color of the goal vertex is and publishes a corresponding *intended command*. In this simulation, the region is always picked correctly.
• *lthmi_model* simulates an LTHMI by randomizing intended commands based on the configured interface matrix. We tested only symmetric matrices with equal diagonal elements, and evenly distributed error elements. *lthmi_model* also introduced a $T_{HMI}=1\text{sec}$ interface delay.

• *robot_model* is a simulation of a holonomic massless point-size robot. Whenever it receives a new desired position message, it starts to "move" towards the new goal with constant speed $V_{robot}=3m/s$, along the shortest path from its current pose to the goal, and publishes its current pose at the rate of $100Hz$.

The timing diagram of a sample navigation process is shown in Figure 5-14.

The intent PDF was initialized with uniform distribution. A navigation experiment starts at the moment when the goal is published. When the probability of a single vertex reaches 99%, the goal vertex is considered inferred. When the robot reaches an inferred goal, the experiment ends. We used the same map as shown in Figure 5-9b (resolution: $500 \times 370$, cell size $10cm$).

### 5.5.2 Simulation experiment results

Four parameters have been varied between the experiments: interface matrix, route (three random routes were defined), best pose selection policy, map segmentation policy. For each combination, 16 runs were executed. The experiments were run on ThinkPad W520 with Intel(R) Core(TM) i7-2760QM CPU @ 2.40GHz CPU and took several days to complete. All experiments were recorded. Figure 5-15 shows a single test run.

As it can be seen from the results of the simulation experiments (Figure 5-16), despite our attempts to speed up *cog2lopt*, the calculation time still noticeably affected the performance, that is why we show both the actual navigation time, and the navigation time with calculation time (spent on map segmentation and pose selection) is subtracted.

Among the expected results, we can observe that, for both HMIs, *no_move* and *ra_maxprob* resulted in the worse navigation time compared to other pose selection methods. For the deterministic HMI, *cog2lopt* expectedly resulted in the shortest navigation time (calculation excluded), even though the 20% difference from *nearcog_obst* is only seen for route 1.
Figure 5-14: Test framework time diagram
Figure 5-15: An example of the evolution of the intent probability distribution function (red/blue color) and corresponding simulated robot motion. The intended destination was inferred at 11s.
Figure 5-16: Results of the automated simulation tests
For the 70% HMI matrix, however, there is no visible difference in the performance. Somewhat surprisingly, the map segmentation policy does not seem to affect the navigation time in these experiments.

5.6 Robot Experiments

In this section we present an integration of the Shared Control unit developed for the POC-system with a robotic wheelchair and results of navigation experiments.

5.6.1 Experimental system setup

The semi-autonomous robot (Figure 5-17) is an upgraded version of the robot system discussed in Chapter 3. It is equipped with two Lidars (Hokuyo UTM-30LX under the seat looking backwards and Hokuyo URG-04LX in the footplate looking forward), wheel-on-wheel encoder modules, and infrared cliff sensors. Using Robotic Operating System (ROS) packages, the robot is capable of SLAM, localization (AMCL) and autonomous navigation (ROS navigation stack).

The wheelchair is a differentially driven platform with non-zero inertia, and, thus, the
Shared Control developed in section 5.4, will not be optimal. Nevertheless, the real robot experiments demonstrated a satisfactory performance. Circular robot footprint was chosen for two reasons: a) for easier integration with Shared Control (that was designed for a point-size robot) b) move_base is known to work better with circular robots.

The overall navigation system consists of the following ROS-nodes (Figure 5-18): Here inference_unit, best_pose_finder, map_divider, and lthmi_model are the same as in Section 5.5. amcl is a ROS localization package that constantly publishes robot pose. move_base is the standard ROS-package that implements autonomous navigation (accepts desired goal via action interface, and moves towards it along a path that is close to the shortest path while avoiding obstacles). To prevent unnecessary rotations, the goal orientation tolerance was set to $2\pi$. mediator ensures smooth goal preemption and eliminates issues caused by localization errors. rviz visualizer shows two maps to the user (Figure 5-19): Intent Estimate PDF with the complete map and a goal marker, and Colored Segmented Map. The operator has to identify the color of his intended destination (goal marker) on the segmented map, and press a corresponding key which is translated by the keyboard selector into intended command. The goal marker intentionally wasn’t displayed on the segmented map, because in real life it doesn’t exist. The user had to refer to the obstacle structure or the grid map instead. As it will be discussed in Section 5.7, that became one of the main challenges for the operator.

