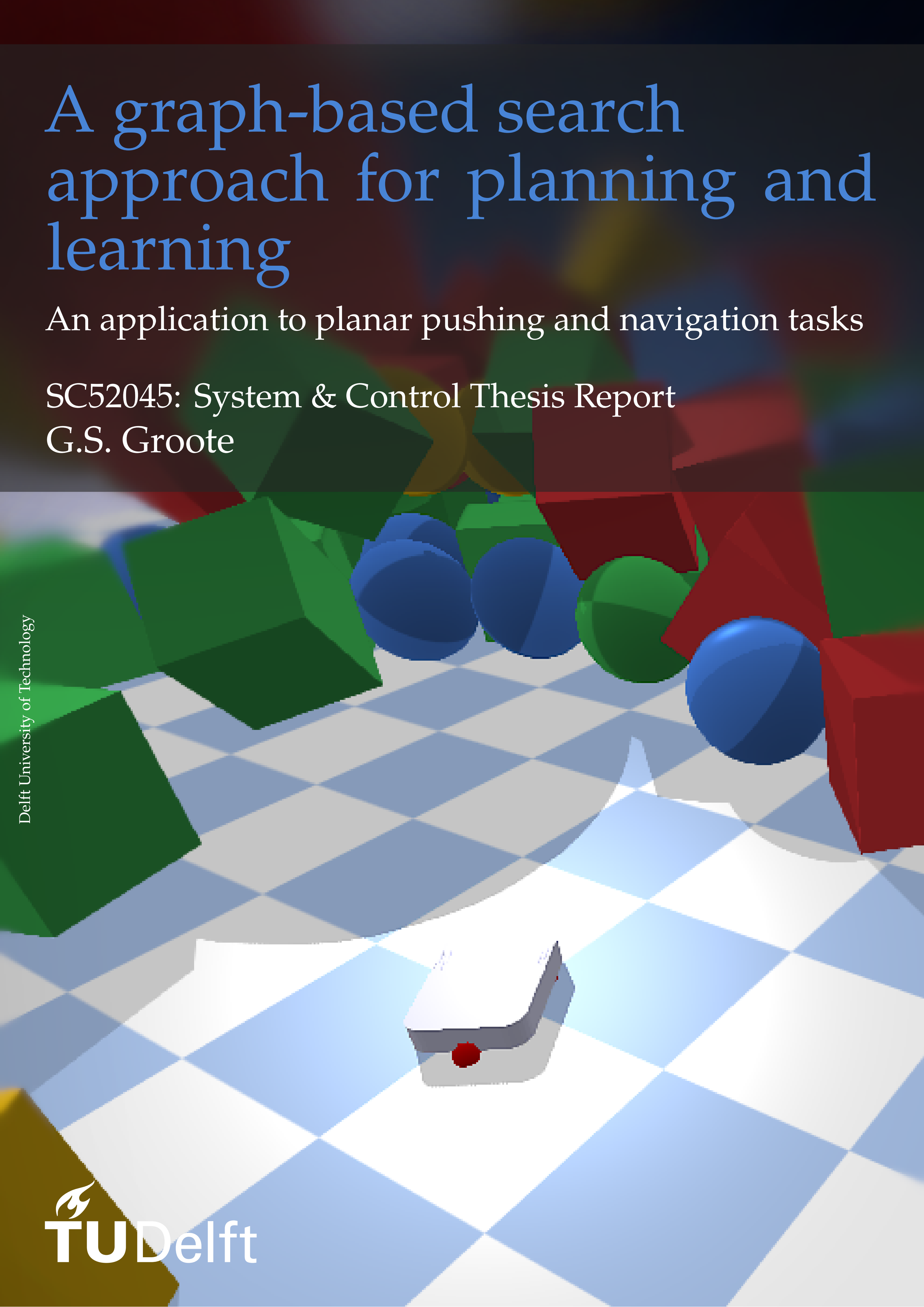
A graph-based search approach for planning and learning

An application to planar pushing and navigation tasks

SC52045: System & Control Thesis Report

G.S. Groote

Delft University of Technology



TUDelft

page intentionally left blank.

A graph-based search approach for planning and learning

An application to planar pushing and navigation tasks

by

G.S. Groote

Student Name	Student Number
Gijs S. Groote	4483987

Supervisors:	C. Smith, M. Wisse
Daily Supervisor:	C. Pezzato
Project Duration:	Nov, 2021 - Feb, 2023
Faculty:	Faculty of Cognitive Robotics, Delft

Cover: Simulation environment used during the thesis [27].

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

Abstract

In the field of robotics, much attention has been given to the research topics *learning object dynamics* [6, 26] , *Navigation Among Movable Objects (NAMO)* [5, 7, 12, 14] and *nonprehensile pushing* [2, 3, 15, 28, 29, 30]. However, because scientific papers that include all three topics are scarce, the combination of these research topics into one robot framework has not been explored sufficiently. A combination of the topics leads to an improvement in planning, faster execution times and rapidly concluding unfeasibility for a robot acting in new and unforeseen environments.

To solve the problem presented in the first paragraph, this thesis proposes a robot framework that combines these three research topics. This framework comprises of three key components: the **hypothesis algorithm**, the **hypothesis graph**, and the **knowledge graph**. The hypothesis algorithm is used to draw a hypothesis on how to relocate an object to a new pose by computing possible action sequences given certain robot skills. In doing so, the hypothesis algorithm creates an hypothesis graph that encapsulates the structure of the action sequences and ensures the robot eventually halts. Once an hypothesis is carried out on the robot, information about the execution, such as the outcome, the type of controller used and other metrics, are stored in the knowledge graph. The knowledge graph is populated over time, allowing the robot to learn for instance object properties and then refine the hypothesis to increase task performance such as success rate and execution time.

The hypothesis algorithm relies on planning to generate hypotheses, for the purpose of planning, and freeing blocked paths that can be encountered, a new planning algorithm is proposed. This planner is an extension of the double tree optimised Rapidly-exploring Random Tree algorithm [5]. The planners both construct a configuration space for an object and are provided with starting and target pose for that object. The planners then convert these poses to points in configuration space and searches for a path connecting the starting point to the target point. A key difference between the newly proposed planner and the existing planner lies in the ability to detect paths that are blocked. For the new planner, objects are initially classified as “unknown” and can later be categorized as either “movable” or “obstacle”. The object type information is then used when constructing the configuration space for the newly proposed planning algorithm. Its configuration space consists of the conventional free- and obstacle space and additionally the unknown- and movable space.

To carry out the investigation, a mobile robot in a robot environment is created with movable and unmovable objects. The robot is given a task that involves relocating a subset of the objects in the robot environment through driving and nonprehensile pushing. The task can be broken down into individual subtasks that consist of an object and a target pose. Planning for a push or drive action occurs with the newly proposed planning algorithm that if successful, provides a path that can be tracked.

It can be concluded that the three topics can be combined into a robot framework because results indicate that task execution improves as the robot gains more experience in the environment it operates. The proposed framework, which covers all three research topics, performs equivalent or better compared to the state-of-the-art frameworks that are specialised in only two out of three research topics [8, 24, 25, 32, 34].

