Investigation of Trajectory Optimization for Multiple Car-Like Vehicles

Semester Thesis

2015

Author: Pascal Fabian Bosshard
Supervisor: Dr. Roland Philippsen
Examiner: Prof. Dr. Roland Siegwart

School of Information Science, Computer and Electrical Engineering
Halmstad University
PO Box 823, SE-301 18 Halmstad
Sweden
Contents

Abstract vii
Acknowledgments ix
Nomenclature xi

1 Introduction 1
  1.1 Problem Setting and Goal of the Project . . . . . . . . . . . . . . . . . . . . 1
  1.2 The Cargo-ANTs Project . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
  1.3 State-of-the-Art in Motion Planning and Trajectory Optimization . . . . 3
  1.4 Structure of the Report . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

2 Principle of CHOMP 5
  2.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
  2.2 Functional Principle of CHOMP . . . . . . . . . . . . . . . . . . . . . . . . 6

3 Objective Functional Designs and Modeling 9
  3.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
  3.2 The Objective Functional for a Planar Case . . . . . . . . . . . . . . . . . . 10
    3.2.1 The Smoothness Objective . . . . . . . . . . . . . . . . . . . . . . . . 10
    3.2.2 The Obstacle Objective . . . . . . . . . . . . . . . . . . . . . . . . . 11
  3.3 Objective Functional Designs . . . . . . . . . . . . . . . . . . . . . . . . . . 13
    3.3.1 The Interference Objective . . . . . . . . . . . . . . . . . . . . . . . . 13
    3.3.2 Curvature Constraints . . . . . . . . . . . . . . . . . . . . . . . . . . 16

4 Simulation Results 19
  4.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
  4.2 Software Setup . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
  4.3 Simulation Study . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
    4.3.1 Robot-to-Obstacle Performance . . . . . . . . . . . . . . . . . . . . . . 21
    4.3.2 Robot-to-Robot Performance . . . . . . . . . . . . . . . . . . . . . . . 21
    4.3.3 Curvature Insights . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31

5 Conclusion and Future Work 33
  5.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
  5.2 Evaluation of the Project . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
  5.3 Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

Appendix 35
  A Partial Derivatives for the 2D Curvature Representation . . . . . . . . . . 35
List of Figures

1.1 Cargo-ANTs project illustration ........................................... 2
3.1 Extended overall objective functional ................................. 9
3.2 Plots of used cost functions .............................................. 13
3.3 Two intersecting trajectories and their connected points .......... 15
3.4 Unit vector representation for the interference objective ........ 15
3.5 Osculating circle of a robot’s position. ............................... 16
4.1 The first and the final version of the GUI ......................... 20
4.2 Drawback using the unit vector approach for the interference objective ................................................................. 22
4.3 Interference objective comparison: Case 1 .......................... 25
4.4 Interference objective comparison: Case 2 ......................... 26
4.5 Interference objective comparison: Case 3 ......................... 27
4.6 Interference objective comparison: Case 4 ......................... 28
4.7 Interference objective comparison: Case 5 ......................... 29
4.8 Interference objective comparison: Case 6 ......................... 30
4.9 Illustration of the curvature .............................................. 32
A.1 Expansion of the curvature formula ............................... 36
A.2 Partial derivatives of the curvature ................................. 37
List of Tables

4.1 Parameter values .......................................................... 21
4.2 Interference objective comparison: obstacle, starting and ending points configurations ........................................... 23
The purpose of the project at hand is to investigate a novel path- and trajectory- adaptation technique called CHOMP [13]. It is hoped that CHOMP simultaneously optimizes the trajectory of multiple robots while obstacle avoidance of each robot should not be neglected. Furthermore, considering car-like steering constraints, the algorithm to be designed should take into account curvature to produce feasible paths.

In a first step, a formulation of CHOMP for a 2D representation is obtained. Based on this adaptation a graphical user interface and simulation tool is implemented. Additionally, two control strategies aiming at avoiding robot-robot collision are developed. The first of these control algorithms is based on the usage of unit vectors for determining the geometric direction whereas the second relies on the usage of the obstacle avoidance functional. As a last step, various methods for incorporating curvature are explored, to investigate whether it is possible to directly integrate steering constraints into CHOMP.

The performance of the adapted control algorithms is tested using simulation studies. It is found that the introduced methods for controlling the steering constraints are not able to optimize the trajectory in a satisfying way. However, CHOMP can be successfully extended for a multiple robot scenario. Both obtained control strategies are feasible with slight differences in robustness and computational effort.
Acknowledgments

My semester thesis at the Center for Applied Intelligent Systems Research (CAISR) provided me with a deeper insight into robot motion planning and a great stay at a foreign university. First of all I would like to thank my supervisor Dr. Roland Philippsen for enabling and supervising this project. His support and expertise in the field of robotics allowed me to overcome numerous practical and theoretical issues. Additionally, I am very thankful for his friendly initiation to the CAISR lab which turned my stay into an unforgettable time.

I especially want to thank the PhD students Jennifer David and Saeed Gholami Shabandi for the inspiring discussions and practical help. Their knowledge in programming were essential in order to implement the simulation program. I also want to thank the whole CAISR members for their open-hearted integration into their lab.

Finally, I would like to thank my tutor Prof. Dr. Roland Siegwart from ETH Zürich who made my stay in Sweden possible.

Zürich, January 2015

Pascal Bosshard
Nomenclature

Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>η</td>
<td>Regularization coefficient</td>
<td>[-]</td>
</tr>
<tr>
<td>κ</td>
<td>Curvature vector of a particular robot body point</td>
<td>[-]</td>
</tr>
<tr>
<td>λ</td>
<td>Smoothness objective weight factor</td>
<td>[-]</td>
</tr>
<tr>
<td>μ</td>
<td>Interference objective weight factor</td>
<td>[-]</td>
</tr>
<tr>
<td>ρ</td>
<td>Curvature objective weight factor</td>
<td>[-]</td>
</tr>
<tr>
<td>ξ</td>
<td>Trajectory function</td>
<td>[m]</td>
</tr>
<tr>
<td>c</td>
<td>Cost Function</td>
<td>[-]</td>
</tr>
<tr>
<td>d</td>
<td>Robot-obstacle or robot-robot distance</td>
<td>[-]</td>
</tr>
<tr>
<td>J</td>
<td>Kinematic Jacobian</td>
<td>[-]</td>
</tr>
<tr>
<td>n</td>
<td>Number of trajectory points</td>
<td>[-]</td>
</tr>
<tr>
<td>q</td>
<td>Robot configuration</td>
<td>[m]</td>
</tr>
<tr>
<td>u</td>
<td>Body point of the robot</td>
<td>[m]</td>
</tr>
<tr>
<td>x</td>
<td>Workspace point</td>
<td>[m]</td>
</tr>
<tr>
<td>U,F</td>
<td>Objective functional</td>
<td>[m]</td>
</tr>
</tbody>
</table>

Indices

<table>
<thead>
<tr>
<th>Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHOMP</td>
<td>Cost function from [13]</td>
</tr>
<tr>
<td>cubic</td>
<td>Cubic cost function</td>
</tr>
<tr>
<td>curv</td>
<td>Curvature</td>
</tr>
<tr>
<td>int</td>
<td>Interference</td>
</tr>
<tr>
<td>obs</td>
<td>Obstacle</td>
</tr>
<tr>
<td>smooth</td>
<td>Smoothness</td>
</tr>
<tr>
<td>t</td>
<td>discrete trajectory point</td>
</tr>
</tbody>
</table>

Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGV</td>
<td>Automated Guided Vehicle</td>
</tr>
<tr>
<td>ASL</td>
<td>Autonomous Systems Lab, ETH Zürich</td>
</tr>
<tr>
<td>CAISR</td>
<td>Centre for Applied Intelligent Systems Research, HH</td>
</tr>
<tr>
<td>CHOMP</td>
<td>Covariant Hamiltonian Optimization for Motion Planning</td>
</tr>
<tr>
<td>ETH</td>
<td>Eidgenössische Technische Hochschule</td>
</tr>
<tr>
<td>HH</td>
<td>Högskolan i Halmstad</td>
</tr>
<tr>
<td>RRT</td>
<td>Rapidly-exploring Random Tree</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Problem Setting and Goal of the Project

Nowadays, mobile robots are of great interest among different application sites like health care, transportation or the arms industry. They can on the one side support the user, on the other side complete complex tasks independently. As Siegwart et al. in [10] demonstrates, a fully autonomous robot system has to deal with aspects like perception, localization, map building, cognition, path planning and motion control. Path planning is an essential part of it, with regard to obstacle avoidance, kinematic constraints and trajectory optimality. Various techniques exist which try to overcome these challenges. Nevertheless, the difficulty of finding feasible solutions and the computational costs for multiple robots systems are still among the most challenging tasks.

The purpose of this project is to investigate a recent technique of trajectory optimization. In particular the CHOMP algorithm, presented by Zucker et al. [13], is used to design optimal trajectories for multiple robots. It is hoped that this trajectory optimization technique can be successfully used for the given overall objective, see therefore section 1.2. In a first step, the motion planning algorithm should lead to robust trajectories for a single robot in a 2D plane. After that, the algorithm should be extended to a multi-robot case. Beside considerations of robot-to-obstacle, also robot-to-robot distances are taken into account at this moment. A main request is that it is possible to integrate this robot-to-robot behaviour into CHOMP, retaining it therefore as a stand-alone algorithm. Furthermore, because of the nonholonomic constraints for two-steered-wheels robots and in some cases for differential drive robots as well, their turning radius is limited. Paying regard to the trajectory’s curvature, it is hoped that the algorithm simultaneously observes the radius limits. Hence the results would become optimal and smooth trajectories which are collision free to obstacles and robots. And even the multi-robot case is explored in the 2D case for this project, the resulting formulation would be able to support an arbitrary number of dimensions.