For each experiment the following data have been recorded: a video from the camera as shown on the bottom of Figure 5-19, a video recording of the laptop screen including the maps shown on the top of Figure 5-19, a ROS bag-file with all critical topics (messages) and parameters, a short note with the status of the experiment. These data with processing tools and several video collages are part of the dissertation contributions. They
Figure 5-19: Robot experiment snapshot: map with *Intent PDF*, robot navigation data, and goal marker (*top right*), zoomed map divided into colored regions as displayed to the operator (goal marker is intentionally not displayed) (*top left*), video camera snapshot (*bottom*).
### Table 5.1: Points of interest on the map are represented as a weighted sum of 2D Gaussians. Gaussian parameters are listed in this table.

<table>
<thead>
<tr>
<th>name</th>
<th>position</th>
<th>standard deviation</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>livroom1</td>
<td>(87, 200)</td>
<td>0.8</td>
<td>2.0</td>
</tr>
<tr>
<td>door1</td>
<td>(118, 19)</td>
<td>1.1</td>
<td>1.7</td>
</tr>
<tr>
<td>office1</td>
<td>(171, 155)</td>
<td>1.3</td>
<td>2.4</td>
</tr>
<tr>
<td>bathroom1</td>
<td>(34, 19)</td>
<td>0.9</td>
<td>2.1</td>
</tr>
<tr>
<td>music1</td>
<td>(138, 234)</td>
<td>1.0</td>
<td>1.9</td>
</tr>
<tr>
<td>bedroom1</td>
<td>(145, 194)</td>
<td>1.3</td>
<td>1.7</td>
</tr>
<tr>
<td>door2</td>
<td>(90, 34)</td>
<td>1.2</td>
<td>1.5</td>
</tr>
<tr>
<td>kitchen1</td>
<td>(84, 91)</td>
<td>1.2</td>
<td>2.1</td>
</tr>
<tr>
<td>storage1</td>
<td>(130, 97)</td>
<td>0.9</td>
<td>1.8</td>
</tr>
<tr>
<td>storage2</td>
<td>(153, 103)</td>
<td>0.9</td>
<td>1.6</td>
</tr>
<tr>
<td>storage3</td>
<td>(153, 87)</td>
<td>0.9</td>
<td>1.6</td>
</tr>
</tbody>
</table>


#### 5.6.2 Parameters of the experiments

A custom environment (27x17m) has been constructed for navigation tests (Figure 5-19). Unless explicitly specified otherwise, the following parameters have been constant in all experiments: shared control grid map cell size: 10cm, robot maximum linear and angular velocities: 0.5m/s and 0.6rad/s, $T_{HMI}=3$ sec, intermediate pose selection policy: $nearnocog_{obst}$ (because it’s fast to calculate and yields almost the same performance as $cog2lopt$), map segmentation policy: $nearnocog_{extremal}$ ($extremal$-policy with the center at $P_{near_{cog}}$). The robot velocity in the Shared Control was configured to be 30% higher than the real robot velocity. This prevented unwanted stops at every intermediate vertex. All experiments were conducted by the same operator.

In some of the experiments, assuming that there is no prior knowledge about the human intended destination, the initial intent PDF estimate is set to a uniform distribution. In other experiments, we model the prior predicted PDF estimate by setting initial PDF to a weighted sum of 2D Gaussians, where each Gaussian represents a Point of Interest (POI). Table 5.1 lists POI Gaussian parameters.
A major challenge for the operator was that when the whole map had been displayed, the individual vertices were barely distinguishable, making it hard for the operator to identify the color of the intended destination vertex. To mitigate this problem, we zoomed the segmented map view to display only those vertices whose probability was higher than a certain threshold. To prevent frequent changes in the view, we limited the view sizes to 16, 32, 64, 128, 256 vertices in length/width, and allowed only view positions that were multiple of the \((\text{viewsise})/2\).

### 5.6.3 Results of robot experiments

More than 250 point-to-point navigation experiments have been recorded to observe the influence of various parameters. The primary goal of these experiments was to demonstrate the feasibility of the proposed navigation framework, as well as to observe the influence of the key parameters on the system performance. An example of a single robot point-to-point navigation experiment (with PDF smoothening enabled) is shown in Figure 5-20.