G.S. Groote
Delft, May 2023

Contents

List of Figures

List of Tables

Symbols

$\mathbb{R}$	Set of Real numbers
$\mathbb{Z}_{\geq 0}$	Set of non-negative integers
n	Number of Degrees Of Freedom
obj	Object in the robot environment
Obj	Set of objects
m	Number of objects in the environment
Origin	Origin of the environment with with x (north to south), y (west to east) and z (down to up) axis
Ground Plane	The robot environment ground plane
Eq	Set of motion equations in the robot environment
k	time step index
ϵ^{pred}	Prediction Error
ϵ^{track}	Tracking Error
C-space	Configuration Space, Dim(C-space) $\in \mathbb{R}^2 \vee \mathbb{R}^3$
c	Configuration, point in configuration space
$\hat{c}$	Estimated configuration
x	distance to Origin over the x-axis
y	distance to Origin over the y-axis
θ	angular distance toward the the Origin's positive x-axis around the z axis
x_{grid}	length of grid in direction of x-axis
y_{grid}	length of grid in direction of y-axis
s_{cell}	length and width of a cell
v	Node in motion or manipulation planner
V_{MP}	Set of nodes for motion or manipulation planner
E_{MP}	Set of edges for motion or manipulation planner
P	Set of paths
S	Task, Tuple of objects and corresponding target configurations.
s	subtask, tuple of an object and an target configuration
h	hypothesis, Sequence of successive edges in the hypothesis graph, an idea to put a object at it's target configuration
$G^{hypothesis}$	hypothesis graph
$G^{knowledge}$	knowledge graph

v	Node in hypothesis or knowledge graph
V_H	Set of nodes for hypothesis graph (hgraph)
V_K	Set of nodes for knowledge graph (kgraph)
e	Edge hypothesis or knowledge graph
E_H	Set of edges for hgraph
E_K	Set of edges for kgraph
NODE_STATUS	Status of a node
EDGE_STATUS	Status of a edge
OBJ_CLASS	classification of an object
ob	observation from the robot environment
α	Success factor for an edge in kgraph

1
Introduction

This chapter narrows the broad field of robotics down to a largely unsolved problem, combining the three topics learning object dynamics, NAMO and nonprehensile pushing. For that problem, the state-of-the-art methods are presented and their shortcomings are highlightedin the upcoming paragraphs. Then the gap in literature regarding the combination of the aforementioned topics is summarized in Section 1.1 in the form of the main and sub research questions. The combination of the three topics is then narrowed down to the scope of this thesis in the problem description, Section 1.2. The chapter finishes by presenting all upcoming chapters in the report structure, Section 1.3.

MARTIJN: you introduce a lot of topics here, and make a lot of bold statements, all of which should be supported by relevant literature references.

Martijn: You should really structure your text so that you have one message per paragraph.

For robots, it remains a hard problem to navigate and act in new, unseen environments. From the emerged challenges robots face in such envirionments, three topics are selected, namely: **learning object dynamics**, **Navigation Among Movable Objects (NAMO)** and **nonprehensile pushing**. The main goal of this thesis is to combine these three topics, a secondary goal is to investigate how these topics can strengthen each other over time. When investigating the influence of the topics on each other, questions arise such as. How does learned environmental knowledge influence planning? Does the execution time decrease and how much for a repeated task? How much can generated action sequences improve as the robot learns more?

Learning object dynamics enables the robot to manipulate unforeseen objects, NAMO allows the robot to move around in an environment even if the robot's target location is blocked by an object, and nonprehensile pushing allows the robot to change the environment. Combining these 3 topics covers any task that involves relocating objects by pushing, which is a wide variety of tasks. Examples are clearing debris in construction sites or war zones or cleaning through pushing trash in one spot.

maximal 2 examples per topic, –> ratjetoe

Learning abilities allow robots to operate in completely new environments and improve robots to adapt to environmental changes. Such learning abilities are crucial when the robot can encounter many different objects. Examples are exploration or rescue missions in collapsed buildings, but also in more everyday robot applications. An unfamiliar environment can emerge from a familiar environment due to some unforeseen change that the robot is not aware of. An example is a leakage that changes the friction coefficient between the floor and everything standing on it. Another example are supermarkets, due to the presence of people in the supermarket the environment changes, providing a slightly new environment for the robots that operate in them.

NAMO and nonprehensile pushing in some cases are the same, make a point that they are not becuase planning is required for the push

NAMO should be here as you have learning above and pushing belwo

Nonprehensile pushing is a form of manipulation that is widely available for robots, even though they are not intentionally designed for pushing. Mobile robots can drive (and thus push) against objects, and a robot arm with a gripper can push against objects even if the gripper is already full. Many robots can push, and pushing is a manipulation action many robots should leverage.

Martijn(who is appeareantly not convinced)I am not convinced that you can really split all relevant research in these two categories, and they also don't sound like mutually exclusive categories.

Research into approaches tackling the 3 topics just described can be split into two categories. The bulk falls into the category of hierarchical approaches [8, 13, 25, 32, 34]. The remainder falls in category locally optimal approaches [17, 23, 24]. Both approaches are elaborated upon later in this chapter. First, the configuration space is discussed that build up to the composite configuration space. Then second, two challenges are highlighted related to the composite configuration space growth of dimension and the fact that the composite configuration space is piecewise-analytic.

Configuration Space Now planning for a single action is investigated, and its relation to the configuration space. Finding a path for a single action (such as robot driving or robot pushing) is known as a *motion-* or *manipulation planning problem* and is planned in configuration space. *Configuration space* for an object *obj* can be described as an *n*-dimensional space, where *n* is the number of degrees of freedom for that single object *obj*. A point in this *n*-dimensional configuration space fully describes where that object is in the workspace. Then the workspace obstacles are mapped to configuration space to indicate for which configurations the object *obj* is in collision with an obstacle in the workspace. The subset of configurations in configuration space for which *obj* is in collistion is called *obstacle space*. The remainder of obstacle space subtracted from configuration space is free space, in which the object can move freely. For every object in the environment, a configuration space can be constructed. A mathematical description of configuration space can be found in **??**. When a configuration space is constructed, dedicated path planners leverage the configuration space to determine if a configuration lies in free or in obstacle space.

Composite Configuration Space Now planning for an action sequence is investigated, and its relation to the composite configuration space. For a robot environment that involves relocating objects among the presence of other movable objects planning for a single action is not enough, because manipulating an object directly to a target position is in many cases, unfeasible. As a simple example, manipulating an object obj_A to a target position is feasible if the object, obj_B that blocks that path is first removed. Clearly, removing blocking object obj_B influences the feasibility of manipulating obj_A to its target position. For planning there must be a connection between the configuration space of obj_A and obj_B, which is where the composite configuration space emerges. A *composite configuration space* or composite space emerges when an objects configuration space is augmented with the configuration space of other objects [31]. A composite configuration in such a composite space fully describes where the robot, and objects are in the workspace. In recent literature the composite configuration space is also named composite configuration space [32], finding "bridges" between configuration spaces [9] or room configuration [24]. The term composite configuration space has been selected because it indicates multiple configuration spaces composed together and does not confuse with joints (or hinges) of a robot. Path planners (in contrast to the planning in configuration space) have great difficulty to connect a starting composite configuration to a target composite configuration [22] for reasons that are discussed in the upcoming paragraph.

Challenges The first challenge with the composite configuration space is that is grows exponentially with the number of movable objects in the environment. An analysis is now provided with a robot environment in which both the robot and objects all have an two dimensional configuration space. The composite configuration space becomes $2(m+1)$-dimensional, with m is the number of objects in the environment. Thus the dimension of the composite space grows linearly with the number of objects in the robot environment, the composite configuration space itself grow exponentially, also known as the *curse of dimensionality*.