The semester thesis “Investigation of Trajectory Optimization for Multiple Car-Like Vehicles” is part of the Mechanical Engineering Master’s programme at Autonomous Systems Lab (ASL), ETH Zürich, Switzerland. It is carried out at Högskolan i Halmstad (HH) and embedded in the Mechatronics Group of the Centre for Applied Intelligent Systems Research (CAISR), HH, Sweden.

The next two sections in this chapter explain in more detail why this project is carried out. Section 1.2 gives an overview of the ongoing European project “Cargo-ANTs” and the tasks of HH therein. Section 1.3 offers a short explanation of different motion plan-
ning and trajectory optimization techniques within the robotics community. Furthermore, benefits and drawbacks of these state-of-the-art strategies are mentioned. Finally, an overview of this report is given in section 1.4.

1.2 The Cargo-ANTs Project

The intention of this semester project is to gain insights of possible applicability of the CHOMP algorithm. In particular this trajectory optimization technique is intended to be used for an ongoing European project called Cargo-ANTs [2], which stands for Cargo handling by Automated Next generation Transportation Systems for ports and terminals. It aims to create smart Automated Guided Vehicles (AGVs) and highly automated trucks that can operate in main ports and freight terminals (see Fig. 1.1). The objectives are an increased performance and throughput of freight transportation and automated shared work of AGVs including perception, positioning and motion planning systems while maintaining a high level of safety.

There are five partners involved, namely VOLVO Technology, Sweden, ICT Automatisering, Netherlands and TNO, Netherlands from the industry side; the Spanish National Research Council (CSIC), Spain and HH, Sweden from the academic side. CAISR is responsible for the task of motion planning which contains task planning, path planning and path adaption. The latter is responsible for executing the path computed by the path planner, while adapting it to the immediate situation encountered during execution. The aim is to use CHOMP as such a trajectory corrector therein. It could run on every vehicle independently or in a centralized way. What the specific inputs and outputs are is not decided yet and part of the ongoing work.

Figure 1.1: Cargo-ANTs works between different terminals with automated trucks (green) as well as within a specific terminal (orange) with AGVs. The figure is given by [1].
1.3 State-of-the-Art in Motion Planning and Trajectory Optimization

In the field of motion planning and trajectory optimization a lot of research has been done already. Several approaches exist and the relationship to this work is listed below. Since the project is based on the application of [13] by Zucker et al., the author of this report has not undertaken a rigorous literature review. For more precise information, please refer to e.g. [13] or the references listed below.

Sampling based motion planners have become well understood in the last years. Furthermore, it is recognised that randomization may not, by itself, account for their efficiency [6]. An often used method in the field of sampling based planners is Rapidly-exploring Random Trees (RRT) presented by LaValle [5]. It has been applied successfully to differential constraints and high-dimensional planning, but still RRT and its extensions lack solution optimality and deterministic completeness, as stated in [10]. A recent advance which overcomes this problem is RRT*, presented by Karaman and Frazzoli in [4]. They introduce a new algorithm which is asymptotically optimal and show that the computational complexity is within a constant factor of the RRT counterpart.

CHOMP does not lie on the approaches introduced above as it can be seen as a gradient descent method. Zucker et al. in [13] distinguishes itself from other methods that CHOMP assumes the availability of the gradient, instead of estimating the gradients using sampling. Prior work in the field of functional gradient was done by Quinlan [9]. The so-called elastic band method models the trajectory as a mass-spring system where motion planning is performed by scanning back and forth along the elastic while moving one point at the time. Similar to CHOMP, the internal forces try to minimize the distance between adjacent points which yields to a smooth trajectory. For example, Philippsen and Siegwart [8] show a successful implementation of elastic bands and other path planning and obstacle avoidance methods.

1.4 Structure of the Report

The remaining chapters of this report are structured as follows: chapter 2 provides a summary of the main points in the investigated paper. It explains the general functional principle and the advantages using this algorithm. Chapter 3 describes how the CHOMP objective functionals are adapted for the project’s representation. Furthermore, chapter 3 presents the new invented objective functional for multiple robots and an investigation of the trajectory’s curvature. In chapter 4 the software setup and the implemented GUI are presented. After that, chapter 4 contains the results of the derived overall objective functional and the curvature study. Finally, chapter 5 draws a conclusion and gives an outlook for future work in the field of motion planning and the CHOMP algorithm.
Chapter 2

Principle of CHOMP

2.1 Introduction

This chapter presents an overview of the CHOMP algorithm and conveys an understanding how this optimization technique works. The project’s achievements are based on the structure of this algorithm, therefore the comprehension of it is essential.

As given in the title of [13], CHOMP stands for Covariant Hamiltonian Optimization for Motion Planning. It is a trajectory optimization technique for motion planning in possible high-dimensional spaces. The main goal is to produce optimal motion, i.e. finding the shortest possible way from a start to a goal point without violating dynamical constraints. To do so the algorithm deals with objective functionals which capture the dynamic of the trajectory and avoidance of obstacles. A functional is a map from a vector space to its field of scalars, see also section 2.2. As mentioned before, CHOMP can be seen as a first order gradient descent method, searching therefore for local minima. The differences to already existing gradient descent methods are explained by Zucker et al. in [13]. At this point it should be highlighted that CHOMP is a trajectory optimizer rather than a path planning algorithm. This implies two differences. First, it already needs a solution in the beginning which connects the start point with the goal point. Second, CHOMP takes time into account. Therefore, CHOMP is capable of finding a valid trajectory even if it is started with an infeasible initial guess.

Zucker et al. states two central tenets as requirements for his trajectory optimization technique, which are displayed below:

- In [13] robot motion is stated as objective functionals. More precisely a smoothness term $U_{\text{smooth}}(\xi)$ which captures the dynamic of the trajectory, and an obstacle term $U_{\text{obs}}(\xi)$ which provides the robot avoiding obstacles are formalized. The first tenet on which CHOMP builds states that gradient information is often available and can be computed inexpensively. Zucker et al. [13] first generalize the smoothness functional in terms of a metric in the space of trajectories. Thereof they are able to include higher-order derivatives. Furthermore, by using the robot’s workspace instead of the configuration space for the obstacle functional’s cost field $c$, they are able to compute functional gradients efficiently for complex real-world tasks.

- The second tenet aims for trajectories where the optimization is unencumbered by the used parametrization. Invariance guarantees identical behaviour independent of the type of parametrization used. As explained in [13], a metric structure of the trajectory space enables to precisely define perturbations of the trajectory. Using a
functional and their gradient makes it covariant to reparametrization.

Built on these guidelines a variational method for optimization is used in [13]. As mentioned above, a functional $U[\xi]$ is a function of the trajectory function $\xi$. This function $\xi$ maps time $t$ to robot configuration $q$ for example. Finally the functional gradient $\nabla U[\xi]$ is the gradient of the functional $U$ with respect to the trajectory $\xi$.

The presented trajectory optimization technique will descend to a local minimum. To sample over a distribution of trajectories, [13] uses the Hamiltonian Monte Carlo algorithm. This method leverages gradient information to efficiently sample from a probability distribution $p(\xi)$. At this point it has to be said that the sampling of trajectories is not a part of the project’s tasks. Therefore it will not be further considered in this report. For more information please refer to chapter 5 in [13].

The following section is intended to give an overview of the functional principle of the CHOMP algorithm. This will help to understand the solutions of the project tasks later on.

### 2.2 Functional Principle of CHOMP

In order to obtain smooth and collision free trajectories, the objective functional measures two complementary aspects. These two terms Zucker et al. denotes in [13] as $F_{\text{smooth}}$ and $F_{\text{obs}}$, and define together the objective functional:

$$U[\xi] = F_{\text{obs}}[\xi] + \lambda F_{\text{smooth}}[\xi]$$  \hspace{1cm} (2.1)

where $\lambda$ denotes a weight factor. As Eq. (2.1) shows, the objective is simply the weighted sum of a smoothness and an obstacle objective. As stated above, $\xi$ maps time to robot configuration. The time is considered to range from 0 to 1 without loss of generality. In the following the two objectives are explained in more detail.

To encourage smooth trajectories, unnecessary motion should be eliminated. $F_{\text{smooth}}$ measures dynamical quantities such as velocity or acceleration across the trajectory. In [13], the smoothness objective is presented as follows:

$$F_{\text{smooth}}[\xi] = \frac{1}{2} \int_0^1 \left\| \frac{d}{dt} \xi(t) \right\|^2 dt.$$  \hspace{1cm} (2.2)

The term inside the integral is the squared velocity norm and could also be replaced by other formulations. In this thesis, Eq. (2.2) will be directly used for deriving the simplified representation and the implementation.