**Effect of HMI accuracy without POIs**

Figure 5-21 demonstrates the effect of the accuracy of HMI when no POIs are defined. We can see that for longer routes (door1→livroom1, office1→bathroom1) (except 1 try), the inference via 94% HMI finishes before the robot arrives to the destination \((T_{\text{task}} > T_{\text{infer}})\). On the other hand, the inference via 70% appears to be 2-4 times slower, and results in \(T_{\text{task}} < T_{\text{infer}}\). A closer look at the distance-to-goal plots allows to see that in the 70% HMI case, when the robot almost reaches the goal, it keeps moving around it for 20–100 sec. Nevertheless, for longer routes, and especially for 94% HMI, the parallelization of the inference and motion makes the robot paths almost indistinguishable from the shortest paths, as if the inference time does not exist. The shorter route (music1→bedroom1), however, shows how the parallelization can result in an unnecessary movement of the robot around the destination, even more so for the 70% HMI.
Figure 5-20: An example of a single robot point-to-point navigation experiment. Here smoothening is enabled. The intended destination is inferred at 37s.
Effect of HMI accuracy when no POIs are defined

Figure 5-21: Effect of HMI accuracy when no POIs are defined
Effect of POIs

Figures 5-22 and 5-23 demonstrate the effect of defining POIs for 94% and 70% HMIs, respectively. As can be observed from the PDF entropy plots, when POIs are defined, the initial entropy of the intent PDF nearly halved (14.2\textit{bits} $\rightarrow$ 7.5\textit{bits}) compared to the case when no POIs are present. Note, that this is still significantly greater than the 3.5\textit{bit} entropy associated with selecting one location out of 11 locations. The entropy is higher due to the uncertainties around the POI locations which are dictated by the Gaussian standard deviation parameters. With the lower initial value of the entropy, the $T_{infer}$ also halved in almost all cases. For the 94% HMI, even on the shortest route music1$\rightarrow$bedroom $T_{infer}$ decreased almost to reach $T_{task}$.

Interestingly, the navigation time on routes door1$\rightarrow$livroom1, office1$\rightarrow$bathroom1 is longer in the POI-scenario. The explanation is that the concentration of POIs in the top right corner of the map happened to be higher, thus, at the start, the robot would tend to move towards that corner, and only after a couple of detected decisions, the navigation system would start moving into the right direction. From the operator’s perspective, navigation with predefined POIs felt much easier.

Effect of PDF smoothening  As it was mentioned above, in the beginning of the inference process, when the complete map is presented to the operator, the individual vertices on it are barely distinguishable. This results in a higher rate of incorrect region selection when the intended destination is close to the boundary between the regions. We attempted to model this additional “noise”, by smoothening the PDF around the boundaries: after every PDF update, the probability values in the vertices close to the boundaries, were set to the average probability value of their $r$-neighborhood, where $r$ corresponded to the view size, as shown in Table 5.2.

Figures 5-24 and 5-25 demonstrate the effect of PDF smoothening for 94% and 70% HMIs, respectively.

<table>
<thead>
<tr>
<th>view size, number of vertices</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$, number of vertices</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 5.2: Relation between view size and smoothening radius $r$
Effect of POIs, 94% HMI

Figure 5-22: Effect of POIs, 94% HMI
Effect of POIs, 70% HMI

Figure 5-23: Effect of POIs, 70% HMI
Figure 5-24: Effect of PDF smoothening, 94% HMI
Figure 5-25: Effect of PDF smoothening, 70% HMI
Smoothening removed abrupt changes from the PDF at the first steps of the inference process and subjectively made the first choices easier for the operator. The plots show that in most runs it increased the inference time, but improved the accuracy of the goal inference (note how the reddish curves in the distance-to-goal plot do not reach 0 at the end. This is a sign that the inferred goal is $10-20\,\text{cm}$ off the intended goal).

**Effect of inference_unit matrix being less accurate than the HMI matrix**

In another attempt to model the human error resulting from an incorrect region selection (not the HMI error), we conducted a set of experiments where the matrix used in the `inference_unit` was set to be less accurate than the matrix used in `lthmi_model`. More specifically, the `inference_unit` assumed a 70% HMI, whereas the `lthmi_model` used a 94% matrix. The additional uncertainty introduced for the `inference_unit` was to account for the region selection error. Figures 5-26 and 5-27 demonstrate the effect of such approach without and with POIs defined, respectively.

This method expectedly increases the inference time, however, no major improvements in the overall navigation have been observed.

**Effect of the goal marker**

This experiment was conducted to observe how much the human error (of incorrectly identifying the region his intended destination belongs to) affects the overall navigation performance. For this purpose, a goal marker was placed on the segmented map. This made it very easy to identify the color under the goal vertex, which essentially eliminated the corresponding human error. In a real navigation task no such goal marker exists.

Figures 5-28 and 5-29 demonstrate the effect of the goal marker for the navigation system with 94% and 70% HMIs, respectively.