The second challenge is due to constraint sets, in configuration space a single constraint set applies. As an example, nonholonomic robots must respect a set of constraints due to the robots inability to drive in any direction. The class of robots that can be described by a bicycle model [19] are nonholonomic. These robots cannot drive sideways, such as the boxer robot in Figure 1.1b. The nonholonomic constraints must be respected during planning, for the planner to yield feasible paths. In configuration space there exist only a single mode of dynamics [9], and path planners such as PRM [10], PRM*, RRT or RRT* [11] connect start to target configuration in configuration space whilst respecting the constraint set. In composite space there exist multiple modes of dynamics that must be respected. A drive action is connected to a different set of constraints than a push action. These different modes of dynamics make the composite configuration space *piecewise-analytic*. Path planners have great difficulty crossing the boundary from one mode of dynamics to another mode of dynamics [32].

Corrado: talk about multi-modal planning

Finding an optimal solution to a NAMO and pushing task requires a search in the composite configuration space. Apart from the fact that the composite configuration space becomes exponentially larger with every increase of objects, there is another problem.

Martijn: It doesn't make sense to me to suddenly talk about dynamics while everything else up to this point was quasistatic because you talked about configuration spaces and planning in those spaces.
Gijs introduce dynamics on top of planning

Corrado: Cite roy, because When you say all this, you can easily refer to papers where they have the same definitions. I think nicoolas Roy uses this term a lot. It will make your claims and definition more robust.

The composite configuration space is , which is explained below. Drive and push actions are translated to the composite configuration space as subspaces. A certain subspace of the composite configuration space is assigned to robot driving and another subspace is assigned to robot pushing. These different subspaces in composite configuration space are called different modes of dynamics. In the driving mode of dynamics, a set of driving constraints must be respected, and the pushing mode has a set of push constraints that must be respected. Such constraints originate from multiple sources, such as the robot being nonholonomic, the robots and objects properties (e.g. geometry, weight distribution), friction coefficient between objects. Multiple modes of dynamics introduce a discontinuity in the constraints, hence the composite configuration space is a piecewise-analytic.

Finding paths for multiple actions (such as first robot driving and then robot pushing) is known as multi-model planning [9] and is planned in composite configuration space.

Appendix A contains an explanation on complexity classes which may be helpful to better understand this paragraph. As mentioned before finding an optimal solution to a NAMO and pushing task requires a search in composite configuration space. Finding an optimal solution falls in category of non-deterministic polynomial-time hard (NP-hard) problems, motivation is provided with the following simplification. If the search for an optimal solution in composite configuration space is simplified by completely removing relocating objects to new positions from the task, a purely NAMO problem is what remains. If the problem is simplified even further, by assuming that every object is an unmovable obstacle, the problem falls in the category of NP-hard problems because a reduction exists from the piano mover's problem which is known to be NP-hard [21]. That a simplified version is NP-hard indicates how difficult it is to find an optimal path in composite configuration space.

The last problem that is introduced is the uncertainty of actions in unknown environments. Planning an action sequence with limited or no environmental knowledge inevitably leads to unfeasible action sequences, such as pushing unmovable obstacles. Updating the environmental knowledge and replanning the action sequence is the cure to the uncertainty introduced by a lack of environmental knowledge. Trying to complete unfeasible action sequences is time and resources lost. Additionally, it can lead to the task itself becoming unfeasible. For example, a pushing robot ends up pushing an object into a dead end due to an action sequence planned with limited environment knowledge. Now that the object is in a stuck position the task has become unfeasible.

Corrado: above and below could be the same paragraph

tasks introduces a introduces a NP-hard planning problem [35]. Such an environment that consists of multiple objects

To summarise, the main challenge is to find an action sequence for a given task to relocate objects that consist of push and drive actions. To find such an action sequence a path from the start configuration to a desired target configuration in the composite configuration space is sought, where all specified objects are at their specified target position. The emerging challenges are the enormity of the composite configuration space, the different modes of dynamics that make the composite configuration space piecewise analytic, and lastly the uncertainty introduced by the lack of environmental knowledge.

Corrado: Why now and not before for instance? Instead of saying this, you can say that a number of researchers tackled this problems. In the following we provide a categorization and a summary of the methods, advantages and disadvantages of the most relevant sota

Now the state-of-the-art methods are discussed that can be categorized into two categories, namely locally optimal and hierarchical approaches, first locally optimal approaches are discussed.

Locally Optimal Approaches As has been indicated in the previous paragraph, finding a path in the composite configuration space cannot computationally be found in reasonable time (orders of magnitude slower than real-time, with no guarantees if no path exists). Only by leveraging simplifications applied to the composite configuration space, a search can be performed, such as considering a heavily simplified probabilistic environment [31], considering a single manipulation action [4], discretization [23] or a heuristic function combined with a time horizon [23]. Such techniques prevent searching in configurations relatively far from the current configuration. Local optimality guarantees can be given and real-time implementations have been shown.

Corrado: sabbagh has 3 papers, connect them: Corrado: Looks like 22, 23 and 18 are all connected. I miss to see the links, in the sense that the explanation is spread. Novin first started with 22 I guess, then built on topo of it in 18 and 23. But you start with 23. This requires better structuring

Corrado: It would be nice to have a short overview of the whole method before diving into details later

The most relevant locally optimal approach is presented by Sabbagh Novin et al. [24]. She presents an optimal motion planner that avoids obstacles in the workspace and respects kinematic and dynamic constraints of a robot arm [23]. Examples of the motion planner are provided using a 3- and 4- Degrees Of Freedom (DOF) planar robot arm. Sampling in the composite configuration space is simplified using discretization (by disjunctive programming)

Corrado: cite disjunctive programming instead of throwing it in here

of the composite configuration space and by using a receding horizon. The disjunctive programming concept is applied for converting the continuous problem of path planning into a discrete form. In other words, a continuous path is made equivalent to some points with equal time distances which represent the entire path. After discretization the composite configuration space remains intracable. Thus a search is performed close to the current configuration by combining a heuristic function with a receding horizon concept. A specially developed heuristic function points *toward* a target configuration, the planner then plans between the current configuration and a point toward the target configuration for a predetermined time horizon. The concept of a receding horizon is used to obtain the optimal path for every time step in the time horizon, but apply only the first term and repeating this process until the end-effector meets the final position.

The optimal motion planner [23] is then converted toward path planning for a nonholonomic mobile robot with a gripper [17]. With the 3-fingered gripper, the robot can grasp legged objects such as chairs or walkers. The targeted workspace is a hospital, where the robot is tasked with handing walkers (or other legged objects) to patients to lower the number of falling patients. The variety of legged objects motivates an object model learning module that learns dynamic parameters from experimental data with legged objects. The dynamic parameters are learned using a Bayesian regression model [25]. An MPC controller then tracks the path and compensates for modeling errors. A key contribution is that the planner can decide to re-grasp one of the object's legs to improve path tracking.

Real world experiments show the effectiveness of Novin's locally optimal approach [24]. She has presented a manipulation planning framework focused on moving legged objects in which the robot has to choose between which leg to push or pull. The framework can operate in real-time, and the local optimality has been shown. From the 3 topics that this thesis focuses on, Sabbagh Novin et al. includes learning object dynamics and prehensile manipulation of objects to target positions, missing only the NAMO problem because a path is assumed to be free during object manipulation. Because Novin uses a gripper to manipulate objects, her research falls into the category of prehensile manipulation. Nonprehensile manipulation bears an additional challenge over prehensile manipulationthat is due to the contact points. With prehensile manipulation a gripper ensured multiple contact points, geometrically locking the object with respect to the gripper. The gripper and gripped object can be threated as a single object untill the gripper opens. With nonprehensile manipulation the object is not geometrically locked into place, adding an additional challenge for nonprehensile manipulation over prehensile manipulation..

check the last sentece above with chat u know what

Hierarchical Approaches The second class of approaches to finding a path in composite configuration space is classified as hierarchical approaches [8, 13, 25, 32, 34] that can be described as follows. A hierarchical structure generally consists of a high-level and a low-level component. The high-level task planner has an extended time horizon which includes several atomic actions and their sequencing. Whilst a low-level controller acts to complete a single action in a single mode of dynamics e.g. drive toward object, push object. The high-level planner has a prediction horizon consisting of an action sequence, a long prediction horizon compared to the low-level planner whose prediction horizon is at most a single action.

