TOPOLOGY-BASED REPRESENTATIONS FOR MOTION SYNTHESIS AND PLANNING

VLADIMIR IVAN

Doctor of Philosophy
School of Informatics
University of Edinburgh

2014

Vladimir Ivan:
Topology-based Representations for Motion Synthesis and Planning
Doctor of Philosophy, 2014

SUPERVISORS:
Sethu Vijayakumar

ABSTRACT

Robot motion can be described in several alternative representations, including joint configuration or end-effector spaces. These representations are often used for manipulation or navigation tasks but they are not suitable for tasks that involve close interaction with the environment. In these scenarios, collisions and relative poses of the robot and its surroundings create a complex planning space. To deal with this complexity, we exploit several representations that capture the state of the interaction, rather than the state of the robot. Borrowing notions of topology invariances and homotopy classes, we design task spaces based on winding numbers and writhe for synthesizing winding motion, and electro-static fields for planning reaching and grasping motion. Our experiments show that these representations capture the motion, preserving its qualitative properties, while generalising over finer geometrical detail. Based on the same motivation, we utilise a scale and rotation invariant representation for locally preserving distances, called interaction mesh. The interaction mesh allows for transferring motion between robots of different scales (motion re-targeting), between humans and robots (teleoperation) and between different environments (motion adaptation). To estimate the state of the environment we employ real-time sensing techniques utilizing dense stereo tracking, magnetic tracking sensors and inertia measurements units.

We combine and exploit these representations for synthesis and generalization of motion in dynamic environments. The benefit of this method is on problems where direct planning in joint space is extremely hard whereas local optimal control exploiting topology and metric of these novel representations can efficiently compute optimal trajectories. We formulate this approach in the framework of optimal control as an approximate inference problem. This allows for consistent combination of multiple task spaces (e.g. end-effector, joint space and the abstract task spaces we investigate in this thesis).

Motion generalization to novel situations and kinematics is similarly performed by projecting motion from abstract representations to joint configuration space. This technique, based on operational space control, allows us to adapt the motion in real time. This process of real-time re-mapping generates robust motion, thus reducing the amount of re-planning. We have implemented our approach as a part of an open source project called the Extensible Optimisation library (EXOTica).

This software allows for defining motion synthesis problems by combining task representations and presenting this problem to various motion planners using a common interface. Using EXOTica, we perform comparisons between different representations and different planners to validate that these representations truly improve the motion planning.

DECLARATION

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Edinburgh, 2014

Vladimir Ivan
February 18, 2015

PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

Ivan, V., Zarubin, D., Toussaint, M., Komura, T., and Vijayakumar, S. (2013). Topology-based Representations for Motion Planning and Generalisation in Dynamic Environments with Interactions. *The International Journal of Robotics Research*, 32(9-10):1151–1163.

Llamazares, Á., Ivan, V., Molinos, E., Ocaña, M., and Vijayakumar, S. (2013). Dynamic Obstacle Avoidance Using Bayesian Occupancy Filter and Approximate Inference. *Sensors*, 13(3):2929–2944.

Llamazares, Á., Ivan, V., Ocaña, M., and Vijayakumar, S. (2012). Dynamic obstacle avoidance minimizing energy consumption. In *Intelligent Vehicles Workshop on Perception in Robotics*, Alcalá de Henares, Spain.

Pauwels, K., Ivan, V., Ros, E., and Vijayakumar, S. (2014a). Real-time Object Pose Recognition and Tracking with an Imprecisely Calibrated Moving RGB-D Camera. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, Illinois, USA.

Pauwels, K., Rubio, L., Ivan, V., Vijayakumar, S., and Ros, E. (2014b). Real-time RGB-D-based Object and Manipulator Pose Estimation. Workshop on RGB-D: Advanced Reasoning with Depth Cameras in conjunction with Conference on Intelligent Robots and Systems (IROS).

Sandilands, P., Ivan, V., Komura, T., and Vijayakumar, S. (2013). Dexterous Reaching, Grasp Transfer and Planning Using Electrostatic Representations. In *Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Atlanta, Georgia, USA.

Zarubin, D., Ivan, V., Toussaint, M., Komura, T., and Vijayakumar, S. (2012). Hierarchical Motion Planning in Topological Representations. In *Proceedings of Robotics: Science and Systems (R:SS)*, Sydney, Australia.

ACKNOWLEDGEMENTS

I would like to thank a number of people for their support, help and advice during conducting this research and writing this thesis.

Firstly, I would like to thank my supervisor Sethu Vijayakumar for providing this opportunity and then encouraging and guiding me through the perils of scientific research.

Secondly, I would like to all the collaborator who I have worked with on various projects, namely Peter Sandilands, Dmitry Zarubin, Ángel Llamazares, Florian Pokorny, Ocaña, Manuel and Eduardo Ros.

I would also like to thank Marc Toussaint, Taku Komura, Subramanian Ramamoorthy, Seth Hutchinson for stimulating discussions and their valuable input, which undeniably helped to improve this research.

I would like to thank Jun Nakanishi, David Braun, Matthew Howard, He Wang and all the people from my research group, SLMC , for their support and friendship.

Lastly, I would like to give special thanks to Joseph Henry, Adam Barnett for countless discussions, proofreading, friendship, numerous coffee breaks and making the past couple of years a pleasant experience.

Good news everyone! This thesis is dedicated to
B. B. R., P. J. F., Dr. J. A. Z. and Professor H. J. F.

CONTENTS

1 INTRODUCTION

Humans and most animals interact with objects that surround them with astonishing levels of skill and dexterity. Even the most common of everyday tasks, such as picking up a glass of water or putting ring on a finger, are trivial for a human but very challenging for an autonomous robotic system. The ease with which we learn how to optimally solve a task is inspiring. For example, successfully completing the non-trivial task of carrying a glass with water and keeping it upright to prevent the water from spilling while maintaining balance and avoiding obstacles. The human nervous system learns and encodes these skills through a process of trial and error. The question is: how can we endow a robot with similar level of robust motion planning capabilities using our expertise, instead of going through the lengthy process of teaching the robot through examples? How can we take goals, that humans easily formulate, and translate them into a language that a complex robot can use for motion synthesis? Some of the most advanced robotic systems we know of today are still relatively simple when compared to biological systems. Yet motion synthesis for such robotic platforms is notoriously hard, especially when it involves close interactions with the environment. What makes motion planning in the real world so challenging?

Curse of dimensionality

Modern robotic systems, such as robot arms and humanoid robots, have a large number of actuators, each of which defines a degree of freedom (DoF) that has to be controlled. A humanoid robot usually has tens of DoF[1]. If we aim to explore the space of such high dimensionality to compute a motion plan, we will soon encounter the practical problem of sufficiently covering the space with samples to ensure that the plan is truly feasible and optimal. This phenomenon is caused by the fact that any two points may appear close together along one dimension but be very far apart on another, which increases the space that needs to be covered. This problem is known as the *curse of dimensionality*. Planning motion for such a high number of DoF has been studied in the context of exploring the space of possible motions (Shkolnik and Tedrake [2009]). In the field of robotics, not only

1 The Atlas robot by Boston Dynamics has 28 DoF (without the hands) and the NASA Valkyrie robot has 44 DoF (including the hands).

the dimensionality but also the way how these degrees of freedom interact poses another very common difficulty.

Redundancies

A robot usually has more DoF than it needs to perform a single task. This means that there are multiple *redundant* ways to complete the task, e.g. when opening a door by pulling the door handle, we could pull the handle using the shoulder and upper arm or we could keep the upper body stiff and take a step back, dragging the door handle behind. Each solution has its advantages. Using only the arm is more energy efficient, while using the weight of the whole body when stepping back allows us to exert more force and to open heavier doors. Redundancies have been studied on systems with low DoF (Aydin and Kocaoglan [1995]) but more complex redundant systems still pose a challenge. One way to address this problem is to define multiple tasks or multiple objectives that would exploit each others' redundancies (Kanoun et al. [2011]).

Multi-objective optimisation

Multi-objective optimisation has been applied in many areas of science on problems with multiple simultaneous objectives. When the objectives are complementary to each other, they may indeed help to resolve redundancies. However, when they are conflicting, a trade-off has to be made to find a solution that satisfies all of the objectives optimally (da Graça et al. [2012]). For example, when reaching for an apple on a tree, we have to make a trade-off between keeping balance and reaching for the apple. To solve this problem, we may choose one of the two approaches: (1) to *prioritize* one objective and satisfy the second one without violating the higher priority objective (Flacco et al. [2012]), e.g. reach for the apple only as far as we can without loosing balance; (2) to decide on relative *weighting* between the two objectives and to satisfy both to a degree set by their respective weights (Siciliano and Khatib [2008]). In both cases, we have to specify, either, to what degree do we care about satisfying an objective, or on the priority of the objectives. After making this decision, we are still faced with the problem of synthesizing a motion that fulfils the task defined by this mixture of objectives. When dealing with multi-objective problems, it is often the case that these objectives impose complex constraints that prevent motion in a specific direction, e.g. if another branch prevents us from reaching any closer to the apple. In this case, an algorithm attempting to minimize the distance to the apple would get stuck.

Local minima

Many optimization algorithms[2] are based on the principle of incrementally improving an existing or initial motion and are therefore susceptible to getting stuck in a *local minima*. We can get around this problem by choosing a different initial

2 See Section 2.2 for overview of optimisation algorithms used for solving problems in robotics.

motion in an attempt to converge to a *globally optimal solution*, e.g. instead of reaching around the branch that is blocking our way to the apple, we can choose to sit on the obstructing branch and try to reach for the apple from there. In this scenario, we are still only considering planning the motion without accounting for the interaction with the environment, such as the branches bending in the wind or simply not knowing the shape of the branch.

Complex dynamics

These interactions between the robot and its surroundings often have *complex dynamics* (Sentis et al. [2010]), where by complex dynamic we refer to dynamical systems within which the tasks depend on *contacts* with objects, *friction* and *collisions*. The complexity arises from the difficulty to model and, often, even measure these properties using sensors, e.g. when picking up an object, the contact area between the robot hand and the object it is grasping is difficult to estimate, which is the reason why grasping is often performed based on predefined object shape classes such as cylinders or spheres (Miller et al. [2003]). When synthesizing motion we may choose to define the motion relative to an object. If the object stays static, the interaction can be evaluated without any further consideration. However, this is not always the case.

Dynamic environments

The objects around the robot may be moving. If we plan a motion in this *dynamic environment*, the plan will quickly become invalid. In that case, *re-planning* is necessary (Gayle et al. [2007]). In some cases, the motion of the objects in the environment is known, e.g. motion of free flying objects can be efficiently estimated using Kalman filter (Siciliano and Khatib [2008]). Not all kinds of motion can be predicted though.

Uncertainty and unpredictability

Some motion is *stochastic* in nature, which has to be accounted for during motion synthesis (Theodorou et al. [2010]). When interacting with a human, however, we may not be able to model their behaviour at all, which will make any motion the human makes *unpredictable*. In this case, re-planning is often necessary. Humans are very skilled at adapting their plans when the environment changes and quickly react to correct their actions. Reacting in this way is challenging (Kröger [2012]) and it requires a way of describing objectives that will capture the interaction with the environment.

Representing goals

Tasks can be defined in terms of simple goals such as reaching for an apple or keeping a hand upright to hold a glass of water. These are examples of classical *goal representations* that are based on geometrical properties such as position and orientation. Some interactions can be represented more directly, e.g. by describing how the fingers wrap around glass or how the palm envelops the apple (see

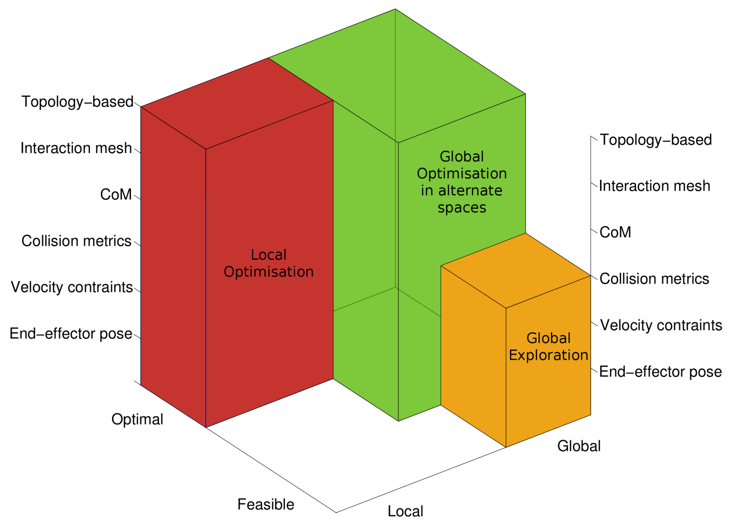


Figure 1: Lattice showing areas of applications of exploratory motion planning algorithms such as RRTs (orange), and optimisation techniques such as iLQG and AICO (red). See Chapter 2 for overview of different motion planning techniques. Our proposed methods are aimed at improving these techniques by extending their areas of application towards the green region by exploiting alternate representations.

Sandilands et al. [2013b] and Ivan et al. [2013]). If the specific property of the interaction is not known, we may choose preserve the spatial relationship of the robot segments to match those of a demonstrator as proposed by Ho et al. [2010b]. There are multiple choices of goal representations for each task, which brings us back to the problem of multi-objective optimisation. How to solve such complex robotic problems for the real world applications then? Figure 1 shows an overview of different classes of motion synthesis algorithms and the ways how they solve the motion planning problem. Our work aims to improve the existing techniques by extending their areas of application towards the green region.

We have, thus far, presented challenges that have to be addressed when solving the problem of motion planning with close interactions in dynamic environments. In this thesis, we address these challenges. The focus of our work is to directly represent the task involving interactions between the robot and its surroundings. After formalizing the *task representations* in this way, we investigate ways to do planning in an abstract task representation in the changing environments before-

Task representations

hand and to then couple this abstract plan with the low level controls. This can be viewed as computing motion plans that generalise to different situations.

The main reason for modelling the interaction directly, in a way that generalises well to novel situations, is to improve the speed of motion synthesis and robustness of the resulting motion. Motion planned in environments that dynamically change often becomes invalid and it has to be recomputed. On the other hand, if our motion is defined using an abstract task representation, we can recompute the mapping into the space of controls when the environment changes, instead of replanning the motion from scratch. The key assumption behind our work is that even though planning the motion in abstract spaces may be computationally costly, the resulting plans will then generalise over a wide variety of practical scenarios and that the coupling between the motion in an abstract representation and the robot controls is less costly. The nature of this coupling dictates how efficient and robust the motion synthesis will turn out to be.

There are several properties we are aiming for when designing task representations. A task representation defines a mapping from the space of controls into an abstract space which we treat as a *task space* (a classical example of a task space is the end-effector position of a robot arm). In doing so, we define representations[3] we use for transferring the motion to different environments or robots by preserving simple metrics in the task space. The simplicity of these metrics may be evaluated based on the gradient in the task space, which would ideally be smooth, differentiable, continuous and in some cases even monotonous. Task spaces with such metrics may then render complex motion in the configuration space to be equivalent to basic interpolation in the task space[4]. The motion synthesis methods differ in the way they utilise the task representations. While exploratory methods, that we review in Section 2.1, often exploit task spaces for validity checking, optimisation methods, discussed in Section 2.2, heavily utilise gradients in the task spaces. Our objective is to exploit spaces that provide gradients suitable for efficient optimisation. Additionally, we propose methods that can combine multiple different task representations at the abstract, as well as the lower (execution) level, and plan simultaneously at these different levels to efficiently generate optimal, collision free motion that satisfies constraints and affordances. Each representation has its own strengths and weaknesses and coupling them enables us solve a wider range of problems than they are capable of solving individually.

3 In Chapter 3, we discuss novel task representations and evaluate their utility using experiments.
4 In Section 3.5, we discuss task space invariants based on homotopy classes, which we utilise for motion planning.

The representations that have the properties we are looking for are often based on invariants that abstract away the geometry of the interaction between the robot and its environment, and capture the topology of this interaction. The topological invariants have a formal definition in topology, as the area of mathematics. These invariants are, however, often impractical due to the computational cost or the assumption of the full knowledge of the state of the environment at all times. This is why we aim to design representations that are based on topological invariants or have similar properties with respect to their ability to abstract away the geometrical details of the interaction [5].

The contributions of this thesis are:

- The introduction of abstract *representations* inspired by ideas from a mathematical branch called topology that directly model the interaction between the robot and its environment: winding number, writhe, electrostatic flux, electric coordinates and interaction mesh.
- A principled extension of a stochastic optimal *control framework* that admits the capability to combine various representations for motion synthesis. This is expressed in a graphical model that couples motion priors at different levels of representations.
- A method for real-time motion *generalization* (remapping) to novel situations using abstract representations and operational space control.
- A set of *sensing techniques* adapted for capturing dynamic environments: vision based detection and tracking using RGB-D features, magnetic tracking, and inertia measurement based tracking.
- A set of *experiments* comparing novel and classical representations using state of the art motion planning methods (AICO, RRT*).
- A software implementation of a framework for planning in and benchmarking task representations and motion planners called *EXOTica*.

5 We formally define a topology-based representation in Chapter 3 and further discuss and justify the use of topological invariants for representing robot tasks.

2 MOTION PLANNING

Motion planning is a fundamental topic in robotics. A multitude of techniques for solving a variety of motion planning problems have been proposed in the literature over past couple of decades. To provide a deeper insight into these problems, we describe the motion planning process in the area of robotics, starting by introducing the building blocks of a planning algorithm.

We describe the configuration of a robot with n DoF using a n dimensional vector x. The space of all possible configurations of the robot is then called the *configuration space*:

Configuration space

$$x \in C \subseteq \mathbb{R}^n. \tag{1}$$

In our work, we consider continuous configuration spaces[1] that are $\mathbb{R}^n$ because they represent the continuous properties of the real world such as position, velocity or rotation angle. We call the space in which the robot operates the *work space*

Work space

$$W \subseteq \mathbb{R}^m, \tag{2}$$

where m is the dimensionality of the work space. Each part[2] of the robot occupies a some subset A_i of the work space such that

$$A = \cup_{i=1}^{n} A_i \subseteq W, \tag{3}$$

where A is the subset of the workspace occupied by the robot as a whole. Similarly, obstacles occupy a subset O of the same work space. The part of the space where the robot and the obstacles are not in collision is called *free space* and we define it as

$$C_{\text{free}} = \{x \in C | A_x \cap O = \varnothing\}. \tag{4}$$

1 Discrete spaces such as grid worlds are commonly used to approximate the real world.

2 Here, we assume that a part occupying space A_i is composed of all robot parts that are rigidly attached to joint i which actuates them.

Assuming that we are given an initial state $x_0 \in C_{\text{free}}$ and a goal state[3] $x_T \in C_{\text{free}}$, we define motion planning as the mapping

$$\tau : [0, 1] \to C_{\text{free}}, \tag{5}$$

subject to

$$\tau(0) = x_0, \tag{6}$$

$$\tau(1) = x_T, \tag{7}$$

where the subscript on x_T denotes the final time when the robot reaches its goal. The planning algorithms differ from one another based on how they compute the mapping τ. We will introduce two approaches: (1) exploratory methods, that sample random states from configuration space until a configuration that satisfies the goal criteria is found, and (2) optimisation based methods, that iteratively improve an initial trajectory until an optimal trajectory is found. Although these methods are very different from each other, both can be applied on the same class of problems.

A classical problem in robotics is to find a feasible path in the free space (LaValle [2006]) but it is often desirable to plan a trajectory that satisfies additional constraints or one that minimizes a cost function in a space, different from the configuration space. We call this space the task space and define it as

$$\phi : X \to Y. \tag{8}$$

Task space

Function ϕ (the *task map*) is the mapping between the configuration space X and the *task space* Y. We aim to exploit different task spaces to improve the performance and robustness of motion planning, but we will start by presenting two classes of planning algorithms to provide deeper insight into how the choice of task spaces affects motion planning.

First, we present a class of algorithms that use exploration in the configuration space to compute the motion plan, where the task spaces are used to bias the exploration to improve convergence. We will then present a class of techniques that use the task space to construct a cost function that is then used for incrementally improving the existing plan. The contributions of this chapter are:

3 The goal state can be a single state or a set of states from a region that satisfies the goal criteria based on some arbitrary metric.

- A principled extension of a *stochastic optimal control framework* that admits the capability to combine various representations for motion synthesis. This is expressed in a graphical model that couples motion priors at different levels of representations.
- Extension of the Approximate Inference Control (AICO) algorithm for real time replanning for mobile platforms in dynamic environments (See Section 2.2.3).

2.1 EXPLORATORY MOTION PLANNING

When the exploratory motion planning was introduced in the 1990s, it enabled solving motion planning problems that had been previously considered infeasible (LaValle [2006]). These techniques are build around the principle of sampling robot configurations and building up a tree of possible transitions between these configurations until one that satisfies the goal criteria is found. The tree is then back-tracked to its root to produce the trajectory for the robot motion. The *rapidly exploring random trees* (RRT) algorithm, for instance, computes feasible collision-free paths given an initial state x_0 and a termination criteria. RRT builds a tree T in the configuration space as described in Algorithm 1, where the sample_free() is a procedure returning a random configuration from the free space. The nearest_neighbour(x, V) procedure returns the nearest neighbour of x from within the set of existing nodes V in the tree T. The steer(x, y) procedure returns a configuration that is closer[4] to x than y. The collision_free(x, y) procedure tests if the path between x and y is free of obstructions, and build_tree() assesses the termination criteria.

RRT

Algorithm 1 RRT (x_0)

```
1: $T.V \leftarrow \{x_0\}$
2: while build_tree do
3:     $x_{rand} \leftarrow$ sample_free()
4:     $x_{nearest} \leftarrow$ nearest_neighbour$(x_{rand}, T.V)$
5:     $x_{new} \leftarrow$ steer$(x_{nearest}, x_{rand})$
6:     if !collision_free$(x_{nearest}, x_{new})$ then
7:         continue
8:     $T.V \leftarrow T.V \cup \{x_{new}\}$
9:     $T.parent(x_{new}) \leftarrow x_{nearest}$
```

4 If the dynamical model of the robot is known, the steer(x, y) procedure usually explots this model to steer the robot closer to state y.

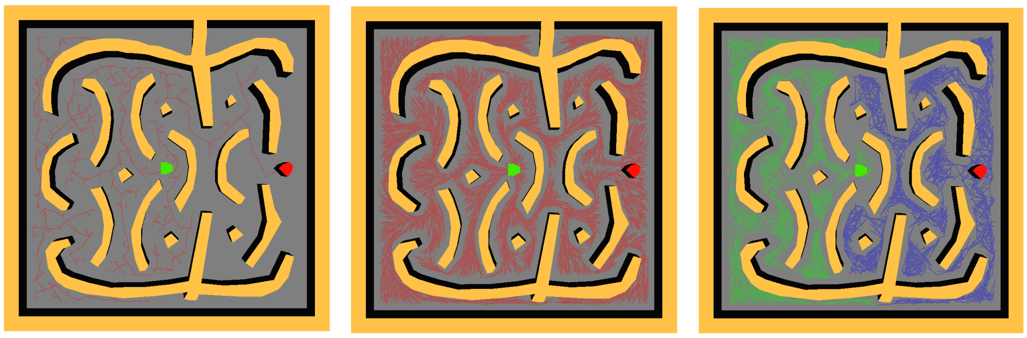


Figure 2: Different search trees produced by RRT (left), RRT* (middle) and Bi-directional KPIECE[6] (right). In three figures, the root of the tree is in the middle of the maze (under the green polygon) and the goal is under the red polygon. The KPIECE algorithm grows a secondary tree from the goal location, which connects to the primary tree in the middle bottom section of the maze.

Unfortunately, common exploratory algorithms such as RRT produce solutions that are sub-optimal or even far from optimal, furthermore, Nechushtan et al. [2011] showed that with probability one, algorithms like RRT will not produce optimal trajectories. This means that high quality trajectories are impossible to construct from entirely random samples, therefore, changing the sampling strategy using a heuristic has been proposed by Urmson and Simmons [2003], to give samples in certain part of the configuration spaces higher probability of being picked. There are, however, multiple approaches to changing the sampling strategy based on the characteristics of the motion planning problem (e.g. differential constraints or complex task spaces). The performance of exploratory algorithms can be severely compromised in the presence of kinodynamic constraints such as planning a swing up task for an under-actuated pendulum or motion of a non-holonomic vehicle (LaValle [2006]). Shkolnik et al. [2009] proposed to change the sampling strategy using a feasibility set estimated based on minimum and maximum commands that can be applied to the robotic system. It is, however, often the case that even a purely kinematic task is too difficult to solve, when the dimensionality of the configuration space is too high or the task constraints are very restrictive. To get around this problem, Shkolnik and Tedrake [2009] proposed *biasing* the exploration towards targets in the task space, thus creating a *motion prior*[5] making selection of certain configurations in this space more probable. This solution dramatically reduces the amount of exploration needed for solving problems on high DoF robots and it indirectly biases the exploration towards more optimal trajectories. Even in this case, the cost of the motion is still not directly optimised.

Motion prior

5 See Section 2.2.2 for formal definition of a motion prior.

A strategy that modifies the tree structure itself has been proposed by Karaman and Frazzoli [2011] to construct a tree that can be used for computing trajectories in the configuration space that are optimal with respect to a cost function. The modified RRT algorithm is called RRT*. The connections between the tree nodes are modified after the new sample is added to the tree (see line 9 in Algorithm 1). At this point, k nearest neighbours of the new node are computed, and if connecting the new node to the nearest node produces a trajectory with a lower cost, the tree gets rewired. Figure 2 shows the tree computed using RRT on left and a tree computed using RRT* in the middle. The cost function used in this example was based on trajectory length. An arbitrary cost function can be specified instead. Motion planning with an arbitrarily complex cost function and in a possibly high dimensional configuration space requires large amount of exploration to compute an optimal trajectory. The computational effort to do this can be reduced by changing the sampling strategy by biasing the exploration using a task space defined using the metric based on the the cost function. Akgun and Stilman [2011] use the change of sampling strategy and additionally grow a second tree rooted at the goal configuration to improve the convergence speed of RRT*. Another approach presented by Salzman and Halperin [2013] solves a similar, *near-optimal*, problem by computing approximate solutions with lower bound on the error. This method allows us to tweak the lower bound of the task cost on the scale between unbounded (equivalent to RRT) and *asymptotically optimal* (equivalent to RRT*), which allows for a trade off between quality and computational cost.

Asymptotically optimal algorithms

Even though proofs of convergence of algorithms like RRT* exist, practical problems often require prohibitive amount of exploration to converge to an optimal solution. The exploration can be biased towards targets in the task space but even in this case, the motion prior in the task space is not directly used to drive the planning. In some cases the motion prior is not available at all (e.g. contacts with the environment). Kalakrishnan et al. [2011] proposed a method that generates (samples) noisy trajectories to explore the space around an initial trajectory, which are then combined to compute a trajectory with a lower cost. This is a useful feature, but even when the motion prior is available, it is not being utilised within this technique. On the other hand, when the motion prior can be computed, and a gradient in the task space exists, it can be used to improve the initial trajectory as proposed by Ratliff et al. [2009]. This technique can be extended with Hamiltonian Monte Carlo sampling scheme to approximate the gradient in the task space and to

6 The Kinodynamic Motion Planning by Interior-Exterior Cell Exploration (KPIECE) has been proposed by Şucan and Kavraki [2010].

alleviate the problem of local minima. This approach is based on gradient descent, which is used to iteratively improve the trajectory. Such iterative improvements are the core of trajectory optimisation algorithms.

2.2 TRAJECTORY OPTIMISATION

Trajectory optimisation is a key problem in robotics. There are two classes of methods that can be used for solving these optimisation problems: (1) using gradient methods, typically with spline-based trajectory encoding (e.g. Yao-Chon [1991] and Zhang and Knoll [1995]) or (2) using *sequential quadratic programming* (SQP) schemes. Solving SQP problems typically involves iterating linear quadratic gaussian (LQG) solution methods. This approach is often used when solving *stochastic optimal control* (SOC) problems. Let us consider a time discrete SOC problem where

SQP

LQG

$$x_{t+1} = f(x_t, u_t) + \xi, \xi \sim \mathcal{N}(0, Q_t), \tag{9}$$

such that the function $f_t(x_t, u_t)$ computes the new deterministic configuration x_{t+1} using the configuration x_t and controls u_t at time t. The Gaussian noise ξ with the covariance Q_t is then added to model the stochasticity. For a sequence of configurations $x_{0:T}$ and controls $u_{0:T}$, we define the cost as

$$C(x_{0:T}, u_{0:T}) = \sum_{t=0}^{T} c(x_t, u_t). \tag{10}$$

In the case of LQG (Stengel [1986]), we approximate Equation 9 using a linear process with gaussian noise

$$P(x_{t+1}|x_t, u_t) = \mathcal{N}(x_{t+1}|A_t x_t + a_t + B_t u_t, Q_t), \tag{11}$$

and the cost from Equation 10 using a quadratic cost

$$c(x_t, u_t) = x_t^\top R_t x_t + 2r_t^\top x_t + u_t^\top H_t u_t. \tag{12}$$

In Equation 11 and the following, we use the notation

$$\mathcal{N}(x|a, A) \propto \exp\left(-\frac{1}{2}(x-a)^\top A^{-1}(x-a)\right) \tag{13}$$

defining a Gaussian over x with mean a and covariance A.