As it can be observed from the entropy plots, in most of the cases, the goal marker significantly shortened the inference process. A closer look reveals that it happens because with the goal marker the entropy decrease is more monotonic. Indeed, without the goal marker the entropy tend to have more ups and downs which seem to be a sign that the operator tries to correct some of his precious choices. The operator also reported that with
Figure 5-26: Effect of *inference_unit* matrix being less accurate than HMI matrix, without POIs
Effect of inference_unit matrix being less accurate than HMI matrix, with POIs

Figure 5-27: Effect of inference_unit matrix being less accurate than HMI matrix, with POIs
Effect of the goal marker, 94% HMI

Figure 5-28: Effect of the goal marker, 94% HMI
Figure 5-29: Effect of the goal marker, 70% HMI
the goal marker inferring the exact goal vertex was much easier, and he felt more confident making his choices.

Navigation to non-POI destination

One of the advantages of the proposed navigation framework is that it allows to define a set of points of interest, but at the same time does not prohibit the operator from navigating to any other vertex on the map. A set of trials have been conducted to demonstrate this capability. POIs office1 and bedroom1 were removed from the list of POIs (Table 5.1), then two routes were tested: a longer route kitchen1→office1, and a shorter route office1→bedroom1. Each route was tested 4 times with the 94% HMI and with the 70% HMI. The results are presented in Figure 5-30.

In all cases the navigation was successful, even though the inference time was expectedly longer compared to the case where no POIs were defined. On the longer route, in 3 out of 4 tries, the pose was inferred before it was reached ($T_{infer} < T_{task}$).

Changing intended destination on the way

Another advantage of the proposed and implemented navigation framework is the ability to change the intended destination on the way. Here we will describe experiments that were conducted to demonstrate this capability on a longer and shorter routes. In this set of experiments we used an 85% HMI. For the longer route kitchen1→(livroom1)→storage1 (upper set of plots in Figure 5-31), the user started to navigate from kitchen1 to livroom1, and then, when exiting the kitchen doorway, "changed his mind" and proceeded to storage1 instead. This was repeated four times with and without POIs defined. In all cases, the operator was able to arrive to the final destination (storage1). On the entropy plot, the intent change can be seen as the entropy increasing between 20 and 40 seconds. We can also observe that for try #2 without POIs, at 14sec, the inference was just one step away from identifying livroom1 as the destination. In all tries without POIs the robot entered the living room, whereas in all tries with POIs the robot did not enter the living room.

For the shorter route storage1→(storage2)→storage3 (lower set of plots in Figure 5-31), the user started to navigate from storage1 to storage2, and then, after 5 selections,
Navigating to Non-POI vertices

Figure 5-30: Navigating to Non-POI vertices
Changing intended destination on the way

Figure 5-31: Changing intended destination on the way
"changed his mind" and proceeded to storage3 instead. In all cases except for try #3 with POIs, the user was able to reach storage3. In the try #3 with POIs case, given the lower initial entropy due to pre-defined POIs, and no incorrect detections, the storage2 pose was inferred before the user was able to convey his new destination to the machine.

**Comparing map segmentation and pose selection policies**

We conducted a set of experiments to observe how the map segmentation and pose selection policies affect the navigation performance of a real robot. Three map segmentation policies (*nearcog_extremal*, *altertile*, *extredist*), and three pose selection policies (*no_move*, *cog2lopt*, *nearcog_obst*) were tested. For each combination the experiment was repeated 2 times. The HMI was configured to use a 91% HMI. While this limited set of data may not be statistically representative, it still allows to observe some regularities.

Figures 5-32, 5-33, and 5-34 demonstrate a comparison of map segmentation policies. In these experiments *altertile* performed the worst. Navigation time with *altertile* was the worst in 9 cases out of 15, moreover in try #1 with *altertile* in Figure 5-33 the destination was inferred incorrectly, and, as a result, the robot arrived into a wrong location. The reason for that was that *altertile* created two separated interest areas (parts of the PDFs with high probability value), and the operator failed to track them both. It was unexpected for the operator when the inferred pose appeared to differ from the intended destination. This experiment was a practical example of how map segmentation policies that ignore the environment topology can negatively affect the navigation. The topology-aware segmentation policies *nearcog_extremal* and *extredist* did not exhibit such problems. From the given set of data, it is hard to conclude which of the two performed better.