The most relevant hierarchical approach is presented by Scholz et al. [25]. He presents a planner for the NAMO problem that can handle environments with under-specified object dynamics. The robot's workspace is split into various regions where the robot can move freely. Such regions can be connected if an object separating two regions and can be manipulated by the robot, to connect both regions. The manipulation action is uncertain because objects have constraints that the robot has to learn, e.g. a table has a leg that only rotates, but cannot translate. A Markov Decision Process (MDP) is chosen as a graph-based structure, where the nodes represent a free space region and objects separating the regions are edges in the MDP. Finding a solution for the MDP, leads to an action sequence consisting of a number of drive and object manipulation actions to eventually drive the robot toward a target position. The under-specified object dynamics introduce uncertainty in object manipulation. During action execution object constraints are captured with a physics-based reinforcement learning framework that results in improving manipulation planning when replanning is triggered.

Scholz et al. presented a NAMO planner that makes use of a hierarchical MDP combined with a learning framework, resulting in online learning of the under-specified object dynamics. The method's effectiveness in learning and driving toward a target location has been shown by an implementation on a real robot. From the 3 topics that this thesis focuses on, Scholz et al. includes learning and the NAMO problem, missing only push manipulation toward target locations. By not including manipulation of objects to target positions Scholz et al. can find a global path without running into high dimensional spaces. In other words, by driving only the robot toward a target location a global path will encounter objects only once. By running into objects only once the manipulation of an object does not affect the feasibility of the global path, hence the simplification.

Corrado: This is the same paper as above right? Why is there a blank line. Merge and avoid repetitions

Both local optimal and hierarchical approaches have been discussed, both having their advantages and disadvantages. Local optimal approaches converge to a local optimal plan. To avoid the curse of dimensionality simplifications must be used to sample the composite configuration space in order to be computationally feasible. Such simplifications determine the quality of solutions found. Hierarchical structures generally provide solutions which are computationally efficient but are hierarchical, meaning the solutions found are the best feasible solutions in the task hierarchy they search. The quality of the solution depends on the hierarchy which is typically hand-coded and domain-specific [32].

The most relevant work for local optimal [24] and hierarchical [25] approaches are discussed. They both combine two of the three topics (one learning, and two NAMO problem). Note both relevant works focus on prehensile manipulation, whilst this thesis focuses on nonprehensile push manipulation. Individually a considerable amount of research is done on these 3 topics (NAMO [5, 7, 8, 12, 14, 34], nonprehensile push manipulation [2, 3, 15, 28, 29, 30], learning object dynamics [6, 26]).

Martijn: YOU are repeating yourself, Corrado: agreed you repeat yourself

Combining two topics received little attention by the scientific community and combining all three topics (to the best of my search) not at all. Table 1.1 presents state-of-the-art-literature and which portion of the three topics they include in their research.

Martijn: Rightkmost column should be checks and marksa

Choose: Specify object target positions or nonprehensile pushing

			NAMO		Specify object target positions	
Author	Citation	Learns object dynamics	prehesile	nonprehesile	prehesile	nonprehesile
Ellis et al.	[8]	✓	✗	✓	✗	✗
Sabbagh Novin et al.	[24]	✓	✓	✗	✓	✗
Scholz et al.	[25]	✓	✓	✗	✗	✗
Vega-Brown and Roy	[32]	✗	✓	✗	✓	✗
Wang et al.	[34]	✗	✗	✓	✗	✗
Groote	Proposed Framework	✗/✓	✗	✓	✗	✓

Table 1.1: Overview of 3 topics in recent literature and their object manipulation, where *grasp-push* and *grasp-pull* refer to prehensile push and pull manipulation, *gripped* refers to fully gripping and lifting objects for manipulation, *pushing* refers to nonprehensile push manipulation. The proposed framework shows ✗/✓for learning system dynamics because it proposes system identification to generate a system model, however for the implementation an hardcoded system model is used.

1.1. Research Question

To investigate the effect of learning on action selection and action planning the following research questions have been selected.

Main research question:

How do learned objects' system models improve global task planning for a robot with nonprehensile push manipulation abilities over time?

The main research question is split into two smaller more detailed subquestions.

Research subquestion:

1. How to combine learning and planning for push and drive applications?
2. How do learning system models and remembering interactions compare to only learning system models? And, how does the proposed framework compare against the state-of-the-art?

The main contribution of this thesis is to combine all three topics. These topics are learning object dynamics, the NAMO problem and nonprehensile push manipulation. The proposed framework combines these 3 topics with the *hypothesis algorithm*. The algorithm builds a graph-based structure with nodes and edges, named the *hypothesis graph*. Planning directly in the composite configuration space is avoided. The hypothesis algorithm plans only in a single mode of dynamics and searches for a global path with a technique known as a backward search [13]. Learned object dynamics are stored in a knowledge base called the *knowledge graph*. The hypothesis algorithm (halgorithm), hgraph and kgraph are introduced in **??**.

1.2. Problem Description

To answer the research questions, tests are performed in a robot environment. A simple environment is desired because that simplifies testing, yet the robot environment should represent many real-world environments in which robots operate, thus a 3-dimensional environment is selected. The environment consists of a flat ground plane since many mobile robots operate in a workspace with a flat floor, such as a supermarket, warehouse or distribution center. An environment with a flat floor and a flat robot can be treated as a 2-dimensional problem because the robot and objects can only change position over x and y axis (xy plane parallel to the ground plane) and rotate around the z axis (perpendicular to the ground plane). A flat robot is selected because it has a low center of gravity, which lowers the chance of tipping over.

Let's start with defining the environment. Let the tuple $\langle \text{Origin}, \text{Ground Plane}, \mathit{Obj}, \text{Eq} \rangle$ fully define a robot environment where:

Origin	Static point in the environment with a x-, y- and z-axis. Any point in the environment has a linear and an angular position and velocity with respect to the origin
Ground Plane	A flat plane parallel with the Origin's x- and y- axis. Objects cannot pass through the ground plane and meet sliding friction when sliding over the ground plane.
Obj	A set of objects, $Obj = (ob_1, ob_2, ob_3, \ldots, ob_i)$ with $i \geq 1$, an object is a 3-dimensional body with shape, can be unmovable or movable. In the latter case, the mass is uniformly distributed. The robot itself is considered an object, an environment thus contains one or more objects. Examples of objects are given in Figure 1.2.
Eq	A set of motion equations describing the behavior of objects.

A configuration consists of the linear position of an object's center of mass with respect to the environment's origin and the angular position of an object's orientation with respect to the environment's origin.

Formally, a **configuration**, $c_{id}(k)$ is a tuple of $\langle pos_x(k), pos_y(k), pos_\theta(k) \rangle$ where $pos_x, pos_y \in \mathbb{R}$, $pos_\theta \in [0, 2\pi)$
k indicates the time step and can dropped to simplify the notation, *id* is short for identifier and indicates the object to which this configuration belongs.