Matrices A_t, B_t and vector a_t linearly approximate the state transition[7] with the normally distributed noise with covariance Q_t as in Equation 9. The matrix H_t is used to model the control effort and the terms R_t and r_t describe task specific cost[8]. We can now compute the optimal control commands[9]

$$u_t^*(x) = -(H_t + B_t \top V_{t+1} B_t)^{-1} B^\top (V_{t+1}(A_t x_t + a_t) - v_{t+1}), \tag{14}$$

such that

$$\begin{aligned}
\mathcal{V}_t(x) &= x_t^\top V x_t - 2x_t^\top v_t + \text{terms independent of } x_t, && (15)\\
V_t &= R_t + (A_t^\top - K) V_{t+1} A_t && (16)\\
v_t &= r_t + (A_t^\top - K)(v_{t+1} - V_{t+1} a_t) && (17)\\
K &= A^\top V_{t+1}^\top (V_{t+1} + B_t^{-\top} H_t B_t^{-1})^{-1},
\end{aligned}$$

where the optimal value function $\mathcal{V}_t$ gives the expected future cost at time t for the best controls, and it obeys the Bellman optimality equation

$$\mathcal{V}_t(x_t) = \min_{u_t} \left[c(x_t, u_t) + \mathcal{V}_{t+1}(f(x_t, u_t)) \right]. \tag{18}$$

The LQG allows us to define exact backward recursion to compute the value function, which will always be a quadratic form of the state x_t. Equations 14-17 are called *Ricatti equations* (Stengel [1986]). If we initialize this system of equations with $V_T = R_T$ and $v_T = r_T$, at the final time step, we can recursively compute the optimal value function. For non-linear systems, we can start with an initial guess for trajectory $x_{0:T}$ and iteratively improve this trajectory by alternating between computing the controls $u_t^*(x)$, that are locally optimal around current trajectory, and evaluationg the value function after executing these controls, as described in Algorithm 2. This method is called iLQG (Todorov and Li [2005]) and it has been used for optimising robot motion with complex dynamics (Nakanishi et al. [2011], Braun et al. [2012]). The iLQG method is suitable for solving non-linear problems because a local linear approximation of the system dynamics, and a local quadratic approximation of the system cost is computed at each time step. This is a powerful technique but it works under the assumption that the system dynamics are smooth and differentiable. Additionally, the trajectory has to be sufficiently

Ricatti equations

7 A_t and a_t describe the motion depending on the state (e.g. drift). B_t approximates the effects of controls.
8 In Section 2.2.2 and Equation 37 we describe how we compute R_t and r_t.
9 See Stengel [1986] for derivation of Ricatti equations.

Algorithm 2 iLQG $(x_{0:T}, H_{0:T}, A_t(x), a_t(x), B_t(x), R_t(x), r_t(x))$

```
1:  $\alpha \leftarrow$ convergence rate
2:  repeat
3:      $V_T \leftarrow R_T(x_T), v_t \leftarrow r_T(x_T)$
4:      for $t = T - 1$ to $0$ do // backward Ricatti recursion
5:          access $A_t(x_t), a_t(x_t), B_t(x_t), R_t(x_t), r_t(x_t)$
6:          compute $V_t$ and $v_t$ using equations 16 and 17
7:      for $t = 0$ to $T - 1$ do // forward controls propagation
8:          compute $u_t^*(x_t)$ using equation 14
9:          $x_{t+1} \leftarrow (1-\alpha)x_{t+1} + \alpha\,[A_t(x_t)x_t + a_t(x_t) + B_t(x_t)u_t^*(x_t)]$
10: until convergence
```

densely discretised to ensure that the local LQG approximations are accurate and the convergence ratio α has to be set to ensure convergence without wasting resources.

The stochastic optimal control problem can also be formulated as Approximate Inference Control (AICO) as proposed by Toussaint [2009]. In this case, the system is modelled using a graphical model and message passing is used compute the maximum likelihood trajectory. This framework naturally leads to iterative updates of local messages, rather than recursive updates of the whole trajectory, thus creating a more flexible algorithm.

2.2.1 *Approximate Inference Control*

Approximate Inference Control (AICO) frames the problem of optimal control as a problem of inference in a dynamic Bayesian network. Let x_t be the state of the system as defined in Equation 1. We will always consider the dynamic case, such that $x_t = (q_t, \dot{q}_t)$, where q_t are the joint positions and $\dot{q}_t$ are the joint velocities. Consider the problem of minimizing (the expectation of) the cost

$$C(x_{0:T}, u_{0:T}) = \sum_{t=0}^{T} c_x(x_t) + c_u(u_t) \tag{19}$$

where c_u describes costs for the control and c_x describes task costs depending on the state (usually a quadratic error in some task space, such as in Equation 12).

The robot dynamics are described by the transition probabilities $P(x_{t+1} \mid u_t, x_t)$. The AICO framework translates this into the probabilistic model

$$p(x_{0:T}, u_{0:T}) \propto P(x_0) \prod_{t=0}^{T} P(u_t) \prod_{t=1}^{T} P(x_t \mid u_{t\text{-}1}, x_{t\text{-}1}) \cdot \prod_{t=0}^{T} \exp\{-c_x(x_t)\}. \tag{20}$$

The task cost is reflected in the model through the exponential term $\exp\{-c_x(x_t)\}$, which can be interpreted as "conditioning on the tasks" in the following sense: We may introduce a binary random variable z_t with $P(z_t = 1 \mid x_t) \propto \exp\{-c_x(x_t)\}$, that is, the probability of $z = 1$ is high when the task costs $c_x(x_t)$ are low in time slice t. The above defined distribution is then the posterior $p(x_{0:T}, u_{0:T}) = P(x_{0:T}, u_{0:T} | z_{0:T} = 1)$. In the LQG case, the costs are quadratic and the controls are linear, we therefore translate the cost terms to a Gaussian motion prior[10]

$$P(z_t = 1 \mid x_t) \propto \mathcal{N}[x_t | r_t, R_t] \tag{21}$$

and the linear controls translate to

$$P(u_t) = \mathcal{N}[u_t | 0, H_t]. \tag{22}$$

We can analytically integrate out the control over time, producing a simpler motion prior

$$\begin{aligned} P(x_{t+1}|x_t) &= \int_u du \mathcal{N}(x_{t+1}|A_t x_t + a_t + B_t u_t, Q_t) \mathcal{N}[u_t|0, H_t] \\ &= \mathcal{N}(x_{t+1}|A_t x_t + a_t, Q_t + B_t H_{-1} B_t^\top) \end{aligned} \tag{23}$$

AICO in general tries to infer the posterior trajectory as the probability distribution $P(x_{0:T} \mid z_{0:T} = 1)$. In Toussaint [2009], this is done using Gaussian message passing (comparable to Kalman smoothing) on a factor graph, based on local Gaussian approximations around the current belief model. Inference on a factor graph (see Figure 3) is a standard backward-forward process, where the posterior marginal belief over a random variable is given by the product of incoming messages. In the

10 The bracket notation $\mathcal{N}[x|a, A] \propto e^{-\frac{1}{2}x^\top A x + x^\top a}$ denoted a Gaussian over x in canonical for with precision matrix A and mean $A^{-1}a$.

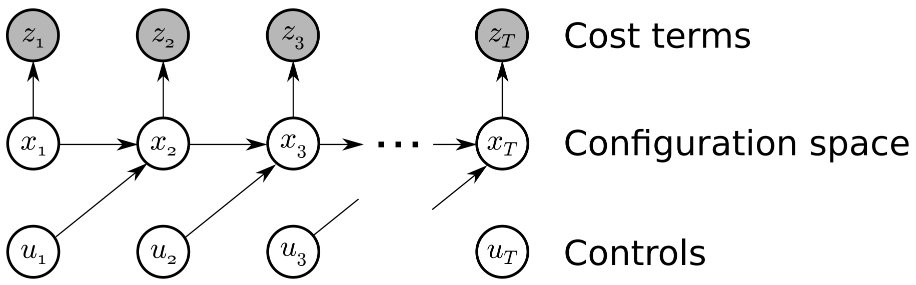


Figure 3: AICO graphical model.

case of the graphical model described by Equation 20 and shown in Figure 3, the belief is computed as

$$b_i(X_i) = \prod_j \mu_{j\to i}(X_i), \tag{24}$$

where we define the following three messages:

$$\mu_{x_{t-1}\to x_t}(x_t) = \mathcal{N}(x_t|s_t, S_t), \tag{25}$$
$$s_t = a_{t-1} + A_{t-1}(S_{t-1}^{-1} + R_{t-1})^{-1}(S_{t-1}^{-1}s_{t-1} + r_{t-1}),$$
$$S_t = Q_t + B_t H_t^{-1} B_t^\top + A_{t-1}(S_{t-1}^{-1} + R_{t-1})^{-1} A_{t-1}^\top,$$
$$\mu_{x_{t+1}\to x_t}(x_t) = \mathcal{N}(x_t|v_t, V_t), \tag{26}$$
$$v_t = -A_t^{-1}a_t + A_t^{-1}(V_{t+1}^{-1} + R_{t+1})^{-1}(V_{t+1}^{-1}v_{t+1} + r_{t+1}),$$
$$V_t = A_t^{-1}[Q_t + B_t H_t^{-1} B_t^\top + (V_{t+1}^{-1} + R_{t+1})^{-1}]A_t^{-\top},$$
$$\mu_{z_t\to x_t}(x_t) = \mathcal{N}[x_t|r_t, R_t]. \tag{27}$$

The messages are then iteratively updated as described in Algorithm 3.

In Rawlik et al. [2012], the theory of the general equivalence of this framework with stochastic optimal control is detailed. Generally, the AICO approach is very similar to differential dynamic programming (Murray and Yakowitz [1984]) or iLQG (Todorov and Li [2005]). In fact, the backward message $\mu_{x_{t+1}\to x_t}(x_t)$ is equivalent to Ricatti equations. To show this, we first define the backward message in canonical form as

$$\bar{V}_{t+1} = V_{t+1}^{-1} + R_{t+1}, \tag{28}$$
$$\bar{v}_{t+1} = V_{t+1}^{-1}v_{t+1} + r_{t+1}. \tag{29}$$

Algorithm 3 AICO $(x_{0:T}, H_{0:T}, A_t(x), a_t(x), B_t(x), R_t(x), r_t(x))$

```
1: $\alpha \leftarrow$ convergence rate
2: $\theta \leftarrow$ time slice stopping threshold
3: Initialise $s_0 \leftarrow x_0$, $S_0^{-1} \leftarrow 1e10$, $v_{0:T} \leftarrow 0$, $V_{0:T}^{-1} \leftarrow 0$, $r_{0:T} \leftarrow 0$, $R_{0:T} \leftarrow 0$
4: $k \leftarrow 0$
5: repeat
6:     for $t = 0$ to $T-1$ do // forward sweep
7:         update $s_t$ and $S_t$ using Equation 25
8:         if $k = 0$ then
9:             $\hat{x}_t \leftarrow s_t$
10:        else
11:            $\hat{x}_t \leftarrow (1-\alpha)\hat{x}_t + \alpha b_t$
12:        access $A_t(\hat{x}_t), a_t(\hat{x}_t), B_t(\hat{x}_t), R_t(\hat{x}_t), r_t(\hat{x}_t)$
13:        update $r_t$ and $R_t$ using Equation 27
14:        update $v_t$ and $V_t$ using Equation 26
15:        update $b_t$ and $B_t$ using Equation 24
16:        if $\|\hat{x}_t - b_t\|^2 > \theta$ then
17:            $t \leftarrow t-1$ // repeat time step
18:    for $t = T-1$ to $0$ do // backward sweep
19:        ... same updates as in forward sweep.
20:    $k \leftarrow k+1$
21: until convergence
```

This allows us to rewrite the backward message using Woodbury identities as

$$\bar{V}_t = R_t + (A_t^\top - K)\bar{V}_{t+1}A_t \tag{30}$$

$$\bar{v}_t = r_t + (A_t^\top - K)(\bar{v}_{t+1} - \bar{V}_{t+1}a_t), \tag{31}$$

$$\tag{32}$$

where

$$K = A^\top \bar{V}_{t+1}^\top (\bar{V}_{t+1} + B_t^{-\top} H_t B_t^{-1})^{-1}, \tag{33}$$

which is equivalent to the Ricatti equations 14-17. Although the backward updates in AICO and LQG methods are equivalent, there is no counterpart for the forward messages ("cost-to-reach functions") in the LQG case. The forward message is, however, necessary for computing proper posterior marginal belief. The similarities and differences between AICO and other methods like iLQG and DDP are explained in more detail in Rawlik et al. [2012] and Toussaint [2009].

2.2.2 *Expressing motion priors in task spaces*

To estimate the posterior, the controls u_t can be marginalized, implying the following *motion prior*:

$$P(x_{t+1} \,|\, x_t) = \int_{u_t} P(x_t \,|\, u_{t\text{-}1}, x_{t\text{-}1})\; P(u_t)\; du_t \;. \tag{34}$$

This motion prior arises as the combination of the system dynamics and our choice of control costs $c_u(u_t)$; for LQ systems it is a linear Gaussian.

The motion prior is a unique view on our motivation for abstract representations. In the introduction, we mentioned the impact of representations on the sampling strategy, the metric, or the topology. In other terms, successful trajectories are likely to be "simpler" (easier to find, shorter, local) in an appropriate space. In Machine Learning terms, this is expressed in terms of a prior. In this view, task spaces are essentially means to express priors about potentially successful trajectories—in our case we employ the linear Gaussian prior in an abstract space to express the belief that trajectories may appear "simple" in such space.

However, using AICO with a linear Gaussian motion prior in topology space is not sufficient to solve general motion synthesis problems: 1) The computed posterior in an abstract task space does not directly specify an actual state trajectory or control law at the joint level. 2) We neglect the problem of minimization of control and task costs originally defined at the joint level. To address these issues, we need mechanisms to couple inference in task space and state space. We do so by coupling task and joint state representations in AICO's graphical model.

Figure 4 displays the graphical model from Equation 20. The bottom layer corresponds to the standard AICO setup, with the motion prior defined in Equation 34 implied by the system dynamics and control costs. Additionally it includes the task costs represented by $P(z_t = 1 \,|\, x_t) = \exp\left(-c_x(x_t)\right)$. The top layer represents a process in task space with a prior given as a linear Gaussian motion prior $P(y_{t+1}|x_t)$. Both layers are coupled by introducing additional factors

$$f(x_t, y_t) = \exp\left(-\frac{1}{2}\rho_t \|y_t - \phi(x_t)\|^2\right), \tag{35}$$

which essentially aims to minimize the squared distance between the task reference[11] y_t and the task coordinate computed from the joint configuration $\phi(x_t)$,

[11] The task reference represents the desired task space trajectory. The task reference is often set by interpolating between start task state $\phi(q_0)$ and the end-state y^*.

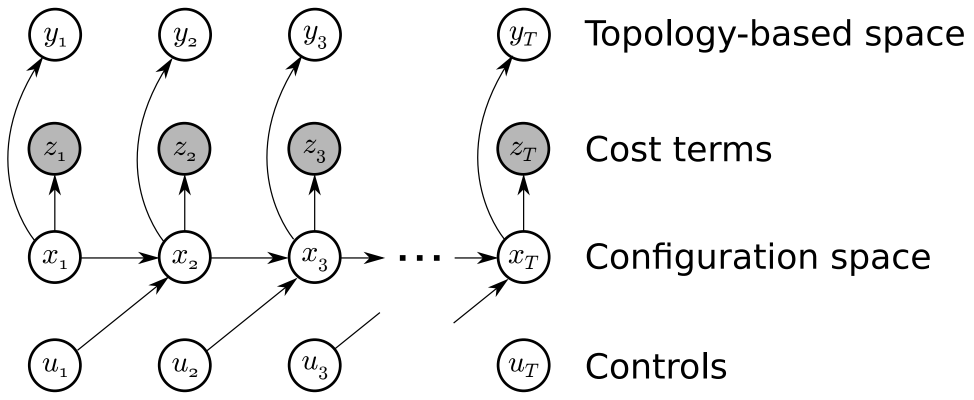


Figure 4: Coupling of AICO trajectories in the configuration space and in the topology-based space. We formally define the *topology-based space* in Chapter 3.

weighted by a precision constant ρ. The precision constant may also be indexed on time. This is useful for certain tasks like grasping, where for example: collision should be avoided during the reaching motion but contacts involving collision are necessary during the actual grasp. Note that using a local linearisation of ϕ (having the Jacobian J_t of the task space[12]) is sufficient for Gaussian message passing between the task and configuration layer of the graphical model which is why we define the cost terms[13] from equation Equation 21 as

$$R_t = \rho_t J_t^\top J_t, \tag{36}$$

$$r_t = \rho_t J_t^\top (y_t - \phi(x_t) + J_t x_t). \tag{37}$$

If there are multiple weighted objectives, the resulting matrix R and vector r can be obtained by accumulating (summing) the weighted contributions from each of these tasks. These factors essentially treat the abstract task space configuration y_t as an additional random variable for the lower level inference. In our setup, we generally distinguish between configuration space, task space and abstract spaces. While the task space generally describes a space sufficient to describe cost or rewards the role of the abstract spaces is to provide alternative metrics (and topology) for trajectory optimization. More precisely, we may define a task space as a

12 Here we use the shorter notation $J_t = J(x_t)$.
13 When solving the generic LQG problem, we can use equations 36 and 37 to compute the cost terms in equations 14-17 as well.

projection of configuration space for which we apriori know that costs (other than transition costs) depend only on this task space.

PLANNING IN ALTERNATE SPACES The choice of task spaces and their weighting depends on the task — in our experiments, these are chosen manually. When combining tasks in this way, the precision constant ρ serves the purpose of a mixing factor, allowing one prior to have more effect than another. When the priors are complementary, the convergence is improved. For example, using classical representations, we can define a reaching task for a redundant robot and resolve the redundancy using a null space motion which keeps the robot pose close to a comfortable position. Here, the reaching task and comfortable pose task would be complementary.

It is, however, possible to arbitrarily combine abstract representations as well. For example, if the task requires both, preserving the nature of interaction as well as planning a wrapping motion, as we proposed in Zarubin et al. [2012]. If, however, the priors are contradictory, they may cancel each other out. An example of this effect would be trying to reach for two objects located at opposite sides of the manipulator at the same time.

In our experiments, we build only two layers of hierarchy between the configuration space and the abstract or classical task spaces. Since the abstract spaces provide simple priors that capture the task well, there is no need for a more structured graphical model. It is, however, possible to build more complex hierarchies by extending the graphical model.

2.2.3 *AICO initialisation for re-planning in dynamic environments*

The AICO algorithm, as proposed in Toussaint [2009], was designed for solving finite time horizon problems on linearly approximated system models with discrete time. The work proposed in Rawlik et al. [2010] extended this algorithm to allow for time optimisation.

One of the strengths of the AICO algorithm is that the convergence speed can be greatly improved by providing a good initialisation of the graphical model. In Rawlik et al. [2012], the authors have used a hierarchy of graphical models of different resolution of the time discretization. The optimisation starts by solving the problem at a low granularity. This is essentially a one step problem. The process then continues by using this solution to initialise the next problem with higher resolution. In practice, the computational speed is decreased even though multiple problems have to solved sequentially.

We propose another way of reusing previous graphical models to initialise a new problem for scenarios with dynamically moving obstacles where replanning is necessary. We initially solve the whole problem given current knowledge of the state of the environment. When the environment changes due to unpredicted movement of obstacles, we re-plan the trajectory, this time using the graphical model used for computing the initial solution as initialisation of the new problem. Since we still assume we are solving a finite horizon problem, we truncate the graphical by removing the part of the model that corresponds to the trajectory that the robot has already executed. The remainder of the trajectory is then optimised using the updated information about the state of the environment.

The AICO algorithm provides the flexibility to solve motion planning problems using alternate representation. We have, however, exploited the AICO framework to re-plan paths for a mobile robot by initialising the probabilistic model using beliefs computed for the previous plan that became invalid due to changes in the environment. The aim of this experiment is to show that our system plans and re-plans collision-free paths in dynamic environments, reaching the goal configuration optimally with respect to energy consumption.

In this experiment we use the following cost function:

$$c(x_t) = \rho \|x^* - x_t\|^2 + \rho_t^{\text{coll}} \|\phi_{\text{coll}}(x_t)\|^2 \,, \tag{38}$$

where x^* is the desired goal location and x is the current robot position. We don't use the task mapping $\phi(x)$ for reaching the goal in this case because we are planning directly in the configuration space. We assume that the configuration space is continuous but we construct a grid representation to record the position of obstacles more efficiently. The collision cost term $\phi_{\text{coll}}(x)$ then utilises this occupancy grid to compute the reciprocal distance to the closest obstacle or occupied grid cell in this case. AICO is initialized using the path computed from the Voronoi graph of the environment including only static obstacles. We then use AICO to compute the initial optimal path from start to goal positions (see Figure 5). Starting and goal positions are marked by the green and red dots respectively. The covariance ellipses are overlaid.

Equation 38 shows the cost function that we aim to minimize but the cost function does not, actually, appear within the AICO algorithm in this form. The cost terms are represented using motion priors instead. The two cost terms in Equation 38 are $\rho\|x^* - x\|^2$ and $\rho_{\mathrm{coll}}\|\phi_{\mathrm{coll}}(x)\|^2$. We substitute these terms into Equation 36 and Equation 37. These equations will will then take the following form:

$$R_t^x = \rho I, \tag{39}$$

$$r_t^x = \rho I(x^* - x_t + Ix_t), \tag{40}$$

$$R_t^{\mathrm{coll}} = \rho_t^{\mathrm{coll}} J_t^{\mathrm{coll}\top} J_t^{\mathrm{coll}}, \tag{41}$$

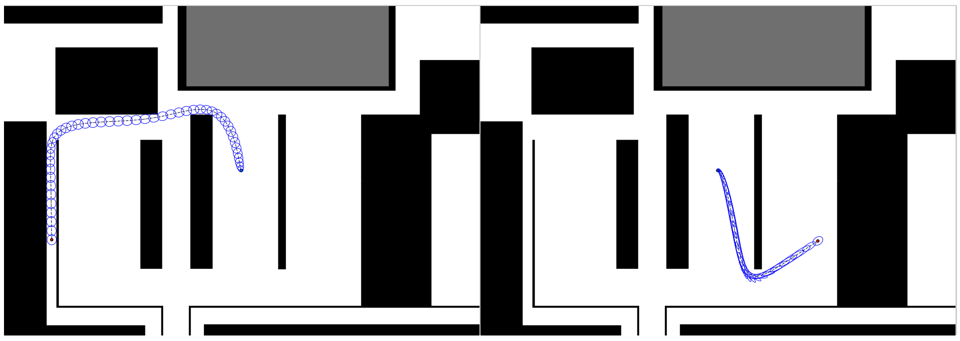
$$r_t^{\mathrm{coll}} = \rho_t^{\mathrm{coll}} J_t^{\mathrm{coll}\top}(-\phi_{\mathrm{coll}}(x_t) + J_t^{\mathrm{coll}} x_t). \tag{42}$$

Since the first term computes cost in the configuration space, there the task map is equal to the robot state $\phi(x_t) = x_t$ and the task Jacobian is an indentity matrix $J_t = I$ (see Equations 39 and 40). The second task term aims to minimize the collision cost, which is why we set task space reference (which is also the desired goal cost) to zero $y_t^{\mathrm{coll}} = 0$ (see Equation 42). Most task cost terms will include a task map and a task space reference, in which case the equations will have the same form as Equation 36 and Equation 37. We compute the mean and covariance of the task cost prior by summing the individual task terms $R_t = R_t^x + R_t^{\mathrm{coll}}$ and $r_t = r_t^x + r_t^{\mathrm{coll}}$. The resulting task terms R_t, r_t are then used to define motion priors within the AICO framework as described in Algorithm 3. Throughout this thesis, we choose to write the down the cost terms in form of a cost function (such as in Equation 38) because it is much easier to identify the individual terms that we use for optimisation. However, when we implement this cost function, we actually define motion priors analogously to the ones we defined in equations 39 to 42.

In order to compare our method to the classical obstacle avoidance algorithms in terms of energy, we only take into account the power demand of the robot's motors. We assume that the rest of the equipment has constant energy consumption and therefore, it cannot be improved any further. We also assume that the power demands of the robot's motors are based on overcoming inertia, road grade, tyre friction, and aerodynamic loss. This road-load methodology was mainly introduced by Sovran and Bohn [1981]. The power demand (in Watts) is the tractive power as defined below:

$$P = mv[a(1+\varepsilon) + gR_G + gK_R] + \frac{1}{2}\rho K_D A_F v^3 \tag{43}$$

where m is vehicle mass in metric tones ($0.077\mathrm{kg}$ in our case), v is vehicle speed (assuming no headwind) in ms^{-1}, a is vehicle acceleration in ms^{-2}, ε is a mass factor accounting for the rotational masses, is assumed to be 0.1 (estimated according

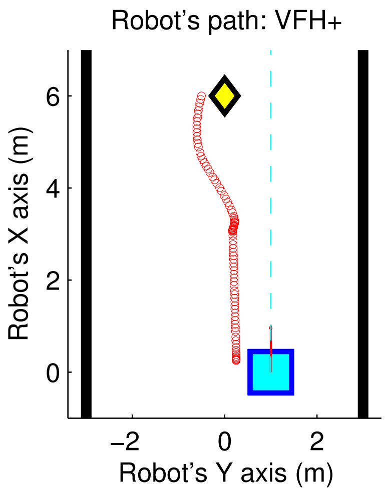


(a) Goal in a narrow corridor (b) Goal in an open area

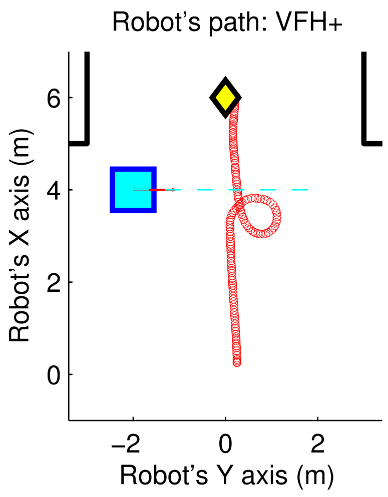
Figure 5: Two examples of initial optimal paths computed using AICO in a static environment.

to Palacios [1999]), g is acceleration due to gravity ($9.8m/s_{-2}$), R_G is road grade (0.0 in our case), K_R is rolling resistance – this value for radial tires can range from 0.008 to 0.013 for a majority of the on-road passenger car tires but can be larger depending on tire pressure, temperature, ground surface, and speed according to GmbH [2004] and Gillespie [1992] (a medium value in the range ≈ 0.009 is estimated based on Sovran and Bohn [1981]), ρ is air density ($\approx 1.2kg/m^3$), K_D is aerodynamic drag coefficient (≈ 0.3 Sovran and Bohn [1981]) and A_F is the frontal area ($\approx 1m^2$ in our case).

We have used the probabilistic model of the environment we proposed in Llamazares et al. [2012] to detect the position and velocity of dynamic obstacles in the robot coordinate frame which we have then mapped into the global coordinate frame. We use this information to predict the movement of these obstacles. AICO is then used to compute the optimal trajectory around the initial path while using the probabilistic model predictions about the dynamic obstacles. The following set of task variables has been used to define the optimality: distance to goal, power demand (Equation 43), turning velocity and collision avoidance. The collision avoidance is achieved by inferring cost for reciprocal distance to the closest obstacle. Inference-based path planning with the linearised motion model and the holonomic constraint is difficult and suffers from problems with local minima due to the velocity constraints. The reasoning behind this is that the Gaussian distribution over the state space can potentially assign probability mass to states that do not satisfy the holonomic constraint which either causes sideways slipping in the model or if we constrain the Gaussian itself the distribution becomes degenerate. For this reason, we have excluded the orientation from the state and we have added an additional cost term to penalise for angular velocity instead. We



(a) Parallel scenario



(b) Perpendicular scenario

Figure 6: Parallel and perpendicular scenarios including dynamic moving obstacles used for experiments. The goal is yellow diamond, the vehicle crossing the robot's path is a blue square and the robot trajectory is red. Figure courtesy of Ángel Llamazares.

assume that arbitrary angular velocities can be executed but optimise for low angular velocities. This reduces the complexity of the state space by turning the hard holonomic constraint into a soft constraint.

AICO works under the assumption that the full state of the world, including the motion of the obstacles is known. This is, however, no longer true when the prediction made by the probabilistic model that we proposed in Llamazares et al. [2013] is inaccurate. Since we keep updating this model in real-time, we can detect when the original prediction diverges from the actual state. We therefore update our prediction using the new observations and discount the occupancy probability over time. We re-plan the path if the prediction error reaches a threshold. The feedback loop between the planner and the sensing model therefore behaves similarly to a Kalman filter.

We only expect small changes of the environment between two time steps. In such situations AICO requires only a small number of iterations to converge. This makes re-planning computationally affordable. The replanning time (3 s on average) is however too long to be used as a reactive controller.

In our experiment, we compare the energy consumption on two scenarios (see Figure 6):

- **Parallel:** The robot is driving along a straight corridor, trying to reach a goal position directly in front of it. Another vehicle representing an obstacle will then drive towards the controlled robot at a higher speed and overtake it, creating a dangerous situation with possible collisions.
- **Perpendicular:** The robot is driving through a crossing, trying to reach a goal position directly in front of it. Another vehicle will cross its path from the side.

The task is in both cases to reach the goal safely, without colliding with the second vehicle. This class of problems is typically solved using reactive controllers. We have compared our method with the following state of the art methods: VHF+ (Ulrich and Borenstein [1998]), CVM (Fox et al. [1997]), LCM (Ko and Simmons [1998]) and BCM (Fernández et al. [2004]). Our methods uses the probabilistic sensor model we proposed in Llamazares et al. [2013]. Table 1 shows that using the probabilistic sensor model reduces the energy consumption of each the tested algorithms. The baseline for these experiments was computed using the raw sensor data from the laser distance scanner. We then show that our method using AICO and the probabilistic model is capable of further improving the performance of the best performing reactive method in each corresponding scenario.