Figures 5-35, 5-36, and 5-37 demonstrate a comparison of pose selection policies. Expectedly, on longer route experiments (livroom1→kitchen1 and storage3→livroom2), and on most of the shorter route experiments *no_move* policy resulted in the longest navigation time. Also, in these experiments, *nearcog_obst* seems to have performed somewhat better than *cog2lopt*. Indeed, out of all 30 comparisons, in 16 cases *nearcog_obst* resulted in a shorter navigation time than *cog2lopt*, in 8 cases the time is comparable, and only in 6 cases *cog2lopt* is faster. These observations, however, should be taken very cautiously, given the
Performance effect of MAP SEGMENTATION policy when pose selection policy is 'no_move'
Performance effect of MAP SEGMENTATION policy when pose selection policy is 'cog2lopt'.

Figure 5-33: Performance effect of MAP SEGMENTATION policy when pose selection policy is cog2lopt
Performance effect of MAP SEGMENTATION policy when pose selection policy is 'nearcog_obst'

Figure 5-34: Performance effect of MAP SEGMENTATION policy when pose selection policy is nearcog_obst
Performance effect of POSE SELECTION policy when division policy is 'altertile'

Figure 5-35: Performance effect of POSE SELECTION policy when division policy is *altertile*
Figure 5-36: Performance effect of POSE SELECTION policy when division policy is *extredist*
Performance effect of POSE SELECTION policy when division policy is 'nearcog_extremal'

Figure 5-37: Performance effect of POSE SELECTION policy when division policy is nearcog_extremal
5.7 Discussion

In this section, we will first discuss the advantages of the proposed and developed NoVeLTI (Navigation Via Low Throughput Interfaces) navigation framework, then its limitations and possible ways to address them, then potential extensions of the system will be proposed, and other directions for future work outlined.

Advantages of the NoVeLTI navigation framework are summarized as follows

- It does not restrict user navigation to a set of predefined destinations. It gives the operator the freedom to go to any vertex on the map, limited only by the map resolution. No other shared position control method for discrete LTIs is known to the author that can do that.

- At the same time it allows for easy integration with a priori knowledge of human intent that can be built based on user habits. Differently from others, the points of interest in our method do not have to be individual destinations, they can be 2D areas of any shape with variable probability. This is a much more accurate representation of the reality. What is important to emphasize is that the definition of POIs does not limit navigation to those POIs. It just makes it easier to navigate to those points. A priori intent PDF (predicted PDF) can also be dynamic, this is discussed below as a potential extension for the system.

- NoVeLTI parallelizes the inference and navigation processes to the extent desired by the operator, effectively reducing the total navigation time. The other positive effect of parallelization is that it gives the operator the feeling of constant control over the process. Which might be even more important for many users than the reduced control time.

- By choosing different pose selection policies the desired robot behavior can be configured as desired by a given user. As it was demonstrated in the robot experiments...
above, when intended destination is close to the current robot location, the time-optimal policy results in a lot of unnecessary movement. This behavior is, however, configurable. In addition to the simple use of no_move pose selection policy, an intermediate solution is possible. First, the operator has to be asked if his destination is far or close, and then, only if it is far we can use the time-optimal policy. Interestingly, this “question” can be asked using the existing functionality. Indeed, equidist map segmentation repeated one or several times will provide the machine with this information.

- The inference in NoVeLTI is probabilistic which allows not to waste any information available through the LTI.

- Our map segmentation approach maximizes the information throughput (entropy decrease) for an LTI with a known interface matrix, but at the same time it leaves the flexibility of the exact geometric shapes to the designer. It is known that maximizing information throughput (entropy decrease) may not necessarily optimize the system performance (Perrin, 2009). Indeed, depending on the structure of the problem some information can be more important than other. And sometimes Our map segmentation approach allows to decouple information throughput from information importance. The throughput is always maximized (accumulated probabilities of map regions are made optimal), but the importance is up to the designer: If the distance to the intended goal is more important, then equidist map segmentation policy can be used first. If the angular position of the goal is more important, then extremal map segmentation policy can be used first. By iterating map vertices in a different geometric order, other map segmentation policies can be designed. It is still an open research question, what map segmentation policy is optimal.

- NoVeLTI supports any number of discrete commands in LTI.

- As a consequence of the previous two features, similar to (Demeester et al., 2007), our system can be configured for any specific user. If the accuracy of detection of a certain command $k$ for a given user is known to be low, the optimal a priori
probabilities for the map segmentation unit just need to be adjusted for optimal ITR. In this case, the $k$-th map region will just be of smaller size.

- Developed map segmentation algorithms *equidist* and *extremal* account for map topology.

- Navigation with NoVeLTI is mostly smooth, no stops are necessary to make choice, but, again, this is configurable.