Gijs: you are a tennis ball, stay on a side, MARTIJN: I think that it is incomplete to talk about dynamic equations (in Eq) yet to ignore velocities here in your definition of the problem. In addition to configuration I think that you should talk about state (which encompasses configuration).

1.2.1. Task Specification

Corrado: This is a bit of a useless sentence if I may say. You can simplify a lot of things in your text by just going to the point without going around in circles.
Just say,
To answer the research questions, a number of tasks is designed

The research questions investigates the effect of learning system models and monitor the effect of learned knowledge over time. Thus the robot needs an incentive to learn object properties, and interactions with the objects in its environment, otherwise it would simply remain standing still in its initial location. Therefore the robot is asked to complete a task. A task is defined as a subset of all objects with associated target configurations

$$\text{task} = \langle Ob_{task}, C_{targets} \rangle$$

this suggest that only the first k objects, from Obj, can take part in the task. The way you write it here and in the previous page, it is not possible to skip ob1, and start assigning targes from ob2 onwards.

Corrado: Ob task = set ofj , see feedback from corrado

where $Ob_{task} = (ob_1, ob_2, ob_3, \ldots, ob_k) \subset Ob$, $C_{target} = (c_1, c_2, c_3, \ldots c_k)$ and $k > 0$.

A task is completed when the robot manages to push every object to its target configuration within a specified error margin.

Corrado: I hope you will write the exact equation for success later in the experimental section

1.2.2. Assumptions

To simplify the pushing and learning problem, several assumptions are taken, which are listed below.

Closed-World: *Objects are manipulated, directly or indirectly only by the robot. Objects cannot be manipulated by influences from outside the environment.*

Perfect Object Sensor: *the robot has full access to the poses and geometry of all objects in the environment at all times.*

Tasks are Commutative: *Tasks consist of multiple objects with specified target positions. The order in which objects are pushed toward their target position is commutative.*

Martijn: this sounds so ad-hoc. At the very least, you should check all simulation results afterward, to ensure that objects did not fall over. Here, you should mention that you will do those checks, and then in the results section you should report that all checks were positive.

Objects do not tip over: *Movable objects slide if pushed.*

The assumptions taken serve to simplify the problem of task completion. Note that in **??** insight is given to remove all assumptions. By removing assumptions completing tasks becomes a harder problem, but a more realistic problem closer to real-world applications.

what is a zero velocity component?

Assumptions might have certain implications, which are listed below. The **closed-world assumption** implies that objects untouched by the robot and with zero velocity component remain at the same position. Completed subtasks are therefore assumed to be completed for all times after completion time.

The **perfect object sensor assumption** simplifies a sensor setup, it prevents Lidar-, camera setups and tracking setups with aruco or other motion capture markers. The existence of a single perfect measurement wipes away the need to combine measurements from multiple sources with sensor fusion algorithms, such as Kalman filtering [33].

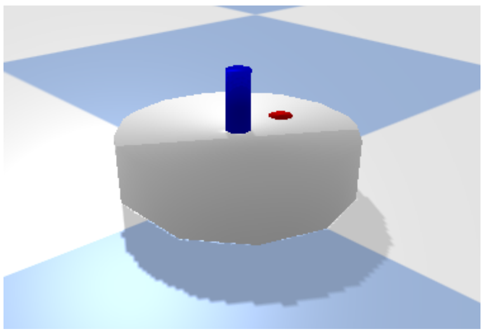
Certain tasks are only feasible if performed in a certain order (e.g. the Tower of Hanoi). The **tasks are commutative assumption** allows focusing only on a single subtask since it does not affect the completion or feasibility of other subtasks.

Corrado: Maybe the assumption should be that a 3d world can be represented as 2d. This connects with what Martijn commented earlier as well and would include objects not tipping over or be lifted from the ground

The **objects do not tip over assumption** ensures that objects do not tip over and suddenly have vastly different dynamics. In practice, objects will not be higher than the minimum width of the object, and spheres are excluded from the environment. This experimental strategy seemed sufficient whilst running experiments but does not guarantee that objects do not tip over.

1.2.3. An Example of Robots and Objects

To get a sense of what the robots and the objects look like, see the two robots that are used during testing in Figure 1.1. And among many different objects, two example objects are displayed in Figure 1.2.


(a) The holonomic point robot the 2 velocity inputs drive the robot in x and in y direction

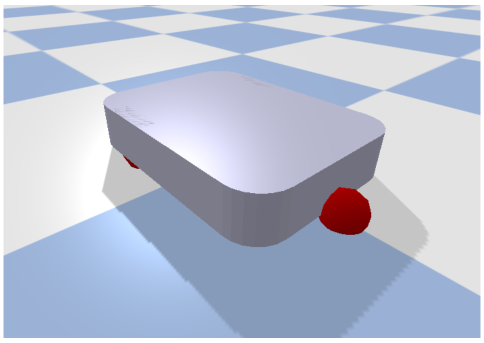
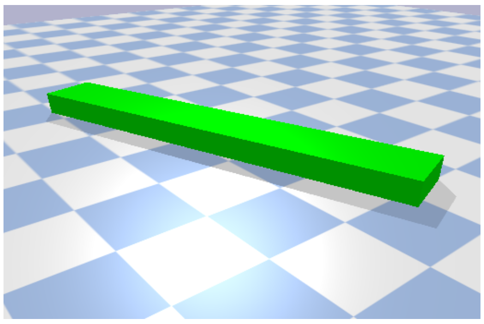

(b) The nonholonomic boxer robot, the input is forward and rotational velocity

Figure 1.1: Robots in the simulation environment


(a) A box object

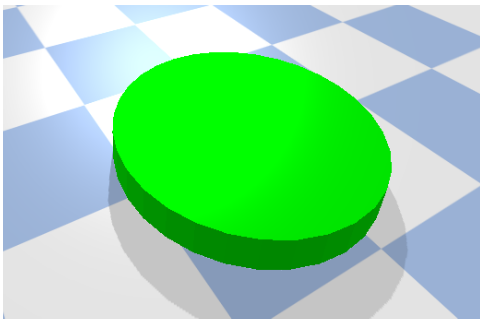

(b) A cylinder object

Figure 1.2: Objects in the robot environment

For complete environments with accompanying tasks, see **??**.

1.3. Report Structure

Corrado: And the performance are tested in Chapter 4.
Simplify the text to avoid repetitions and to shorten it

The proposed framework heavily relies on a number of methods and functions. These methods and functions are conveniently grouped in **??**. Then the proposed framework is presented and discussed in **??**. Testing the proposed framework is presented in **??**. The last chapter draws conclusions on tests and answering the research questions.

Glossary

List of Acronyms

MPPI	Model Predictive Path Integral	ii
MPC	Model Predictive Control	ii
NAMO	Navigation Among Movable Objects	i
NP-hard	non-deterministic polynomial-time hard	3
NP	non-deterministic polynomial-time	16
halgorithm	hypothesis algorithm	7
hgraph	hypothesis graph	vi
kgraph	knowledge graph	vi
RRT	Rapidly-exploring Random Tree	
RRT*	optimised Rapidly-exploring Random Tree	
PRM	Probabilistic Road Map	
PRM*	optimised Probabilistic Road Map	
DOF	Degrees Of Freedom	4
MDP	Markov Decision Process	5

References

[1] Ian Abraham et al. "Model-Based Generalization Under Parameter Uncertainty Using Path Integral Control". In: *IEEE Robotics and Automation Letters* 5.2 (Apr. 2020), pp. 2864–2871. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2020.2972836. URL: https://ieeexplore.ieee.org/document/8988215/ (visited on 01/31/2022).