AICO solves the finite horizon optimisation problems which means that the duration of the trajectory needs to be specified a priori. It is not within the scope of this work to optimise for time, we have therefore set the trajectory duration to the respective average durations as computed using the reactive methods with our probabilistic sensing model.

This work shows that we can reuse previous solutions of the the AICO algorithm to improve convergence during the subsequent runs. We, however, want to approach the replanning from a different angle. In the next chapter, we will introduce novel representations that will allow us to compute plans that are robust under perturbations and geometric changes in the environment. We then use these plans and *re-map* the trajectory in the abstract space into a trajectory in the configuration space. This process replaces the costly re-planning by a more efficient re-mapping at a cost of local approximation.

	Prallel scenario		Perpendicular scenario	
Raw laser data	CVM	0.2261W	BCM	0.1659W
	VHF+	0.2140W	VHF+	0.1249W
	BCM	0.2054W	CVM	0.1045W
	LCM	0.2012W	LCM	0.0909W
Probabilistic model	CVM	0.2091W	CVM	0.0999W
	LCM	0.2070W	BCM	0.0904W
	BCM	0.1904W	VHF+	0.0883W
	VHF+	0.1743W	LCM	0.0835W
	AICO	**0.1455W**	AICO	**0.0789W**

Table 1: Summary of energy consumption results. AICO using the probabilistic sensing model outperforms all reactive methods in both parallel and perpendicular scenarios.

3 TASK REPRESENTATION

The result of motion planning is a trajectory in configuration space that can be executed on the robot. Although, this trajectory is constructed in the configuration space, we often use other metrics in alternate task spaces (as defined in Equation 8) during motion planning. Lets assume that every trajectory in configuration space can be mapped into an arbitrary task space. This task space may have very different properties and therefore the trajectory in this space will have a very different shape. Figure 7 illustrates the use of a task space where a collision free trajectory is easier to find than in the configuration space. In fact, a simple interpolation in this task space produces a collision free trajectory in this hypothetical case. Therefore, the way how we represent the task can have a big impact on motion planning.

There are several formal views on the implication of an alternate task representation: 1) In the context of randomized search, such representations alter the Voronoi *bias* or more generally the sampling strategy and therefore, the efficiency of rapidly exploring random trees (RRT) or probabilistic road maps (PRM). Lindemann and LaValle [2004] demonstrate this effect in the case of RRTs. 2) An alternate representation may imply a different *topology*, such that a trajectory that is a complex path in one space becomes a simple geodesic in another. 3) An alternate representation may change the metric of the space, such that local *optimization* in one space is sufficient for finding a solution whereas global optimization (randomized search) would be needed in the other. 4) Finally, different representations may allow us to express different *motion priors*, for instance, a prior preferring "wrapping-type motions" can be expressed in one space as a simple Brownian motion or Gaussian process prior, whereas the same Brownian motion prior in configuration space renders wrapping motions extremely unlikely.

There are task representations that are defined purely based on the robot model. However, many tasks require additional parameters such as positions of obstacles or shapes of colliding objects. We assume that these parameters may be indexed on time but they are not a function of the robot configuration. We define a notion of a *scene* to be the collection of the robot model description and all parameters

Scene

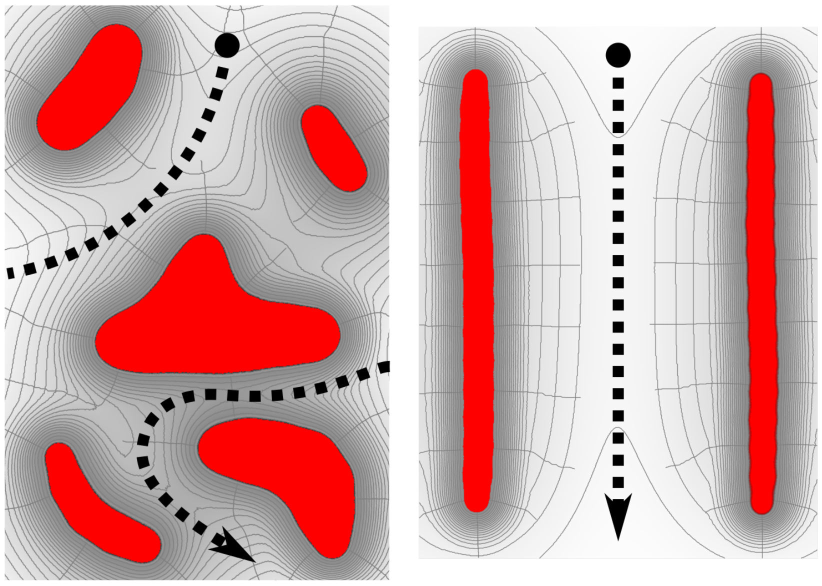


Figure 7: Illustration of a trajectory in the configuration space (left) and a the same trajectory mapped into a task space (right) with different metrics and topology. Obstacles are red and the trajectory is a black dashed line.

external to the robot which are required for computing the task representations. We then define the task representation as:

$$\phi(x_t, \alpha_{\text{scene}}) : X \to Y, \tag{44}$$

where x_t is the robot configuration and α_{scene} is the set of parameters external to the robot[1] at time t. In the robotics literature, these parameters usually get treated as part of the work space and won't get modelled explicitly or they get added into the configuration space as unactuated degrees of freedom. In both cases, the motion synthesis algorithm gets presented with task representation as defined in Equation 8, which makes the separation of the robot configuration and the scene parameters impractical. Such separation is useful for explicitly modelling the interaction between the robot (x_t) and the scene (α_{scene}). The parameter α_{scene} is treated as a constant rather than as a variable within the function $\phi(x_t, \alpha_{\text{scene}})$. This allows us to use a task map in the same way, as we defined it in Equation 8 but to model the parameter explicitly at the same time. We will now present several task representations that model these interactions explicitly and group them based on the type of problems they are most suitable to be applied to.

1 In Appendix A, we describe a *kinematic scene* which implements this functionality. Although the scene may contain parameters that are not kinematic in nature, such as external forces, all of the tasks that we use are based on kinematic state of the robot and other objects in the scene.

WINDING AND WRITHING Choosing correct task representation for a given problem is a crucial part of designing motion synthesis techniques. For example, in order to cope with problems of close interactions, we can represent the task using spatial relations between the body parts and objects. Multiple tasks involve wrapping and winding motion, e.g. untangling knots, wrapping fingers around a handle or passing arm through a sleeve[2]. Several representations have been proposed to solve this type of problems. For representing knots, Dowker and Morwen [1983] proposed describing the configuration of a 3D strand based on its overlap with itself, when viewed from a specific direction. This approach has been later used by Takamatsu et al. [2006], Wakamatsu et al. [2006], Matsuno et al. [2006] and Saha and Isto [2007] for planning knotting motion. The obvious disadvantage of such representations is its view-dependence and the difficulty in generating commands to actually manipulate the elements of the string.

These problems have been addressed by Bhattacharya et al. [2010] where the authors proposed using winding numbers to compute homotopy classes of paths to classify trajectories in 2D configuration spaces. Further work by Bhattacharya et al. [2011] extended this work to classifying 3D paths using the Ampere's law. This work, however, stops at classifying configuration space paths into homotopy groups, which limits this approach to systems with 3 DoF. Ho et al. [2010a] proposed to use the same representation for describing interactions between skeletons of robots and other (possibly articulated) objects in their workspace. This representation called *topology coordinates* is based on the Gauss Linking Integral and it is suitable for representing tangling motion. In Tamei et al. [2011], the same representation is applied for controlling the movement of a robot that puts a shirt on a human. This type of winding and writhing motion is suitable for manipulating string like objects and controlling the skeletons of robots. Some tasks, however, require interaction between the surface of the robot and the object, not just between their skeletal approximations. Such representations are useful for planning grasping motion.

ENVELOPING AN OBJECT Grasping is an active area of research within robotics and many methods for achieving stable grasps under various constraints have been proposed in the literature. These approaches have been successfully applied to solve the grasping problem but they encode the interaction with the grasped object through grasping primitives such as spheres and cylinders (Miller et al. [2003]),

2 Passing an arm through a sleeve is equivalent to wrapping the arm around the inside of the edge of the sleeve (see Figure 28).

a proxy object (Gioioso et al. [2012]) or fingertip positions (Peer et al. [2008]). Another class of techniques uses learning from demonstration (Kang and Ikeuchi [1994]) or the hand is controlled via teleoperation (Haiying et al. [2005]) to achieve successful grasping motion. Common interactions between the hand the object, such as the power grasps, usually involve enveloping the object with the surface of the hand. In Sandilands et al. [2013b], we proposed a representation based the electrostatic field of the virtually charged manipulated object. This method utilises electric flux to compute how much does the surface of the hand envelop the object. The same electrostatic field can be also used to define more generic metrics.

GENERIC INTERACTION REPRESENTATIONS The electrostatic field can be probed at different points around the virtually charged object. In Sandilands et al. [2013b], we also proposed to use the field potential to create an object centric coordinate system called *electric coordinates*. The potential can be tracked along the electric field gradient to the surface of the object, which makes it possible to compute properties, such as finger tip position on the object and force closure, even when the hand is not in contact with the object[3]. This representation is useful for grasping objects, but it is requires the knowledge of the exact shape of the object. Many interaction with the environment can, however, be done at a more coarse level by preserving the spacial relationships with objects surrounding the robot.

Another representation called *interaction mesh*, that captures such spatial relationship, was introduced by Ho et al. [2010b]. The relationship is quantified by the Laplacian coordinates of the volumetric mesh whose points are sampled on the links of the robot and the objects in the scene. This methods is compatible with the definition of the task representation as defined in Equation 44. However, it is a discontinuous representation which is only valid in the neighbourhood of the posture from which the volumetric mesh is computed. This is due to the way the connectivity of the mesh is defined.

All these representations share a common property, which is, that they can render two very different motions that involve complex interaction as very similar to each other in a analogous to the way that the topology invariants render different geometrical shapes as topologically equivalent. In the context of this thesis, we define the term *topology-based space* as any space, that in general abstracts away the

Topology-based space

3 The force closure is computed using the fingertip positions as query points in the electrostatic field with the assumption that these points will become contact points when the hand gets closer to the object. See Section 3.3 for more details.

geometric detail of the work space. Not all the topology-based spaces, therefore, strictly relate to a space with a novel topology — most of these spaces are in fact $\mathbb{R}^n$ with the usual topology and they can be classified as universal covering spaces of the joint space (Munkres [2000]). For example, the writhe scalar described in Section 3.1 is a homology invariant (see Section 3.5) but the interaction mesh space defined in Section 3.6 is a metric space designed to represent interaction between objects by preserving their relative spacial relationships.

As opposed to the computer animation domain, where topology-based representation have recently been used (Ho et al. [2010b]), synthesizing motion in such abstract spaces for planning and control of robotic systems comes with additional challenges. Typically, control tasks are specified in world (or end-effector) coordinates, the obstacles may be observed in visual (or camera) coordinates, and the joint limits of the actuators are typically described in configuration space. Therefore, the general challenge is to devise motion synthesis methods that combine the benefits of reasoning in topology-based coordinates while preserving consistency across the control coordinates and managing to incorporate dynamic constraints from alternate representations seamlessly. This is where the flexibility of the AICO framework plays a crucial role. The graphical model can be easily extended with new random variables as we discussed in Section 2.2.2. We will now introduce several task representations that fit within this framework and create motion priors with an interesting metric or topology.

The contributions of this chapter are:

- Applying and combining representations that directly model the interaction between the robot and its environment. These include: winding number, writhe, electrostatic flux, electrostatic coordinates and interaction mesh.
- A set of *experiments* comparing novel and classical representations using state of the art motion planning methods (AICO, RRT*).
- Experiments showing utility of motion planning using winding numbers for folding motion in 3D and navigation with winding constraints in 2D.
- Electric field based representations suitable for transferring reaching and grasping motion.
- Validation that the proposed representations are simple enough so that local optimisation methods such as AICO can be used.
- Experiments showing grasp transfer/teleoperation with different bias/style.

- Extension of interaction mesh representation by adding per-edge *weighting*.
- Application of electric flux and electrostatic coordinates to problems of *grasping and grasp transfer*.
- A method for real-time motion *generalization* (re-mapping) to novel situations using abstract representations and operational space control.

3.1 WRITHE REPRESENTATION

The writhe is a property of the configuration of two kinematic chains (or in the continuous limit, of two strings). Intuitively the writhe describes to what degree (and how and where) the two chains are wrapped around each other, which is a well suited property for representing winding and wrapping motion (see Klenin and Langowski [2000]).

Let us describe two kinematic chains by positions $\vec{p}^{\,1,2}_{1:K}$ of their joints, where $\vec{p}^{\,i}_{k} \in \mathbb{R}^3$ is the kth point of the ith chain. Using standard kinematics, we know how these points depend on the configuration $x \in \mathbb{R}^n$, that is, we have the forward map $\phi^i_k(x)$ and Jacobian $J^i_k := \frac{\partial \vec{p}^{\,i}_k}{\partial x}$ for each point. The writhe is a function of the link positions $\vec{p}^{\,1,2}_{1:K}$. More precisely, the *writhe matrix* W_{ij} describes the relative configuration of two points $(\vec{p}^{\,1}_{i}, \vec{p}^{\,1}_{i+1})$ on the first chain and two points $(\vec{p}^{\,2}_{j}, \vec{p}^{\,2}_{j+1})$ on the second chain where i, j are indexes of points along the first and the second chain respectively. For brevity, let us denote two consecutive points on the first chain by $(\vec{a}, \vec{b}) = (\vec{p}^{\,1}_{i}, \vec{p}^{\,1}_{i+1})$ and similarly $(\vec{c}, \vec{d}) = (\vec{p}^{\,2}_{j}, \vec{p}^{\,2}_{j+1})$ on the second chain (see Figure 8). We then define the *writhe* using the task map $\phi_{\text{writhe}}(x)$ which computes writhe matrix

Writhe matrix

$$W_{ij}=\left[\sin^{-1}\frac{\vec{n}_a^\top \vec{n}_d}{|\vec{n}_a||\vec{n}_d|}+\sin^{-1}\frac{\vec{n}_b^\top \vec{n}_c}{|\vec{n}_b||\vec{n}_c|}+\sin^{-1}\frac{\vec{n}_c^\top \vec{n}_a}{|\vec{n}_c||\vec{n}_a|}+\sin^{-1}\frac{\vec{n}_d^\top \vec{n}_b}{|\vec{n}_d||\vec{n}_b|}\right] \operatorname{sign}\left[\vec{ab}^\top(\vec{ac}\times\vec{cd})\right] \tag{45}$$

where $\vec{n}_a, \vec{n}_b, \vec{n}_c, \vec{n}_d$ are normals at the points $\vec{a}, \vec{b}, \vec{c}, \vec{d}$ with respect to the opposing segments, defined as

$$\vec{n}_a=\vec{ac}\times\vec{ad}\ ,\ \ \vec{n}_b=\vec{bd}\times\vec{bc}\ ,\ \ \vec{n}_c=\vec{bc}\times\vec{ac}\ ,\ \ \vec{n}_d=\vec{ad}\times\vec{bd}\ . \tag{46}$$

The above equations compute the Gauss linking integral (GLI) along two segments. The solution of this integral is based on an analogy with the solid angle formed by

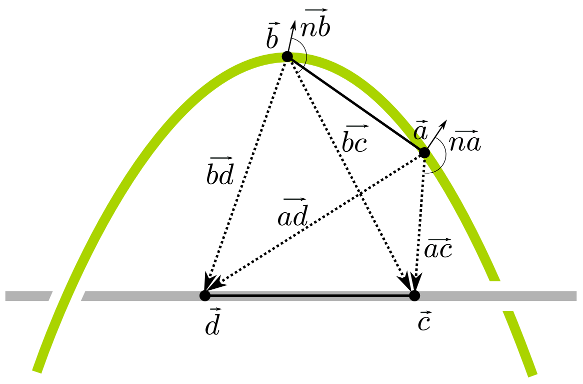


Figure 8: Illustration of the definition of writhe for two segments. $\overrightarrow{ab}$ belongs to the manipulator skeleton (green curve) and $\overrightarrow{cd}$ is a part of the obstacle skeleton (gray line).

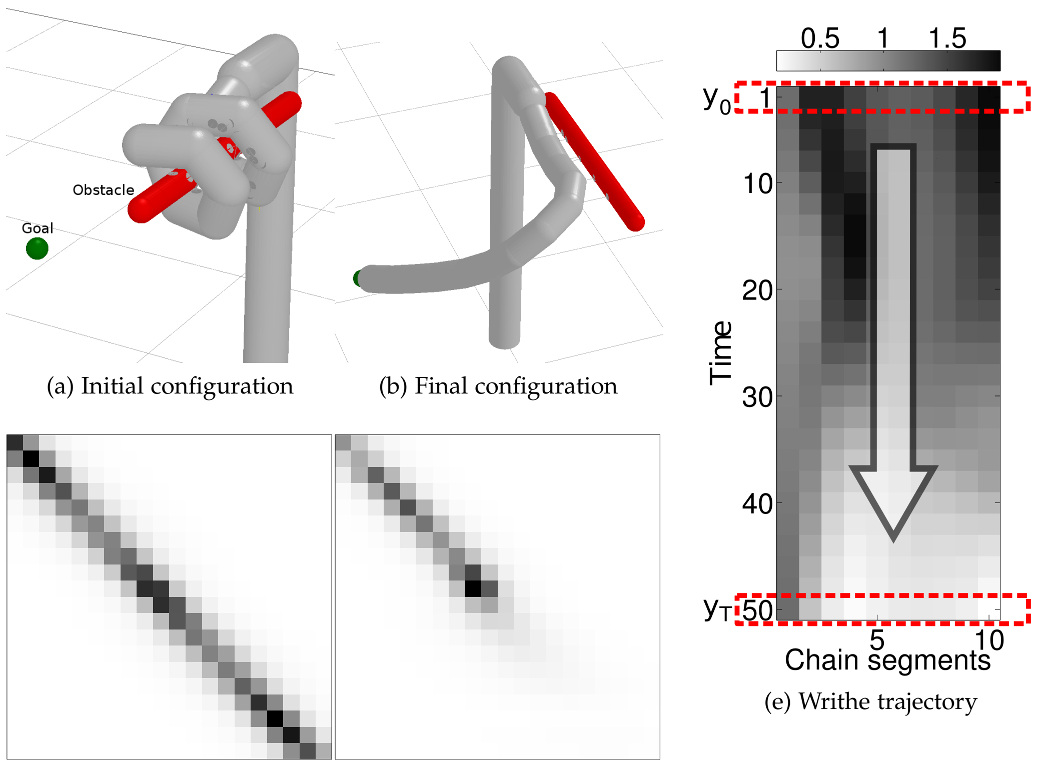
all view directions in which segments $\overrightarrow{ab}$ and $\overrightarrow{cd}$ intersect multiplied by an appropriate sign (Klenin and Langowski [2000]). Since the writhe matrix is a function of the link positions $p^{1,2}_{1:K}$ we can compute its Jacobian using the chain rule[4].

Figure 9 illustrates two configurations together with their writhe matrix representation. The amplitude of the writhe (shading) along the diagonal illustrates which segments are wrapped around each other. We can derive simpler metrics from the full writhe matrix, usually by summing over writhe matrix elements. For instance, the Gauss linking integral, which counts the mean number of intersections of two chains when projecting from all directions, is the sum of all elements of the writhe matrix. In our experiments, we will also use the vector $\phi^j_{\text{writhe}}(x) = w_j = \sum_i W_{ij}$ as a representation of how much the links of the robot wrap around the entire skeleton of an obstacle.

3.1.1 *Planning winding motion with Writhe*

The utility of the writhe representation is in scenarios where winding motion is required. In the following experiment, we will demonstrate how motion planning with a prior in the writhe space improves the performance of both exploratory and optimisation-based planners. We simulate a generic manipulator consisting of 10

4 See Appendix B.2 for full derivation of the writhe.



(a) Initial configuration

(b) Final configuration

(e) Writhe trajectory

(c) Initial writhe matrix

(d) Final writhe matrix

Figure 9: The experimental task is to grasp the object without collisions. Corresponding writhe matrices (c, d) are depicted below the configurations (a, b) - the darkness represents the amplitude of the writhe value. Each row in writhe space evolves over time as shown in (e). The red boxes indicate the initial and final time slices.

segments connected together using 20 revolute joints[5], making 20 DoF in total. Initially, this snake-like manipulator is coiled two times around a pole-shaped obstacle, giving us approximately 720° of writhe density (See Figure 9a). The task is to reach for the target rendered as a green sphere and avoid colliding with the red cylindrical obstacle (Figure 9b show the desired final configuration). Figure 9e illustrates an example of a unwrapping trajectory in the writhe space when all rows of the writhe matrix are summed up into one column to representing the configuration more compactly. This compact representation can be justified when the skeleton of the obstacle has an uniform shape, such as the red pole in our experiments.

5 Each pair of segments is connected through 2 revolute joint with axes perpendicular to each other. Each pair of joints simulates a ball-socket joint.

AICO WITH CLASSICAL REPRESENTATIONS We first solved the problem using AICO and classical representations. In this case, we constructed the following cost function (as the state cost part of the overall cost function defined in Equation 19)

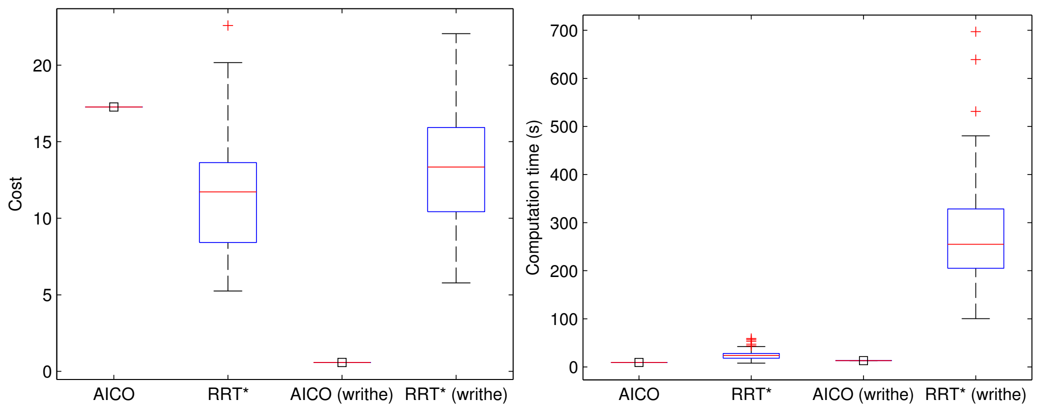
$$c(x_t) = \rho_{\text{eff}} \|y_{\text{eff}} - \phi_{\text{eff}}(x_t)\|^2 + \rho_{\text{coll}} \|\phi_{\text{coll}}(x_t)\|^2 \tag{47}$$

using a combination of the end-effector position cost term $\|y_{\text{eff}} - \phi_{\text{eff}}(x_t)\|^2$ and a collision cost term $\|\phi_{\text{coll}}(x_t)\|^2$. Equation 47 is a cost function in the quadratic form as the defined in Equation 35. The end-effector position cost is the distance to the reaching target (the red sphere). We use $\phi_{\text{eff}}(x_t)$ to compute the end-effector position using forward kinematics. To achieve a smoother trajectory, we created a reference trajectory y_{eff} for the end-effector by interpolating between the initial and target position in the work space coordinates. The collision cost $\phi_{\text{coll}}(x_t)$ was computed as the reciprocal distance between the robot body and the closest obstacle when the obstacle is closer than a safety distance. The collision cost term is ignored if the obstacles are too far to cause collisions.

We combined the two cost terms by manually assigning relative weights ρ_{eff} and ρ_{coll}. Then, we attempted to compute an optimal trajectory[6] using AICO. Our first attempts failed because the initial large amount of coiling of the robot around the obstacle creates a deep local minimum which caused AICO to get stuck. However, we managed to fine tune the task weighting parameters ρ to produce a successful trajectory. The resulting weights gave high importance to the collision cost term throughout the whole trajectory. Then, we weighted the end-effector position term using ρ_{eff} an order of magnitude lower than ρ_{coll}, except for the last time step which was then weighted higher than the collisions cost term. This is a very specific weighting scheme that was fine-tuned for this particular winding problem. Since this was the only cost function that allowed us to solve the problem using AICO and the classical representation, we have used this cost function to also evaluate trajectories computed using the competing methods.

RRT* WITH CLASSICAL REPRESENTATIONS The local minima that significantly affects the local trajectory optimizers has a very different effect on exploratory methods. In our second scenario, we used the RRT* algorithm (see Section 2.1 which explains the modifications to RRT necessary for finding asymptotically optimal paths) to solve the winding problem. We used collision detection for rejecting samples that were in collision with the obstacle and we used the end-effector po-

6 We have time discretized the trajectory using 50 time steps.



(a) Cumulative cost (b) Computation time

Figure 10: Comparison of AICO and RRT* algortims on an unwrapping task with and without using the writhe representation. In case of RRT* we show statistics over 100 runs. The cumulative cost is computed using Equation 47 for all trajectories.

sition cost term and the collision cost term from Equation 47 as the optimisation criteria for RRT*. We have additionally used the end-effector cost term to bias the sampling as proposed by Shkolnik and Tedrake [2009]. We manually tuned the ratio between random exploration and movement toward the goal to minimize the computation time. Figure 10 shows the comparison of both the total cost and the computation time of RRT* and AICO. The cumulative costs of trajectories computed using RRT* are slightly lower than the cost of the trajectory computed using AICO but the AICO trajectory performs very similar to the RRT* trajectory, showing that both methods have computed trajectories close to the global optimum. The computation time that the RRT* algorithm required was slightly higher than the time required by AICO (see Figure 10b).

The RRT* algorithm computes the optimal trajectory as the number of samples tends to infinity. In practice this means that we have to limit the amount of time RRT* is allowed to use to improve the trajectory. To make the computation time comparison more objective, we report the time RRT* required for finding the feasible solution[7]. We, however, continue the sampling process until the trajectory cost converges based on the same termination criteria as AICO. The trajectory cost we report in Figure 10 is the optimal cost, rather than the cost of the feasible trajectory. The RRT* algorithm can be improved in multiple ways in terms of convergence rate and cost optimality beyond the methods we selected (as we discussed in Section 2.1). We justify comparing the planning methods using the feasible trajectory computation time and the optimized trajectory cost by the fact that regardless of

7 This is equivalent to running the RRT algorithm.

the choice of the exploratory algorithm, the feasible trajectory will always be the fastest to compute and optimal trajectory will always produce the best cost. The trade off between the time needed to compute the optimal trajectory and its cost can be affected by the algorithm implementation and by further tuning their parameters. An exhaustive evaluation of the state of the art exploratory algorithms is outside of the scope of this thesis. We are interested in the effect of planning in alternate spaces instead.

AICO WITH WRITHE REPRESENTATIONS In the next scenario we start with the same setup as we used for solving the winding problem using AICO and classical representations. We, however, replace the collision cost term in Equation 47 with a writhe cost term

$$c(x) = \rho_{\text{eff}} \| y_{\text{eff}} - \phi_{\text{eff}}(x) \|^2 + \rho_{\text{writhe}} \| \phi_{\text{writhe}}(x) \|^2 \,. \tag{48}$$

Since the task requires unwinding of the robot links, we simply aim to minimize the amount of writhe. We can afford to remove the collision term from the cost function entirely because states that would cause collisions would also produce high writhe values and would therefore be avoided as any other sub-optimal solutions. As in the previous experiment, we have manually selected the task weights ρ. Choosing the same weights for the end-effector position and the write was sufficient to successfully compute a trajectory. Figure 10 shows that the cost of this trajectory is significantly lower than the trajectory planned using classical representations. Even though the collision cost was not optimized in this case, the writhe motion prior rendered smooth unwinding motion away from the obstacle more likely than any motion towards the obstacle. Once the robot was completely unwound, the reaching motion towards the target exploited the null space of the writhe representation. While the trajectory cost was reduced by an order of magnitude, the computational cost was increased only slightly. The writhe representation is, however, more costly to compute. The small impact on the total computational cost was due to the fast convergence of the AICO algorithm. As a result, the number of AICO iterations was lower because the writhe prior was fully utilised and each iteration monotonically improved the trajectory.

RRT* WITH WRITHE REPRESENTATIONS In the last set of experiments, we replicated the same setup as in the scenario with RRT* using classical representations. In this case, we have added a sampling bias in the writhe space to the existing bias in the end-effector space. Analogously to the design of the cost function for AICO (see Equation 48), we used a weighted combination of the writhe bias and the end-effector bias with equal relative weighting. We have, again, fine tuned the ratio of random sampling and movement towards the goal. Figure 10a shows that the trajectory cost remained the same as the cost produced by RRT* with classical representations. On the other hand, the computation cost was significantly increased. RRT* does not fully utilise the bias in the writhe space. Certain amount of time is always spent doing randomized exploration. Even though we have fine tuned the ratio of exploration, the RRT* algorithm still required roughly the same number of samples to find the feasible solution using writhe as before, when we used classical representations. As a result, additional time was required to compute the writhe mapping for biasing the exploration.