The other objective introduced in (2.1) is the obstacle objective $F_{\text{obs}}$. It penalizes parts of the robot that are too close to obstacles or already in collision. As a result the consequent trajectory is unencumbered by any obstacles. Zucker et al. defines $F_{\text{obs}}$ in [13] as an integral that gathers the cost encountered by each workspace body point on the robot across the trajectory:

$$F_{\text{obs}}[\xi] = \int_0^1 \int_B c \left( x(\xi(t), u) \right) \left\| \frac{d}{dt} x(\xi(t), u) \right\|^2 du dt.$$  \hspace{1cm} (2.3)

$B$ denotes the set of points on the exterior robot’s body. $c$ is the workspace cost function which penalizes close obstacles. $x(\xi(t), u)$ is defined as the forward kinematics, which maps a robot configuration $q \in Q$ and a particular body point $u \in B$ into the workspace.
2.2. Functional Principle of CHOMP

Through the multiplication of the cost function with the norm of the workspace velocity, Eq. (2.3) is transformed into an arc length parametrized line integral. This ensures that the obstacle objective is invariant to re-timing the trajectory.

As mentioned earlier, CHOMP uses a gradient method to optimize its trajectory. In this case the functional gradient \( \bar{\nabla}U \) is the perturbation \( \phi : [0,1] \rightarrow \mathcal{C} \subset \mathbb{R}^d \) that maximizes \( U[\xi + \epsilon \phi] \) as \( \epsilon \rightarrow 0 \), given in [13]. For a Euclidean norm and a differentiable objective of the form \( \mathcal{F}[\xi] = \int v(\xi(t)) dt \), the functional gradient is given as Quinlan developed in [9]:

\[
\bar{\nabla} \mathcal{F}[\xi] = \frac{\delta v}{\delta \xi} - \frac{d}{dt} \left( \frac{\delta v}{\delta \xi} \right).
\]  

\( \text{(2.4)} \)

Since the objective functional is the sum of the prior and the obstacle term, Eq. (2.1) can easily be written for the functional gradient as

\[
\bar{\nabla} U = \bar{\nabla} \mathcal{F}_{\text{obs}} + \lambda \bar{\nabla} \mathcal{F}_{\text{smooth}}.
\]  

\( \text{(2.5)} \)

Applying the definition of the functional gradient to the two objective terms (2.2) and (2.3), the result presented in [13] is:

\[
\bar{\nabla} \mathcal{F}_{\text{smooth}}[\xi](t) = -\frac{d^2}{dt^2} \xi(t),
\]  

\( \text{(2.6)} \)

\[
\bar{\nabla} \mathcal{F}_{\text{obs}}[\xi] = \int_B J^T \| \xi' \| \left( (I - \hat{x}' \hat{x}'^T) \nabla c - c \kappa \right) du,
\]  

\( \text{(2.7)} \)

where \( \kappa = \| \xi' \|^2 (I - \hat{x}' \hat{x}'^T) \hat{x}'' \).

\( \text{(2.8)} \)

\( J \) in Eq. (2.7) is the kinematic Jacobian at the particular body point. \( \xi' \) and \( \xi'' \) are the velocity and acceleration of a body point and \( \hat{x}' \) is the normalized velocity vector. \( \kappa \) denotes the curvature vector of a particular body point. Eq. (2.8) will be important again in section 3.3.2.

Given the gradient of \( l \), it is possible to carry out gradient descent. In fact also a parameterization of \( \mathcal{F}_{\text{smooth}} \) and \( \mathcal{F}_{\text{obs}} \) is needed which depends on the given problem to solve. This is done in section 3.2. Zucker et al. [13] defines an iterative update rule that starts from an initial trajectory \( \xi_0 \), for example a straight line between start and end point. With the actual trajectory \( \xi_i \), a refined trajectory \( \xi_{i+1} \) is then computed. The derivation of the update rule is done by solving the Lagrangian form of an optimization problem (see therefore [13]). The resulting update rule is

\[
\xi_{i+1} = \xi_i - \frac{1}{\eta} A^{-1} \bar{\nabla} U[\xi_i].
\]  

\( \text{(2.9)} \)

\( \eta \) is a regularization coefficient which specifies the trade-off between step size and minimizing \( l \). Applying a steepest descent algorithm, unfortunately, gets dependent on the trajectory’s representation. To get rid of this dependence on the parametrization the gradient \( \bar{\nabla} l \) is multiplied by the inverse matrix \( A \). For the used representation later on, \( A \) measures the total amount of acceleration in the trajectory. A detailed derivation for the planar case is given in section 3.2.1. In general, \( A \) is not related to the identically named matrix \( A \) in Eq. (3.9).

Given the parametrization of the trajectory and differentiable objective functionals, it is possible now to optimize the trajectory by using Eq. (2.9). At the end, a termination criteria may be used if the trajectory has to be fixed before execution. There are many
possibilities, but a straightforward one is to terminate when the magnitude of $\nabla u[\xi]$ falls below a predefined threshold. At this moment it is not sure if the encountered solution is optimal in a global sense. With a higher-level planner one may decide if the trajectory is kept or overruled, based on the quality and feasibility of the path.
Chapter 3

Objective Functional Designs and Modeling

3.1 Introduction

A key essence of a well-performing algorithm is the reasonable mathematical description of the physical system. One of the great advantages of CHOMP lies in its generality and relief of a particular parametrization. One of the goals of this project is to adapt the algorithm for a planar case. Reducing the problem to a two dimensional case allows to deploy several simplifications. These derivations are shown in this chapter. Moreover, [13] does not treat the question of how to use CHOMP in a multi-robot case. Also, constraints in curve radii of wheeled robots are disregarded. Another important goal is to solve these questions by designing additional objective functionals which treat these new tasks. Fig. 3.1 should depict the desired solution. While the original objectives introduced in the previous chapter are bordered by the smaller block, the new overall objective functional shall be formed by two additional new objectives.

In a first step, the desired model of the project is stated. Out of this the given smoothness and obstacle objective are adapted, discretized and necessary terms derived. In a third step, an interference objective which deals with robot-to-robot behaviour and allows to use the algorithm for a multi robot case is derived. The fourth and last step considers several explorations about the trajectory’s curvature. Different possible objective functionals are deduced to overcome such limitations. Even if the implementation results are not successful (see section 4.3.3), a lot of insights are gained.

The following sections describe the four steps outlined above in more detail.

\[ U[\xi] = \mathcal{J}_{\text{obs}}[\xi] \oplus \lambda \mathcal{F}_{\text{smooth}}[\xi] \oplus \mu \mathcal{J}_{\text{int}}[\xi] \oplus \rho \mathcal{F}_{\text{curv}}[\xi] \]

Figure 3.1: This illustrates the desired shape of an extended objective functional. The inner block shows the objective functional presented in [13]. Additionally to this, two new objectives are introduced: an interference objective \( \mathcal{J}_{\text{int}} \) and a curvature objective \( \mathcal{F}_{\text{curv}} \).
3.2 The Objective Functional for a Planar Case

As this project serves as a first investigation for the use of CHOMP in the Cargo-ANTs project, the representation can be relatively simple. A reasonable possibility is to consider a point robot which moves freely in a 2D plane. This simplifies various things like clearer equations, easier implementation of a trajectory representation and shorter runtime. Nevertheless, it is still a good approximation to the encountered situation in the Cargo-ANTs project.

Before the two objective terms $F_{\text{smooth}}$ and $F_{\text{obs}}$ are derived for the desired robot representation, a particular parametrization of the trajectory $\xi$ has to be chosen. For this case, the same like in [13] is used. A uniform discretization samples the trajectory function over equal time steps of length $\Delta t$:

$$\xi \approx (q_1, q_2, ..., q_n)^T$$  \hspace{1cm} (3.1)

Each robot configuration $q_i$ represents a point in the discretized trajectory and contains by itself an x- and y-coordinate, $\vec{q}_i = (x_i, y_i)$. As already done before, the vector arrow is eschewed for the sake of simplicity. Furthermore it is assumed that the starting and ending points are fixed, given as $q_0$ and $q_{n+1}$ respectively.

3.2.1 The Smoothness Objective

As initially explained in section 2.2, the smoothness objective seeks after an optimal path. First the waypoint parametrization introduced above is applied and turns (2.2) into a series of finite differences:

$$F_{\text{smooth}}[\xi] = \frac{1}{2(n+1)} \sum_{t=0}^{n} \left\| \frac{q_{t+1} - q_t}{\Delta t} \right\|^2.$$ \hspace{1cm} (3.2)

Eq. (3.2) and (3.3) are presented\(^1\) by Zucker et al. in [13] and can be rewritten with a finite differencing matrix $K$ and vector $e$ as

$$F_{\text{smooth}}[\xi] = \frac{1}{2} \left\| K\xi + e \right\|^2 = \frac{1}{2} \xi^T A \xi + \xi^T b + c$$ \hspace{1cm} (3.3)

with $A = K^T K$, $b = K^T e$ and $c = e^T e / 2$. Matrix $A$ can be seen as a measurement of the total amount of acceleration in the trajectory. The equation on the right hand side of (3.3) brings a big advantage along in contrast to (3.2). The computation of the smoothness gradient is straightforward:

$$\nabla F_{\text{smooth}}[\xi] = A\xi + b.$$ \hspace{1cm} (3.4)

If it is possible to state the matrix $A$ and vector $b$ for the given problem, the functional gradient of $F_{\text{smooth}}[\xi]$ is computed very fast. In the following the derivation of these terms are shown.