[2] Ermano Arruda et al. "Uncertainty Averse Pushing with Model Predictive Path Integral Control". In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids). Birmingham: IEEE, Nov. 2017, pp. 497–502. ISBN: 978-1-5386-4678-6. DOI: 10.1109/HUMANOIDS.2017.8246918. URL: http://ieeexplore.ieee.org/document/8246918/ (visited on 04/29/2022).

[3] Maria Bauza, Francois R. Hogan, and Alberto Rodriguez. "A Data-Efficient Approach to Precise and Controlled Pushing". Oct. 9, 2018. arXiv: 1807.09904 [cs]. URL: http://arxiv.org/abs/1807.09904 (visited on 03/01/2022).

[4] Dmitry Berenson et al. "Manipulation Planning on Constraint Manifolds". In: *2009 IEEE International Conference on Robotics and Automation*. 2009 IEEE International Conference on Robotics and Automation (ICRA). Kobe: IEEE, May 2009, pp. 625–632. ISBN: 978-1-4244-2788-8. DOI: 10.1109/ROBOT.2009.5152399. URL: http://ieeexplore.ieee.org/document/5152399/ (visited on 12/05/2022).

[5] Long Chen et al. "A Fast and Efficient Double-Tree RRT\$$\hat{*}$\$-Like Sampling-Based Planner Applying on Mobile Robotic Systems". In: *IEEE/ASME Transactions on Mechatronics* 23.6 (Dec. 2018), pp. 2568–2578. ISSN: 1083-4435, 1941-014X. DOI: 10.1109/TMECH.2018.2821767. URL: https://ieeexplore.ieee.org/document/8329210/ (visited on 04/14/2022).

[6] Lin Cong et al. "Self-Adapting Recurrent Models for Object Pushing from Learning in Simulation". July 27, 2020. arXiv: 2007.13421 [cs]. URL: http://arxiv.org/abs/2007.13421 (visited on 04/06/2022).

[7] Mohamed Elbanhawi and Milan Simic. "Sampling-Based Robot Motion Planning: A Review". In: *IEEE Access* 2 (2014), pp. 56–77. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2014.2302442. URL: https://ieeexplore.ieee.org/document/6722915/ (visited on 11/30/2022).

[8] Kirsty Ellis et al. "Navigation among Movable Obstacles with Object Localization Using Photorealistic Simulation". In: July 2022. URL: https://www.researchgate.net/publication/362092621_Navigation_Among_Movable_Obstacles_with_Object_Localization_using_Photorealistic_Simulation.

[9] Kris Hauser and Jean-Claude Latombe. "Multi-Modal Motion Planning in Non-expansive Spaces". In: *The International Journal of Robotics Research* 29.7 (June 2010), pp. 897–915. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364909352098. URL: http://journals.sagepub.com/doi/10.1177/0278364909352098 (visited on 12/05/2022).

[10] D. Hsu, J.-C. Latombe, and R. Motwani. "Path Planning in Expansive Configuration Spaces". In: *Proceedings of International Conference on Robotics and Automation*. International Conference on Robotics and Automation. Vol. 3. Albuquerque, NM, USA: IEEE, 1997, pp. 2719–2726. ISBN: 978-0-7803-3612-4. DOI: 10.1109/ROBOT.1997.619371. URL: http://ieeexplore.ieee.org/document/619371/ (visited on 05/03/2023).

[11] Sertac Karaman and Emilio Frazzoli. "Sampling-Based Algorithms for Optimal Motion Planning". In: *The International Journal of Robotics Research* 30.7 (June 2011), pp. 846–894. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364911406761. URL: http://journals.sagepub.com/doi/10.1177/0278364911406761 (visited on 03/15/2022).

[12] Zachary Kingston, Mark Moll, and Lydia E. Kavraki. “Sampling-Based Methods for Motion Planning with Constraints”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 1.1 (May 28, 2018), pp. 159–185. ISSN: 2573-5144, 2573-5144. DOI: 10.1146/annurev-control-060117-105226. URL: https://www.annualreviews.org/doi/10.1146/annurev-control-060117-105226 (visited on 05/06/2022).

[13] Athanasios Krontiris and Kostas Bekris. “Dealing with Difficult Instances of Object Rearrangement”. In: July 2015. DOI: 10.15607/RSS.2015.XI.045.

[14] Steven Michael LaValle. *Planning Algorithms*. Cambridge ; New York: Cambridge University Press, 2006. 826 pp. ISBN: 978-0-521-86205-9.

[15] Tekin Meriçli, Manuela Veloso, and H. Levent Akın. “Push-Manipulation of Complex Passive Mobile Objects Using Experimentally Acquired Motion Models”. In: *Autonomous Robots* 38.3 (Mar. 2015), pp. 317–329. ISSN: 0929-5593, 1573-7527. DOI: 10.1007/s10514-014-9414-z. URL: http://link.springer.com/10.1007/s10514-014-9414-z (visited on 01/31/2022).

[16] neuromorphic tutorial. *LTC21 Tutorial MPPI*. June 7, 2021. URL: https://www.youtube.com/watch?v=19QLyMuQ_BE (visited on 05/31/2022).

[17] Roya Sabbagh Novin et al. “Dynamic Model Learning and Manipulation Planning for Objects in Hospitals Using a Patient Assistant Mobile (PAM)Robot”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Madrid: IEEE, Oct. 2018, pp. 1–7. ISBN: 978-1-5386-8094-0. DOI: 10.1109/IROS.2018.8593989. URL: https://ieeexplore.ieee.org/document/8593989/ (visited on 05/05/2022).

[18] Manoj Pokharel. “Computational Complexity Theory(P,NP,NP-Complete and NP-Hard Problems)”. In: (June 2020).

[19] Philip Polack et al. “The Kinematic Bicycle Model: A Consistent Model for Planning Feasible Trajectories for Autonomous Vehicles?” In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017 IEEE Intelligent Vehicles Symposium (IV). Los Angeles, CA, USA: IEEE, June 2017, pp. 812–818. ISBN: 978-1-5090-4804-5. DOI: 10.1109/IVS.2017.7995816. URL: http://ieeexplore.ieee.org/document/7995816/ (visited on 05/03/2023).

[20] James Rawlings, David Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation and Design*. 2nd edition. Santa Barbara: Nob Hill Publishing, LLC, 2020. ISBN: 978-0-9759377-5-4.

[21] John Reif and Micha Sharir. “Motion Planning in the Presence of Moving Obstacles”. In: *26th Annual Symposium on Foundations of Computer Science (Sfcs 1985)*. 26th Annual Symposium on Foundations of Computer Science (Sfcs 1985). Portland, OR, USA: IEEE, 1985, pp. 144–154. ISBN: 978-0-8186-0644-1. DOI: 10.1109/SFCS.1985.36. URL: http://ieeexplore.ieee.org/document/4568138/ (visited on 05/05/2022).

[22] Nicholas Roy. “Hierarchy, Abstractions and Geometry”. May 6, 2021. URL: https://www.youtube.com/watch?v=mP-uK1PFqfI (visited on 05/03/2023).

[23] Roya Sabbagh Novin, Mehdi Tale Masouleh, and Mojtaba Yazdani. “Optimal Motion Planning of Redundant Planar Serial Robots Using a Synergy-Based Approach of Convex Optimization, Disjunctive Programming and Receding Horizon”. In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 230.3 (Mar. 2016), pp. 211–221. ISSN: 0959-6518, 2041-3041. DOI: 10.1177/0959651815617883. URL: http://journals.sagepub.com/doi/10.1177/0959651815617883 (visited on 05/05/2022).