3.1.2 *Discussion*

Our experiments have shown that utilising topology-based spaces within the AICO framework does, indeed, improve the convergence and overall cost when applied to a suitable problem. In the case of the writhe representation, a suitable problem would involve large amount of winding of the end effector, such as the unwrapping motion of a snake like robot in our experiment, wrapping fingers around a strap of a backpack, or passing a robot arm through a hoop. The writhe captures the interaction between two 1D chains embedded in 3D space and it's invariant to the relative transformations of the two chains. This means that the chains can deform but maintain the same amount of writhe relative to each other at the same time. Writhe can also be used to prevent the two chains from intersecting with each other by avoiding high amount of writhe that is generated when they get close to each other. This is not a substitution for collision avoidance based on geometric distance metrics because writhe does not take into account the actual shape (thickness) of the objects that are being approximated by the kinematic chains.

The writhe representation has a null space that can be exploited during motion planning. Motion which does not affect the amount of writhe, such as reaching for a target far away from the second kinematic chain, cannot be controlled using writhe. In such cases writhe can be used as a complementary task definition, until the robot moves closer to the object of interest and the interaction becomes the dominant element in the cost function.

The computational complexity of calculating the writhe of two chains composed of K_1 and K_2 linear segments respectively is $O(K_1K_2)$. The computation can be therefore parallelised to improve the computational time. The choice of K_1 and K_2 affects how finely the articulated or deformable object is approximated by a chain of linear segments. In the case of articulated objects composed of rigid bodies, such as robot links, the approximation can be rough with one linear segment per robot link (or less in most cases). When interacting with deformable objects or with rigid objects that have complex shapes and require multiple linear segments to capture the shape accurately, we may require to define a higher number of linear segments. How to approximate the objects of interest using the minimum number of linear segments is outside of the scope of this thesis.

We have also discovered that existing implementations of motion planning algorithms have been designed for benchmarking standard motion problems. Our experiments required changing the definition of the problem by changing the cost terms and fine tuning parameters such as task weights ρ and the exploration ratio. These can be abstracted as a set of parameters defining the planning problem. However, we have not found an existing software framework which would implement this abstraction. We have therefore developed a new motion planning library that defines an extensible language for describing motion planning problems. See Appendix A for more details.

In the rest of this chapter, we will present topology-based representations that can be applied to different types of tasks. Since we have shown that a suitable representation can improve the performance of motion planning algorithms, we will now use experiments to demonstrate suitability of different representations for specific problems in robotics.

3.2 WINDING NUMBERS

If the problem can be simplified to planning a wrapping motion in 2D, we can use the *winding numbers* to represent the interaction. The winding number is a measure of how many times a curve is wound around a point on a 2D plane (Figure 11). We compute the Winding number using the approximate algorithm derived by O'Rourke [1998] which is based on calculating inverse trigonometric functions of

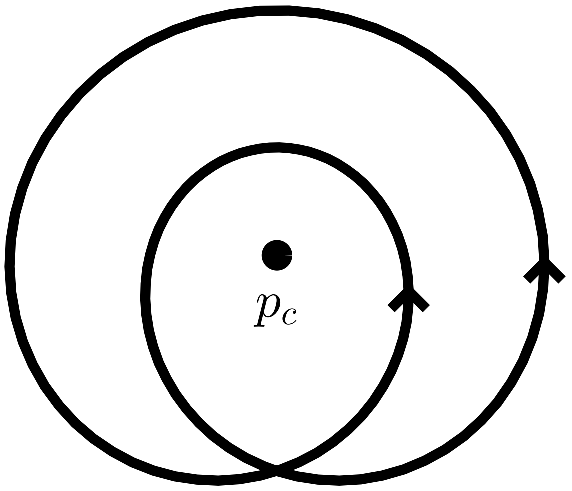


Figure 11: Winding number of a point p_c surrounded by the doubly wound curve: $\phi_{\text{winding}}(x) = 2$.

the scalar product of two normalized vectors, formed by consequent points $\vec{p}_i$ and $\vec{p}_{i+1}$ on a curve and a central point $\vec{p}_c$:

$$\phi_{\text{winding}}(x) = \frac{1}{2\pi} \sum_{i=1}^{n-1} \arccos\left(\frac{(\vec{p}_i - \vec{p}_c)^{\top}(\vec{p}_{i+1} - \vec{p}_c)}{|\vec{p}_i - \vec{p}_c|\ |\vec{p}_{i+1} - \vec{p}_c|} \right) \tag{49}$$

where n is the number of points along the curve.

This scalar continuous function can be thus viewed as a simplification of a writhe representation defined in Section 3.1.

WINDING IN 3D We have designed a toy experiment to demonstrate how we count winding numbers in 3D. The task is to fold a carton sheet into the shape of a box. For simplicity, we do not consider controlling a robot to fold the carton, we instead control the box directly by moving the carton segments. We model these segments as six rigid bodies connected by six revolute joints. Figure 12 shows the carton box in its initial unfolded state (left), and also half way to the goal state (right). In this experiment with the winding numbers, we assume that all the points $\vec{p}_i$ are embedded in the 3D space but we project them onto a common 2D plane. We have projected control points on the surface of the carton along the red and green lines as shown Figure 12 and we set the position of the central point p_c to be the centre of the box (depicted as a small blue cube in Figure 12). We use the standard kinematics to compute the positions of the points $\vec{p}_i$ depending on the

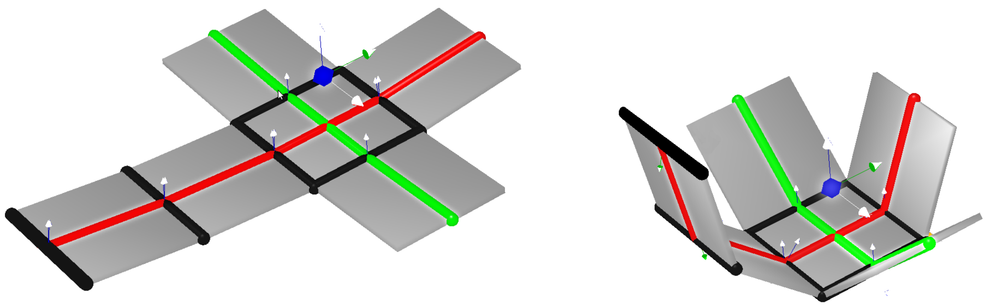


Figure 12: Representing carton box folding using winding numbers. The winding number is computed using a virtual query point (blue) and a projection onto the surface of the box (red and green). The amount of winding is simply increased from the initial configuration (left), to a goal value (right).

configuration $x \in \mathbb{R}^n$. We can therefore use the chain rule to compute the Jacobian of the winding number $\frac{\partial \phi_{\text{winding}}}{\partial x}$. Using this technique, we can directly control the amount of winding.

We apply AICO to compute the carton folding trajectory using the following cost function

$$c(x) = \|y_{\text{winding}} - \phi^{\text{red}}_{\text{winding}}(x)\|^2 + \|y_{\text{winding}} - \phi^{\text{green}}_{\text{winding}}(x)\|^2 \,, \tag{50}$$

where we set the winding goal $y_{\text{winding}} = 1$ which translates into winding each of the two curves around the central point exactly once. We have iterated AICO to convergence producing the carton folding motion. This example is very simplistic and the same result can be achieved by moving each of the revolute joints connecting the carton segments in positive direction (see video at http://youtu.be/L0AG5Vmmt04). The utility of winding number representation is in capturing the interaction between carton and the central point. This interaction may be much more complex, such as wrapping fingers around a door handle. Winding numbers have, however, another application in the area of mobile robotics.

PATH PLANNING WITH WINDING CONSTRAINTS Vernaza et al. [2012] proposed to use winding numbers to define winding constraints for surveillance using mobile robots. Similarly to their approach, we have used the Kuka YouBot mobile platform (see Figure 13) and performed path planning with winding constraints. We have used a discrete grid to represent the configuration space of the

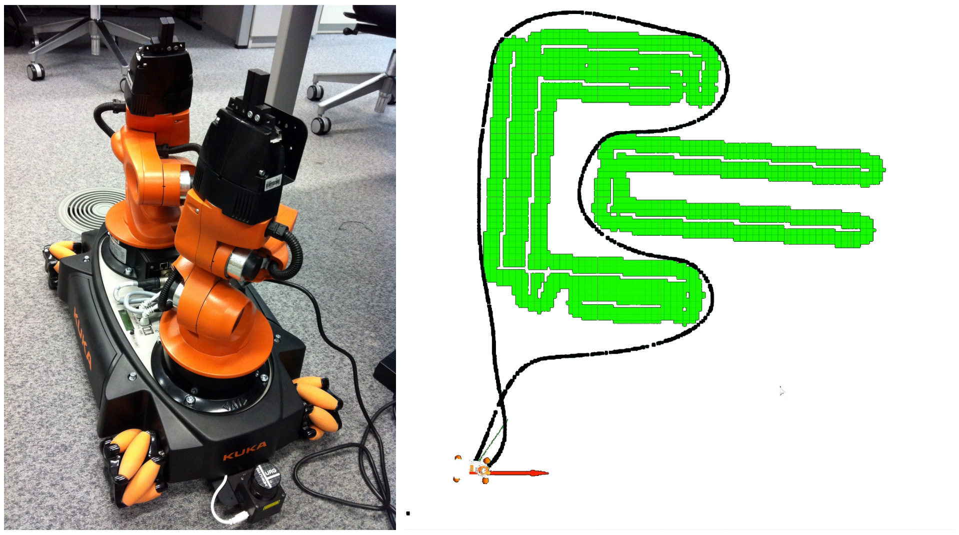


Figure 13: KUKA YouBot robot used in the path planning experiments (left) and a real world scenario with a concave obstacle (right). Figure courtesy of Yiming Yang.

robot. We have assigned costs to each grid cell. Some areas that may be potentially dangerous have been assigned a high cost. The task is to survey the area by driving around each of the points of interest and return to the starting position. We have used the A* algorithm to plan an optimal path. By augmenting the connectivity of the search graph, as described in Vernaza et al. [2012], we were able to define winding constraints around obstacles in the scene. The winding constraint is computed using the central point p_c but the obstacle does not have to be a point. In fact, the the shape of the obstacle does not affect the winding constraint as long as we place the central point inside of the obstacle. Figure 13 (right) shows the results of path planning with a winding constraint around a concave obstacle. We have added another obstacle into the scene to show that the winding constraint is satisfied in presence of other objects[8].

The winding constraint can easily be combined with other metrics and cost terms by augmenting the grid we use for approximating the environment. It is also possible to constrain the path to wind around multiple points of interest. There are, however, multiple solutions that satisfy this constraint when there are two or more points of interest. Figure 14 (left and right) shows two possible solutions for surveying an area with three points of interest. Without specifying any

8 The second obstacle was not used for winding constraint computation to highlight that other cost terms affect the shape of the trajectory and that the winding constraint does not completely define its shape.

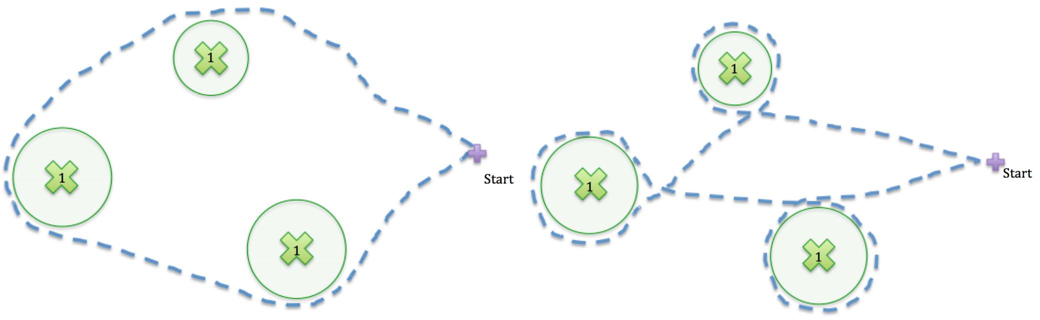


Figure 14: The difference between applying the winding constraint to all obstacles together (left) and to each point of interest individually (right). The number on top of each point of interest is the desired amount of winding. Figure courtesy of Yiming Yang.

additional constraints, adding multiple points of interest will result in a trajectory that winds around the whole group of points because such path is the optimal solution to the problem with respect to the path length. If we wish to compute a path that winds around the points of interest individually, we have to consider the individual winding constraints and ignore the remaining points of interest.

When we plan a path that winds around each point individually, we first specified the order in which we want to visit the points. We then computed a shortest path from current location to the point of interest. This path usually terminates when the robot is within a specified distance from the obstacle. This is where we defined a temporary start point. We then computed a path with a winding constraint around the point of interest. This paths starts and ends at the temporary start point. We repeated this process for each point of interest in order we have specified previously.

We ran an experiment with four points of interest as shown in Figure 15. In this case, we constrained the path to wind around points of interest 1 and 4 individually and we have grouped points 2 and 3 together. We have used the A* algorithm to plan a collision free path that additionally avoids high cost areas. This experiment demonstrates that the winding number allows us to compute paths which wind around areas of interest. This representation is invariant to the shape of these areas since we define the constraint using just a single point inside the area of interest. The winding number representation can only be used in 2D. In 3D, it is possible to either use projection onto a surface, or to use the writhe representation.

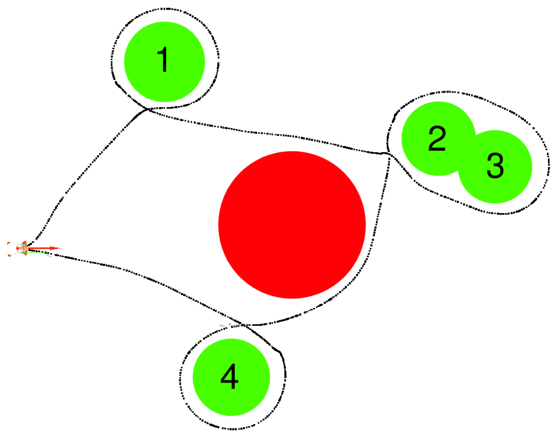


Figure 15: An arbitrary grouping points of interest with respect to the winding constraint. The numbers show the desired order in which the points of interest should be approached. The obstacles are green and the high cost area is rendered red. Figure courtesy of Yiming Yang.

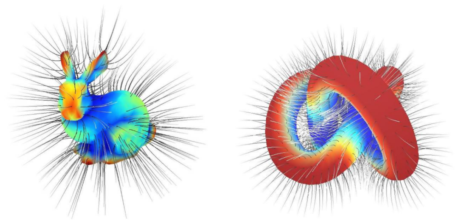
3.3 ELECTROSTATIC COORDINATES

Synthesizing reaching and grasping motion is very challenging due to complex contacts between the gripper and the grasped object, multiple redundant solutions, grasp stability issues, and the complex shape of the open space, especially when concave objects are involved. In this case, motion planning is computationally expensive, because collision detection and global path-planning are required. Furthermore, the movement is no longer valid once the geometry of the hand or the shape of the object change.

Our work is based on the electrostatic parametrisation proposed by Wang et al. [2013], which computes an object-centric curvilinear coordinate system that resembles polar coordinates. Such coordinate system is useful for defining reaching and grasping motion with respect to the object, as it gives a relative spatial relationship between a point and an object.

In order to compute this coordinate system, we virtually charge the object as a conductor. We use a triangulated mesh[9] to approximate the surface of the object. Using the principle of superposition and the fact that the potential around a charged object is proportional to the charge, we can construct a dense linear system of L equations, each representing the notion that the potential must be 1 volt at some probe point on the surface of the object. Using L number of variables

9 Uniformly triangulated surfaces produce more accurate results (Goto et al. [1992]).


(a) Electric field lines of charged objects

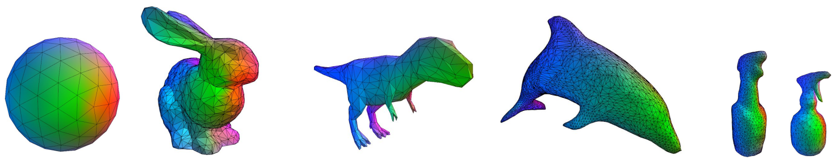

(b) Spherical Mapping

Figure 16: An example of charged objects. (a) The charge is mapped from blue to red, representing low charge and high charge respectively. (b) Projection of 2D spherical coordinates computed using electric field showing correspondence between different objects. Figure courtesy of Peter Sandilands.

denoting the unknown charges Q_l for each of the triangles, we write down the system of linear equations

$$\begin{cases} V_1(\vec{x}_1)Q_1 + V_2(\vec{x}_1)Q_2 + \cdots + V_n(\vec{x}_1)Q_L = 1 \\ \qquad\qquad\qquad\qquad\qquad\qquad \cdots \\ V_1(\vec{x}_L)Q_1 + V_2(\vec{x}_L)Q_2 + \cdots + V_n(\vec{x}_L)Q_L = 1, \end{cases} \tag{51}$$

where $V_l(\vec{x}_m)Q_l$ denotes the potential V_l at point $\vec{x}_m$ due to the l-th triangle carrying charge Q_l in the analytical form (as defined in Goto et al. [1992]). For the probe points $\vec{x}_m$, we select the barycentres of the mesh triangles[10]. Equation 51 defines a dense linear system that can be written down in matrix notation as $PQ = 1$. This system is typically ill-conditioned, and therefore we solve it using pseudo-inverse of P in the least squares sense.

Figure 16a visualizes the distribution of charges over the surface of different objects and the resulting electrostatic fields around them by plotting their field lines. In our system we calculate the potential of each control point $\vec{p}_i(\mathbf{x})$ attached to the hand (related to robot configuration $\mathbf{x}$ through forward kinematics). We compute the potentials at the control points from L uniformly charged triangles defined by points $\vec{p}_k$ with surface charges P

$$\phi_p(\mathbf{x}) = f_p(\vec{p}_i(\mathbf{x}), P, \vec{p}_k) \tag{52}$$

10 Any L points inside or on the surface of the object are suitable as probe points.

as described in Goto et al. [1992].

After completing the charge simulation, the resulting electric field can be used to parametrise the space. Indeed, the field lines emanating from each point on the surface never intersect (or else the conservation of energy would be violated), and since the potential *harmonically* decreases with distance from the object, each point along a particular field line has a unique potential (see Figure 17a). A harmonic field has field lines which do not intersect with each other, making it a field with a small (minimum) number of saddle points and singularities. By following the field lines from the surface outwards, we can map all the points on the object to an infinitely large sphere outside, giving us a 2 dimensional parametrisation of the surface of the object. Here we parametrise the 3D space by mapping the spherical coordinates of the infinitely large sphere to every location in the space by following the field lines outwards. We compute this parametrisation numerically by evaluating the path integral of the electrostatic field along the field line C that passes through a point $\vec{x}$ via

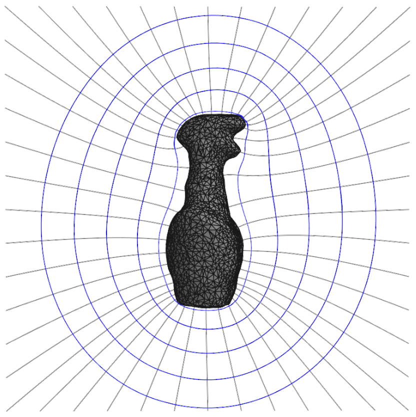
Harmonic field

$$f_{uv}(\vec{x}) = \int_C^{\infty} \vec{E}(\vec{x})d\vec{x}, \text{where } \vec{x} \in C. \tag{53}$$

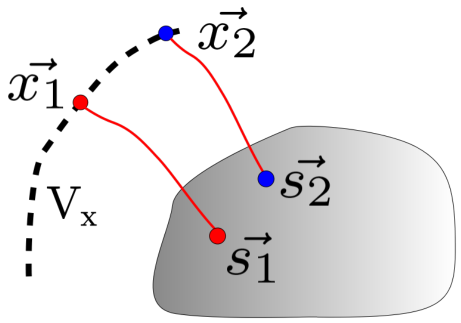
$$\phi_{uv}(x) = \begin{bmatrix} f_{uv}(\vec{p}_0(x)) \\ f_{uv}(\vec{p}_1(x)) \\ \vdots \\ f_{uv}(\vec{p}_M(x)) \end{bmatrix} \tag{54}$$

In practice the sphere is not infinite, as we follow the lines only to a low potential, ensuring the endpoints are far from the object. We use the azimuth u and elevation v (from the endpoints of projection $f_{uv}(\vec{x})$) as the first two coordinates in our parametrisation as ϕ_{uv}, and the electric potential ϕ_p as the final coordinate. Following Wang et al. [2013], we call this parametrisation *electric coordinates*. Additionally, we also compute the inverse map from electric coordinates $\phi_p(x)$ and $\phi_{uv}(x)$ to the position on the surface $\vec{s_i}$ of the object by following the field lines inwards by numerically evaluating the integral $\vec{s_i}(x) = \int_C -\vec{E}(\vec{x})d\vec{x}$ (see Figure 17b). This gives us a projection onto the surface of the object for any point in the outer space of the object. This is a bijective mapping defined for all points on the surface of the object when there are no saddle points. This mapping exists even for object of dissimilar shapes and arbitrary topology due to the non-intersection property

Electric coordinates



(a) Electric field lines



(b) Projection of points onto the surface of the object

Figure 17: (a) The grid (field lines and equipotentials) of our curvilinear electric coordinate system. (b) An illustration of the projection of points $\vec{x}_i$ onto the surface of an object using the electric coordinates. V_x is the isosurface (surface at equal potential, dashed line) of the electric field that $\vec{x}_i$ lies on, $\vec{s}_i$ is the point on the surface of the object corresponding to $\vec{x}_i$ obtained my increasing the potential along the electric field line that intersects $\vec{x}_i$ (shown as a solid red line). Figure courtesy of Peter Sandilands.

of the field lines. If the shape of the charged object has genus larger than 1, saddle points and singularities will appear, and the map between the UV sphere and the surface of the object will be missing the singular points where the field line at the limit of the singularity would intersect the sphere. We compute the *correspondence* between different objects by the mapping of the surface points of the first object onto the sphere and then mapping these points back to the surface of the second object. Figure 16b shows examples of such mapping using the colour of the mesh surface. The mapping is fully defined between all six objects because all of these shapes have the same topology or genus which makes them homeomorphic to a sphere. The projection onto the surface of the object also allows us to compute *force closure* directly from surface points projected using electric coordinates as

Object correspondence

Force closure

$$\phi_{fc}(\mathbf{x}) = f_{fc}(\vec{s}_{1..M}(\mathbf{x})). \tag{55}$$

Function f_{fc} is the force closure measure defined in Miller and Allen [1999]. Here we assume that the projected points will become contact points when distance to the surface of the object decreases. This is a valid assumption if the controlled points are, for example, the fingertips of a robot hand, which will eventually become contact points when we move them closer to the surface of the object by decreasing their electric potential coordinate. When exact placement of the fingers is required, the points projected onto the surface of the object can be used directly as goals. Although these projections can be computed at runtime, we pre-compute the electric coordinates at the vertices of a 3D grid structure surrounding the object. We then look up the electric coordinates for points $\vec{p}_i(x)$ in the 3D grid in order to decrease computational cost at runtime.

The electric field computational complexity is $O(L)$ (where L is the number of triangles comprising the object), as defined by the superposition principle and discussed in Wang et al. [2013]. This lookup grid reduces the computational complexity of the potential calculation for a single point from $O(L)$ to an $O(1)$ lookup and trilinear interpolation. It also reduces the complexity of computing the corresponding point on the surface of the object given a point in space, as this is performed by using the Euler method of integration along the field line until the surface is reached. This means that ϕ_{uv} and ϕ_p are also precomputed and stored in the 3D lookup grid.

INTERIOR DISTANCE FIELD The approximation of the electric field using the 3D grid and the Equation 54 are only valid for the outer space of the object. Although theoretically the field is a consistent 1 volt both on the surface and inside of the object, sampling this field inside the object leads to values less than the surface value of 1 volt due to various approximations we made while computing the field. This not only causes a local maxima at the surface of the object but means that the local potential value does not define whether we are 'inside' the object or not.

To alleviate this problem, we post-process our computed voxel grid to store the values of a distance field as the potential for interior points, which computes $(1 + ||\vec{x} - \vec{b}||)^3$ where $\vec{x}$ is the point in question and $\vec{b}$ is the closest surface point. In this case, the post-processed field potential increases in the direction of the medial axis of the object and decreases in the direction of the nearest surface. If a control point is placed inside of the object, it would be able to follow the gradient of this field towards a potential value of 1, bringing it to the closest point on the surface of the object.

In practise we compute the interior distance field by using a physics engine to emit a ray in an arbitrary direction from the point and return the first point on the object we hit. If no object is hit, we can safely ignore the point as it is outside of the object[11]. If the object is hit, we compute the dot product between the ray and the

11 We assume that the object is closed.

normal vector for the triangle at the point we hit. If it is positive (the ray has hit on the inside of the triangle), we then recalculate the potential of this point using the interior distance field calculation as previously specified.

3.3.1 *Grasp planning in electrostatic coordinates*

To validate our choice of representation, we designed an experiment where we use the electric coordinates for planning motion of KUKA LWR4 robotic arm in combination with the Schunk hand. The object we use here is a spray bottle, which has a standard way of holding[12]. We manually specify the finger placement on the surface of the spray bottle based on a typical human grasp. Since we have specified the finger placement based on a human grasp, we select points on the robot hand that most closely match the corresponding locations on a human hand[13] as well. This allows us to use the electric coordinate representation for motion planning in the AICO framework. In addition to the electric coordinates, we add collision cost to allow the robot to avoid obstacles while reaching for the spray bottle. We use the following cost function:

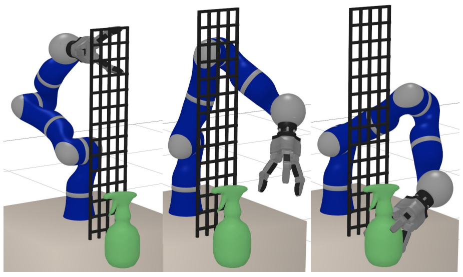
$$c(x) = \rho_p \|y_p - \phi_p(x)\|^2 + \rho_{fc} \|\phi_{fc}(x)\|^2 + \rho_{coll} \|\phi_{coll}(x)\|^2 \,, \tag{56}$$

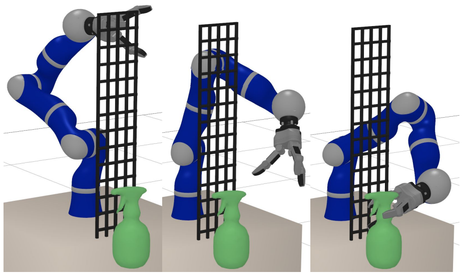
where $\phi_p(x)$ computes the electric potential using Equation 52 with the aim to guide the fingertips towards the surface of the object, $\phi_{fc}(x)$ computes the force closure metric using Equation 55 which aids with optimizing the position of the fingers on the surface of the object, such that the resulting grasp is stable, and $\phi_{coll}(x)$ is the reciprocal collision distance metric we have also used in our previous experiments. In this case, the collision term is used to avoid collisions of the hand with the object during the approach. We used the reference electric potential y_p which we compute by interpolating between the initial and goal coordinate in the electric potential space. We have manually fine tuned the task weighting parameters ρ_p, ρ_{fc} and ρ_{coll}. We disable the force closure cost term by setting $\rho_{fc} = 0$ throughout the whole trajectory, except for the final time step where we use the the force closure cost to evaluate the stability of the grasp.

We now add an obstacle into the environment and perform motion planning in the electric coordinate space (see Figure 18). Since all of the task variables are

12 The fingers being wrapped around the bottle neck and one finger placed on the trigger.

13 These points are fingertips, interphalangeal joints and the wrist. Our experiments confirmed that these points capture the motion sufficiently.


(a) Force closure grasp


(b) Precision grasp

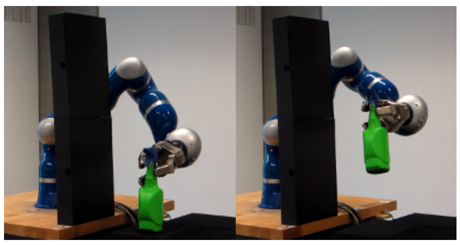

(c) Precision grasp and pick up

Figure 18: Planning successful grasps while reaching around an obstacle. (a) Force-closure grasp maximising stability. (b) Precision grasp with style defined using electric coordinates. (c) Precision grasp implemented on hardware and picking up the spray bottle.

well defined everywhere in the workspace, we were able to iterate AICO to convergence. Figure 18 shows the result of successfully planning a reaching and grasping trajectory. The planner computed a stable generic grasp for the spray bottle (see Figure 18a).

We have then added a task cost term for computing error in the spherical coordinates producing the following cost function:

$$c(x) = \rho_{uv}\|y_{uv} - \phi_{uv}(x)\|^2 + \rho_p\|y_p - \phi_p(x)\|^2 + \rho_{fc}\|\phi_{fc}(x)\|^2 + \rho_{coll}\|\phi_{coll}(x)\|^2 \,, \tag{57}$$

where $\phi_{uv}(x)$ is the electrostatic UV projection and y_{uv} goal position for the fingertips that we have chosen manually based on typical human finger placement. These spherical coordinates arise from the parametrisation of the electrostatic field around the object as defined in Equation 54. While the electric potential allows us to move closer and further away from the object, the spherical coordinates allow us to move along the equipotential surfaces of the electrostatic field, thus giving a full control over the finger placement. We specify a target in spherical coordinates only at the last time step of the trajectory and set the task weight ρ_{uv} to zero for rest of the time steps. This allows us to encode a bias (or a style) while the remainder of the trajectory relies predominantly on potential and force closure motion priors. We have successfully computed a stable grasp using AICO. Figure 18b shows the resulting precision grasp around the neck of the spray bottle and Figure 18c shows the robot executing this plan and subsequently picking up the spray bottle.