Assume $n$ discretized trajectory points with a starting point $q_0$ and ending point $q_{n+1}$. If the summation in (3.2) is split up in its individual parts one gets

$$F_{\text{smooth}}[\xi] = \frac{1}{2(n+1)} \left( \frac{\left\| q_1 - q_0 \right\|}{\Delta t}^2 + \frac{\left\| q_2 - q_1 \right\|}{\Delta t}^2 + \frac{\left\| q_3 - q_2 \right\|}{\Delta t}^2 + ... + \frac{\left\| q_{n+1} - q_n \right\|}{\Delta t}^2 \right).$$ \hspace{1cm} (3.5)

\(^1\)The actual presented equation in [13] differs slightly from (3.2). The lower bound of summation starts at $t = 1$ and the upper bound of summation ends at $t = n + 1$. Since this violates the definition of the ending point, the author of this report assumes a typing error in [13].
Using the Euclidean norm and a uniform step size $\Delta t$, (3.5) can be written as

$$F_{\text{smooth}}[\xi] = \frac{1}{2(n+1)\Delta t^2} \left( (q_1 - q_0)^2 + (q_2 - q_1)^2 + (q_3 - q_2)^2 + \ldots + (q_{n+1} - q_n)^2 \right)$$

(3.6)

For a better understanding the number of trajectory points is set on $n = 3$. Applied to (3.6) and expanding the binomial terms one obtains:

$$F_{\text{smooth}}[\xi]_{n=3} = \frac{1}{2(n+1)\Delta t^2} \left( q_0^2 + 2q_1^2 + 2q_2^2 + q_3^2 - 2q_0q_1 - 2q_1q_2 - 2q_2q_3 - 2q_3q_4 \right)$$

(3.7)

In this case, $\xi$ is now defined as $\xi = (q_1, q_2, q_3)^T$. Out of Eq. (3.7) the part inside the brackets can be written with a coefficient matrix in the manner of the desired Eq. (3.3):

$$F_{\text{smooth}}[\xi]_{n=3} = \frac{1}{2(n+1)\Delta t^2} \left[ \begin{array}{c} q_1 \\ q_2 \\ q_3 \end{array} \right] \left[ \begin{array}{ccc} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{array} \right] \left[ \begin{array}{c} q_1 \\ q_2 \\ q_3 \end{array} \right] + \left[ \begin{array}{c} -2q_0 \\ 0 \\ -2q_4 \end{array} \right] \frac{q_0^2 + q_4^2}{c}$$

(3.8)

Matrix $A$, vector $b$ and scalar $c$ correspond therefore exactly to the terms in (3.3). For the sake of clarity one important point should be mentioned. A robot configuration point $q_i$ is a 2x1 vector. Consequently every coefficient inside the matrix $A$ is a 2x2 identity matrix multiplied by its coefficient. For example, the first top left entry in $A$ of (3.8) is $2 \cdot I = \left[ \begin{array}{cc} 2 & 0 \\ 0 & 2 \end{array} \right]$. Finally, the result of (3.8) can be expanded for the general case with $n$ trajectory points. Also for $n$ points there are only dependencies to neighbouring points which leads to a band diagonal matrix:

$$A = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 2 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} -2q_0 \\ 0 \\ \vdots \\ 0 \\ -2q_{n+1} \end{bmatrix}, \quad c = q_0^2 + q_4^2$$

(3.9)

Vector $b$ only consists the starting and ending point, regardless how many trajectory points are considered. Scalar $c$ is also given in (3.9) but will not have any influence further on. For the optimization step, the gradient of the smoothness objective is used and therefore only $A$ and $b$ are needed.

### 3.2.2 The Obstacle Objective

To obtain collision free trajectories, the robot is penalized when near to obstacles. Therefore the obstacle objective and its gradient presented in [13] are used. Therefore, Eq. (2.3)
only has to be slightly modified to be used for the project’s desired manner. Since only a point robot is considered, the workspace body point \( x(\xi(t), u) \) simplifies to \( x(\xi(t)) \). In addition, the configuration space and the workspace are the same. This makes it possible to define \( x \equiv q \). Using the same discretized waypoint parametrization as for \( F_{\text{smooth}}[\xi] \), the gradient of the obstacle objective derived from (2.7) is

\[
\bar{\nabla} F_{\text{obs}}[\xi] = \sum_{t=0}^{u} \left( \frac{q_{t+1} - q_{t}}{\Delta t} \right) \left( (I - \partial \partial^T) \nabla c - c \kappa \right), \quad \text{where} \quad \partial = \frac{q_{t+1} - q_{t}}{\Delta t}.
\] (3.10)

The Jacobian maps the projection between configuration and geometrical space. In our representation this is the same and therefore the Jacobian is equal to the identity matrix, \( J \equiv I \). The curvature vector (2.8) can be written as:

\[
\kappa = \left( \frac{q_{t+1} - q_{t}}{\Delta t} \right)^2 \left( I - \partial \partial^T \right) : \nabla F_{\text{smooth}}.
\] (3.11)

In this case, the acceleration \( x'' \) is equal to the gradient of the smoothness objective. This makes sense because \( \bar{\nabla} F_{\text{smooth}} \) is defined in (2.6) as the second derivative of \( \xi \).

**Cost Function Variations**

For applying Eq. (3.10) a cost function \( c \) which penalizes the robot for being near obstacles is also needed. Given a trajectory through space, the cost is defined by the distance between the robot’s body and the obstacles. A trajectory \( \xi \) is collision-free if for every configuration \( q_i \in \xi \) the distances from the robot to any obstacles are greater than some threshold \( \epsilon > 0 \). Beside of that, Zucker et al. [13] determines a distance field where all distances from any point in the workspace to the surface of obstacles are mapped. For the implementation of the simplified 2D representation the obstacles are assumed to be circles with known radii. The cost function depends therefore on the Euclidean distances between every trajectory point and the center of the obstacles.

For the implementation two different cost functions are used. The first one is a cubic function where close obstacles lead to a fast increase in the cost function

\[
c_{\text{cubic}} = \begin{cases} 
\frac{kD}{3} \left(1 - \frac{d}{D}\right)^3 & \text{if} \quad 0 < d < D \\ 0 & \text{otherwise}
\end{cases}
\] (3.12)

where \( D \) is the distance at which an obstacle starts influencing a waypoint, and \( k \) is a gain parameter. \( d \) stands for the Euclidean norm between two points, i.e. \( d = \| \Delta d \| \).

The second cost function is an adapted version of the presented function in [13]. Instead of searching for the minima in the distance field, the Euclidean distance \( d \) is taken directly:

\[
c_{\text{CHOMP}}(d) = \begin{cases} 
-d + \frac{1}{2} \epsilon, & \text{if} \quad d < 0 \\
\frac{1}{2\epsilon} (d - \epsilon)^2, & \text{if} \quad 0 < d \leq \epsilon \\
0, & \text{otherwise}
\end{cases}
\] (3.13)

Between 0 and a user-defined threshold \( \epsilon \), \( c_{\text{CHOMP}} \) drops as a quadratic function to zero. For values \( d > \epsilon \), the cost remains zero. For negative distances the trajectory is penalized only with a linear increase of \( c \). Fig. 3.2 shows how the two cost functions differ for small distances.
3.3. Objective Functional Designs

The goal of this section is to design new functional objectives which treat additional requirements concerning the path planning. As seen in figure 3.1 the new objectives should work in the same manner as the original ones in CHOMP [13]. This implementation has a big advantage. The resulting overall objective functional \( U \) optimizes all desired aspects simultaneously. Thus the algorithm could be seen as a all-in-one optimizer. No cascaded control is necessary which makes the algorithm fast and compact.

Two different requirements are observed in the following. With the first one, a collision of two robots should be avoided. With the second one, turning constraints given in real robots should be taken into account.

3.3.1 The Interference Objective

Before an objective can be derived, the given representation has to be extended to two robots. In that case, one deals now with two independent trajectories and four starting and ending points in total. Nevertheless, CHOMP should not be divided into two separate optimizers. Therefore every additional robot and its trajectory is added to the
Chapter 3. Objective Functional Designs and Modeling

Existing trajectory:

\[ \xi = (q_1, q_2, \ldots, q_n, q_{n+3}, q_{n+4}, \ldots, q_{2n+2})^T. \]  

Eq. (3.14) is valid when two robots are considered which have the starting points \(q_0, q_{n+2}\) and the ending points \(q_{n+1}, q_{2n+3}\). Obviously the matrix \(A\), vector \(b\) and scalar \(c\) of (3.9) change when \(\xi\) gets bigger. For the assumption that the trajectories of both robots contain the same quantity of points, \(A\) and \(b\) can be stacked:

\[
A_{2R} = \begin{bmatrix} A_0 & 0 \\ 0 & A \end{bmatrix}, \quad b_{2R} = \begin{bmatrix} -2 \cdot q_0 \\ \vdots \\ -2 \cdot q_{n+1} \\ 0 \\ \vdots \\ 0 \\ -2 \cdot q_{2n+3} \end{bmatrix}, \quad c_{2R} = q_0^2 + q_{n+1}^2 + q_{n+2}^2 + q_{2n+3}^2.
\]

If more than two robots are treated, then the terms in (3.15) are enlarged in the same way. The change from one to two robots doesn’t change a lot in the implementation. The most adjustments have to be done in the adaption of the GUI to display all trajectory points correctly (see also section 4.2).