[24] Roya Sabbagh Novin et al. “A Model Predictive Approach for Online Mobile Manipulation of Non-Holonomic Objects Using Learned Dynamics”. In: *The International Journal of Robotics Research* 40.4-5 (Apr. 2021), pp. 815–831. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364921992793. URL: http://journals.sagepub.com/doi/10.1177/0278364921992793 (visited on 05/05/2022).

[25] Jonathan Scholz et al. “Navigation Among Movable Obstacles with Learned Dynamic Constraints”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Daejeon, South Korea: IEEE, Oct. 2016, pp. 3706–3713. ISBN: 978-1-5090-3762-9. DOI: 10.1109/IROS.2016.7759546. URL: http://ieeexplore.ieee.org/document/7759546/ (visited on 04/29/2022).

[26] Neal Seegmiller et al. "Vehicle Model Identification by Integrated Prediction Error Minimization". In: *The International Journal of Robotics Research* 32.8 (July 2013), pp. 912–931. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364913488635. URL: http://journals.sagepub.com/doi/10.1177/0278364913488635 (visited on 02/16/2022).

[27] Max Spahn. *Urdf-Environment*. Version 1.2.0. Aug. 8, 2022. URL: https://github.com/maxspahn/gym_envs_urdf.

[28] Jochen Stüber, Marek Kopicki, and Claudio Zito. "Feature-Based Transfer Learning for Robotic Push Manipulation". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)* (May 2018), pp. 5643–5650. DOI: 10.1109/ICRA.2018.8460989. arXiv: 1905.03720. URL: http://arxiv.org/abs/1905.03720 (visited on 03/15/2022).

[29] Jochen Stüber, Claudio Zito, and Rustam Stolkin. "Let's Push Things Forward: A Survey on Robot Pushing". In: *Frontiers in Robotics and AI* 7 (Feb. 6, 2020), p. 8. ISSN: 2296-9144. DOI: 10.3389/frobt.2020.00008. arXiv: 1905.05138. URL: http://arxiv.org/abs/1905.05138 (visited on 02/24/2022).

[30] Marc Toussaint et al. "Sequence-of-Constraints MPC: Reactive Timing-Optimal Control of Sequential Manipulation". Mar. 10, 2022. arXiv: 2203.05390 [cs]. URL: http://arxiv.org/abs/2203.05390 (visited on 04/29/2022).

[31] Jur van den Berg et al. "Path Planning among Movable Obstacles: A Probabilistically Complete Approach". In: *Algorithmic Foundation of Robotics VIII*. Ed. by Gregory S. Chirikjian et al. Red. by Bruno Siciliano, Oussama Khatib, and Frans Groen. Vol. 57. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 599–614. ISBN: 978-3-642-00311-0 978-3-642-00312-7. DOI: 10.1007/978-3-642-00312-7_37. URL: http://link.springer.com/10.1007/978-3-642-00312-7_37 (visited on 06/24/2022).

[32] William Vega-Brown and Nicholas Roy. "Asymptotically Optimal Planning under Piecewise-Analytic Constraints". In: *Algorithmic Foundations of Robotics XII*. Ed. by Ken Goldberg et al. Vol. 13. Cham: Springer International Publishing, 2020, pp. 528–543. ISBN: 978-3-030-43088-7 978-3-030-43089-4. DOI: 10.1007/978-3-030-43089-4_34. URL: http://link.springer.com/10.1007/978-3-030-43089-4_34 (visited on 06/23/2022).

[33] M. Verhaegen and Vincent Verdult. *Filtering and System Identification: A Least Squares Approach*. Cambridge ; New York: Cambridge University Press, 2007. 405 pp. ISBN: 978-0-521-87512-7.

[34] Maozhen Wang et al. "Affordance-Based Mobile Robot Navigation Among Movable Obstacles". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Las Vegas, NV, USA: IEEE, Oct. 24, 2020, pp. 2734–2740. ISBN: 978-1-72816-212-6. DOI: 10.1109/IROS45743.2020.9341337. URL: https://ieeexplore.ieee.org/document/9341337/ (visited on 06/28/2022).

[35] Gordon Wilfong. "Motion Planning in the Presence of Movable Obstacles". In: *Annals of Mathematics and Artificial Intelligence* 3.1 (Mar. 1991), pp. 131–150. ISSN: 1012-2443, 1573-7470. DOI: 10.1007/BF01530890. URL: http://link.springer.com/10.1007/BF01530890 (visited on 05/01/2023).

[36] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. "Model Predictive Path Integral Control Using Covariance Variable Importance Sampling". Oct. 28, 2015. arXiv: 1509.01149 [cs]. URL: http://arxiv.org/abs/1509.01149 (visited on 05/02/2022).

2 Appendix

The appendix contains additional information that may help better understand the thesis.

A
Complexity Classes

Problems in class P have a solution which can be found in polynomial time, problems in non-deterministic polynomial-time (NP) are problems for which no polynomial algorithms have been found yet, and of which it is believed that no polynomial time solution exist. For problems in NP, when provided with a solution, verifying that the solution is indeed a valid solution can be done in polynomial time. NP-hard problems are a class of problems which are at least as hard as the hardest problems in NP. Problems that are NP-hard do not have to be elements of NP. They may not even be decidable [18]. This thesis or other recent studies in the references do not attempt to find an optimal solution. Instead, they provide a solution whilst guaranteeing properties such as near-optimality or probabilistic completeness. As the piano's mover problem can be reduced to the NAMO problem combined with relocating objects to target positions, the conclusion can be drawn that this NAMO problem is NP-hard.

B

Control Methods

B.1. MPC Control

In recent literature involving predictive methods Model Predictive Control (MPC) methods are dominating, before moving on to MPC and variations of MPC, MPC will briefly be explained. The basic concept of MPC is to use a dynamic model to forecast system behaviour and optimise the forecast to produce the best decision for the control move at the current time. Models are therefore central to every form of MPC. Because the optimal control move depends on the initial state of the dynamic system [20]. A dynamical model can be presented in various forms, let's consider a familiar differential equation.

$$\frac{dx}{dt} = f(x(t), u(t))$$
$$y = h(x(t), u(t))$$
$$x(t_0) = x_0$$

In which $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the input, $y \in \mathbb{R}^p$ is the output, and $t \in \mathbb{R}$ is time. The initial condition specifies the value of the state x at $t = t_0$, and a solution to the differential equation for time greater than t_0, $t \in \mathbb{R}_{\geq 0}$ is sought. If little knowledge about the internal structure of a system is available, it may be convenient to take another approach where the state is suppressed, no internal structure about the system is known and the focus lies only on the manipulable inputs and measurable outputs. As shown in figure B.1, consider the system $G(t)$ to be the connection between u and y. In this viewpoint, various system identification techniques are used, in which u is manipulated and y is measured [20]. From the input-output relation, a system model is estimated or improved. The system model can be seen inside the MPC controller block in figure B.1.