The reaching and grasping motions are traditionally planned as separate tasks (Goldfeder et al. [2007]), where a set of viable stable grasps is computed using only the gripper, by virtually separating it from the arm at the wrist, producing a set of targets for the wrist pose. The reaching trajectory for the arm is then planned separately with the aim to reach one of these wrist target poses. Using electric coordinates, we were able to represent both, the reaching and grasping motion as a single task and plan both motions at the same time. Although the force closure measure that ensures stability of the grasp is computed using only approximate contact points, the resulting grasps were stable and robust as demonstrated on the real robot (see Figure 18c). We have shown that this is a suitable technique for planning precision grasps where the point contacts between the fingers and the object dominate the interaction. Power grasps, however, require larger contact surfaces, in which case, point contact are not suitable any more, and the way how the object gets enveloped by the hand becomes more important.

3.3.2 *Discussion*

The electric coordinates represent the position of a point around the object of interest using the electric potential (analogous to the distance to the surface) and the UV coordinates on the sphere at the infinite distance (analogous to position on the surface of the object). The electric potential decreases harmonically but non-linearly as the distance from the object increases. This makes the electric potential space a smooth task space suitable for motion planning. This is however the case only when the shape of the object of interest is convex and has genus 1 (no holes). When the shape contains concave areas, the electric field still remains harmonic but the electric field lines starting in these areas get much closer together in the area surrounding the object than the ones starting in the convex areas. This causes a distortion of the UV coordinates around concave areas. In practice, controlling an end-effector position around a concave area of the object requires a precise motion using smaller steps. Additionally, when the shape has genus higher than 1 (the object contains holes, such as a doughnut), the electric field will contain a singularity and the end-points of the field line passing through this singularity

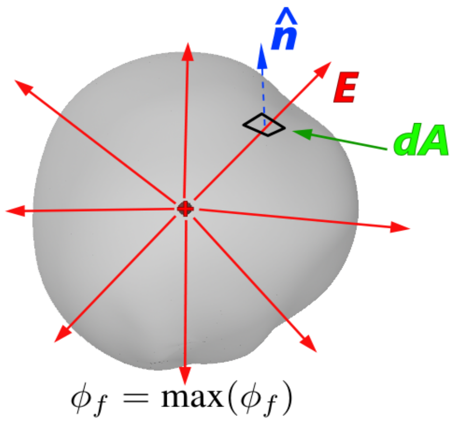


Figure 19: Illustration of Gauss's law for a charged point (red) entirely surrounded by an arbitrary surface (grey). Figure courtesy of Peter Sandilands.

will intercept with the sphere on which we define the UV coordinates. The UV coordinates at these points will not be defined. During motion planning, we would like to avoid these areas with discontinuities. In our experiments, we deliberately choose objects of genus 1 to avoid these problems.

We have also proposed to compute the force closure measure under the assumption that the fingertip contact position on the surface of the object can be approximated by following the field lines towards the surface. This assumption is valid when the fingertips are just points. The fingers of any gripper are, however, rigid bodies and the contact positions between the fingers and the object will change depending on the shape of the two bodies. This approximation is still reasonable when both, the object and the fingers, are convex bodies.

The force closure measure is a function of the positions of all the fingertips we consider for motion planning. This creates a complex, non-smooth, landscape of the force closure cost function. This can create local minima that optimisation planners, such as AICO, may struggle to overcome.

3.4 USING ELECTROSTATICS TO COMPUTE COVERAGE

This electrostatic simulation on the object can also be used to evaluate the amount the object is surrounded by the surface of the hand using a physical property called flux. Electric flux is the surface integral of an electric field passing through a surface. In terms of a representation for planning, this is interesting as it encodes the relative *envelopment* of a surface around an electrically charged object.

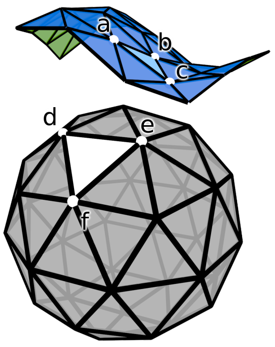


Figure 20: Virtual surface (blue - negative side, green - positive side) with a triangle $\triangle abc$ and a virtually charged object (grey) with a triangle $\triangle def$.

According to Gauss's law, a closed surface surrounding a charged object will always have a constant flux value, no matter the relative transformation or deformation of the outer surface:

$$\oint_S \vec{E} \cdot d\vec{A} = \oint_S \vec{E} \cdot \vec{n} dA = \frac{Q}{\varepsilon_0} = \text{const}, \tag{58}$$

where $\vec{E}$ is the electric field being integrated over the surface S surrounding the charged object (with charge Q). $d\vec{A}$ is a small region of S with a vector with magnitude dA pointing in the normal direction $\vec{n}$, and ε_0 is the electric constant (see Figure 19). This feature makes the flux suitable for representing the coverage of the hand around the object. Here, we use the term "electrostatics" as an analogy to the physical phenomenon. We choose to use this term due the fact that the property of flux was historically first modelled based on electrostatic flux and because this analogy is easier to understand for the reader.

Examples of different configurations in which a deformable object is surrounding the reference object are shown together with the flux value in Figure 21. You can observe that as the object gets more surrounded, the flux value increases.

In practice, we are interested in computing the flux of an arbitrary 2D surfaces modeled using triangulation. Such method was first proposed in Van Oosterom and Strackee [1983] and later used in Wang et al. [2013] to compute the flux of a deformable surface. Given a triangulated model of the closed shape of the virtually charged object and the triangulated open or closed surface in the vicinity of the charged object, we define the approximate electric flux through triangle $\triangle abc$ (see Figure 20) due to uniformly charged triangle $\triangle def$ as:

$$\begin{aligned}\phi_{\text{flux}}(x) =& \frac{|\vec{de} \times \vec{df}|}{2} \sum_{i=1}^{4} g(\vec{x}_i, \triangle abc),\\ \vec{x}_1 =& \frac{4\vec{d}+\vec{e}+\vec{f}}{6}, \vec{x}_2 = \frac{\vec{d}+4\vec{e}+\vec{f}}{6}, \vec{x}_3 = \frac{\vec{d}+\vec{e}+4\vec{f}}{6}, \vec{x}_4 = \frac{\vec{d}+\vec{e}+\vec{f}}{3}\end{aligned} \tag{59}$$

where $g(\vec{x}, \triangle abc)$ is the electric flux through triangle $\triangle abc$ due to charged point $\vec{x}$ defined as[14]:

$$\begin{aligned}g(\vec{x}, \triangle abc) =& 2\,\text{atan2}(J, K),\\ J =& (\vec{ax} \times \vec{bx}) \cdot \vec{cx},\\ K =& |\vec{ax}||\vec{bx}||\vec{cx}| + \vec{ax} \cdot \vec{bx}|\vec{cx}| + \vec{ax} \cdot \vec{cx}|\vec{bx}| + \vec{cx} \cdot \vec{bx}|\vec{ax}|.\end{aligned} \tag{60}$$

Each triangle of the virtually charged object ($\triangle def$) contributes to generating the electric field around the object and each triangle of the virtual surface contributes to the total flux. We compute the total flux using superposition. Because of this, the flux computation has the complexity of $O(lm)$, where l is the number of triangles of the virtually charged object and m is the number of triangles of the virtual surface. The flux formula and its Jacobian are ideal for implementation on parallelized systems, such as GPUs, due to the independent contribution of each triangle to the total flux.

We refer to the object from which we draw triangles $\triangle def$ as virtually charged object. This analogy to objects generating an electric field is useful for visualising the concept of the virtual electric flux, but notice that we don't any actual electrical charges on the the surface of the object, apart from assuming that each triangle is charged uniformly with a unit charge.

If the points a, b, c are attached to a kinematic structure of the robot and controlled via joint angles $q \in \mathbb{R}^n$, then the analytical Jacobian of the flux with respect to the joint angles can be derived using the chain rule (see Appendix B.3). The dimensionality of electric flux space can be chosen arbitrarily anywhere between 1 and D dimensional space, where D is the number of triangles of the hand surface that we use to measure amount of flux. To obtain a lower dimensional flux space, we simply sum the flux over multiple triangle of the hand surface. Lower dimensional space allows for simple control, while higher dimensional flux spaces

14 We denote a triangle defined by edges a, b, and c as $\triangle abc$, a 3D vector as $\vec{x}$, and a vector between two points as $\vec{ax} = \vec{x} - \vec{a}$.

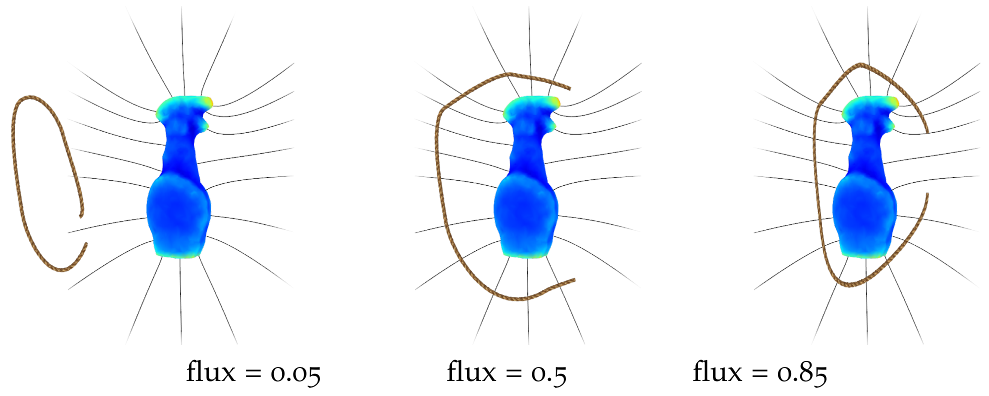


Figure 21: A 2 dimensional illustration of different configurations of a deformable object and a charged reference object labelled with the corresponding flux value of the deformable object. Figure courtesy of Peter Sandilands.

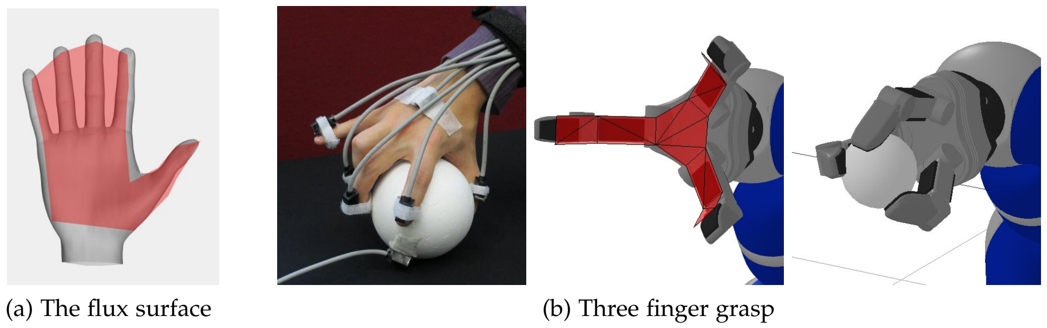


Figure 22: Grasping objects using electric flux. The motion is semantically similar and adapts gracefully to new relative positioning of the hand and the object. Figure courtesy of Peter Sandilands.

allows us to deform parts of the hand independently. This is crucial when controlling style of grasping where each finger moves differently based on the style. We then control the style by weighting each triangle separately. These weights can either be defined manually by an expert or they can be learnt from demonstration.

The electric flux allows us to control the robot hand at a more coarse level than electric coordinates, which makes these representations complementary to each other. A particularly useful property of the electric flux representation is the smoothness of the gradient in the flux space, as this space has very few local minima, which makes it suitable for local optimisation methods or even direct motion transfer and control.

3.4.1 *Grasp transfer between a human hand and a robotic hand*

Using the electric flux representation, we can significantly reduce the complexity of motion planning and transfer by offloading the computational effort onto the mapping between spaces, which can be easily done by the electrostatic parameters. Using these methods, we transfer captured human grasping motion to robotic hands with different kinematics, such as the KUKA LWR4 robotic arm in combination with the Schunk hand, the Shadow hand, and the KCL Metamorphic hand[15].

First, we manually[16] select control points on the surface of the hand and record their electric coordinates from a trajectory provided by a human demonstrator. This gives us a reference trajectory in the space of electric coordinates. In addition to this, we also compute the total flux passing through a mesh defined over the surface of the hand, giving us a measure of coverage and overall orientation towards the object. We define the mapping between the fingers and the different robotic hands manually, based on anthropomorphic similarities with the human hand. We construct the following cost function within the AICO framework:

$$c(x) = \rho_{flux}\|\phi_{flux}(x)\|^2 + \rho_{uv}\|y_{uv} - \phi_{uv}(x)\|^2 + \rho_p\|y_p - \phi_p(x)\|^2 \,, \tag{61}$$

where $\phi_{flux}(x)$ computes the amount of flux passing through the surface of the robot hand using Equation 59, and $\phi_{uv}(x)$ and $\phi_p(x)$ compute the electric coordinates. In this case, we use a constant, non-zero task weight ρ_{uv} for every time step and provide the corresponding reference coordinates y_{uv} for every time step as well. We then iterate AICO until it converges to a trajectory that minimises these cost terms. To show that our method generalizes over robots of dissimilar kinematics, we transfer the motion to Shadow Dexterous Hand, KCL Metamorphic Hand and Schunk Robot Arm (see Figure 23). The resulting motion is semantically similar in terms of relative final grasp locations and approach to the object, which allows us to transfer grasping motion using a natural interface such as our own body.

We have also adapted the grasping motion to different objects. This is possible, because we are able to define correspondences between arbitrary objects by projecting the surface points onto a virtual sphere and then following the electric field gradient outwards of the object as described in Section 3.3 (see Figure 16b). During

15 The KCL Metamorphic hand is presented in Wei et al. [2011].
16 We use the same set of control points as in Section 3.3.1.

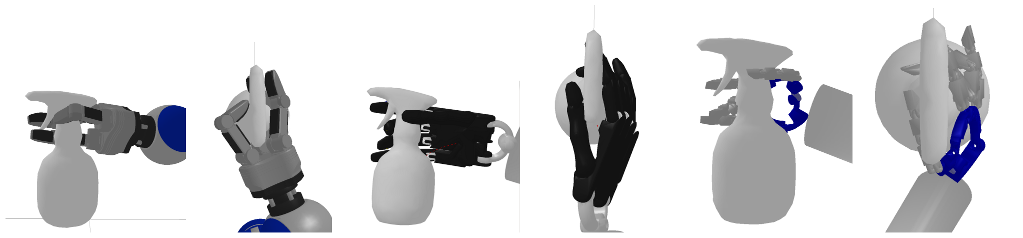

(a) Schunk Grasp (b) Shadow Hand Grasp (c) KCL Metamorphic Hand Grasp

Figure 23: Transferred motion is applicable to different robot hand morphologies. The same electric coordinate sequence is applied to the Schunk hand (a), the Shadow hand (b), and the KCL Metamorphic hand (c). The final grasp is shown here. Figure courtesy of Peter Sandilands.

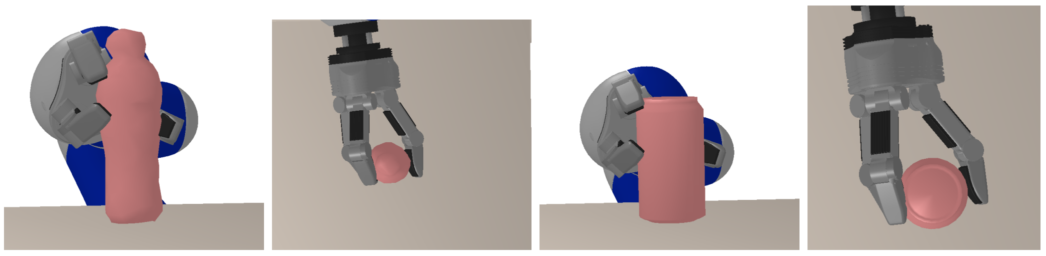

(a) Initial bottle grasp (front and top view) (b) Transferred grasp to soda can (front and top view)

Figure 24: A grasp sequence transferred from a bottle to a soda can. These two objects are different but the grasp can be transferred without any tuning of parameters. The final grasp is shown here. Figure courtesy of Peter Sandilands.

motion transfer to a different object, the electric coordinates remain the same and the mapping onto the surface of the new object results in similar motion adapted to this novel shape. Figure 24 shows grasp transfer between different types of bottles. Notice that the shape of these bottles is reasonably similar (e.g. none of the bottles has a handle nor is any of the bottles significantly larger), which is why the grasps look similar to the human observer. The correspondence does exist for any pair of arbitrarily shaped objects, but the similarity of the actual grasp to the demonstrated grasp and its quality is not guaranteed. The fact that the correspondence can be defined for any pair of shapes is an interesting property of the electric field, that is well known in the area of mathematics called topology.

3.4.2 *Discussion*

Having an oriented surface through which we measure the flux, we can now deform this surface to produce three kinds of behaviour: 1) increase the flux in a positive direction, 2) increase the flux in a negative direction, and 3) maintain zero flux. The positive and negative flux changes can be interpreted as opposite wrapping motions, i.e. wrapping and unwrapping an object with the virtual surface or wrapping the object in clockwise and in the anticlockwise directions. Maintaining zero flux is a bit harder to visualise, since this involves deforming the virtual surface in such way, that it remains perpendicular to the surface of the charged object. We don't use this type of motion for grasping.

We have chosen to compute the flux as a scalar to measure the total flux over the virtual surface generated by the whole charged object. It is also possible to compute the the flux over a subset of the triangles making up the surface, such as the the individual fingers. Another option is to measure flux contribution by each triangle of the charged object. In this case, we can measure and plot the flux on the surface and control the motion to move the flux robot hand from one side of the object to another.

The flux through the virtual surface does not depend on transformations and deformations of the surface which do not cause wrapping (increasing the area covered). Because of this, the flux, as a task representation for motion planning, has a null space which we can utilise to satisfy additional constraints, such as joint limits, collision avoidance, or end-effector position and orientation. Additionally, the proximity of the virtual surface to the object has to be controlled using another task term (such as the electric coordinates), otherwise the virtual surface may collapse to the surface of the charged object. How the flux task gets combined with the other tasks depends on the choice of motion planning algorithm.

3.5 HOMOLOGY INVARIANTS OF ABSTRACT SPACES

We have demonstrated how the position of control points in the electric field can be characterized using object centric electrostatic coordinates regardless of topology and shape of the object. Similarly, we have demonstrated that the geometric shape of obstacles is also not affecting the winding constraints as we have utilised it in surveillance tasks. The writhe representation is also invariant to the geometric shape shape of the robot and the obstacles as long as the relative amount of writhe between the two is the same. The writhe scalar is a topological invariant Edelsbrunner and Harer [2010] based on homotopy that exists between kinematic chains. Let us assume we have one kinematic chain p^i_k that depends on the robot configuration $x \in \mathbb{R}^n$ such as the skeleton of a robot arm. We define m as the number of chains p^j_k that do not depend on the configuration but interact with the robot such as skeletons of obstacles. We can now compute the writhe scalar w_{p^i,p^j} between chains p^i_k and p^j_k. If there exists a *homotopy* between the shapes of the skeleton of the robot at arbitrary two configurations, then we can control the robot to smoothly move between these two configurations without intersecting with the skeletons of the obstacles and we say that these configurations belong to the same homotopy class. *If two chains p^1 and p^2 describing two configurations of a robot belong to the same homotopy class and connect the same two end-points in space then the value of their respective writhe scalars is the same: $w_{p^1,p^j} = w_{p^2,p^j}$.* To prove this we concatenate these two chains to create a closed loop $p^1 \cup -p^2$. Here $-p^2$ indicates that the orientation of the curve is reversed. The writhe scalar of $w_{p^1 \cup -p^2,p^j}$ is non-zero if the closed loop encloses the chain p^j and it is zero otherwise. Therefore, the writhe scalars of the chains p^1 and p^2 must be equal if the chains do not enclose a third chain p^j describing the obstacle. This also means that one of the chains can be smoothly deformed into the other without intersecting p^j. This argument extends to m chains describing the obstacles *individually*. The consequences of using this representation for motion planning are: (1) We can constrain the planning to a particular homotopy class (reach inside a loop as described in Section 3.6.2). (2) Collision avoidance can also be achieved by avoiding high values of writhe that occur near the intersecting configurations.

We call the writhe a topology-based representation because it is directly related to homotopy classes. The writhe scalar is, however, not a homotopy invariant but rather a homology invariant. Here we use homology as an approximation of homotopy (see Rotman [1988] for more details). Homology is a weaker topological invariant which means that if two configurations are homotopic they are also homologous but not necessarily the other way around. As a result, the statement: *"If two configurations of a kinematic chain have the same writhe scalar they belong to the same homotopy class."* is not true universally. An example of such scenario is shown in Figure 25 where the configurations p^1 (solid black) and p^2 (dashed) have the same value of writhe scalar but they belong to different homotopy classes with respect to the two skeletons of the obstacles (circles). In a motion planning scenario, if the task was to reach the goal configuration while maintaining the end-effector

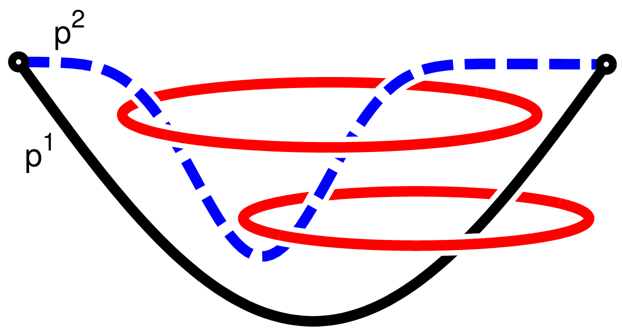


Figure 25: An example of two configurations of a kinematic chain represented by a solid black curve (p^1) and a dashed curve (p^2) that are homologous but not homotopic with respect to both obstacles (circles). As a result it is not possible to deform the black curve into the dashed one without intersecting at least one of the circles. When p^1 and p^2 are concatenated, the three curves form Borromean rings.

position constraint, it would be necessary to break this constraint in order to reach the goal configuration, with a potential danger of getting stuck in a local minima.

We argue that homotopy and homology invariants are desirable properties of task spaces but at the same time, they are not necessary for designing useful representations. In fact, we used the winding numbers (except for defining winding constraints), writhe and electric flux to compute amount of winding, writhing and area coverage respectively, rather than computing the homology classes using these methods. This also means that we can use these techniques on robots that do not form closed curves or enveloping surfaces. The key property of these methods which makes them suitable for motion planning is the way how they allow us to abstract away the geometric details. For example, we will now introduce the interaction mesh, which is a representation that is not based on a topological property but it allows us to preserve spacial relationships between objects in the scene in a similar manner as do the representations derived from topological invariants.

3.6 INTERACTION MESH

When transferring motion between two robots, we may choose to preserve the spacial relationships between the robot and the objects around it. Since we are interested in *transferring motion that involves interactions* with the environment, it is important to consider the transfer of the motion between two scenes, where the scene is the term we use to describe the robot model together with the environment (as defined in Equation 44).

Interaction transfer

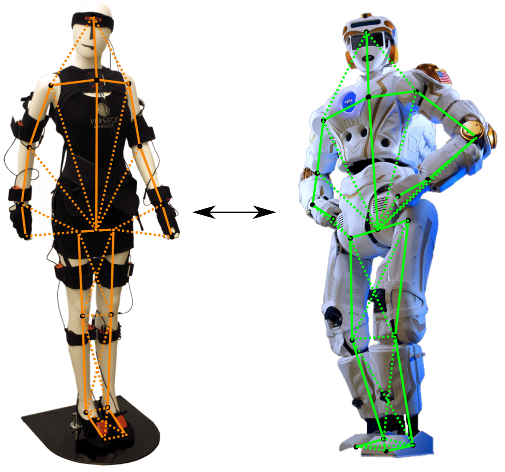


Figure 26: Correspondence between a human model and robot model defined manually by an expert.

First, we choose a set of landmark points from the scene. These points may be attached to the robot or to any object on robot's work space. The landmark points on the robot are usually the joint and end-effector positions. The landmark attached to the objects in the robot's environment usually represent obstacles or other objects that the robot may need to interact with, such as door handles, points on the surface of a table or edges of a shelf. Same number of corresponding points has to be chosen to perform motion transfer between two scenes, where the two scenes may differ in type of robot that is working in them and shape and location of obstacles. Our objective is to select corresponding landmark points such as joint positions and obstacle centroids manually[17], using expert knowledge. See Figure 26 for example set of points on two dissimilar robots. Even though the robot kinematics are different, a human expert can easily define the correspondence.

17 In the field of computer animation, Al-Asqhar et al. [2013] have analysed the scene throughout the motion to automatically generate the points that the virtual character interacts with.

The interaction mesh is a function of a graph connecting a set of the landmark points $P = \{\vec{p}_i\}_i$. Let G be a (bi-directional fully or partially connected) graph on P. To each vertex $\vec{p} \in G$ in the graph, we associate the Laplace coordinate

$$L_G(\vec{p}) = \vec{p} - \sum_{\vec{r} \in \partial_G \vec{p}} \frac{\vec{r}\, w_{pr}}{\sum_{\vec{s} \in \partial_G \vec{p}} w_{ps}} \tag{62}$$

$$w_{pr} = \frac{W_{pr}}{|\vec{r} - \vec{p}|}, w_{ps} = \frac{W_{ps}}{|\vec{s} - \vec{p}|} \tag{63}$$

where $\partial_G \vec{p}$ is the neighbourhood of $\vec{p}$ in the graph G and w_{pr} is the weight inversely proportional to the distance of points $\vec{p}, \vec{r}$ and multiplied by the manually chosen edge importance weighting[18] W_{pr}. The weights are then normalized over the neighbouring nodes in the graph $\vec{s} \in \partial_G \vec{p}$. The collection of Laplace coordinates of all points,

$$\phi_{\text{imesh}}(x) = \left(L_G(\vec{p})\right)_{\vec{p} \in P} , \tag{64}$$

is a 3|P|-dimensional vector which we denote as the *interaction mesh*. We assume the Jacobian of all landmarks in P is given through forward kinematics. The Jacobian $\frac{\partial \phi_{\text{imesh}}}{\partial x}$ of the interaction mesh is given via the chain rule (see Appendix B).

We would like to point out that the squared metric in interaction mesh space has a deformation energy interpretation ([Ho et al., 2010b]). To see this, consider a change of position of a single vertex $\vec{p}$ to a new position p'. The deformation energy associated with such change is defined based on the neighbourhood in a tetrahedronisation T of the point set as

$$E_T(\vec{p}\,') = \frac{1}{2} \|L_T(\vec{p}\,') - L_T(\vec{p})\|^2 \tag{65}$$

where $L_T(\vec{p})$ are the Laplace coordinates of $\vec{p}$ w.r.t. the tetrahedronisation T. This definition is different from our definition of the interaction mesh, because we consider Laplace coordinates L_G w.r.t. the fully connected graph G instead of graph with the connectivity of T, which is a sub-graph of the fully connected graph G computed using tetrahedronisation of points P. Since different configurations lead to topologically different graphs T, using a fully connected graph with topology L_G has the benefit of more continuous measures (deformation energies as well as

18 The edge importance weight matrix allows us further parametrise the representation. For example, high weighting between the target and the end-effector allows us to perform accurate reaching and grasping. If all the elements of the edge importance matrix are set to 1, Equation 63 is identical to the one provided in [Ho et al., 2010b]

the Jacobians). Neglecting this difference, minimizing squared distances in interaction mesh space (as is implicit, e.g., in inverse kinematics approaches as well as the optimal control approaches detailed below) therefore corresponds to minimizing deformation energies. The choice of a particular graph connectivity G and the edge importance weights W_{pr} reflects the desired interaction of the robot with the environment. Similarly to selection of the mesh vertices, we choose the graph connectivity manually[19].

Transferring motion between a human and a robot or between two robotic systems is a very common problem. Solving this problem creates tools for teleoperation, learning by demonstration and other practical applications. It is, however, an ill defined problem. This is the case, because in general, it is difficult to define objective metrics for comparing motion of two dissimilar systems, unlike transferring grasping motion, that can be evaluated based on stability of the grasp. Additionally, different types of motion are often best defined using different soft or hard constraints on position, orientation, proximity or other properties. These constraints may vary within one motion as well as between motions. We use the interaction mesh to approximately preserve these constraints between two kinematic systems.

3.6.1 *Motion transfer using topology-based representations*

The optimal control approach presented in Section 2.2 exploits the additional topology-based spaces to generate optimal trajectories. However, the computed optimal trajectories may no longer be valid when there is a change in the environment, e.g. the obstacles have moved. In order to cope with this issue, we need a dynamic replanning method. Since the topology-based representations are invariant to certain changes in the environment, the replanning at the topology level is not necessary when the representation generalizes the desired motion. We propose a per-frame re-mapping approach in which the optimal trajectory in the topology-based representation $y^*_{0:T}$ is inverse-mapped to the configuration space according to the novel condition of the environment.