**The challenge of remote destination selection.** One of the main observations made during the experiments was that accurately identifying the colored region with the intended destination is not always easy and takes time. Admittedly, selecting a colored region of a complex shape is harder than, let’s say, choosing a steering command (forward, right, left, or stop). In addition to the HMI latency, there was a delay for rationalizing about the correct map region. That was the primary reason for increasing $T_{HMI}$ to 3 sec. Moreover the region selection became an additional source of errors. Based on a single user experience, however, practice made it easier.

There are two main factors hindering the remote destination selection: (1) lack of information about the remote area, (2) necessity to relate the information presented on the computer screen with the real surroundings.

Regarding the first factor, if someone is given a map of a building and asked to point to a specific destination where he wants to go with the $10\text{cm}$ accuracy, it may be challenging for him to do so. He may pick a certain vertex, but on the arrival he is likely to correct the wheelchair pose, because often it is only when the operator is in a close proximity to the desired area, that he can effectively rationalize about the destination with such accuracy. This is situation-dependent: sometimes on the arrival, the operator may notice some new, unexpected factors that he could not take into account when selecting the destination remotely. For example, he may find out that there is an open window and he may or may not want to stay close to it, or there could be a person in the room, and again the operator may or may not want to adjust his goal pose. The quality of the remote destination selection is highly dependent on the amount of context information that is provided to the operator.
about the remote locations. The latter also depends on the quality of visualization. Indeed if the map presented to the user reflects the details of the surrounding environment (may be in 3D), such as furniture, floor pattern, the presence of people, (acquired with the cameras, for example), he is likely to be able to choose the remote destination remotely with a better accuracy.

The second factor also plays an important role. Even on arrival to the destination area and having all necessary information about the surroundings, it is still challenging to relate the displayed colored regions on the screen with the actual physical environment. Augmented reality can be a great solution to this problem. Instead of displaying the colored regions on a 2D screen, when the operator is close to the destination area, they can be displayed as projected directly on the floor, making the user much more confident about his further corrective choices. A quick improvement would be to add some markers on the map that would be easy to relate to with any zoom factor.

The challenge of autonomous navigation. We had to modify ROS `move_base` package that was used for autonomous navigation in this system to achieve smooth navigation. The problem was that regularly, a waypoint identified with a pose selection policy would fall into a vertex that was detected as inaccessible on the local costmap. This can be caused by either: (1) an obstacle that is not present on the environment map (non-static obstacle), or (2) a localization error. When the goal pose is in an inaccessible area, the default policy in `dwa_local_planner` and `base_local_planner` is to drop the goal and stop the robot. But when the the robot is not moving the localization often would not make the necessary correction to the robot pose. In the initial tests, it sometimes resulted in regular stops that made driving experience rather unpleasant, other times it would freeze the robot in once pose, because the next pose would always fall into the same inaccessible vertex. To address the issue, we modified the planners to allow the robot to pursue goals even when it seems inaccessible. This solved the problem, but made it harder to navigate around non-static obstacles (obstacles not present on the static map). The robot would not collide with them, but would approach them pretty close, sometimes making it hard to move away. This problem needs further investigation. Overall a better local planning is necessary for a
reliable navigation for cluttered dynamic environments.

**More research on pose selection and map segmentation policies** is necessary. More intelligent policies and combinations of policies need to be developed to address the issues of remote destination selection, unnecessary motion when intended destination is close to the current pose. New policies for map segmentation can be designed. Luckily, NoVeLTI makes it easy to add new policies. As it was discussed in Section 5.4, it is still not clear what map segmentation policy is time-optimal. The problem can be addressed from two directions: mathematical analysis and automated simulation experiments on large sets of routes. Additionally, discrete map segmentation has another issue: it does not guarantee that each region will be simply-connected. When certain parts of a region form narrow areas, individual vertices may appear on the display as separated from the region. This can be addressed by either adopting a better method of visualization for the discretely segmented maps or by developing a method for continuous map segmentation.

**More statistical observations** can be made from the collected real robot experimental data. Indeed, several important characteristics, such as the average distance error of the inferred destination from the intended goal, the average time required for the human to select the region, the average number of user selection errors (excluding the errors artificially introduced by *ithmi_model*) can be extracted from the collected experimental data. By comparing these characteristics one may have a better understanding of which combination of pose selection and map segmentation policies is more efficient. However, as it was previously noted, to have a more objective knowledge of this, more experiments have to be conducted with more subjects operating the wheelchair.