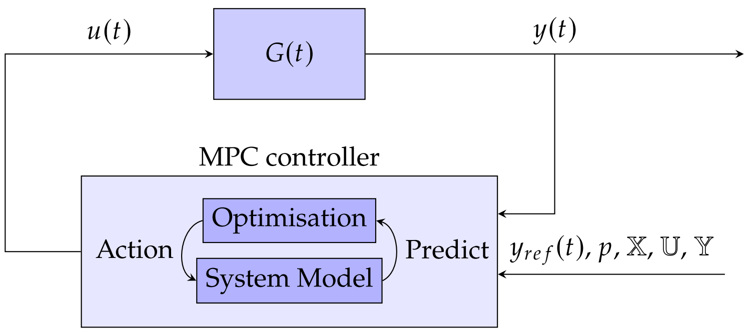


Figure B.1: System $G(t)$ with input $u(t)$, output $y(t)$ and MPC controller with input $y(t)$, reference signal $y_{ref}(t)$, parameterisation p and constraint sets $\mathbb{X}$, $\mathbb{U}$, $\mathbb{Y}$

Some states of the system might be inside an obstacle region, such a region is undesirable for the robot to be in or go toward. The robot states are allowed in free space, which is all space minus the obstacle region. The free space is specified as a state constrained set $\mathbb{X}$. Allowable input can be restricted by the input constraint set $\mathbb{U}$, a scenario in which input constraints are required is for example the maximum torque an engine produces at full throttle. Lastly, the set of allowed outputs is specified in the output constraint set $\mathbb{Y}$. State, input and output constraints must be respected during optimisation, the optimiser takes the state-, input- and output constraint sets $\mathbb{X}$, $\mathbb{U}$, $\mathbb{Y}$ and if feasible, finds an action sequence driving the system toward the reference signal while constraints are respected. The MPC system model predicts future states where the system is steered toward as a result of input actions.

The optimisation minimises an objective function $V_N(x_0, y_{ref}, \mathbf{u}_N(0))$, where $\mathbf{u}_N(k) = (u_k, u_{k+1}, \dots, u_{k+N})$. The objective function takes the reference signal as an argument together with the initial state and the control input for the control horizon. The objective function then creates a weighted sum of some heuristic function. States and inputs resulting in outputs far from the reference signal are penalised more by the heuristic function than outputs closer to the reference signal. Because the objective function is a Lyapunov function, it has the property that, it has a global minimum for the optimal input $\mathbf{u}_N^*$. If the system output reaches the reference signal y_{ref}, x_{ref} then u_{ref} will be mapped to the output reference signal as such $y_{ref} = h(x_{ref}, u_{ref})$. As a result solving the minimisation problem displayed in equation (B.1) gives the optimal input which steers the system toward the output reference signal while at the same time respecting the constraints.

$$\begin{aligned} \underset{u_k, u_{k+1}, \dots, u_{k+N}}{\text{minimize}} \quad & V_N(x_0, y_{ref}, u_k, u_{k+1}, \dots, u_{k+N}) \\ \text{subject to} \quad & x(k+1) = f(x(k), u(k)), \\ & x \in \mathbb{X}, \\ & u \in \mathbb{U}, \\ & y \in \mathbb{Y}, \\ & x(0) = x_0 \end{aligned} \tag{B.1}$$

Figure B.2 displays the predicted output converging toward the constant output reference. After solving the minimisation problem, equation (B.1), the optimal input sequence is obtained $\mathbf{u}_N^*$ (given that the constraints are respected for such input), from which only the first input is executed for time step k to $k+1$. Then all indices are shifted such that the previous time step $k+1$ becomes k, the output is measured and the reference signal, parameterisation, and constraints sets are updated and a new minimisation problem is created, which completes the cycle. Note that figures B.1 and B.2 is an example MPC controller, which hardly scratched the surface of MPC, there are many variations and additions such as deterministic and stochastic MPC, stage and terminal cost, distributed MPC, etc. which [20] visits extensively.

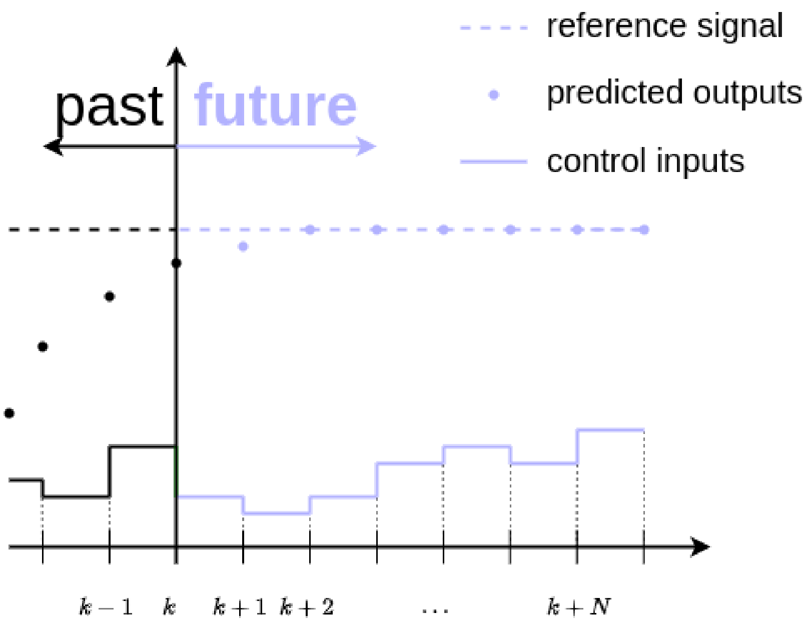


Figure B.2: A discrete MPC scheme tracking a constant reference signal. k indicates the discrete time step, N the control horizon

B.2. MPPI Control

Introduced by [36] MPPI control arose. Which was followed by MPPI control combined with various system models, identification methods [1, 6, 2]. The core idea is from the current state of the system with the use of a system model and randomly sampled inputs to simulate in the future a number of "rollouts" for a specific time horizon, [16]. These rollouts indicate the future states of the system if the randomly sampled inputs would be applied to the system, the future states can be evaluated by a cost function which penalised undesired states and rewards desired future states. A weighted sum over all rollouts determines the input which will be applied to the system. If a goal state is not reached, the control loop starts with the next iteration. An example is provided, see figure B.3.

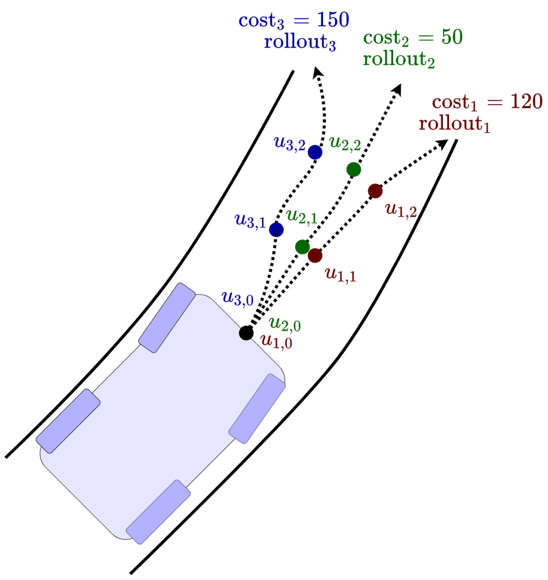


Figure B.3: MPPI controlled race car using a control horizon of 3 time steps, with 3 rollouts all having their respected inputs as $u_{i,j}$ where i is the rollout index and j indicates the time step [16].

Here 3 rollouts are displayed, The objective function is designed to keep the car driving on the center of the road by penalising rollouts which are further away from the center of the road relatively more. resulting in a high cost for rollout_1 and rollout_3 compared to rollout_2. As a result, the input send to the system as a weighted sum of the rollouts is mostly determined by rollout_2. The weighted sum determining the input is displayed in equation (B.2), from [16].

$$u(k+1) = u(k) + \frac{\sum_i w_i \delta u_i}{\sum_i w_i} \tag{B.2}$$

Where δu_i is the difference between $u(k)$ and the input for rollout i, the weight of rollout_i is detemined as: $w_i = e^{-\frac{1}{\lambda}\text{cost}_i}$, λ is a constant parameter.