In this section a new interference objective \(\mathcal{F}_{int}\) is derived which keeps intersecting robots in a safe distance from each other. With the given parametrization, every robot configuration point \(q\) is also a specific timestamp of the robot’s motion. Closer points mean slower velocity, and further apart points imply fast movement. The advantage with this parametrization is, that only one point of each trajectory has to be compared to each other. Fig. 3.3 illustrates this well. The computational effort is therefore strikingly smaller.

Two different approaches are derived to tackle the given problem. The first one uses normalized vectors of the distance between two trajectory points. Consider a configuration point \(q_i\) from the first trajectory and the corresponding point \(q_{n+2+i}\) from a second trajectory. The distance between these two points is given as \(d_i = q_i - q_{n+2+i}\). This is illustrated in Fig. 3.4. The attached unit vector can be computed for every point with

\[
n_i = \frac{d_i}{\|d_i\|}.
\]

The point of the second robot possesses the same unit vector but with a negative sign. These vectors indicate the direction the objective should push the points into if they are too close to each other. Thus a cost function which controls how much the points are penalized is also needed. All together the interference objective is built:

\[
\mathcal{F}_{int}[\xi] = \sum_{i=0}^{n} [c(d_i) \cdot n_i].
\]
Taking the functional gradient of (3.17) one gets the desired objective:

$$\nabla F_{int}[\xi] = \sum_{t=0}^{N} J^T[\nabla c(d_t) \cdot n_t].$$  

(3.18)

The working principle of (3.18) is simple. The unit vector provides the geometrical direction in which the trajectory should be optimized. If the two robots are too close to each other, they have to be pushed away to a certain distance. This is done by the cost function which is a function of the distance. Even though (3.18) looks simple it shows nice results as seen in section 4.3.2. As a main advantage it converges faster than the approach presented in the following.

The second approach uses the obstacle objective (Eq. (3.10)) to avoid collision between robots. It looks evident to reuse this objective because it treats a similar problem. Robots can be seen as obstacles to each other which they mutually have to avoid. Therefore only
small changes in Eq. (3.10) have to be done:

$$\hat{\nabla}\mathcal{J}_{int}[\xi] = \sum_{t=0}^{n} J^T \left\| \frac{q_{t+1} - q_t}{\Delta t} \right\| \left[ (I - \hat{\theta}\hat{\theta}^T) \nabla c(d_t) - c(d_t)\kappa \right].$$

(3.19)

As in the first approach the cost function $c(d_t)$ is now a function of the distance between two corresponding robot points. $\hat{\theta}$ is still the normalized velocity and $\kappa$ is the curvature vector. The working principle of (3.19) is similar to (3.10). The normalized workspace velocity is multiplied by a difference of two terms. In the first one, $(I - \hat{\theta}\hat{\theta}^T)$ is a projection matrix that projects workspace gradients orthogonally to the trajectory’s direction of motion. As explained in [13], Zucker et al. also mentions that it ensures that the workspace gradient does not directly manipulate the trajectory’s speed profile. From this, the weighted curvature is subtracted. As the first term pushes the trajectory straight away from an obstacle or other robot, the second term corrects this if the curvature gets too big. The complexity of this approach is reflected in a slower convergence and more sensitive reaction of the trajectory when near to obstacles. But as a big advantage it is more robust to specific trajectory configurations, as seen in section 4.3.2. This is also the reason why another approach to (3.18) was derived.

### 3.3.2 Curvature Constraints

For front or back wheel driven robots, the turning radius is limited. Because of nonholonomic constraints they cannot drive any desired curve in instantaneous time. Especially thinking of the Cargo-ANTs project, these AGVs have a big turning radius. The goal is to derive a curvature objective $\mathcal{J}_{curv}$ which looks after trajectories with possible curve radii.

![Curvature Constraints Diagram](image)

Figure 3.5: A robot drives on a curve $C$. On every point the osculating circle (blue) approximates the curve near his position $P$. $r$ is the radius from his position to the instantaneous center of rotation (ICR). The green vector illustrates the robot’s current velocity. The figure is a modification of the original one in [11].

The best mathematical number to describe the bending of a curve is the curvature $\kappa$. The curvature measures the amount by which a curve deviates of being straight. Fig. 3.5 shows a robot with its osculating circle at point $P$. This circle matches the curvature at this point. If the reciprocal of $\kappa$ is taken one gets the radius $r$ of the circle, i.e. $\kappa = 1/r$. Therefore, smaller radii lead to higher curvature and thus sharper curves. For bigger radii it is vice versa.

For the derivation of a curvature objective $\mathcal{J}_{curv}$ and its gradient two different versions are established and tried out. First the curvature has to be parametrized and adapted
3.3. Objective Functional Designs

for the 2D planar case. For a plane curve with only x- and y-coordinates the curvature is given in [11] as
\[
\kappa = \frac{q' \times q''}{\|q'\|^3} = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{3/2}}.
\] (3.20)

The primes in (3.20) refer to the time derivatives, i.e. these are the velocity and acceleration in their specific direction. \(\kappa\) is here, in contrast to the formulation in Eq. (2.8), a scalar value.

For the first gradient objective the curvature is discretized in its velocity and acceleration components. The objective function itself is designed as a quadratic function of \(\kappa\):
\[
\mathcal{F}_{\text{curv}}[\xi] = \frac{1}{2(n+1)} \sum_{t=0}^{n} \kappa_t^2
\] (3.21)
\[
\nabla \mathcal{F}_{\text{curv}}[\xi] = \frac{1}{2(n+1)} \sum_{t=0}^{n} 2\kappa_t \cdot \nabla \kappa_t.
\] (3.22)

The gradient \(\nabla \kappa_t\) is derived as the partial derivatives for all involved trajectory points, as presented in appendix A. The gradient objective is finally added to the overall functional objective as shown in Fig. 3.1. There \(\rho\) is a weight factor similar to \(\lambda\) for the smoothness objective. The simulation results unfortunately did not show a desired behaviour and so another approach is tried out and shown below.

This time the derivative with respect to time is taken first. Thereby, it is intended to use the formulation of the curvature given in Eq. (2.8). After that, the function is discretized with the same parametrization used before. To penalize sharp curves, a cost function as used for the interference objective is added. Instead of the distance, the cost function is a function of the curvature’s norm.
\[
\mathcal{F}_{\text{curv}}[\xi] = \int_0^1 c(\|\kappa\|) \cdot \kappa \, dt
\] (3.23)
\[
\nabla \mathcal{F}_{\text{curv}}[\xi] = \frac{1}{2(n+1)} \sum_{t=0}^{n} \left[ \nabla c(\|\kappa_t\|) \cdot \kappa_t + c(\|\kappa_t\|) \cdot \kappa_t \right] \] (3.24)

Here \(\kappa\) is not a scalar but it is a vector as in Eq. (2.8). For the gradient functional also the derivative with respect to time for the curvature is needed. Using the product and quotient rule, the curvature gradient of Eq. (2.8) is:
\[
\kappa_t = \frac{(x'' \cdot x'^T + x' \cdot x'''T)}{\|x'\|^3} \cdot x'' + \frac{(I - \frac{1}{\|x'\|^2} x' \cdot x'^T)}{\|x'\|} \cdot x'''
\] (3.25)

Note that \(x\) is the workspace point and not the specific x-direction. \(x'\) is the velocity, \(x''\) the acceleration and \(x'''\) the jerk of the discrete trajectory point. They are derived via finite differences, similar to the velocity \(v\) in Eq. (3.10).

As in the case before, the simulation showed that the trajectory is not adjusted to the positions where the curvature gets too big. Additionally, the trajectory is very sensitive to deflections.

Also other studies like an approximation of the 1D case are tried out. Unfortunately none of the results lead to a satisfying behaviour of the trajectory. In section 4.3.3, the possible error sources and insights of the curvature investigation are explained in more detail. Other possibilities to overcome this problem are shown in this and section 5.3 as well.
Chapter 4

Simulation Results

4.1 Introduction

In order to better understand the way CHOMP works, a graphical representation is useful. While numbers are hard to transfer into a physical meaning, a graphical user interface (GUI) directly shows solutions or problems. This chapter shows the derived GUI for the stated project’s model and the simulation results tested on it. Section 4.2 explains how the graphical representation is built and how it can be used. As the standard CHOMP algorithm had been extended, the GUI was also customized to illustrate the new features. The progress in this is also shown. In section 4.3 results of the new functional objectives are discussed. This contains for one thing the obstacle objective with its different cost functions, for another thing the two robot-to-robot behaviour approaches. Finally, the takeaways from the curvature study are presented.

4.2 Software Setup

The GUI has three main tasks to solve. First, it should deliver a graphical answer of how the algorithm works. Second, the robustness of the derived approaches should be comparable. And at last, it should serve as a platform to experiment with different scenarios. At the beginning of this project a simple GUI was already available, programmed by the supervisor of this project. The GUI is coded in C++ and platform independent. A good reference which the author of this text also used to get familiar with the programme language is given by Arnold Willemer [12]. For the implementation, for one thing Eigen [3] is used, a C++ template library for linear algebra which makes it more user-friendly to deal with matrices and vectors. For another thing, GTK+ [7] is used, which is needed to create the graphical user interface.

Fig. 4.1 shows how the actual GUI looks like. On the left side, Fig. 4.1a presents how the parametrized CHOMP algorithm derived in section 3.2 looks like. In the bottom left corner the fixed starting point is marked with a red dot. In the top right corner the ending point is shown in green. In between, the grey circles mark the discretized trajectory positions. In the middle a light purple circle symbolizes a spherical obstacle. As for the straight and optimal way the trajectory would touch the obstacle, the algorithm optimizes it to the best possible solution.

A more complex situation is shown in Fig. 4.1b. There, also the interference objective from section 3.3 is applied. Two independent trajectories try to find their best way from
Chapter 4. Simulation Results

(a) The first version of the GUI

(b) The final GUI

Figure 4.1: This figures show the first and the final version of the implemented GUI. In 4.1a a trajectory of one robot is shown. On the right hand side, 4.1b depicts the extended CHOMP algorithm with two robots, avoiding obstacles and each other.

the starting points (red) to the ending points (green). The trajectory of the right starting robot may seem unlikely and not optimal at all. This comes from the fact that the obstacles, the starting and the ending points are moveable. The trajectories are thus pushed into the displayed positions. As mentioned earlier, no global path planner is considered in this project. Consequently the trajectories are not replanned and such S-curves are possible. At the bottom of the GUI different buttons are placed. With the "step" button the algorithm does one iteration and stops for the next input. By clicking "run" the program runs continuously and adapts the trajectory directly to displacements. With "jumble" the trajectory points are spread, with a certain maximal distance, randomly on the plane. With this button it is practical to test how fast and robust the trajectory finds a feasible path.

The GUI represents a 2D plane with an origin of ordinates in the middle. While in the beginning the starting and ending points were fixed, in the final version these points as well as multiple obstacles can be moved. This led to a lot of changes in the code, but gives more flexibility to testing and experimenting with real problem setups.

4.3 Simulation Study

In this section the results of the different cost functions, the interference objectives and the curvature studies are presented. Since the duration of the project was limited, only special cases were considered. This counts in a certain way for the parameter tuning as well. Tab. 4.1 lists the used values for different parameters. These values are found by trial and error, rather than by an automated tuning. The author is aware of the importance of an iterative search for the right parameter values. But the focus of this project lies more on the understanding of what good or bad parameter sets are, and not on their exact values. For the basic CHOMP algorithm weight factors, one sees that the smoothness objective and the obstacle objective are equally weighted. The obstacle objective has
4.3. Simulation Study

no parameter because it is taken as a reference, with a value equal to one. Interestingly, the interference objective weight factor $\mu$ has to be smaller than one, i.e. the other two objectives get more influence in terms of optimizing the trajectory. If $\mu$ is $\mu = 1$ too, the trajectories begin to oscillate if they are close to each other. For $\eta$ and $dt$ also other values can be taken without problem. The provided ones in Tab. 4.1 are a good trade-off in the required time of finding a local minimum and the robustness of the computed trajectory. For example, if $\eta = 10$ a path is computed in only a couple of steps. But the points do not find an equilibrium and they vibrate around the optimal path. In contrast, for $\eta = 500$ the final trajectory is reached very slowly.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>Smoothness objective weight factor</td>
<td>1.0</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Interference objective weight factor</td>
<td>0.40</td>
</tr>
<tr>
<td>$\eta$</td>
<td>Regularization coefficient for gradient descent</td>
<td>100.0</td>
</tr>
<tr>
<td>$n$</td>
<td>Number of waypoints per trajectory</td>
<td>20</td>
</tr>
<tr>
<td>$dt$</td>
<td>Time step size</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 4.1: Parameter values

4.3.1 Robot-to-Obstacle Performance

Considering the two functional objectives $F_{obs}$ and $F_{smooth}$ presented in [13], the focus lies on how the trajectory reacts to obstacles. For one robot CHOMP produces smooth trajectories within a few numbers of iterations. The simulations with the programmed GUI are very satisfying. Hence the derivations made in section 3.2 are correct and useful for the project’s purpose. The applied discretization is intuitive but provides still flexibility for changes or extensions.

A decision every user can make is the choice of the cost function for $F_{obs}$. In section 3.2.2 two possible obstacle cost functions are presented. Primarily both cost functions work impeccably for the given representation. Differences appear if the trajectory’s propagation gets close to obstacles. While $c_{CHOMP}$ already pushes the points away, $c_{cubic}$ doesn’t react at all. On the other hand, if the robot points are very close, $c_{cubic}$ suddenly pushes them away even harder. Therefore, this cost function is more aggressive. This leads to more oscillation of the trajectory, but on the other hand it may be helpful in a cluttered environment.

Besides the two cost functions from above also other weighting policies are possible. For example in a higher dimensional case, a more problem-specific cost function could be necessary.

4.3.2 Robot-to-Robot Performance

In a multiple robot situation, every robot seeks for his own optimal trajectory. To handle upcoming collisions, an interference objective $F_{int}$ was derived earlier in section 3.3.1. There, two different approaches are presented without advising which one suits the given representation better. At this point, it is firstly emphasized that both approaches are working nicely and either of them has its advantages and drawbacks. Nevertheless, in some cases one is stretched to its limits. An impression therefore is given in Fig. 4.2.
Chapter 4. Simulation Results

(a) Unit vector approach: both robots drive the exact path with the same speed. They would collide on the vertical symmetry line below the obstacle.

(b) Obstacle function approach: the robot starting on the left goes a little slower and more downwards than the robot which starts from the right. They thus pass each other slightly to the left of the symmetry line.

Figure 4.2: Both figures show an identical scenario, where an obstacle is placed on the symmetrical line. While the obtained solution is not feasible with the unit vector approach (4.2a), the trajectory is more sophisticated with the obstacle function approach (4.2b).

There, two robots have to circle an obstacle first and head to the ending points afterwards. While both approaches find a solution, the unit vector one in Fig. 4.2a is not possible. The robots would collide in the middle directly under the obstacle. The reason why the algorithm fails is because the unit vectors are orthogonal to the obstacle functional gradient vector. With the obstacle function approach shown in Fig. 4.2b this does not happen since the gradient objective vectors stand with an angle to each other. This leads to an offset which can be enlarged by a bigger robot-to-robot distance threshold.

In the following, different scenarios are investigated. They should represent common possible scenarios as well as special configurations. Applying them two things can be investigated. For one thing, the GUI displays the shape of the trajectories and how feasible they are. For another thing, the required computational effort is compared to each other. For the second evaluation point the first 1000 iterations are examined. As a benchmark the functional gradient $\nabla U(\xi)$ is used with a predefined threshold of $\nabla U(\xi) = 0.1$. For this value the trajectory is well formed while keeping the computational time short. In Tab. 4.2 the configuration parameters of the different cases are listed. Note that for all cases initially all waypoints lie at the starting position. CHOMP thus needs to pull them out towards the goal point. This is done in this way because it simulates the best a robot advances in a real environment. A robot only reacts to obstacles when they are detected by the distance sensors. In [13], it is started with a straight-line initial guess.

In the first case in Fig. 4.3 a normal situation is simulated. Both robots drive in different directions and have a different length of trajectory. When their path crosses each other they are at different positions, highlighted with the coloured dots of each robot’s position. Positions with the same colour denote the same moment in time. As seen in the end
4.3. Simulation Study

Table 4.2: Configuration parameters for starting points $q_{s1}$, $q_{s2}$ of the first and second robot, ending points $q_{e1}$, $q_{e2}$ and the obstacle position. The units for the given x- and y-coordinates are in pixels and dynamically scaled to the size of the GUI window.

<table>
<thead>
<tr>
<th>Case</th>
<th>$q_{s1}$</th>
<th>$q_{e1}$</th>
<th>$q_{s2}$</th>
<th>$q_{e2}$</th>
<th>Obstacle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-4.0, -7.0</td>
<td>3.0, 8.0</td>
<td>3.0, -3.0</td>
<td>-5.0, 2.0</td>
<td>1.0, 1.0</td>
</tr>
<tr>
<td>2</td>
<td>-5.0, -5.0</td>
<td>7.0, 7.0</td>
<td>5.0, -5.0</td>
<td>-7.0, 7.0</td>
<td>1.0, 3.0</td>
</tr>
<tr>
<td>3</td>
<td>-5.0, -5.0</td>
<td>7.0, 7.0</td>
<td>-5.0, -8.0</td>
<td>7.0, 4.0</td>
<td>3.0, -1.0</td>
</tr>
<tr>
<td>4</td>
<td>0.0, -5.0</td>
<td>0.0, 5.0</td>
<td>-5.0, 0.0</td>
<td>5.0, 0.0</td>
<td>1.0, 1.0</td>
</tr>
<tr>
<td>5</td>
<td>5.0, 0.0</td>
<td>-5.0, 0.0</td>
<td>-5.0, 0.0</td>
<td>5.0, 0.0</td>
<td>0.0, 0.0</td>
</tr>
<tr>
<td>6</td>
<td>5.0, 0.0</td>
<td>-5.0, 0.0</td>
<td>-5.0, 0.0</td>
<td>5.0, 0.0</td>
<td>0.0, -0.50</td>
</tr>
</tbody>
</table>

position Fig. 4.3b, the robots do not disturb each other. Thus the curves of the gradient descent are identical.

Case 2 (Fig. 4.4) shows symmetric starting and ending points and an obstacle shifted slightly to the right. The right starting robot reaches the obstacle first and is pushed downward. Because the left robot has to pass right after, the right robot is strongly pushed back, seen in Fig. 4.4b. This leads to strong corrections in the trajectory optimization and to the peak in Fig. 4.4d. In the end, for both approaches results the same trajectory. Fig. 4.4c shows nicely how the right robot drives more slowly in the beginning to pass the other robot in accordance with the predefined minimal distance.

Fig. 4.5 points out that the solutions for both approaches might be different. In this case the inlying robot needs to avoid the obstacle first and because of the shift of his trajectory, the outer robot has to avoid the other one. Fig. 4.5b and 4.5c show nicely how this can be done in two ways. In Fig. 4.5b the outer robot drives faster in the beginning to go past before the other robot. As an advantage both robots drive their optimal path. On the other hand for the obstacle function approach (Fig. 4.5c) the total energy needed is less because there is no acceleration or deceleration along the trajectory. In return the outer robot drives a longer route. One remark to the obstacle functional gradient curve. One notices that the curve does not fall to zero after $\sim 400$ iterations. The reason therefore is a continuous harmonic oscillation of near trajectory points in the first half.

In case 4, Fig. 4.6a shows a situation which has a positive diagonal symmetry. Here it gets clear why the unit vector approach struggles with symmetric situations. While for this case the obstacle function approach finds a feasible solution (Fig. 4.6d), the unit vector approach gets caught by the other robot. The reason, therefore, is the same as explained earlier. The update step falls in a bad equilibrium, where the smoothness, the obstacle and the interference objective gradient cancel each other out. The two peaks of the obstacle function approach in Fig. 4.6e are, on the one hand, for when the robots hit the obstacle and, on the other hand, when they have to avoid each other, seen in Fig. 4.6b. In this case the termination threshold is too high and should be set below $\nabla U[\xi] = 0.03$.

A special scenario shows case 5 (Fig. 4.7). There, each robot’s starting point lies on the others ending point. The obstacle is located exactly in the middle. As both end positions (Fig. 4.7b and Fig. 4.7c) illustrate, both approaches cannot handle two starting lines with a straight angle between them and and an obstacle centred in the middle. Both robots ignore the obstacle in this case. While with the unit vector approach the trajectory stops when the robots get close to each other, the trajectories go straight through with the obstacle function approach.

Finally case 6 is a small modification of case 5. The robot configurations are the same
but the obstacle is shifted slightly downward. Because of this, both robots tend to circuit the obstacle above. As the trajectory progress is axisymmetric, the unit vector approach in Fig. 4.8b gets stuck like it has already been seen in Fig. 4.6c. On the other hand, the obstacle function approach finds, although not very smooth, a possible solution.

With several simulations and the cases showed above it is proved that a robot-to-robot collision avoidance works adequate when integrated into the functional objective $U$. The robots automatically adjust their trajectory when they get within critical distance to another robot by acceleration, deceleration or making a detour around. To assess which approach is the better one is not easy. Overall the obstacle function approach is more robust and can handle almost every scenario. In contrast the unit vector approach needs less iterations to reach a local minimum. But since for both cases the convergence time is small, the obstacle function approach should be preferred.
4.3. Simulation Study

(a) Start position

(b) End position

(c) Functional gradient curves

Figure 4.3: Interference objective comparison: Case 1
Chapter 4. Simulation Results

(a) Start position

(b) Intermediate position obstacle function approach

(c) End position

(d) Functional gradient curves

Figure 4.4: Interference objective comparison: Case 2
4.3. Simulation Study

(a) Start position

(b) End position unit vector approach

(c) End position obstacle function approach

(d) Functional gradient curves

Figure 4.5: Interference objective comparison: Case 3
Figure 4.6: Interference objective comparison: Case 4
4.3. Simulation Study

(a) Start position

(b) End position unit vector approach

(c) End position obstacle function approach

(d) Functional gradient curves

Figure 4.7: Interference objective comparison: Case 5
Chapter 4. Simulation Results

(a) Start position

(b) End position unit vector approach

(c) End position obstacle function approach

(d) Functional gradient curves

Figure 4.8: Interference objective comparison: Case 6
4.3.3 Curvature Insights

In section 3.3.2 different curvature objectives were derived and explained. Looking at Fig. 4.8c, one can see a good example where a constraint in the curvature would be necessary. The turn done by the robot is not possible for higher velocity or two-steered-wheels robots. The curve radius has to be bigger. Thus, in the beginning and end of the trajectory the lines cannot be straight anymore. The presented curvature objectives should have been able to cope with this.

Unfortunately, the simulations do not show the desired behaviour. When the trajectory gets close to the obstacle, it starts to oscillate badly or it does not react at all. The reasons for this could be various ones. Potentially a simplification of the curvature for a two dimensional case is not valid. The discretization and their partial derivatives yield to unequal correcting values for trajectory points around the optimized point. As only one obstacle is considered, they should have the same size. As a conclusion it may not be possible to treat the curvature in the easier discrete case and it should be approached in continuous time. But the derived curvature objectives do not have to be completely wrong. As with this objective, another optimization parameter comes into play. Therefore, the right weighting of each objective gets more important. Thus, the applied weight factors found by trial and error are probably wrong. An automatic, iterative parameter search could yield to better overall performance where all functional objectives are in balance. Because of the limited time this was not able to test in this project.

Even though the curvature objective did not work, several insights were found. Now it is clear how the curvature looks like and how it changes when a trajectory gets bent. Fig. 4.9 shows the magnitude of the curvature when the robot circles an obstacle. The red line is the curvature vector of the particular x- and y-magnitudes. The longer the line, the stronger the curvature is. It attracts attention that the curvature grows fast at the beginning and the end of the curve. This is a matter of the trajectory points resolution. As the robot circles the obstacle in a constant curve, the curvature magnitude is almost constant in the middle.

There are also other approaches to handle the curvature of a trajectory. One idea is to set a predefined maximal curvature magnitude, given for example by a minimal turning radius of a robot. Whenever the curvature of a point is higher than the threshold, the curvatures of the neighbouring points are increased. If the neighbouring points have a higher curvature, then the points with too high curvature are flattened out by the smoothness objective. Another possibility is to run two separate CHOMP algorithms. The first one computes a trajectory based on the robot’s steering constraints. Then, starting with the obtained trajectory, the second CHOMP optimizes the trajectory to allowable obstacle distances. If within the update step the curvature gets smaller, then the optimization is done. Otherwise, the old trajectory point is kept. Another alternative is avoiding excessive curvature by an adaptive weighting of the smoothness objective. As curvature increases at a waypoint, its $\lambda$ is also increased and the trajectory gets flattened in this region. The expected difficulty with this approach lies in the risk of instability due to the changing weights.

If the presented ideas might work could not be tested anymore. They are separated from the initial idea to include the curvature into the functional objective. Nevertheless, these alternative approaches sound promising and might be part of a future work.
Figure 4.9: A trajectory circles an obstacle clockwise. The blue and green lines are the curvature’s magnitude in x- and y-direction. The resulting vector is highlighted in red.
Chapter 5

Conclusion and Future Work

5.1 Introduction

In this chapter, a conclusion of the presented results from chapter 4 is given. The goals of the project are reflected again and compared with the obtained results. Additionally, an outlook on possible future work in the field of motion planning and the CHOMP algorithm in particular is provided.

5.2 Evaluation of the Project

Based on the stated goals in section 1.1 and the simulation results in section 4.3 the work is summarized below.

- In a first step the CHOMP algorithm was successfully adapted to the stated representation. Therefore, the presented theory in [13] had to be investigated and understood. The functional objectives are now given in an adapted form for a 2D plane representation. For this, discrete gradient functional objectives and necessary matrices and vectors are derived.

- Out of the adapted trajectory optimization technique a graphical representation was implemented. It represents simple robot motions in a flat terrain. With this GUI the investigated optimization technique is understood much better. While it is easy to use, simulations and testing can be done with it. Therefore, it serves as a handy platform to investigate new ideas or combinations with other techniques. The results show that CHOMP is highly valuable for use in the underlying project of this work. This algorithm performs in a fast way and with less computational costs than other techniques. It provides smooth and collision free paths while seeking for an optimal trajectory.

- An important goal of this thesis was the extension to a multiple robot representation. Therefore the required matrices and vectors were adapted and a new functional objective was built directly into the CHOMP algorithm. Two different approaches were derived and tested against each other. While the unit vector approach needs less computational cost to converge to a local minimum, the obstacle function approach handles more tricky situations better. As a consequence, the obstacle function approach is to prefer if the number of robots is small. For higher
number of robots this may not be the case anymore. In general the presented tech-
nique to deal with robot-to-robot avoidance works very well and is robust as many
simulations have shown. The use of this extended algorithm for future projects can
be highly recommended.

- Various attempts tried to integrate curvature constraints of nonholonomic robots.
  Even if this did not work, the graphical representations and various simulations led
to a deeper understanding of it. As a result, possible alternatives are presented. A
final refusal to integrate the curvature into the objective functional should not be
done as long as other approaches still are possible.

In summary, the existing path adaption technique is successfully extended for multiple
robots. A graphical representation is implemented and used for simulation and testing.
This GUI can be used for further developments and validations. With the acquired feeling
for curvature behaviour, possibilities and limits are more clear and other approaches are
devised.

5.3 Future Work

During the time of this project, new and interesting questions arose which could not be
answered up to this moment. Possible future work in the field of trajectory optimization
are the following.

- It is desirable to obtain a method which successfully treats curvature constraints
  of wheeled robots. The possible approaches presented in section 4.3.3 are ready for
  implementation and testing. A new derivation of the functional gradient by solving
  the Lagrangian form of an optimization problem may lead to the desired behaviour.

- So far, for every trajectory the number of trajectory points are the same. As a con-
  sequence, the robots reach their ending point at the same time. While a robot with
  a long path drives fast, a robot with a short route has to drive slowly. With a dis-
  engagement of this constraint the robots could move closer to their velocity and
  acceleration limits. The goal points are reached faster and the possibility of a col-
  lision is smaller. Whether one should implement the variable number of trajectory
  points inside CHOMP or deal this as an outer motion planner has to be determined.

- For the simulation always two robots were considered. With this it has been proven
  that the algorithm also works for multiple robots. To simulate a variable number
  of robots, the established code has to be changed in order to automatically adjust
  matrix and vector sizes. The derivations presented in chapter 3 remain the same
  and can be directly modified in the explained way.
Appendix

A Partial Derivatives for the 2D Curvature Representation

In section 3.3.2 two different approaches for a possible curvature objective are presented. For the first one, Eq. (3.22) expresses the discrete gradient objective. Within this equation the gradient $\nabla \kappa_t$ is needed. This appendix presents the partial derivatives which are used for the implementation. They have been computed with Mathematica®, a computational software program.

In the discrete case, the velocity and the acceleration are computed via finite differences. For the velocity, one can choose between forward, average or backward velocity. Starting from the actual trajectory point $q_i$, the forward velocity would be $q'_i = \frac{q_{i+1} - q_i}{\Delta t}$, where $q_{i+1}$ is the trajectory point after $q_i$. This leads to a drift of the trajectory points into the direction of the goal point because only the path in front of $q_i$ is regarded and not what is behind $q_i$. Therefore the average velocity and the average acceleration is used for implementation:

$$q'_i = \frac{q_{i+1} - q_{i-1}}{2\Delta t}, \quad \text{(A.1)}$$

$$q''_i = \frac{q'_i^+ - q'_i^-}{\Delta t} = \frac{q_{i+1} - 2q_i + q_{i-1}}{\Delta t^2}. \quad \text{(A.2)}$$

In Eq. (A.2), $q'_i^+$ stands for the forward velocity and $q'_i^-$ for the backward velocity. This representation of the velocity and acceleration is used in Eq. (3.20) to form the discrete curvature. First, in Fig. A.1 one can see how the curvature looks like when the expression is expanded. Out of this, the partial derivatives are computed and presented in Fig. A.2.
Expand the curvature formula

\[
k = \left(\left(\frac{(xp1 - xml) / (2 * dt)}{(yp1 - 2 * xi + xml) / (dt^2)}\right) - \left(\frac{(yp1 - yml) / (2 * dt)}{(xp1 - 2 * xi + xml) / (dt^2)}\right)\right) / \left(\left(\frac{(xp1 - xml) / (2 * dt)}{2 * dt}\right)^2 + \left(\frac{(yp1 - yml) / (2 * dt)}{2 * dt}\right)^2\right) \bigg)^{3/2}
\]

Simplify[k]

\[
k = \frac{8}{dt^3} \frac{(xpl^2 - 2 * xml^2 + xpl^2 + yml^2)^{3/2}}{dt^3} \left(\frac{\text{xm}^2 - 2 * \text{xm} * \text{xp} + \text{xp}^2 + \text{ym}^2}{\text{dt}^2}\right)
\]

Expands[k]

\[
\begin{align*}
\text{xml yml} & \quad \text{dt}^3 \left(\frac{\text{xm}^2 - 2 * \text{xm} * \text{xp} + \text{xp}^2 + \text{ym}^2}{\text{dt}^2}\right)^{3/2} - \quad \text{xpl yml} \\
\text{dt}^3 \left(\frac{\text{xm}^2 - 2 * \text{xm} * \text{xp} + \text{xp}^2 + \text{ym}^2}{\text{dt}^2}\right)^{3/2} - & \quad \text{xpl yml}
\end{align*}
\]

Figure A.1: The 2D curvature representation of Eq. (3.20) is expanded in a first step. "xp1" stands for the first entry in \(q_{i+1}\) and "yp1" the second entry respectively. "xi" and "yi" are the entries in \(q_i\) and "xm1" and "ym1" are the entries in \(q_{i-1}\).
Partial derivatives

\[
D[\mathbf{kk}, \mathbf{xml}] = \frac{8 (y_1 - y_{pl})}{(x_2 - 2 x_1 x_{pl} + x_{pl}^2 + (y_{ml} - y_{pl})^2)^{3/2}} - \frac{12 (-2 x_1 + 2 x_{pl}) (x_{pl} (y_1 - y_{ml}) + x_1 (y_1 - y_{ml}) + x_{pl} (y_1 - y_{pl}) + x_1 (y_1 - y_{pl}))}{(x_2 - 2 x_1 x_{pl} + x_{pl}^2 + (y_{ml} - y_{pl})^2)^{5/2}}
\]

\[
D[\mathbf{kk}, \mathbf{xi}] = \frac{8 (-y_{ml} - y_{pl})}{(x_2 - 2 x_1 x_{pl} + x_{pl}^2 + (y_{ml} - y_{pl})^2)^{3/2}}
\]

\[
D[\mathbf{kk}, \mathbf{xpl}] = \frac{8 (-y_1 + y_{pl})}{(x_2 - 2 x_1 x_{pl} + x_{pl}^2 + (y_{ml} - y_{pl})^2)^{3/2}} - \frac{12 (-2 x_1 + 2 x_{pl}) (x_{pl} (-y_1 + y_{ml}) + x_1 (-y_1 + y_{ml}) + x_{pl} (-y_1 + y_{pl}) + x_1 (-y_1 + y_{pl}))}{(x_2 - 2 x_1 x_{pl} + x_{pl}^2 + (y_{ml} - y_{pl})^2)^{5/2}}
\]

\[
D[\mathbf{kk}, \mathbf{yml}] = \frac{8 (-x_1 + x_{pl})}{(x_2 - 2 x_1 x_{pl} + x_{pl}^2 + (y_{ml} - y_{pl})^2)^{3/2}} - \frac{24 (-y_{pl}) (x_{pl} (-y_1 + y_{ml}) + x_1 (-y_1 + y_{ml}) + x_{pl} (-y_1 + y_{pl}) + x_1 (-y_1 + y_{pl}))}{(x_2 - 2 x_1 x_{pl} + x_{pl}^2 + (y_{ml} - y_{pl})^2)^{5/2}}
\]

\[
D[\mathbf{kk}, \mathbf{yi}] = \frac{8 (x_1 - x_{pl})}{(x_2 - 2 x_1 x_{pl} + x_{pl}^2 + (y_{ml} - y_{pl})^2)^{3/2}}
\]

\[
D[\mathbf{kk}, \mathbf{ypl}] = \frac{8 (x_1 - x_{ml})}{(x_2 - 2 x_1 x_{pl} + x_{pl}^2 + (y_{ml} - y_{pl})^2)^{3/2}} - \frac{24 (-y_{pl}) (x_{pl} (-y_1 + y_{ml}) + x_1 (-y_1 + y_{ml}) + x_{pl} (-y_1 + y_{pl}) + x_1 (-y_1 + y_{pl}))}{(x_2 - 2 x_1 x_{pl} + x_{pl}^2 + (y_{ml} - y_{pl})^2)^{5/2}}
\]

Figure A.2: Partial derivatives of the curvature given in Fig. A.1.
Bibliography


Halmstad University Student Project Description
Investigation of trajectory optimization for multiple car-like vehicles

July 19, 2014

Overview

In scenarios where multiple robots move among objects, it can be advantageous to explicitly take into account interactions between robots at the planning and control stage [5, 4]. As a concrete example, consider a cargo goods terminal where a fleet of automated guided vehicles (AGVs) transport containers to and from ships, trains, trucks, and storage stacks. Halmstad University is involved in the European project Cargo-ANTS which addresses this scenario 1.

Of particular interest is the use of path- and trajectory- adaptation techniques to serve as bridge between planning and control. This has been shown to be an effective way of dealing with partially known and unpredictable environments [2, 1]. Recently, techniques based on numerical optimization have emerged that promise to provide more effective solutions to the trajectory adaptation problem, in particular Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [3, 6].

Description of Work

In this project, we will investigate the use of CHOMP for simultaneously optimizing the trajectory of multiple mobile robots. Furthermore, we will conduct an exploration into considering car-like steering constraints directly inside CHOMP. We expect work to progress according to the following steps.

1. Formulate a simple multi-robot trajectory representation. Start by considering point robots that can move freely in the 2D plane. Extensions will depend on progress in the more experimental parts of the project.
2. Use CHOMP to implement an optimization of the formulated representation, such that robot-to-robot and robot-to-obstacle distances are taken into account. Graphically demonstrate the optimization process with a simple interactive program.
3. Compute the curvature along the optimized trajectory in order to get a feeling for how it evolves depending on optimization parameters.
4. Investigate formulations for path-curvature objective functions inside CHOMP with the goal of directly taking into account steering constraints of car-like vehicles.

1http://www.cargo-ants.eu/
Supervision

The project will be supervised by Roland Philippsen and one of his PhD students. Weekly coaching sessions will be set up. Additional supervision will be available as required on demand.

References