**The speed of the calculations for the time-optimal pose selection policy** can potentially be increased. Two primary approaches in this direction are possible. (1) Calculate probabilistic distance for several vertices at the same using multithreading. (2) Precompute all mutual distances on the map. The number of accessible vertices on the map in the discussed robot experiments is \( \sim 17 \) thousands. If each distance value takes 4 bytes, the amount of memory needed to store information about the mutual distances between all
vertices is \(17 \times 17 \times 4/2 = 578\) MiB. On a modern computer, this can easily be loaded into RAM.

**Robot orientation** is not currently controlled with NoVeLTI. There are two aspects of this problem. One is the need for a full support for nonholonomic noncircular robots (discussed in the next paragraph), and the other is lack simple orientation control at the destination. The latter can be achieved relatively easy by using a pie diagram where circular sectors are mapped to LTI commands. Once the position \((x, y)\) of the robot is inferred, the pie diagram can be displayed to the operator to iteratively identify the desired orientation. In such implementation, it make sense to extend human intent estimate to a 3D array that would also include the orientation. That will speed up the inference of orientation.

**Extension to nonholonomic noncircular robots** is critical for practical adoption of the navigation framework by real users since most of the wheelchairs are both noncircular and nonholonomic (differentially driven). As it was demonstrated in the experiments, NoVeLTI can still work with nonholonomic robots, but is not optimal in this case and results in unnecessary rotations. A major research effort is needed to adopt the NoVeLTI approach for nonholonomic robots. First, the human intent estimate will have to be modeled with a 3D array to include the orientation. Second, probabilistic distance calculation will become more complicated and will require newer possibly approximate methods. Thirdly, effective map segmentation for such robots will be even more challenging. On the other hand, once the orientation is integrated into the intent estimate and environment map, and NoVeLTI is adapted for nonholonomic robots, modifying the system to support noncircular robots should be less problematic.

**Integration with real LTI**s is an important direction for future work. One interesting method of integrating NoVeLTI with an SSVEP-based BCI is as follows. Instead of using colors, the map regions can be the stimuli themselves. Each region may flicker with a unique different frequency (or modulated by another parameter), thus allowing for a natural method of selecting the destination. It is not clear how effective such stimulation model will be since the stimuli in such design are very close to each other, but it was already
demonstrated (D. Zhang et al., 2010) that even overlapping stimuli can be effectively used for BCI. Alternatively the map region colors can simply be mapped to external stimuli. For BCIs based on oddball paradigm (such as P300 and AEP), instead of using different colors the regions can be highlighted one after another. Whenever the region with the intended destination is displayed P300 signal should be triggered. The same method can be used for the single-switch interface. Spontaneous BCI (for example, those based on motor imagery) and other asynchronous LTIs, can be forced to work in the synchronous way as it was discussed in Section 5.1.

**Human intent prediction** is another promising research direction. Within the scope of this project, some preliminary results on modeling human behavior have been achieved in collaboration with Junqing Qiao (Qiao, 2016). Namely, manual wheelchair navigation data were collected by the author of this dissertation who used the wheelchair as the only way to move within WPI AtWater Kent Laboratories building for 5 days, the data were clustered to identify several points of interest (Figure 5-38) that were further modeled with Gaussian distributions. Then a Bayesian network model to predict likely user destinations based on time of the day and the time of previous events was proposed, but not fully implemented. This or a similar dynamic prediction model can significantly increase improve the performance of NoVeLTI navigation framework.

**User satisfaction** is yet another factor that needs further investigation. Only one subject has participated in the experiments described in this chapter, and even his feedback has not been formally recorded. This yields a very limited amount of user experience data for analysis. Nevertheless, sometimes, selecting the destination felt like fighting and other times it felt rather natural. With human-robot interaction, subjective human perception is one of the key aspects that has to be taken into account. A set multi-user experiments with formally defined user experience criteria, similar to those used in (Perrin, 2009), is necessary to further evaluate the proposed navigation framework.

**Extension to systems with manipulators** can make people suffering severe disabilities capable of doing most of activities of daily living (ADL). Consider the following scenario.
A quadriplegic user or an ALS patient is headed to the kitchen to make a cereal for breakfast. Among the huge number of objects in the kitchen that can be picked up, he should be able to choose the bowl, then cereal box, than spoon, and so on. While the scene is now 3D, the human perception with eyes is still 2D, the view can again be divided into several areas, and the operator can be asked iteratively to select one of the areas. On every update the robot can make an intelligent move, such as to approach a certain area first, than move the manipulator end effector closer to the place where several pick-up candidate objects are located. The parallelization of inference and motion in this case again reduces the total control time, and, maybe even more importantly, gives the operator the feeling of constant control and freedom to do anything that he wants. While certain actions are easier to execute because of higher prior probabilities, the operator is still not stuck with several preprogrammed options for breakfast, he can pick any object and place it anywhere. LTIs, such as BCI have already been used to select objects for picking (Bell et al., 2008), however, they do not yet implement such optimized strategy as described here.
Chapter 6

Conclusions

This dissertation was initially motivated by the problem of controlling a robotic wheelchair with a brain-computer interface. Such systems are designed for people with limited upper- and lower-body mobility and particularly people diagnosed with Locked-in Syndrome for many of whom a brain-computer interface is the only way to interact with the world.