In practise, this is done by computing the configuration of the system per-frame, such that the re-mapping error relative to the original optimal topology-based trajectory $y^*_{0:T}$ is minimized. However, topology-based representations such as the writhe matrix or the interaction mesh are very high-dimensional—often higher di-

19 In all our experiments, we use the fully connected graph.

mensional than the configuration space itself. This is in strong contrast to thinking of $y^*_{0:T}$ as a lower dimensional task space like end-effector position or the electric flux space (which is an exception). Therefore, following $y^*_{0:T}$ exactly is generally infeasible and requires a regularisation procedure that minimizes the 1-step cost function:

$$f(x_{t+1}) = \|x_{t+1} - x_t - h\|^2 + \|\phi(x_{t+1}) - y^*\|^2_C \ , \tag{66}$$

$$\underset{x_{t+1}}{\operatorname{argmin}} f(x_{t+1}) = x_t + J^\sharp(y^*_t - \phi(x_t) + (I - J^\sharp J)h \tag{67}$$

$$\text{with } J^\sharp = J^\top(JJ^\top + C^{-1})^{-1}$$

where C describes a cost metric in y-space and h is an arbitrary motion in the null-space of the primary task.

For the case of the interaction mesh, we mentioned the relation of a squared metric C to the deformation energy. Therefore, using the per-frame re-mapping to follow an interaction mesh reference trajectory $y^*_{\text{imesh}_{0:T}}$ essentially tries to minimize the deformation energy between the reference $y^*_{\text{imesh}_t}$ and the actual $\phi_{\text{imesh}}(x_t)$ at each time step. This implies generalizing to new situations by approximately preserving relative distances between interacting objects instead of directly transferring joint angles. In conjunction with the use of feedback gains, the methodology proposed here is able to cope with dynamic environments (see Section 3.6.2) and bounded unpredictable changes.

The bottleneck for a feedback controller using this methodology is the computational cost of the forward mapping of the topology-based representations. The forward mapping of the winding numbers has the computational complexity of $O(n)$, n being the number of linear segments approximating the curve being wound. Computing the writhe coordinates requires $O(nm)$ number of operations (each operation is defined by Equation 45), where n and m are number of segments of the two kinematic chains respectively. Computing the interaction mesh has the computational complexity of $O(2n)$ where n is the number of vertices of the mesh. The performance is therefore in all these cases mainly defined by granularity of the approximation of the geometry of the robot and the environment. The experiment in Section 3.1.1 shows that computation overhead (e.g. the average running time is about 10 seconds on standard 4 core 2.40GHz computer with 4 GB memory) becomes affordable when the task is relatively complex. On the other hand, the experiments in Section 3.6.2 and Section 3.6.3 show that an interaction mesh with a small number of vertices can be used in a feedback loop to provide real time control.

3.6.2 *Dynamic reaching through a loop using writhe and interaction mesh*

Writhe space is a suitable representation for tasks that involve interactions with chains or loops of obstacles. We have altered the task from Section 3.1.1 to reaching through a hollow box, where the rim of the box forms a loop of segments, see Figure 28b. Classically this problem would be addressed by exploiting the end-effector position and collision avoidance cost terms. This is, however, a classical example of a bug trap problem. The advantage of using writhe as a description of the interaction is in defining the task as a *relative* configuration of the robot and the loop. This relative description remains effective also when the box is moved dynamically. The writhe matrix corresponding to the final configuration peaks around the last link which passes through the box (see Figure 28a). This target in writhe space does not uniquely define the task for all arbitrary positions of the box (unlike the unwrapping task in Section 3.1.1) which allows for defining sub-goals, such as precisely controlling the endeffector position via another task variable. We can therefore achieve accurate manipulation within a spatially constrained dynamic environment. In this case, the explicit collision avoidance was then superfluous.

The computation of the optimal trajectory using this methods required 3 to 6 AICO iterations on average using the following cost function:

$$c(x) = \|y_{\text{writhe}} - \phi_{\text{writhe}}(x)\|^2 \,, \tag{68}$$

where y_{writhe} is the desired target configuration in the writhe space. We then used the same target in writhe space and randomly displaced the hollow box. Our method was able to plan collision-free trajectories for all test scenarios. We compared our method with AICO, using only classical representations utilising the following cost function:

$$c(x) = \rho_{\text{eff}}\|y_{\text{eff}} - \phi_{\text{eff}}(x)\|^2 + \rho_{\text{coll}}\|\phi_{\text{coll}}(x)\|^2 \,. \tag{69}$$

This required 20 to 30 AICO iterations on average and in 90% of the trials the algorithm failed to find a collision-free path. The remaining 10% of trials were suc-

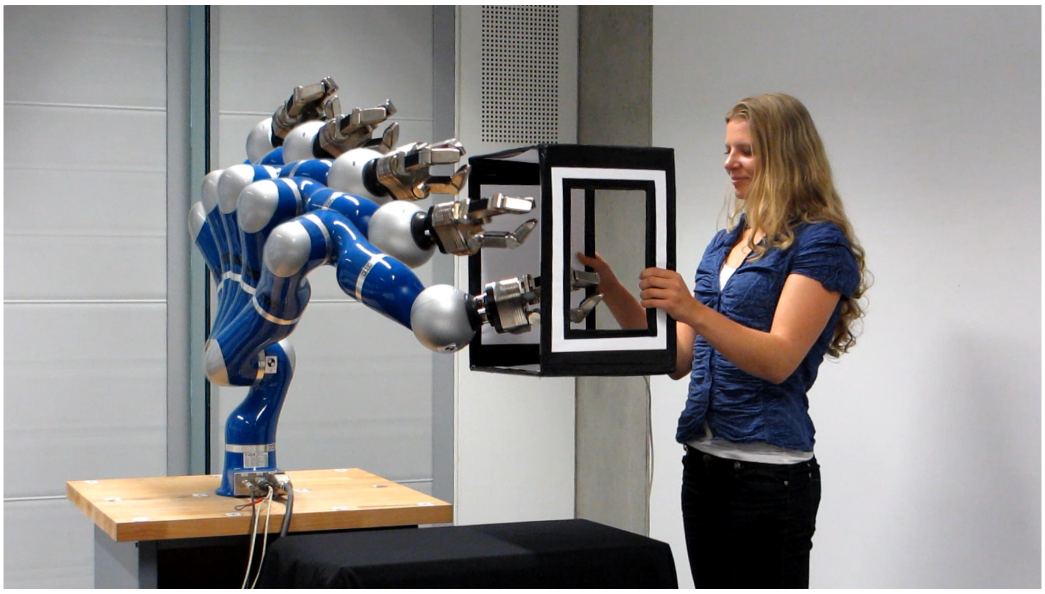


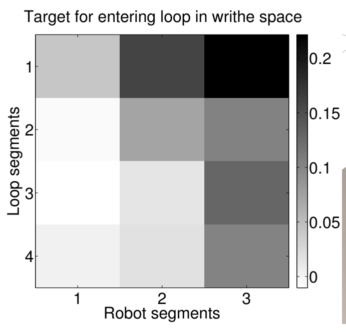
Figure 27: KUKA LWR 4 robotic arm reaching through a hollow box with task being defined in combined Writhe and interaction mesh space, showing an example of planning and dynamic re-mapping using topology-based representations as described in Section 3.6.2.

cessful because the the box was placed in a position where the bug trap problem did not manifest, e.g. the open side of the box was facing the robot.

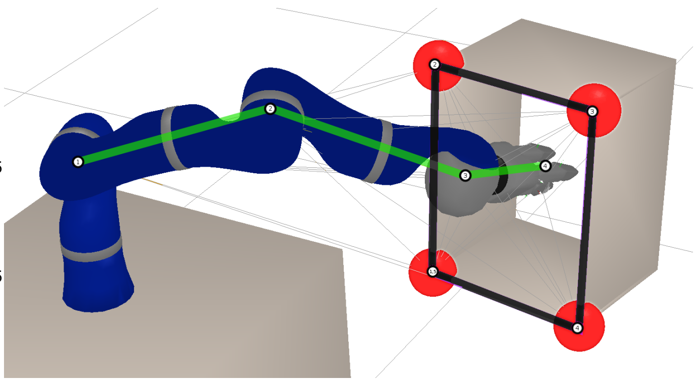
We have then implemented online re-mapping as described in Section 3.6.1 using the interaction mesh space, to test behaviour of the system when the box position is changed dynamically on the fly. In this demonstration we use interaction mesh to represent reaching movement originally planned in the writhe space, while maintaining relative positions of robot links w.r.t. the obstacles. We initially recorded the full trajectory in a static environment in the interaction mesh space, producing reference trajectory $y_{\mathrm{imesh}_{0:T}}$. We have then dynamically updated the robot's position in real time using gradient based IK as discussed in Section 3.6.1. We used the following objective function to minimize within the IK algorithm:

$$c(x) = \|y_{\mathrm{imesh}} - \phi_{\mathrm{imesh}}(x)\|^2 , \tag{70}$$

where $\phi_{\mathrm{imesh}}(x)$ computes the interaction mesh coordinates using Equation 63. We were able to reach inside of the hollow box without any collisions, even when the box was moving. In this experiment, we tracked the position of the hollow box using magnetic motion tracking system. We demonstrate successful motion re-mapping in the following video: `http://youtu.be/LOAG5Vmmt04`.



(a)



(b)

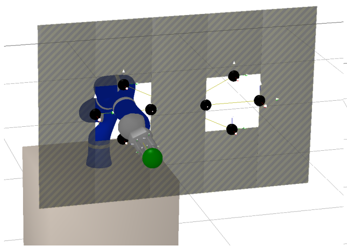
Figure 28: (a) Writhe space target for passing the last link of a robotic arm through a loop. (b) The configuration corresponding to the target in Writhe space. The loop was built by connecting the corners of the rim of the hollow box. The chains representing the robot and the obstacle are overlaid.

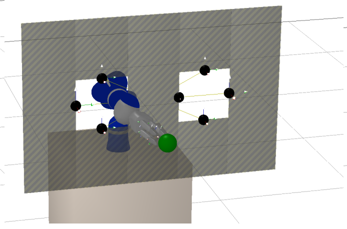
This experiments shows that interaction mesh is capable of representing interactive motion. In this case, the reaching motion can be planned using the writhe representation as well, but we use the interaction mesh as a substitute for real time applications.

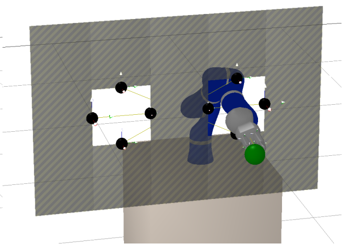
3.6.3 *Motion generalisation using interaction mesh space*

Finally, we present an experiment in which we show examples of motion generalisation when using topology-based representations. We use the KUKA LWR4 arm with 7 DOF and create a scenario common for factory environment, where the task is to reach between items moving on a conveyor belt. The obstacle is a wall with two windows. The wall is moving in front of the robot, thus obstructing the path to the goal (see Figure 29d). We compute the initial trajectory in the static environment using AICO and the classical representations including end-effector position and collision avoidance (we use the cost function defined in Equation 69). These representations are suitable for the simplified case where one of the windows is located in front of the robot. We use the forward mapping defined in Equation 63 to compute the reference trajectory in the interaction mesh space.

MOTION ADAPTATION We then use the re-mapping technique described in Section 3.6.1 to update the motion in real time while still fulfilling the task using Equation 70 for IK computation. Figure 29b shows that the interaction mesh gen-


(a)


(b)


(c)

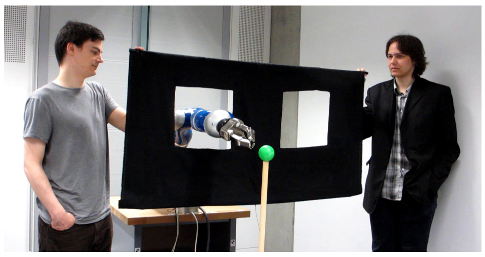

(d)

Figure 29: (a) Initial plan for the robot arm reaching for a goal (sphere) avoiding the wall (striped). (b) Remapping the interaction mesh trajectory after moving the wall to the left. (c) Replanning when the generalisation fails due to the wall being moved too far from its original position. (d) Real time re-mapping on the real robot.

eralizes the motion well. When the wall moves to the left, the path eventually gets obstructed and the task cannot be fulfilled any more. We detect this by measuring the distance of the end-effector from the goal and using collision detection[20]. In this case, re-planning is necessary. Figure 29c shows the motion after re-planning for reaching through the second window. By using the interaction mesh, we were able to delay the re-planning until the obstacle makes it physically impossible to reach the target through the first window. This level of generalization allows us to increase the robustness of the robot motion against perturbations.

MOTION RE-TARGETING In the second part of this experiment, we replaced the KUKA LWR4 arm with a generic 14DOF manipulator with a different kinematic structure. In order to demonstrate that plans in topology-based space can generalize across kinematic differences, we have manually defined correspondences of the landmarks points between the KUKA LWR4 and the generic manipulator. Figure 30 shows the result of re-using the plan in interaction mesh space. We have used the same reference trajectory $y_{\text{imesh}_{0:T}}$ as the one we used for the JUJA LWR4

20 We only use the collision detection to compute when the task cannot be completed any more. We don't use the collision measure to avoid collisions in this case.

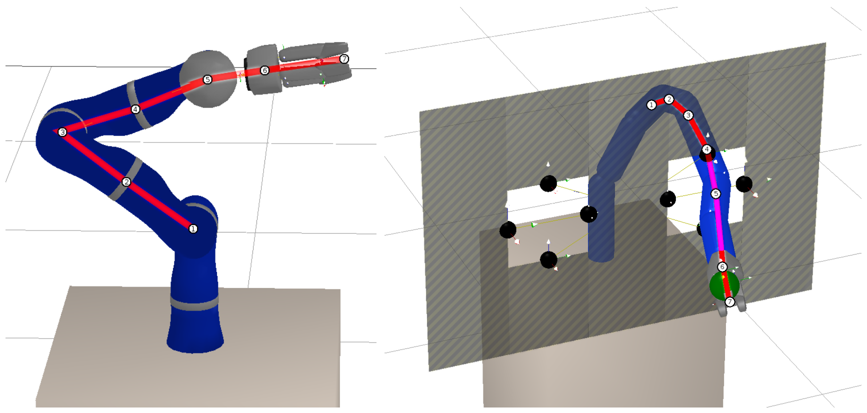


Figure 30: (left) Kinematic chain of the KUKA LWR4 robot used for planning. (right) Remapping the reaching trajectory onto robot with different kinematics. The points used for defining the interaction mesh and the corresponding part of the kinematic chain of the robot are overlaid.

arm. The higher DOF manipulator is also able to perform the assigned task when the wall/window is moving. We demonstrate successful motion re-targeting in the following video: `http://youtu.be/LOAG5Vmmt04`.

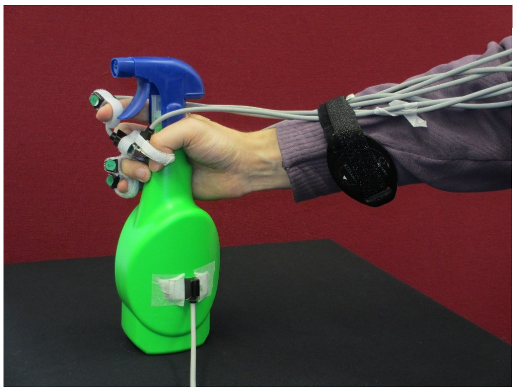
Although, the interaction mesh does not exploit the well known invariants, such as homotopy classes (which was the case with winding numbers, writhe and electric coordinates), the experimental results show that this representation generalises the robot motion well in practise. The real-time re-mapping of the trajectory in the interaction mesh space even allows us to react to unpredictable motion of the objects surrounding the robot. This is, however, only possible if these objects can be tracked in real time. We address the problem of real time sensing of objects with close interactions in the next chapter.

4 SENSING FOR ABSTRACT SPACES

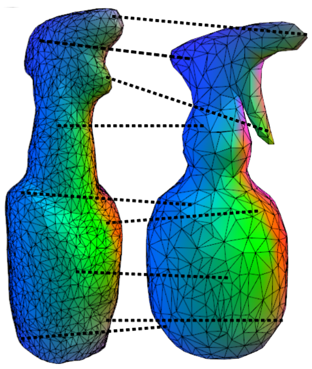
Real time motion planning and control in changing environments is only possible if we are able to keep the state of the scene up to date. The state of the scene is provided to the planning and control algorithms through the scene parameters α_{scene} using the Equation 44 but we have not discussed how to obtain these parameters. In simulated environments, we use the ground truth directly from the simulator. However, in real world applications, we use sensors to detect and track objects in the scene. The representations introduced in Chapter 3 rely on the parameter's α_{scene} to be updated before we compute the task map. This is only possible if we provide the full model of the environment. Occlusions and missing data often change the context of interaction because the topology of the scene changes when data is missing in an occluded area. For this reason, we require sensing methods which can track the state of the whole scene and exploit any prior knowledge about the objects in the scene, such as the shapes of the objects scanned apriori. There is a variety of techniques capable of detecting and tracking objects in the environment using magnetic (Sandilands et al. [2013a]), inertial (Stanton et al. [2012]) or vision sensors (Pauwels et al. [2014a]). It is beyond the scope of this thesis to address the problem of real time sensing for robotics. We will, however, describe techniques that we have utilised in our experiments.

The contributions of this chapter are:

- Application of magnetic tracking techniques to problems of tracking human and robot hand motion while closely interacting with objects.
- Experiments with inertial tracking systems for transfer of human motion to robots using interaction mesh in real time.
- Experiments demonstrating the utility of combining visual cues and robot forward kinematics for accurate real time tracking of objects with close interactions with the robot.



(a) Motion Capture



(b) Correspondences

Figure 31: Motion capture for synthesizing a grasping motion. (a) An interaction of an actor grasping the object is captured . (b) A dense correspondence between the captured object (left) and reference object (right) is produced using the electric field. Figure courtesy of Peter Sandilands.

4.1 MAGNETIC TRACKING

An important part of real time online motion transfer is capturing the interactions between the demonstrator and objects they interact with. In order to do this, we use a magnetic motion-capture system called Polhemus Liberty (introduced by Krieg [1993]), a Microsoft Kinect, and the technique proposed by Sandilands et al. [2013a] for capturing both the hand configuration and the object's positions. The motivation for using this technique is it allows us to capture close interactions with objects. Such interactions usually cause occlusions, which the magnetic sensors do not suffer from.

First, we capture the approximate object geometry using aligned point cloud data from the Kinect camera (see Izadi et al. [2011]). This will produce a detailed triangulated mesh we can use for computing representations such as electrostatic coordinates, which we presented in Section 3.3. We then attach magnetic markers to the objects which shapes we have recorded and (see Figure 31a and Figure 31b) compute their transformations in the scene. No further processing is required for rigid objects, which means that we can use the computed transformations to position the triangulated models in the scene. When tracking articulated objects, such as the human hand, we use inverse kinematics (IK) to reconstruct the motion of the articulated structure. For this, we use the magnetic marker transformations as reference positions for segments of the articulated structure. In the case of hand motion capture, we place one marker on the back of the hand and one marker on each fingertip. Since the magnetic tracking system computes the 6D transforma-

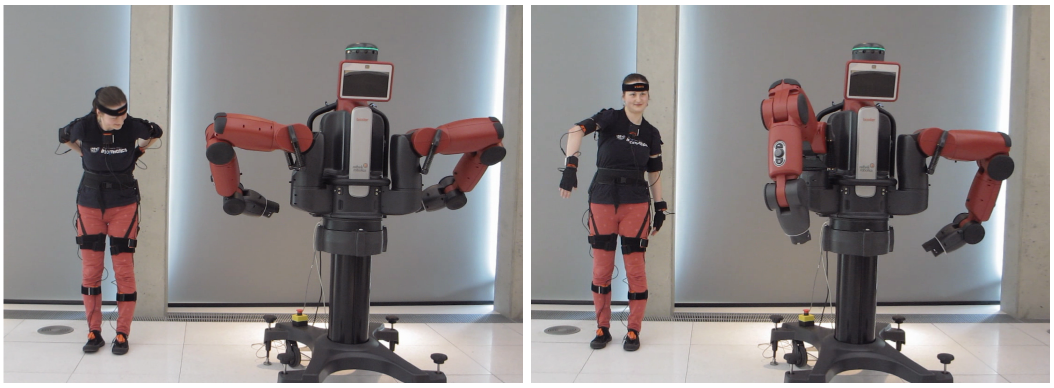


Figure 32: Real time motion transfer using interaction mesh. The human motion was captured using the XSens MVN Biomech system.

tion (position on rotation) of the marker and hand motion is constrained by the human hand kinematics, there exists exactly one solution to the IK problem.

This results in a digitized scene with both geometry and motion data. See Sandilands et al. [2012] and Sandilands et al. [2013a] for further details. If the geometry of the objects in the scene is readily provided (using a CAD model or through other means), the initial geometry capture can be skipped. Tracking objects in the scene for problems with close interactions and occlusions is also possible using inertial tracking systems.

4.2 INERTIAL TRACKING

Inertial measurement units (IMU) are sensors that use a combination of accelerometers and gyroscopes to compute velocity, orientation and gravitational forces. Tracking motion of an object therefore involves integrating the velocities and forces over time to provide relative object poses. Roetenberg et al. [2009] developed a system to accurately estimate and track human motion using IMU measurements which exploits a kino-dynamic model of the human body. We have used this method for transferring human motion onto robots in real time (see Figure 32). In this experiment, we used the motion re-targeting technique we presented in Section 3.6.3. We adapted the cost function defined in Equation 70 within the IK algorithm as follows:

$$c(x) = \|\phi_{\text{imesh}}(z) - \phi_{\text{imesh}}(x)\|^2 \, . \tag{71}$$

We used the human motion capture data z to compute the reference configuration in the interaction mesh space $\phi_{\text{imesh}}(z)$.

This method is suitable for tracking the whole body human motion in real time, without considering other objects in the environment. Although this technique is not able to capture the state of the whole scene, it is a suitable complement to other capture methods (such as robust vision techniques), as it does not suffer from occlusions.

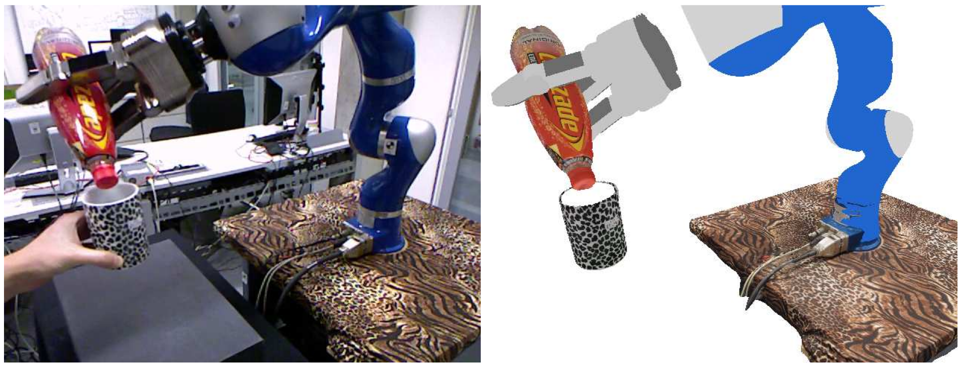
4.3 TRACKING OF KNOWN OBJECTS USING A RGB-D CAMERA

Robust and accurate machine vision techniques have been developed for tracking objects in real time (Romea et al. [2011], Brox et al. [2010] and Drummond and Cipolla [2002]), but they do not scale well to problems with close interactions due to a large amount of occlusion of the object by the manipulator. However, if we consider the vision input as a complementary source of data for a sensor fusion algorithm, the occlusion problem can be alleviated.

In Pauwels et al. [2014a] we propose such method combining visual cues with any additional tracking information available through magnetic tracking, inertial tracking or through forward kinematics of the robot. We have adopted this method and within this approach, we consider an environment with multiple freely moving objects, such as the robot, objects for grasping, and obstacles. The RGB and depth data is captured using a freely moving Kinect camera at a 30 Hz frame rate and any additional tracking data and robot configuration is provided at the same rate.

The object tracking is performed under the assumption that if we are able to generate hypotheses of how the input data from the camera should look like, based on estimated poses of the tracked object, we can iteratively improve this estimate. To generate such hypotheses, we have to create a virtual scene, which contains detailed textured 3D models[1] of all the tracked objects in the scene. By rendering the virtual scene, we obtain synthetic RGB data (see Figure 33a right) and depth data (see Figure 33b right). We then compute visual cues, such as depth cues and augmented reality flow based motion cues, in the synthetic image. These cues are then compared with corresponding cues from the real image. The estimate of

1 To obtain the detailed 3D models, we either use available CAD models or capture the shape and texture of the object using the commercial product Autodesk 123d catch as we proposed in Pauwels et al. [2014b].


(a) RGB data

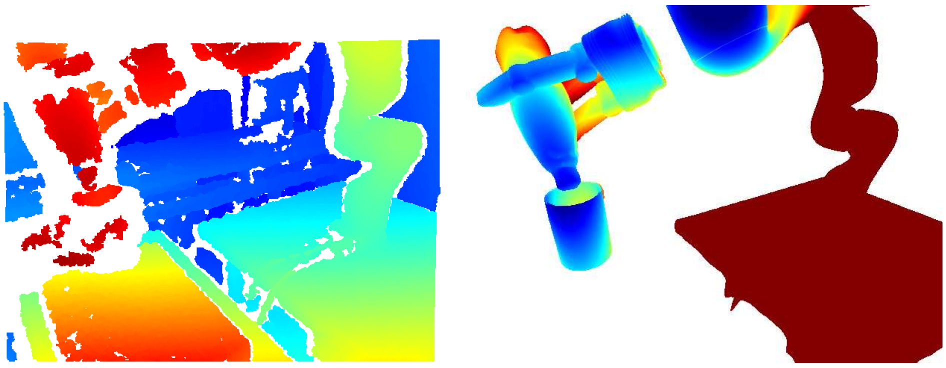
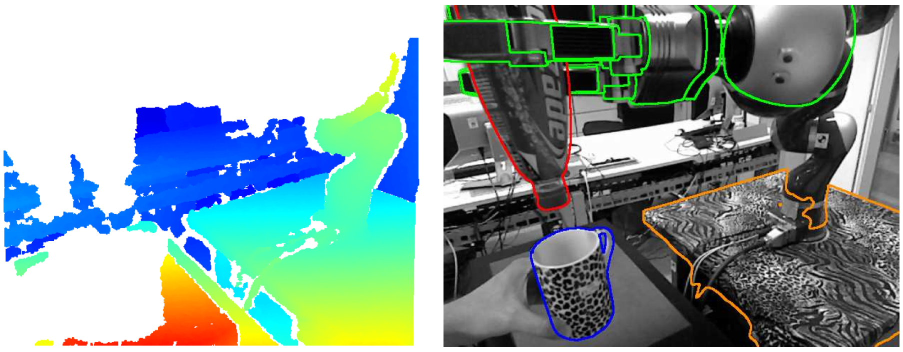

(b) Depth data

Figure 33: Object tracking using visual cues. (a) Captured RGB image (left) and synthesized RGB image (right). (b) Captured Kinect depth data (left) and synthesized depth data (right). Figure courtesy of Karl Pauwels.

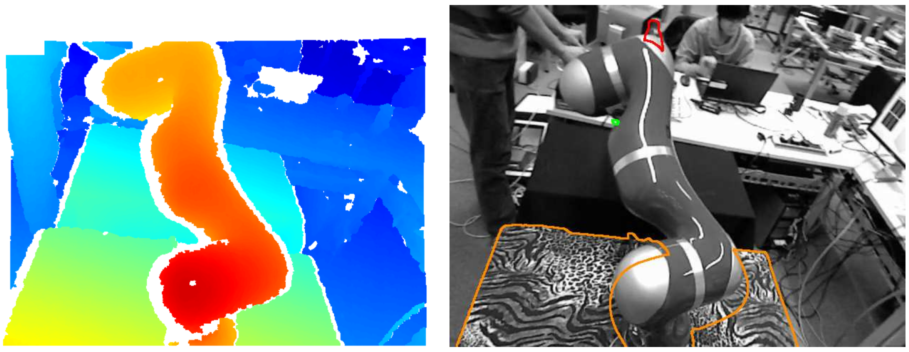
objects' poses is then iteratively improved, until the synthetic cues explain a high enough percentage of the real image cues.

The robustness of this technique can be improved by providing additional priors and pose constraints obtained from other motion tracking techniques or using forward kinematics. Every pose prior or pose constraint then gets translated into a soft constraint on the relative position of two objects in the scene. These additional constraints bias the search for valid object pose estimates and therefore improve both the accuracy and convergence rate of the estimation process. Figure 34 shows that this method is robust when not all of the cues are available in the real image (such as the depth cue) or when the occlusion is severe, and the tracked object is barely visible.

The occlusion, which is a difficult challenge for most vision based tracking systems, is handled implicitly by our method. When producing the synthetic image, the rendering algorithm uses the Z-buffer to remove occluded surfaces. Therefore, the resulting synthetic image contains only the visible parts of all the objects in the



(a) Missing depth data



(b) Occlusion

Figure 34: Motion tracking results with the outlines showing the pose estimate: (a) depth data is missing due to the robot being close to the camera; (b) the robot arm occludes the bottle that is being manipulated. Figure courtesy of Karl Pauwels.

scene. The comparison of the cues from the synthetic and real images then does not need to consider occlusions at all, which results in a more robust and stable tracking.

The estimation of the object poses is a local technique that works well for keeping track of objects that have been previously detected, but it requires a reasonable initialization when the object is lost. In this case, we compute the scale-invariant feature transforms (SIFT) using the same detailed 3D models of the tracked objects that we used for rendering the virtual scene. We detect the approximate pose of object by matching its SIFT features with the features computed from the real image data. This process is an order of magnitude slower than the real time tracking[2]. Therefore, we run the object detection in parallel at a slower rate (about 1Hz). Combination of robust vision-based object tracking with other tracking methods allows us to capture motion with close interaction with high accuracy.

2 The real time tracking runs at 30Hz, due to the update rate of the Kinect sensor. Pauwels et al. [2013] have demonstrated performance of this method using offline data processing at rates exceeding 60Hz.

4.4 DISCUSSION

The representations we introduced in Chapter 3 rely on knowledge of the state of the environment. All of these representations are robust enough to be used in applications where the exact shape of the object is not known but an approximate model is provided. On the other hand, this approximate model has to be provided at all times. This is the reason why we exploit sensing methods which can track objects in the environment in real time, such as the ones presented in Sections 4.1, 4.2, and 4.3.

In comparison, each sensing method has its advantages over the others. The magnetic tracking based method presented in Section 4.1 provides superior accuracy, update rates and it does not suffer from occlusion. On the other hand, the magnetic sensors have to be rigidly attached to the tracked objects, their position on the object has to be measured, the sensors have to be connected to the base station via a wire, and the position of the base station with respect to the robots base frame must be measured. Additionally, the magnetic field gets distorted by metal objects, such as the robot itself. For this reason, we have used this technique mainly for recording the human motion in an environment without any metal objects.

The XSens system is based on inertia measurement sensors. These sensors don't need to be connected to the base station, as they transfer the measurement data wirelessly. Since this system also does not suffer from occlusion, the range at which the motion can be tracked is much larger (up to 100m away from the base station). The update rate and the accuracy of the XSens system are high, particularly for the motion between the body parts. However, the global position suffers from drift which accumulates over time. The drift has to be eliminated using additional sensing. Because of these limitations, we mainly use the XSens system to record the pose of the human for motion transfer. We integrate the data from other sensing methods afterwards to add objects in the environment to the virtual scene.

The machine vision approach to motion tracking presented in Section 4.3 tracks the motion in the camera frame. We translate the tracked positions to the robot base frame by tracking the base of the robot or by calibrating for the camera position. The tracking accuracy depends on the resolution of the camera, the distance of the objects from the camera, the amount of occlusion and reflections, and the lighting conditions. However, this technique does not require any modifications to the tracked object, as long as it's textured richly enough and it can be modelled beforehand.

The comparison of these methods with other state of the art techniques is outside the scope of this theses but you can refer to such evaluation in Sandilands et al. [2013a] for the magnetic sensor based tracking, Roetenberg et al. [2009] for the XSens system, and Pauwels et al. [2013] for the RGB-D camera based tracking.

5 CONCLUSION

In this thesis, we demonstrated that motion planning in dynamic environments, and for scenarios where close interaction with the environment can be made more robust by utilising metrics in alternate task spaces. We model the interaction directly, in a way that generalises well to novel situations and improves the speed and robustness of motion synthesis. To model such interactions explicitly, we exploited topology-based task representations, because they render two very different motions that involve complex interaction close to each other in a similar way the topology invariants render different geometrical shapes as topologically equivalent. We utilised wining numbers, writhe, electric coordinates, electrostatic flux and interaction mesh for solving a range of tasks involving reaching, grasping, winding and path planning for surveillance.

To compute optimal trajectories, we formulated the motion planning problem in the framework of optimal control as an approximate inference problem. We demonstrated how this formulation allows us to couple the motion in the topology-based representations with the motion in the configuration space. We also demonstrated that fully replanning the motion in the topology-based space is not required unless the environment significantly changes. This allowed us to use re-mapping instead of replanning to produce safe motion in dynamic environments with objects with unpredictable motion in real time.

The re-mapping was, however, only possible because we have exploited real time sensing techniques to update the internal model of the environment. These techniques exploited magnetic, inertia and vision based sensors to capture motion of objects with close interactions and occlusions, which is a difficult problem even for state of the art sensing methods. Although it was not within the scope of this thesis to develop novel sensing methods, we have utilized the available methods.

FUTURE WORK In our experiments, we have used the AICO framework to plan robot motion that was complex in configuration space but relatively simple in the topology-based spaces. This approach computed the trajectory in configuration space and coupled it with the trajectory in the topology-based spaces. We have not, however, investigated planning directly in these spaces, nor have we systematically analysed the complexity of these representations. In our future work, we would like to *decouple planning* in the abstract representations from the execution in the configuration space. This will allow us to re-plan tasks in topology-based space, with the objective of computing a reference trajectory in this space. Such trajectory can be then used as a reference for re-mapping, in the same way as we demonstrated in Section 3.6.3. This approach will then perform *re-mapping and re-planning in parallel*, with the objective of replanning the existing abstract trajectory before the re-mapping fails, thus improving the robustness and responsiveness.

Parallel re-mapping and re-planing

The winding numbers, writhe and electric field invariants have been applied for classifying trajectories in the configuration space (Bhattacharya et al. [2011]). We have computed these representations from points and skeletons of the robot and objects in the work space. However, we have not considered using these representations to *characterize trajectories in the work space*. By analysing the trajectories of robot links in the work space, we would be able to classify the motion of these robot links based on which homotopy group their trajectories belong to. This will allow us to explore the work space based on all the possible homotopy classes of work space trajectories and optimize the motion within each of these classes, thus finding a global solution by utilizing only a limited number of local initializations.

Homotopy classes of work space trajectories

Lastly, the interaction mesh representation can be further developed to capture both *position and orientation* of the landmark points. The relationship descriptors proposed by Al-Asqhar et al. [2013] do consider orientation of the landmark points in a similar way that would be useful for interaction mesh. By combining the orientation and position preservation properties with the weighting scheme we proposed, it will be possible to encode a much larger variety of interactions. Additionally, in the future, we intend to learn the weighting from demonstrated trajectories to provide a way to further specialise a generic interaction mesh graph for a specific task.

Position and orientation preserving interaction mesh

A EXTENSIBLE OPTIMISATION FRAMEWORK

The robotics community has developed a multitude of motion planning algorithms over couple of decades. We have reviewed some of these algorithms in Section 2.1 and Section 2.2 but very little has been said about how these algorithms are implemented. Majority of the exploratory algorithms presented in Section 2.1 have been implemented as a part of the Open Motion Planning Library (OMPL). Trajectory optimisation techniques such as the ones proposed by Kalakrishnan et al. [2011] and Ratliff et al. [2009] are included in the Robot Operating System's MoveIt! package (MoveIt!). There are couple more software libraries for robotics, such as the Open Robot Control Software (OROCOS) and Open Architecture Robot Controller (Ford [1994]) but the implementation of algorithms used in robotics research is often confined to software developed by research labs such as Fallon et al. [2014] and Kanehiro et al. [2002].

Integration of robotics software has been encouraged in recent years by development of robotics tool within the Robot Operating System (ROS) but motion planning algorithms still share very little code even within this software project. Commonly used tools, such as kinematic solvers, distance and penetration solvers for collision checking, robot models and task definitions, get often re-implemented for each motion planning algorithm. Comparison of motion planning techniques is therefore a task that requires software engineering skills to interface these techniques. It is also challenging and time consuming to validate that the performance of a motion planner is improved because of quality of the algorithm, rather than the quality of the implementation. If we hope to compare motion planning techniques objectively, we require a platform that will allow us to define motion planning problems using a common description language and present the problem to an arbitrary motion planner together with standard tools for computing kinematics, collision queries and other task related properties.

EXOTica

We have developed the Extensible Optimisation Framework (EXOTica). A central part of this software package is a modular problem definition language that allows us to specify a problem using a set of terms called task definitions. Our main design objectives were *modularity* and *extensibility*, which is why new task maps can be easily added to the framework and immediately used for motion planning. The EXOTica library is aimed at robotics researchers, allowing them to experiment with novel motion planning algorithms and task representations and running comparisons with existing techniques objectively.

Software libraries with similar functionality already exist (i.e. OMPL, OROCOS and MoveIt!), so why is there a need for for another framework? Both the OMPL and MoveIt! implement an interface between multiple motion planning algorithms that allow us to run benchmarks on common navigation and pick-and-place problems. Solving such problems has practical applications but in robotics research we often address a wider variety of problems. Software libraries such as OMPL and MoveIt! do allow us to define arbitrary cost terms and constraints but the problem definition is subjugated to the requirements of the motion planner implementation. For this purpose we require a common language to define arbitrary task representations in a systematic way. The amount of software engineering required to interface task representations with multiple implementations of motion planning algorithms is often underestimated, which why we require a modular and extensible framework within which we make only minimal assumptions about both the problem and solver.

We will now describe the elements of the EXOTica framework using the experiment we presented in Section 3.1.1 as an example. In this experiment, we have used a robot arm that was coiled around an obstacle and we have specified a reaching task that required unwinding the robot arm to make the target reachable without collisions with the obstacle. We have performed trajectory optimisation using the AICO algorithm and the writhe task representation.

Motion solver

We have implemented AICO as a motion *problem solver* within EXOTica (see system diagram Figure 35). A motion solver only requires an initial configuration of the robot and a problem definition. After successfully solving the planning problem, AICO returns a trajectory as a time discretized[1] sequence of robot configurations that we can command to the robot.

Task definition

The motion problem that gets presented to AICO is a collection of parameters[2] specific to the AICO algorithm and a collection of *task definitions* (see Figure 35). The task in this experiment was to unwind the robot and reach for a target. For this purpose, we have created a squared distance error in the work space, for minimizing distance between the end-effector and the target, and a squared distance

1 The AICO algorithm as described in Section 2.2.1 was designed for discrete time problems. However, an EXOTica motion problem solver is not limited to discrete time problems.

2 E.g. number of time steps and time step duration for time discretisation, and control state transition covariance Q_t.

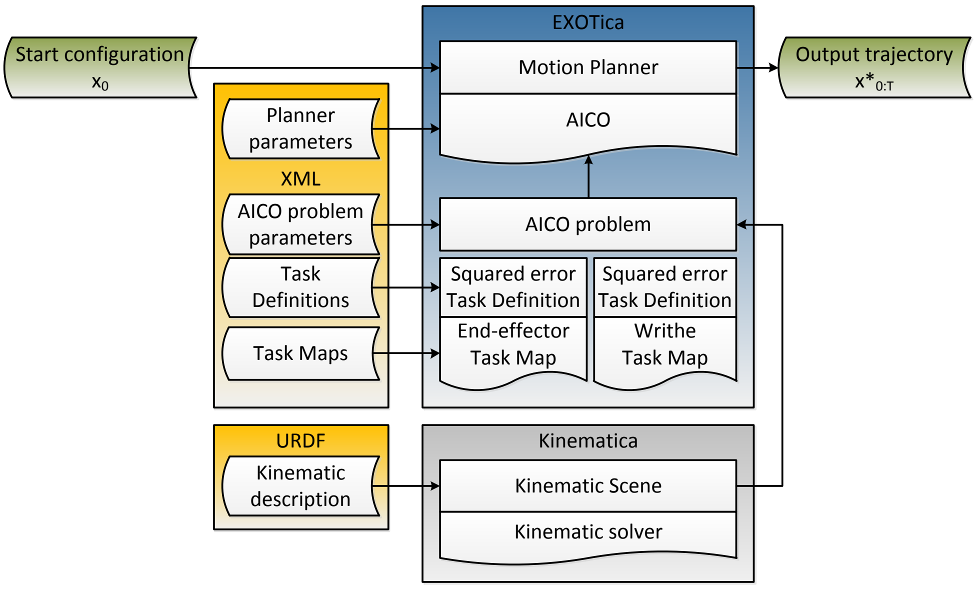


Figure 35: Overview of the EXOTica framework.

error in the writhe space to create a motion prior that will render unwinding more likely. These two cost terms are implemented using task definition objects. A task definition object only specifies that the cost term is a squared metric[3] and provides a point of reference, such as the position of the target to reach.

The space in which the squared metric is measured is then defined within a *task map* (see Figure 35). The task map implements forward task mapping we formally defined in Equation 44. The writhe task map therefore implements Equation 45 and it also computes the Jacobian using Equation 86. We handle the scene parameters α_{scene} using a module called *kinematic scene* (see Figure 35). The kinematic scene contains the robot model as well as a representation of the robot's environment (e.g. the position and shape of the obstacle). Although there may be dynamic elements in the scene, we currently only support kinematic objects and use a module called *Kinematica* which handles forward kinematics (FK) and Jacobian computations of kinematic frames in the work space. The reaching task map therefore only forwards the FK solution and the Jacobian computed by Kinematica without any additional computations.

Task map

Kinematic scene

Kinematica

3 It is also possible to create task definitions for soft and hard constraints and termination criteria.

Algorithm 3 is implemented as a problem solver which we use to solve the motion problem. We query the motion problem in each iteration of the forward sweep loop at line 6 and the backward sweep at line 18. Within this query, we update the kinematic scene using the configuration $\hat{x}_t$ and we then compute $r_t(\hat{x}_t)$ and $R_t(\hat{x}_t)$ using task maps and task definitions. Using this seemingly complex hierarchy of planning problems, task definitions and task maps we fully specify the motion planning problem. Furthermore, we use this structured problem specification as a problem definition language and we store it in an XML file (see code listing 1). The XML file is human readable and it contains an explicit definition of the motion planning problem which allows us to clearly see all the parameters, and objectively compare them with other problem definitions.

Notice that the problem definition is specialised for the AICO solver on line 9. The problem definition in listing 1 can be reused for any motion planner but we need to know the capabilities of this hypothetical algorithm and create a specification of the motion problem that the planner can understand and use for computations. Every time we design a new motion planner, we also design a motion problem specification that will present the task definitions and task maps to the planning algorithm. For example, the *AICOProblem* only accepts task definitions that compute squared error metrics. If the problem was to contain a hard constraint task definition (e.g. end-effector has to always point upwards) the AICO solver wouldn't be able to interpret it, we therefore use the *AICOProblem* specification which will notify us that hard constraints are not supported. On the other hand, any other motion solver which supports only squared error metrics can readily use the AICO problem. In general, all algorithms that solve the LQG class of problems (such as iLQG) won't require any changes to the problem definition.

```xml
<EXOTicaConfiguration>
  <AICOsolver name="MyAICOsolver"> <!--AICO solver parameters-->
    <sweepMode>Symmetric</sweepMode>
    <max_iterations>100</max_iterations>
    <tolerance>1e-2</tolerance>
    <damping>0.01</damping>
  </AICOsolver>

  <AICOProblem name="MyAICOProblem"> <!--AICO problem-->
    <!-- AICO problem parameters -->
    <T>100</T> <!-- Number of time steps -->
    <duration>5.0</duration> <!-- Motion duration (seconds) -->
    <Qrate>1e-10</Qrate> <!-- System noise amplitude -->
    <Hrate>1.0</Hrate> <!-- Control noise amplitude -->
    <W> 7 6 5 4 3 2 1 </W> <!-- Control effort per joint -->

    <Map type="Writhe" name="WindingMap">
      <kscene name="ExampleKinematicScene"/>
      <!-- Map Parameters -->
    </Map>

    <Task name="Unwind" type="TaskSqrError">
      <map name="WindingMap"/>
      <Rho>1.0</Rho> <!-- Task precision -->
      <Goal>0.0</Goal> <!-- Goal writhe value (unwind completely) -->
    </Task>

    <Map type="EffPosition" name="ReachingMap">
      <kscene name="ExampleKinematicScene"/>
    </Map>

    <Task name="Reach" type="TaskSqrError">
      <map name="ReachingMap"/>
      <Rho>10.0</Rho> <!-- Task precision -->
      <Goal>0.5 1.0 0.0</Goal> <!-- Target to reach (meters) -->
    </Task>

    <KScene name="ExampleKinematicScene"> <!--Kinematic scene-->
      <Kinematica> <!--Kinematic solver-->
        <Urdf>robot.urdf</Urdf> <!--Kinematic structure file-->
        <Root segment="root">
          <vector>0 0 0</vector>  <!-- x y z-->
          <quaternion>1.0 0 0 0</quaternion> <!-- w x y z-->
        </Root>
        <Update> <!--List of controlled joints-->
          <joint name="jnt1"/>
          <joint name="jnt2"/>
          <joint name="jnt3"/>
          <joint name="jnt4"/>
          <joint name="jnt5"/>
          <joint name="jnt6"/>
          <joint name="jnt7"/>
        </Update>

        <EndEffector> <!--List of end-effectors-->
          <limb segment="link7"></limb>
        </EndEffector>
      </Kinematica>
    </KScene>

  </AICOProblem>
</EXOTicaConfiguration>
```

Listing 1: EXOTica XML configuration file.

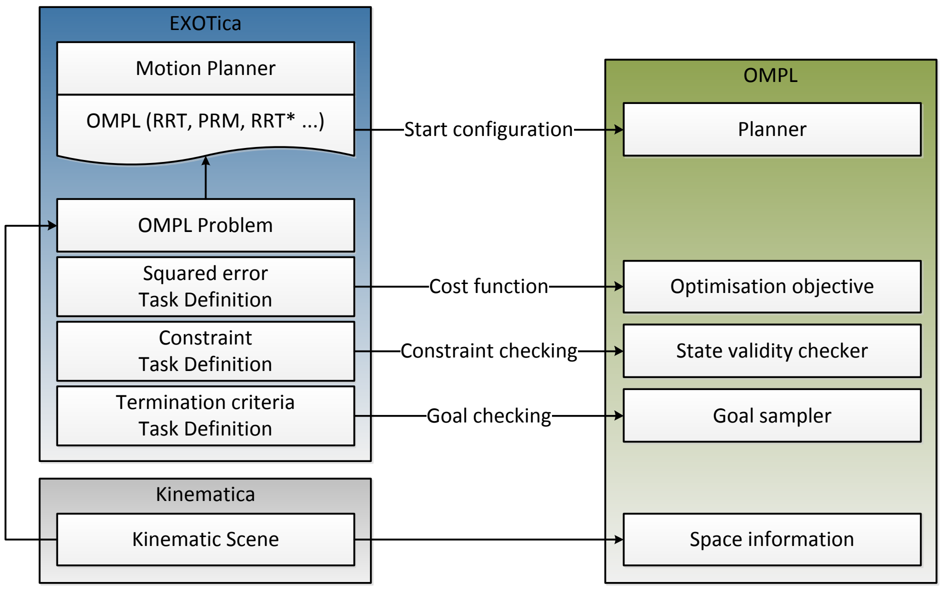


Figure 36: Interface between EXOTica and OMPL library. See Figure 35 for overview of the EXOTica components.

The EXOTica framework is very flexible and easily extensible. Solvers, problems, task definitions, task maps, even the kinematic scene and its underlying kinematic solver can be extended. A typical user will usually design new task maps (such as interaction, mesh, winding numbers or electric flux) and implement new motion planning algorithms (e.g. iLQG or RRT*). Existing implementations can be reused, for example, we have added exploratory planning algorithms implemented within the OMPL library as motion solvers within EXOTica. By doing this, we have crated an interface that translates EXOTica problem description into termination criteria, goal regions, biased sampling strategies and optimisation cost functions within OMPL (see Figure 36). We are therefore able to run comparisons between AICO and a range of exploratory motion planners at the cost of translating problem descriptions between the OMPL and EXOTica frameworks. We are yet to analyse the performance of EXOTica but the initial experiments show no loss of performance compared to native libraries.

EXOTica is very light weight in terms of computational overhead and dependencies. The only dependencies of the core library are Eigen and Boost libraries. However, to open up EXOTica to a wider community of users, we have written interfaces between EXOTica and the ROS MoveIt! package. Current version of our

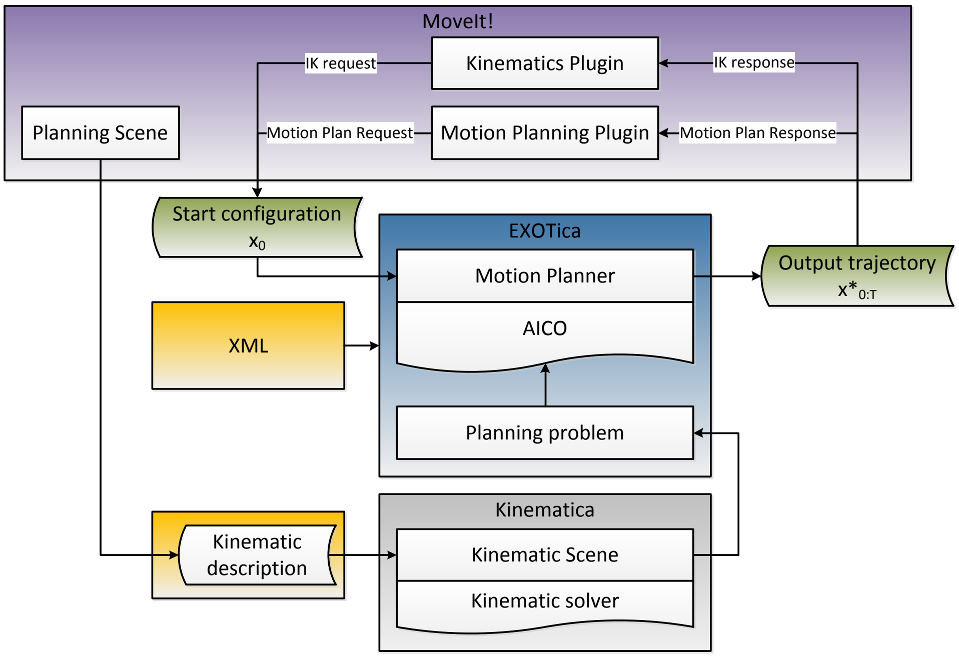


Figure 37: Interface between EXOTica and MoveIt! library.

library is fully integrated with MoveIt! at several levels. We have implemented a motion planner plugin, an inverse kinematics solver plugin and we have tested the code on a real time motion transfer task between human and a Baxter robot (see Section 4.2). We have verified that the core components of the library are working properly but we are yet to use exotica on large scale comparisons. Our next step in developing this library is to release it to the robotics community and extend it with implementations latest motion planning algorithms and task representations.

B ANALYTICAL JACOBIANS OF TOPOLOGY-BASED REPRESENTATIONS

This appendix contains analytical Jacobian derivations of interaction mesh, writhe and electrostatic flux representations. Please refer to the corresponding sections in Chapter 3 for more details about these representations.

B.1 INTERACTION MESH

We have defined the Laplace coordinate of a weighted interaction mesh as

$$L_G(\vec{p}) = \vec{p} - \sum_{\vec{r} \in \partial_G \vec{p}} \frac{\vec{r} w_{pr}}{\sum_{\vec{s} \in \partial_G \vec{p}} w_{ps}}, \tag{72}$$

$$w_{pr} = \frac{W_{pr}}{|\vec{r} - \vec{p}|}, w_{ps} = \frac{W_{ps}}{|\vec{s} - \vec{p}|}. \tag{73}$$

We then compute the analytical Jacobian of the interaction mesh as

$$\begin{aligned} \frac{\delta L(\vec{p}_j)}{\delta x_i} &= \frac{\delta \vec{p}_j}{\delta x_i} - \sum_{l=1, l \neq j}^{M} \left(\frac{\delta w_l^j}{\delta x_i} \vec{p}_l + w_l^j \frac{\delta \vec{p}_l}{\delta x_i}\right) \\ &= \vec{p}'_{ji} - \sum_{l=1, l \neq j}^{M} \left(\frac{\delta w_l^j}{\delta x_i} \vec{p}_l + w_l^j \vec{p}'_{li}\right), \end{aligned} \tag{74}$$

$$\frac{\delta w_l^j}{\delta x_i} = -\frac{W_{lj} \frac{\delta A}{\delta x_i}}{A^2}, \tag{75}$$

$$A = \sum_{k=1, k \neq j}^{M} \frac{W_{kj} P_l}{P_k}, \tag{76}$$

$$\frac{\delta A}{\delta x_i} = \sum_{k=1, k \neq j}^{M} \frac{W_{kj} \frac{\delta P_l}{\delta x_i} P_k - P_l \frac{\delta P_k}{\delta x_i}}{P_k^2}, \tag{77}$$

$$P_l = \|\vec{p}_j - \vec{p}_l\|, P_k = \|\vec{p}_j - \vec{p}_k\|, \tag{78}$$

$$\tag{79}$$

$$\begin{aligned} \frac{\delta p}{\delta x_i} &= \frac{\delta\sqrt{\vec{p}_x^2 + \vec{p}_y^2 + \vec{p}_z^2}}{\delta x_i}, \\ &= \frac{\vec{p}_x \frac{\delta \vec{p}_x}{\delta x_i} + \vec{p}_y \frac{\delta \vec{p}_y}{\delta x_i} + \vec{p}_z \frac{\delta \vec{p}_z}{\delta x_i}}{\sqrt{\vec{p}_x^2 + \vec{p}_y^2 + \vec{p}_z^2}} \\ &= \frac{\vec{p}_x \vec{p'}_x + \vec{p}_y \vec{p'}_y + \vec{p}_z \vec{p'}_z}{\sqrt{\vec{p}_x^2 + \vec{p}_y^2 + \vec{p}_z^2}}, \end{aligned} \tag{80}$$

$$\begin{aligned} \frac{\delta P_l}{\delta x_i} =& \frac{(\vec{p}_{j_x} - \vec{p}_{l_x})(\vec{p'}_{ji_x} - \vec{p'}_{li_x})}{\|\vec{p}_j - \vec{p}_l\|} + \frac{(\vec{p}_{j_y} - \vec{p}_{l_y})(\vec{p'}_{ji_y} - \vec{p'}_{li_y})}{\|\vec{p}_j - \vec{p}_l\|} \\ &+ \frac{(\vec{p}_{j_z} - \vec{p}_{l_z})(\vec{p'}_{ji_z} - \vec{p'}_{li_z})}{\|\vec{p}_j - \vec{p}_l\|}, \end{aligned} \tag{81}$$

$$\begin{aligned} \frac{\delta P_k}{\delta x_i} =& \frac{(\vec{p}_{j_x} - \vec{p}_{k_x})(\vec{p'}_{ji_x} - \vec{p'}_{ki_x})}{\|\vec{p}_j - \vec{p}_k\|} + \frac{(\vec{p}_{j_y} - \vec{p}_{k_y})(\vec{p'}_{ji_y} - \vec{p'}_{ki_y})}{\|\vec{p}_j - \vec{p}_k\|} \\ &+ \frac{(\vec{p}_{j_z} - \vec{p}_{k_z})(\vec{p'}_{ji_z} - \vec{p'}_{ki_z})}{\|\vec{p}_j - \vec{p}_k\|}, \end{aligned} \tag{82}$$

where $\vec{p'}$ is the Jacobian of the of the point $\vec{p}$.

B.2 WRITHE

We compute the writhe matrix of two strings (γ_1 and γ_2) linearly approximated by a series of vectors using the Gauss linking integral (GLI)

$$\mathrm{GLI}(\gamma_1, \gamma_2) = \frac{1}{4\pi} \int_{\gamma_1} \int_{\gamma_2} \frac{d\gamma_1 \times d\gamma_2 \cdot (\gamma_1 - \gamma_2)}{\|\gamma_1 - \gamma_2\|^3}. \tag{83}$$

Each element of the writhe matrix is linearly approximated as

$$W_{ij} = \left[\sin^{-1} \frac{\vec{n}_a^\top \vec{n}_d}{|\vec{n}_a||\vec{n}_d|} + \sin^{-1} \frac{\vec{n}_b^\top \vec{n}_c}{|\vec{n}_b||\vec{n}_c|} + \sin^{-1} \frac{\vec{n}_c^\top \vec{n}_a}{|\vec{n}_c||\vec{n}_a|} + \sin^{-1} \frac{\vec{n}_d^\top \vec{n}_b}{|\vec{n}_d||\vec{n}_b|} \right] \mathrm{sign}\left[\overrightarrow{ab}^\top (\overrightarrow{ac} \times \overrightarrow{cd}) \right], \tag{84}$$

where

$$\vec{n}_a = \overrightarrow{ac} \times \overrightarrow{ad}\ ,\ \vec{n}_b = \overrightarrow{bd} \times \overrightarrow{bc}\ ,\ \vec{n}_c = \overrightarrow{bc} \times \overrightarrow{ac}\ ,\ \vec{n}_d = \overrightarrow{ad} \times \overrightarrow{bd}\ . \tag{85}$$

Vectors $\overrightarrow{ac}$, $\overrightarrow{ad}$, $\overrightarrow{bd}$ and $\overrightarrow{bc}$ are computed from the vectors along the strings γ_1 (points a and b) and γ_2 (points c and d) as shown on figure 8.

Assuming the position of vectors $\overrightarrow{a}, \overrightarrow{b}$ is a function of the robot configuration and $\overrightarrow{a'}, \overrightarrow{b'}$ are their respective Jacobians, we derive the writhe matrix Jacobian as

$$\begin{aligned}
J^{W_{ij},q_k} =& \frac{\overrightarrow{n'}_a \cdot \overrightarrow{n}_b + \overrightarrow{n}_a \cdot \overrightarrow{n'}_b}{\sqrt{1-(\overrightarrow{n}_a \cdot \overrightarrow{n}_b)^2}} + \frac{\overrightarrow{n'}_b \cdot \overrightarrow{n}_c + \overrightarrow{n}_b \cdot \overrightarrow{n'}_c}{\sqrt{1-(\overrightarrow{n}_b \cdot \overrightarrow{n}_c)^2}} \\
&+ \frac{\overrightarrow{n'}_c \cdot \overrightarrow{n}_d + \overrightarrow{n}_c \cdot \overrightarrow{n'}_d}{\sqrt{1-(\overrightarrow{n}_c \cdot \overrightarrow{n}_d)^2}} + \frac{\overrightarrow{n'}_d \cdot \overrightarrow{n}_a + \overrightarrow{n}_d \cdot \overrightarrow{n'}_a}{\sqrt{1-(\overrightarrow{n}_d \cdot \overrightarrow{n}_a)^2}}, \\
\overrightarrow{n'}_a =& \frac{\overrightarrow{A}\|\overrightarrow{ac} \times \overrightarrow{ad}\| - (\overrightarrow{ac} \times \overrightarrow{ad})\|\overrightarrow{A}\|}{\|\overrightarrow{ac} \times \overrightarrow{ad}\|^2}, \\
\overrightarrow{n'}_b =& \frac{\overrightarrow{B}\|\overrightarrow{ad} \times \overrightarrow{bd}\| - (\overrightarrow{ad} \times \overrightarrow{bd})\|\overrightarrow{B}\|}{\|\overrightarrow{ad} \times \overrightarrow{bd}\|^2}, \\
\overrightarrow{n'}_c =& \frac{\overrightarrow{C}\|\overrightarrow{bd} \times \overrightarrow{bc}\| - (\overrightarrow{bd} \times \overrightarrow{bc})\|\overrightarrow{C}\|}{\|\overrightarrow{bd} \times \overrightarrow{bc}\|^2}, \\
\overrightarrow{n'}_d =& \frac{\overrightarrow{D}\|\overrightarrow{bc} \times \overrightarrow{ac}\| - (\overrightarrow{bc} \times \overrightarrow{ac})\|\overrightarrow{D}\|}{\|\overrightarrow{bc} \times \overrightarrow{ac}\|^2}, \\
\overrightarrow{A} =& \frac{\delta(\overrightarrow{a'c} \times \overrightarrow{a'd})}{\delta x_k}, \overrightarrow{B} = \frac{\delta(\overrightarrow{a'd} \times \overrightarrow{b'd})}{\delta x_k}, \\
\overrightarrow{C} =& \frac{\delta(\overrightarrow{b'd} \times \overrightarrow{b'c})}{\delta x_k}, \overrightarrow{D} = \frac{\delta(\overrightarrow{b'c} \times \overrightarrow{a'c})}{\delta x_k}, \\
\|\overrightarrow{A}\| =& \frac{\delta\|\overrightarrow{a'c} \times \overrightarrow{a'd}\|}{\delta x_k}, \|\overrightarrow{B}\| = \frac{\delta\|\overrightarrow{a'd} \times \overrightarrow{b'd}\|}{\delta x_k}, \\
\|\overrightarrow{C}\| =& \frac{\delta\|\overrightarrow{b'd} \times \overrightarrow{b'c}\|}{\delta x_k}, \|\overrightarrow{D}\| = \frac{\delta\|\overrightarrow{b'c} \times \overrightarrow{a'c}\|}{\delta x_k}.
\end{aligned} \tag{86}$$

B.3 ELECTRIC FLUX

We approximate the electric flux through triangle $\triangle abc$ due to uniformly charged triangle $\triangle def$:

$$\begin{aligned} f_{\text{flux}}(\triangle abc, \triangle def) &= \frac{|\overrightarrow{de} \times \overrightarrow{df}|}{2} \sum_{i=1}^{4} g(\overrightarrow{x}_i, \triangle abc), \qquad (87)\\ \overrightarrow{x}_1 = \frac{4\overrightarrow{d} + \overrightarrow{e} + \overrightarrow{f}}{6}&, \overrightarrow{x}_2 = \frac{\overrightarrow{d} + 4\overrightarrow{e} + \overrightarrow{f}}{6},\\ \overrightarrow{x}_3 = \frac{\overrightarrow{d} + \overrightarrow{e} + 4\overrightarrow{f}}{6}&, \overrightarrow{x}_4 = \frac{\overrightarrow{d} + \overrightarrow{e} + \overrightarrow{f}}{3}, \end{aligned}$$

where $g(\overrightarrow{x}, \triangle abc)$ is the electric flux through triangle $\triangle abc$ due to charged point $\overrightarrow{x}$ defined as:

$$\begin{aligned} g(\overrightarrow{x}, \triangle abc) =& 2\,\text{atan2}(J, K), \qquad (88)\\ J =& (\overrightarrow{ax} \times \overrightarrow{bx}) \cdot \overrightarrow{cx},\\ K =& |\overrightarrow{ax}||\overrightarrow{bx}||\overrightarrow{cx}| + \overrightarrow{ax} \cdot \overrightarrow{bx}|\overrightarrow{cx}| + \overrightarrow{ax} \cdot \overrightarrow{cx}|\overrightarrow{bx}| + \overrightarrow{cx} \cdot \overrightarrow{bx}|\overrightarrow{ax}|. \end{aligned}$$

If points a, b, c are attached to a kinematic chain and controlled via joint angles $q \in \mathbb{R}^n$, then the Jacobian of the electric flux with respect to the joint angles can be obtained using the chain rule:

$$\begin{aligned} \frac{\partial f(\triangle abc, \triangle def)}{\partial x} =& \frac{|\overrightarrow{de} \times \overrightarrow{df}|}{2} \sum_{i=1}^{4} \frac{\partial g(\overrightarrow{x}_i, \triangle abc)}{\partial x}, \qquad (89)\\ \frac{\partial g(\overrightarrow{x}, \triangle abc)}{\partial x} =& 2\frac{\frac{\partial J}{\partial x}K - J\frac{\partial K}{\partial x}}{J^2 + K^2}, \qquad (90)\\ \frac{\partial J}{\partial x} =& (\frac{\partial \overrightarrow{ax}}{\partial x} \times \overrightarrow{bx} + \overrightarrow{ax} \times \frac{\partial \overrightarrow{bx}}{\partial x}) \cdot \overrightarrow{cx} + (\overrightarrow{ax} \times \overrightarrow{bx}) \cdot \frac{\partial \overrightarrow{cx}}{\partial x}, \qquad (91)\\ \frac{\partial K}{\partial x} =& \frac{\partial |\overrightarrow{ax}|}{\partial x}|\overrightarrow{bx}||\overrightarrow{cx}| + |\overrightarrow{ax}|\frac{\partial |\overrightarrow{bx}|}{\partial x}|\overrightarrow{cx}| + |\overrightarrow{ax}||\overrightarrow{bx}|\frac{\partial |\overrightarrow{cx}|}{\partial x}\\ &+ (\frac{\partial \overrightarrow{ax}}{\partial x} \cdot \overrightarrow{bx} + \overrightarrow{ax} \cdot \frac{\partial \overrightarrow{bx}}{\partial x})|\overrightarrow{cx}| + \overrightarrow{ax} \cdot \overrightarrow{bx}\frac{\partial |\overrightarrow{cx}|}{\partial x}\\ &+ (\frac{\partial \overrightarrow{ax}}{\partial x} \cdot \overrightarrow{cx} + \overrightarrow{ax} \cdot \frac{\partial \overrightarrow{cx}}{\partial x})|\overrightarrow{bx}| + \overrightarrow{ax} \cdot \overrightarrow{cx}\frac{\partial |\overrightarrow{bx}|}{\partial x}\\ &+ (\frac{\partial \overrightarrow{cx}}{\partial x} \cdot \overrightarrow{bx} + \overrightarrow{cx} \cdot \frac{\partial \overrightarrow{bx}}{\partial x})|\overrightarrow{ax}| + \overrightarrow{cx} \cdot \overrightarrow{bx}\frac{\partial |\overrightarrow{ax}|}{\partial x}, \qquad (92) \end{aligned}$$

where the partial derivatives of all the combinations of the edges of the triangle are computed in a similar manner as $\frac{\partial \overrightarrow{ax}}{\partial x} = -\frac{\partial \overrightarrow{a}}{\partial x}$ and $\frac{\partial |\overrightarrow{ax}|}{\partial x} = \frac{\overrightarrow{ax} \cdot \frac{\partial \overrightarrow{a}}{\partial x}}{|\overrightarrow{ax}|}$.

BIBLIOGRAPHY

B. Akgun and M. Stilman. Sampling heuristics for optimal motion planning in high dimensions. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2640–2645, San Francisco, California, USA, 2011. doi: 10.1109/IROS.2011.6095077. (Cited on page 11.)

R. A. Al-Asqhar, T. Komura, and M. G. Choi. Relationship Descriptors for Interactive Motion Adaptation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 45–53, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2132-7. doi: 10.1145/2485895.2485905. (Cited on pages 61 and 80.)

K. K. Aydin and E. Kocaoglan. A knowledge-based system for redundancy resolution and path planning, using self-motion topology of redundant manipulators. In *Proceedings of International Conference on Tools with Artificial Intelligence*, pages 282–285, Herndon, Virginia, USA, 1995. IEEE. ISBN 0-8186-7312-5. doi: 10.1109/TAI.1995.479615. (Cited on page 2.)

S. Bhattacharya, V. Kumar, and M. Likhachev. Search-based Path Planning with Homotopy Class Constraints . In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Atlanta, Georgia, USA, 2010. AAAI Press. (Cited on page 29.)

S. Bhattacharya, M. Likhachev, and V. Kumar. Identification and Representation of Homotopy Classes of Trajectories for Search-based Path Planning in 3D. In *Proceedings of Robotics: Science and Systems (R:SS)*, Los Angeles, California, USA, 2011. (Cited on pages 29 and 80.)

D. Braun, M. Howard, and S. Vijayakumar. Optimal variable stiffness control: formulation and application to explosive movement tasks. *Autonomous Robots*, 33:237–253, 2012. ISSN 0929-5593. (Cited on page 13.)

T. Brox, B. Rosenhahn, J. Gall, and D. Cremers. Combined Region and Motion-Based 3D Tracking of Rigid and Articulated Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):402–415, March 2010. ISSN 0162-8828. doi: 10.1109/TPAMI.2009.32. (Cited on page 74.)

M. M. da Graça, T. J. A. Machado, and T. P. Azevedo-Perdicoúlis. A Multi-objective Approach for the Motion Planning of Redundant Manipulators. *Applied Soft Computing*, 12(2):589–599, 2012. ISSN 1568-4946. doi: 10.1016/j.asoc.2011.11.006. (Cited on page 2.)

C. H. Dowker and B. T. Morwen. Classification of knot projections. *Topology and its Applications*, 16(1):19–31, 1983. doi: 10.1016/0166-8641(83)90004-4. (Cited on page 29.)

T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):932–946, Jul 2002. ISSN 0162-8828. doi: 10.1109/TPAMI.2002.1017620. (Cited on page 74.)

H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. AMS Press, 2010. ISBN 9780821849255. (Cited on page 59.)

EXOTica. Extensible optimisation framework. `http://wcms.inf.ed.ac.uk/ipab/slmc/research/EXOTica`, 2014. Accessed: 08/09/2014. (Cited on pages 81, 82, 83, 86, and 87.)

M. Fallon, S. Kuindersma, S. Karumanchi, M. Antone, T. Schneider, H. Dai, C. Perez-D'Arpino, R. Deits, M. DiCicco, D. Fourie, T. Koolen, P. Marion, M. Posa, A. Valenzuela, K. Yu, J. A. Shah, K. Iagnemma, R. Tedrake, and S. Teller. A Summary of Team MIT's Approach to the Virtual Robotics Challenge. Technical Report MIT-CSAIL-TR-2014-003, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 2014. (Cited on page 81.)

J. L. Fernández, R. Sanz, J. A. Benayas, and A. R. Diéguez. Improving collision avoidance for mobile robots in partially known environments: the beam curvature method. *Robotics and Autonomous Systems*, 46(4):205–219, 2004. (Cited on page 25.)

F Flacco, A De Luca, and O Khatib. Prioritized Multi-Task Motion Control of Redundant Robots under Hard Joint Constraints. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3970–3977, Algarve, Portugal, 2012. doi: 10.1109/IROS.2012.6385619. (Cited on page 2.)

W. E. Ford. What is an open architecture robot controller? In *Proceedings of the International Symposium on Intelligent Control (ISIC)*, pages 27–32, 1994. doi: 10.1109/ISIC.1994.367846. (Cited on page 81.)

D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, 4(1):23–33, 1997. ISSN 1070-9932. doi: 10.1109/100.580977. (Cited on page 25.)

R. Gayle, K. R. Klingler, and P. G. Xavier. Lazy Reconfiguration Forest (LRF) - An Approach for Motion Planning with Multiple Tasks in Dynamic Environments. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1316–1323, Rome, Italy, 2007. IEEE. doi: 10.1109/ROBOT.2007.363167. (Cited on page 3.)

T. Gillespie. *Fundamentals of Vehicle Dynamics*. Society of Automotive Engineers, 1992. ISBN 9781560911999. (Cited on page 23.)

G. Gioioso, G. Salvietti, M. Malvezzi, and D. Prattichizzo. An Object-Based Approach to Map Human Hand Synergies onto Robotic Hands with Dissimilar Kinematics. In *Proceedings of Robotics: Science and Systems (R:SS)*, Sydney, Australia, 2012. (Cited on page 30.)

Robert Bosch GmbH. *BOSCH Automotive Handbook*. Bosch Handbooks. Wiley, 2004. ISBN 9781860584749. (Cited on page 23.)

C. Goldfeder, P.K. Allen, C. Lackner, and R. Pelossof. Grasp Planning via Decomposition Trees. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4679–4684, Rome, Italy, 2007. doi: 10.1109/ROBOT.2007.364200. (Cited on page 51.)

E. Goto, Y. Shi, and N. Yoshida. Extrapolated Surface Charge Method for Capacity Calculation of Polygons and Polyhedra. In *Journal of Computational Physics*, volume 100, pages 105–115, 1992. (Cited on pages 44, 45, and 46.)

H. Haiying, L. Jiawei, X. Zongwu, W. Bin, L. Hong, and G. Hirzinger. A robot arm/hand teleoperation system with telepresence and shared control. In *Proceedings of Advanced Intelligent Mechatronics (AIM)*, pages 1312 –1317, Monterey, California, USA, 2005. doi: 10.1109/AIM.2005.1511192. (Cited on page 30.)

E. S. L. Ho, T. Komura, S. Ramamoorthy, and S. Vijayakumar. Controlling humanoid robots in topology coordinates. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 178–182, Taipei, Taiwan, 2010a. doi: 10.1109/IROS.2010.5652787. (Cited on page 29.)

E. S. L. Ho, T. Komura, and Ch. Tai. Spatial relationship preserving character motion adaptation. *ACM Transactions on Graphics*, 29:1–8, 2010b. doi: 10.1145/1778765.1778770. (Cited on pages 4, 30, 31, and 62.)

V. Ivan, D. Zarubin, M. Toussaint, T. Komura, and S. Vijayakumar. Topology-based Representations for Motion Planning and Generalisation in Dynamic Environments with Interactions. *The International Journal of Robotics Research*, 32(9-10): 1151–1163, 2013. doi: 10.1177/0278364913482017. (Cited on page 4.)

S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 559–568, Santa Babara, California, USA, 2011. ACM Symposium on User Interface Software and Technology. doi: 10.1145/2047196.2047270. (Cited on page 72.)

M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. STOMP: Stochastic trajectory optimization for motion planning. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4569–4574, Shanghai, China, 2011. doi: 10.1109/ICRA.2011.5980280. (Cited on pages 11 and 81.)

F. Kanehiro, K. Fujiwara, S. Kajita, K. Yokoi, K. Kaneko, H. Hirukawa, Y. Nakamura, and K. Yamane. Open architecture humanoid robotics platform. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 24–30 vol.1, 2002. doi: 10.1109/ROBOT.2002.1013334. (Cited on page 81.)

S. B. Kang and K. Ikeuchi. Robot Task Programming by Human Demonstration: Mapping Human Grasps to Manipulator Grasps. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 97–104, Munich, Germany, 1994. (Cited on page 30.)

O Kanoun, F Lamiraux, and P. B. Wieber. Kinematic Control of Redundant Manipulators: Generalizing the Task Priority Framework to Inequality Tasks . *IEEE Transactions on Robotics*, 27(4):785–792, 2011. (Cited on page 2.)

S. Karaman and E. Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011. ISSN 0278-3649. doi: 10.1177/0278364911406761. (Cited on page 11.)

K. Klenin and J. Langowski. Computation of writhe in modeling of supercoiled DNA. *Biopolymers*, 54(5):307–317, 2000. doi: 10.1002/1097-0282(20001015)54:5<307::AID-BIP20>3.0.CO;2-Y. (Cited on pages 32 and 33.)

N. Y. Ko and R. G. Simmons. The lane-curvature method for local obstacle avoidance. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 161–1621 vol.3, Victoria, Canada, oct 1998. doi: 10.1109/IROS.1998.724829. (Cited on page 25.)

J. Krieg. Motion tracking: Polhemus technology. *Virtual Reality Systems*, 1(1):32–36, 1993. (Cited on page 72.)

T Kröger. On-Line Trajectory Generation: Nonconstant Motion Constraints. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2048–2054, Saint Paul, Minnesota, USA, 2012. doi: 10.1109/ICRA.2012.6225186. (Cited on page 3.)

S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. ISBN 9781139455176. (Cited on pages 8, 9, and 10.)

S. R. Lindemann and S. M. LaValle. Incrementally reducing dispersion by increasing Voronoi bias in RRTs. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3251–3257, New Orleans, Luisiana, USA, 2004. doi: 10.1109/ROBOT.2004.1308755. (Cited on page 27.)

Á. Llamazares, V. Ivan, M. Ocaña, and S. Vijayakumar. Dynamic obstacle avoidance minimizing energy consumption. In *Intelligent Vehicles Workshop on Perception in Robotics*, Alcalá de Henares, Spain, 2012. (Cited on page 23.)

Á. Llamazares, V. Ivan, E. Molinos, M. Ocaña, and S. Vijayakumar. Dynamic Obstacle Avoidance Using Bayesian Occupancy Filter and Approximate Inference. *Sensors*, 13(3):2929–2944, 2013. ISSN 1424-8220. doi: 10.3390/s130302929. (Cited on pages 24 and 25.)

T. Matsuno, D. Tamaki, F. Arai, and T. Fukuda. Manipulation of Deformable Linear Objects Using Knot Invariants to Classify the Object Condition Based on Image Sensor Information. *IEEE/ASME Transactions on Mechatronics*, 11:401–408, 2006. doi: 10.1109/TMECH.2006.878557. (Cited on page 29.)

A.T. Miller and P.K. Allen. Examples of 3D grasp quality computations. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1240–1246, Detroit, Michigan, USA, 1999. doi: 10.1109/ROBOT.1999.772531. (Cited on page 47.)

A.T. Miller, S. Knoop, H.I. Christensen, and P.K. Allen. Automatic grasp planning using shape primitives. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1824–1829 vol.2, Taipei, Taiwan, 2003. doi: 10.1109/ROBOT.2003.1241860. (Cited on pages 3 and 29.)

MoveIt! Moveit! `http://moveit.ros.org/`, 2014. Accessed: 08/09/2014. (Cited on pages 81 and 82.)

J. R. Munkres. *Topology*. Prentice Hall, Incorporated, 2000. ISBN 9780131816299. (Cited on page 31.)

D. M. Murray and S. J. Yakowitz. Differential dynamic programming and Newton's method for discrete optimal control problems. *Journal of Optimization Theory and Applications*, 43:395–414, 1984. doi: 10.1007/BF00934463. (Cited on page 16.)

J. Nakanishi, K. Rawlik, and S. Vijayakumar. Stiffness and temporal optimization in periodic movements: An optimal control approach. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 718 –724, San Francisco, California, USA, 2011. IEEE. (Cited on page 13.)

O. Nechushtan, B. Raveh, and D. Halperin. Sampling-Diagram Automata: A Tool for Analyzing Path Quality in Tree Planners. In *Algorithmic Foundations of Robotics IX*, volume 68, pages 285–301. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-17451-3. doi: 10.1007/978-3-642-17452-0_17. (Cited on page 10.)

OMPL. Open motion planning library. `http://ompl.kavrakilab.org/`, 2014. Accessed: 08/09/2014. (Cited on pages 81, 82, and 86.)

OROCOS. Open robot control software. `http://www.orocos.org/`, 2014. Accessed: 08/09/2014. (Cited on pages 81 and 82.)

J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1998. ISBN 0521640105. (Cited on page 39.)

J. L. J. Palacios. *Understanding and quantifying motor vehicle emissions with vehicle specific power and TILDAS remote sensing*. PhD thesis, Massachusetts Institute of Technology, 1999. (Cited on page 23.)

K. Pauwels, L. Rubio, J. Diaz, and E. Ros. Real-Time Model-Based Rigid Object Pose Estimation and Tracking Combining Dense and Sparse Visual Cues. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2347–2354, 2013. doi: 10.1109/CVPR.2013.304. (Cited on pages 76 and 78.)

K. Pauwels, V. Ivan, E. Ros, and S. Vijayakumar. Real-time Object Pose Recognition and Tracking with an Imprecisely Calibrated Moving RGB-D Camera. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, Illinois, USA, 2014a. (Cited on pages 71 and 74.)

K. Pauwels, L. Rubio, V. Ivan, S. Vijayakumar, and E. Ros. Real-time RGB-D-based Object and Manipulator Pose Estimation. Workshop on RGB-D: Advanced Reasoning with Depth Cameras in conjunction with Conference on Intelligent Robots and Systems (IROS), 2014b. (Cited on page 74.)

A. Peer, S. Einenkel, and M. Buss. Multi-fingered telemanipulation - mapping of a human hand to a three finger gripper. In *Proceedings of Robot and Human Interactive Communication (RO-MAN)*, pages 465 –470, Munich, Germany, 2008. doi: 10.1109/ROMAN.2008.4600710. (Cited on page 30.)

N. Ratliff, M. Zucker, A. J. Bagnell, and S. Srinivasa. CHOMP: Gradient Optimization Techniques for Efficient Motion Planning. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4030–4035, Kobe, Japan, 2009. IEEE Press. ISBN 978-1-4244-2788-8. (Cited on pages 11 and 81.)

K. Rawlik, M. Toussaint, and S. Vijayakumar. An Approximate Inference Approach to Temporal Optimization in Optimal Control. In *Proceedings of Neural Information Processing Systems (NIPS)*, Vancouver, Canada, 2010. (Cited on page 20.)

K. Rawlik, M. Toussaint, and S. Vijayakumar. On Stochastic Optimal Control and Reinforcement Learning by Approximate Inference. In *Proceedings of Robotics: Science and Systems (R:SS)*, Sydney, Australia, 2012. (Cited on pages 16, 17, and 20.)

D. Roetenberg, H. Luinge, and P. Slycke. Xsens MVN: full 6DOF human motion tracking using miniature inertial sensors. Xsens Motion Technologies BV, Tech. Rep, 2009. (Cited on pages 73 and 78.)

A. C. Romea, M. M. Torres, and S. Srinivasa. The MOPED framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research*, 30(10):1284 – 1306, 2011. (Cited on page 74.)

ROS. Robot operating system. http://www.ros.org/, 2014. Accessed: 08/09/2014. (Cited on page 81.)

J. J. Rotman. *An Introduction to Algebraic Topology*. Springer-Verlag, 1988. ISBN 9780387966786. (Cited on page 59.)

M. Saha and P. Isto. Manipulation Planning for Deformable Linear Objects. *IEEE Transaction on Robotics*, 23:1141–1150, 2007. doi: 10.1109/TRO.2007.907486. (Cited on page 29.)

O. Salzman and D. Halperin. Asymptotically near-optimal RRT for fast, high-quality, motion planning. CoRR, 2013. (Cited on page 11.)

P. Sandilands, Myung-Geol Choi, and T. Komura. Capturing Close Interactions with Objects Using a Magnetic Motion Capture System and a RGBD Sensor. In *Motion in Games*, volume 7660, pages 220–231. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-34709-2. doi: 10.1007/978-3-642-34710-8_21. (Cited on page 73.)

P. Sandilands, Myung-Geol Choi, and T. Komura. Interaction Capture using Magnetic Sensors. *Computer Animation and Virtual Worlds*, 2013a. ISSN 1546-427X. doi: 10.1002/cav.1537. URL http://dx.doi.org/10.1002/cav.1537. (Cited on pages 71, 72, 73, and 78.)

P. Sandilands, V. Ivan, T. Komura, and S. Vijayakumar. Dexterous Reaching, Grasp Transfer and Planning Using Electrostatic Representations. In *Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Atlanta, Georgia, USA, 2013b. (Cited on pages 4 and 30.)

L Sentis, J Park, and O Khatib. Compliant Control of Multi-Contact and Center of Mass Behaviors in Humanoid Robots. *IEEE Transactions on Robotics*, 26(3): 483–501, 2010. (Cited on page 3.)

A. Shkolnik and R. Tedrake. Path planning in 1000+ dimensions using a task-space Voronoi bias. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2892–2898, Kobe, Japan, 2009. IEEE. (Cited on pages 1, 10, and 36.)

A. Shkolnik, M. Walter, and R. Tedrake. Reachability-guided Sampling for Planning Under Differential Constraints. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4387–4393, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-2788-8. (Cited on page 10.)

B. Siciliano and O. Khatib. *Springer Handbook of Robotics*. Springer, Berlin, Heidelberg, 2008. ISBN 978-3-540-23957-4. (Cited on pages 2 and 3.)

G. Sovran and M. Bohn. Formulae for the Tractive-Energy Requirements of Vehicles Driving the EPA Schedules. SAE Technical Paper, 1981. (Cited on pages 22 and 23.)

C. Stanton, A. Bogdanovych, and E. Ratanasena. Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning. In *Proceedings of Australasian Conference on Robotics and Automation (ARAA)*, Wellington, New Zealand, 2012. ISBN 978-0-9807404-3-1. (Cited on page 71.)

R. F. Stengel. *Optimal Control and Estimation*. Dover Publications, 1986. ISBN 9780486682006. (Cited on pages 12 and 13.)

J. Takamatsu, T. Morita, K. Ogawara, H. Kimura, and K. Ikeuchi. Representation for Knot-Tying Tasks. *IEEE Transactions on Robotics*, 22:65–78, 2006. doi: 10.1109/TRO.2005.855988. (Cited on page 29.)

T. Tamei, T. Matsubara, A. Rai, and T. Shibata. Reinforcement learning of clothing assistance with a dual-arm robot. In *Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 733–738, Bled, Slovenia, 2011. IEEE. doi: 10.1109/Humanoids.2011.6100915. (Cited on page 29.)

E. Theodorou, J. Buchli, and S. Schaal. A Generalized Path Integral Control Approach to Reinforcement Learning. *The Journal of Machine Learning Research*, 11: 3137–3181, 2010. ISSN 1532-4435. (Cited on page 3.)

E. Todorov and Weiwei Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of American Control Conference*, pages 300–306, Portland, Oregon, USA, June 2005. doi: 10.1109/ACC.2005.1469949. (Cited on pages 13 and 16.)

M. Toussaint. Robot Trajectory Optimization using Approximate Inference. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1049–1056, New York, NY, USA, 2009. ACM. doi: 10.1145/1553374.1553508. (Cited on pages 14, 15, 17, and 20.)

I. Ulrich and J. Borenstein. VFH+: reliable obstacle avoidance for fast mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1572 –1577 vol.2, 1998. doi: 10.1109/ROBOT.1998.677362. (Cited on page 25.)

C. Urmson and R. Simmons. Approaches for heuristically biasing RRT growth. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 1178–1183 vol.2, Las Vegas, Nevada, USA, 2003. doi: 10.1109/IROS.2003.1248805. (Cited on page 10.)

A. Van Oosterom and J. Strackee. The Solid Angle of a Plane Triangle. *IEEE Transactions on Biomedical Engineering*, BME-30(2):125–126, 1983. ISSN 0018-9294. doi: 10.1109/TBME.1983.325207. (Cited on page 53.)

P. Vernaza, V. Narayanan, and M. Likhachev. Efficiently finding optimal winding-constrained loops in the plane. In *Proceedings of Robotics: Science and Systems (R:SS)*, Sydney, Australia, 2012. (Cited on pages 41 and 42.)

H. Wakamatsu, E. Arai, and S. Hirai. Knotting/Unknotting Manipulation of Deformable Linear Objects. *The International Journal of Robotics Research*, 25(4):371–395, 2006. doi: 10.1177/0278364906064819. (Cited on page 29.)

He Wang, K. Sidorov, P. Sandilands, and T. Komura. Harmonic Parameterization by Electrostatics. *ACM Transactions on Graphics (TOG)*, 2013. ISSN 0730-0301. (Cited on pages 44, 46, 48, and 53.)

G. Wei, J. S. Dai, S. Wang, and H. Luo. Kinematic Analysis and Prototype of a Metamorphic Anthropomorphic Hand with a Reconfigurable Palm. *International Journal Of Humanoid Robotics*, 8(3):459 – 479, 2011. (Cited on page 56.)

C. Yao-Chon. Solving robot trajectory planning problems with uniform cubic B-splines. *Optimal Control Applications and Methods*, 12(4):247–262, 1991. ISSN 1099-1514. doi: 10.1002/oca.4660120404. (Cited on page 12.)

D. Zarubin, V. Ivan, M.c Toussaint, T. Komura, and S. Vijayakumar. Hierarchical Motion Planning in Topological Representations. In *Proceedings of Robotics: Science and Systems (R:SS)*, Sydney, Australia, 2012. (Cited on page 20.)

J. Zhang and A. Knoll. An Enhanced Optimization Approach for Generating Smooth Robot Trajectories in the Presence of Obstacles. In *Proceedings of the European Chinese Automation Conference*, pages 263–268, 1995. (Cited on page 12.)

I. A. Şucan and L. E. Kavraki. Kinodynamic Motion Planning by Interior-Exterior Cell Exploration. In *Algorithmic Foundation of Robotics VIII*, volume 57, pages 449–464. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-00311-0. doi: 10.1007/978-3-642-00312-7_28. (Cited on page 11.)