We addressed this problem from the controls perspective. While the main contributions of this dissertation are the novel shared position control method developed in Chapter 5 and the fast path planning algorithm CWave (a critical component of the shared position control) developed in Chapter 4, the contributions made in this work cover all levels of the hierarchical navigation system: motion control, local planning, and global planning.

On the level of motion control, an adaptive controller with online parameter estimation has been developed for a differentially driven wheelchair. The distinguishing characteristics of the proposed controller are the use of motor currents as system inputs, and an arbitrary position of the wheelchair center of gravity. Mathematical and computer models of the closed-loop trajectory tracking system were developed to demonstrate simulation results with different initial conditions and wheelchair parameters.

On the level of local planning, a shared steering control method for electric wheelchairs was implemented and integrated with various low throughput human-machine interfaces, such as facial expression control with Emotiv EPOC, voice control with Google Glass, voice control with CMU Sphinx, and Brain-computer interface.

To enable fast probabilistic reasoning on the level of global navigation, a novel high-
performance algorithm for single-source any-angle path planning on 2D grids was developed. The problem was addressed from a fresh perspective, and the graph representation of the grid was abandon, while discrete geometric primitives were introduced to represent the propagating wave front. By utilizing efficient Bresenham algorithms, the single-source any-angle path planning can now be done using only integer addition and bit shifting operations. The mathematical analysis was developed to calculate distance error bounds, while experiments comparing the performance of the algorithm to alternatives demonstrated a significant advantage of the proposed solution. Several modifications of the algorithm, including an optimal version and a multithreaded implementation, were implemented and their performance is compared.

Finally, as the main result of this work a novel approach to shared position control in a known indoor environments using a low throughput human-machine interface was proposed, implemented and tested. A general theoretical formalism was developed for control systems with low throughput human-machine interfaces that are aimed at minimizing the control time. The formalism is used to design a proof-of-concept navigation system (POC-system) for a massless point-size holonomic robot in a known map. An automated test framework was designed to search for the optimal configuration of the POC-system. The POC-system was then adopted for a real differentially driven wheelchair, and more than 250 navigation experiments have been conducted to demonstrate the capabilities of the navigation system. The experimental data have also been collected and are one of the contributions of this work.

**Bibliography**


213


Jiang, Z.-P. and H. Nijmeijer. “Tracking Control of Mobile Robots: A Case Study in Backstepping”. Automatica 33.7. ISSN: 0005-1098. DOI: 10.1016/S0005-1098(97)


“Locked-in syndrome | Genetic and Rare Diseases Information Center (GARD) – an NCATS Program”. *Genetic and Rare Diseases Information Center (GARD) – an NCATS Program | Providing information about rare or genetic diseases*. (Cit. on p. 23). Web. 21/04/2017. <https://rarediseases.info.nih.gov/diseases/6919/index#ref_13009>.


Merletti, R., D. Farina, and Wiley-Blackwell Online Books. Surface electromyography: physiology, engineering and applications. 00004. Hoboken, New Jersey; Piscataway, NJ; IEEE Press, 2016. (Cit. on p. 30). Web. <http://wpi.summon.serialssolutions.com/2.0.0/link/0/eLvHCXMwfV09C8lwED38AHHzEz_ByU0xTZRmcFEUQcRBB7fSpPhdww204T4Wqvf3U5Mgon_zg-1su13vskT3Oj[j2x2GKDPvxpGtEHOJkxOissmh4rKJRI_LMRWJlIIXIO_meUIYMNnitQ2Niir8So0der9hX2rLHN6WmTUdegtVzs56sBDeVZtcbjDiEK4h3xwb0KKCiYAZoEFjgkiir1DgZM65CobUTNKHyY4AmdNO18V2WZ_y7Tda-1s9ebSgSIRaAqwoo/sdu9FcX-3KfMw>.


—-. “Neural control of cursor trajectory and click by a human with tetraplegia 1000 days after implant of an intracortical microelectrode array”. *Journal of Neural Engineering* 8.2. ISSN: 1741-2552. DOI: 10.1088/1741-2560/8/2/025027 (2011). 00000:


