Robotics and Computer Integrated Manufacturing 53 (2018) 162–177

Contents lists available at ScienceDirect

Robotics and Computer Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

Full length Article

Robust optimization with applications to design of context specific robot solutions

Troels Bo Jørgensen[*,a], Adam Wolniakowski[b], Henrik Gordon Petersen[a], Kristian Debrabant[c], Norbert Krüger[a]

[a] Maersk McKinney Møller institute, University of Southern Denmark, Odense M, 5230, Denmark
[b] Faculty of Mechanical Engineering, Białystok University of Technology, Wiejska 45C, 15-351 Białystok, Poland
[c] Department of Mathematics and Computer Science, University of Southern Denmark, Odense M, 5230, Denmark

ARTICLE INFO

Keywords:
Robust optimization
Dynamic simulation
Robotic manipulation

ABSTRACT

This paper presents an investigation of five optimization algorithms for simulation-based optimization for robotic tasks, where robust solutions are required. We evaluate the optimization methods on three use cases. The use cases involve using a robot for handling meat, optimizing gripper design for aligning objects and optimizing gripper design for table picking in cluttered scenes. We use dynamic simulations to model the use cases, where the most important physical aspects are captured. We have a focus on the robustness with respect to crucial system uncertainties, which is important in an industrial setting. The choice of parameterization and objective scores is also discussed since this choice has some impact on the performance of the optimization algorithms. For all problems, we find feasible solutions ready for real world testing, and overall the optimization method RBFopt has the best performance in terms of finding robust solutions within the fewest amount of simulations.

1. Introduction

Ensuring low setup times for industrial robotic systems which contain uncertainties is a difficult problem. For instance, it is a fairly simple task to build a robotic setup for picking objects whose poses are exactly known, but if there is an uncertainty in the pose of the objects (e.g. when computer vision is used to determine the pose) it might become difficult to build such a system. The reason for this is that a potential solution has to be validated for a substantial number of object poses, to ensure that it is satisfactory from an industrial viewpoint. Furthermore, if the solution is designed in an iterative process it might be difficult to ensure each iteration doesn't cause a new problem for a pose that was successfully grasped in an earlier iteration. A simple example could be that the gripper tend to fail on grasps where the object is placed to the left of the believed pose, because the gripper collides with the objects. A solution could be to move the gripper more to the left, but this might results in the gripper failing on objects that are placed to the right of the believed pose. This way it can be difficult to determine appropriate design modifications unless many grasps are tested in every iteration of the design process.

When analyzing the problem abstractly, four key concepts become apparent: Firstly, the "system", which is the robotic setup and the mechanical interactions within the setup; secondly, the "parameterized solution", which is the way the problem is solved, formulated such that it is tunable to a specific task; thirdly, the "objective", which expresses the quality of the individual solutions; and lastly, the "uncertainties" which cover the variations in the system.

To automatically design solutions ready for integration, we propose to combine simulation with deterministic numerical optimization, enhanced by Monte-Carlo simulation to account for uncertainty in the parameters, in order to determine potential candidate solutions. In this paper, we propose to use dynamic simulation to model the "system" and evaluate the "objective", since experience shows that evaluation based on well designed dynamic simulation is comparable to real world testing [1] and [2]. Simulation also enables a more advanced analysis of the system and gives complete control of the experiments. Furthermore, as long as the dynamic simulation is set up in a general manner, it is often a simple task to make slight modifications to the simulation, in order to model similar tasks. This last part is important if simulation is to be used to reduce set-up times when designing solutions for similar tasks. This problem occurs frequently in industrial assembly, e.g. in case of slight design changes of the product to be assembled.

After the simulation is designed, the "parameterized solution" has to be defined, such that an optimization algorithm can find a good instance of the solution. The parameterization should be made in a general manner, such that the same parameterization can solve multiple

* Corresponding author.
E-mail address: trjoe@mmmi.sdu.dk (T. Bo Jørgensen).

https://doi.org/10.1016/j.rcim.2018.04.005
Received 14 May 2017; Received in revised form 4 April 2018; Accepted 5 April 2018

similar tasks as long as the parameters are properly tuned. This is again beneficial if the goal is to solve several similar industrial tasks, such as setting up robots for picking different kinds of cylindrical objects.

The next step is to model and parameterize the "uncertainties" in the system, such that Monte-Carlo simulation can be used to capture the effects of varying the uncertain parameters. Lastly, an evaluation procedure for the task has to be defined, in order to determine the "objective". This is done by determining appropriate quality indices based on the simulation. These indices are then combined into a single objective score, and a robust version of this score can be determined based on Monte-Carlo simulation where the uncertain parameters are varied. Since the procedure described above requires a Monte-Carlo simulation in every iteration of the optimization algorithm, it becomes computationally expensive. This is undesirable since it is likely to affect the overall setup-time of the system. Therefore we propose to minimize the amount of simulations used in the Monte-Carlo simulation during the optimization, and ideally one only uses one simulation. We then propose to find multiple candidate solutions, based on optimizations with different starting conditions. Lastly, we propose to determine the best of these candidate solutions, based on more thorough Monte-Carlo simulations using a substantial amount of simulations for each evaluation.

To explore this approach, we investigate three use cases, which are illustrated in Fig. 1. The use cases are inspired by real industrial problems that contain a substantial amount of uncertainty. In the first case, a robot has to move meat pieces, but the size, shape and deformability of the meat pieces vary significantly. This uncertainty has to be considered in order to ensure solutions that almost always work well. Such a solution is considered a robust solution, and finding such solutions is considered robust optimization. The other two cases are about designing grippers for handling operations, which are common operations in robotic systems [3]. In both of these cases, the pose of the object to be grasped is uncertain, and this has to be considered in order to produce robust solutions.

In order to reduce the amount of simulations required in the Monte-Carlo simulations during optimization, it is advantageous to design the objectives such that they are favoring robust solutions. This way, for the meat handling case, only one simulation was used per evaluation. In the gripper design cases 50 and 100 simulations were necessary.

In order to minimize the amount of Monte-Carlo simulations, it is advantageous to pick an optimization algorithm that improves quickly and tends to find robust solutions. Therefore we have evaluated multiple optimization algorithms for all problems, in order to find the most suited algorithm for the use cases presented in this paper.

The main concepts and methodology of the paper have now been introduced. In the remainder of this section, we will discuss the use cases in more detail, describe concepts relevant for optimization and summarize key contributions of this work.

In the first use case [4], a robot has to pick a deformable meat piece and place it on a conveyor belt, this use case will be referred to as "deformable manipulation", (see Fig. 1 to the left). In the second use case [5,6], a robot has to pick a cylindrical object, in a way such that the orientation of the cylindrical object is controlled. This case will be referred to as "cylindrical alignment" (see Fig. 1 in the middle). In the last use case, a robot has to pick a T-shaped object reliably, while ensuring a large coverage of the object. A high coverage is important to ensure multiple grasp options, in case some of them are infeasible due to clutter. This case will be referred to as "cluttered table picking" (see Fig. 1 to the right).

In Fig. 1, it can be seen that all cases have some control parameters, some uncertainties regarding the objects and some criteria for how well the task is solved, and based on this, numerical optimization can be used to determine satisfactory solutions.

A key difference between the use cases is the number of actions modeled to determine an objective score. In the "deformable manipulation" use case, it is important that the placement of the objects match a desired placement. Thus only one pick and place action is required per evaluation, to determine the quality of a solution. When one action is used for the evaluation, the determined objective score will be sensitive to variations in the task, and thus the score is not particularly robust. In the "cylindrical alignment" use case, it is important that the objects are aligned similarly after the grasp. Thus roughly 50 grasps are required per evaluation, in order to compute a similarity of alignment score. This makes the objective function more robust. For the "cluttered table picking" scenario it is important to avoid collision with other objects. Therefore it is important that the gripper can grasp the objects from multiple angles. Because of this, it is necessary to evaluate a gripper based on multiple grasp directions. Thus, roughly 100 grasps are required per simulation in order to ensure an objective function that captures the quality of the gripper design. This also makes the objective function substantially more robust to uncertainties in the task.

All three use cases are representatives of different classes of tasks. So the goal is not just to solve a single use case, but to find methods for solving a large range of these problems, such that hardware and robot motions can be automatically designed in a context specific manner. The results also indicate good practices for solving robotic optimization problems, where robustness is important.

In order to apply optimization to these use cases, the solutions have been parameterized and during the simulation, various objective criteria are evaluated and combined to a single objective score. The solution parameters are bounded in order to capture the feasible solutions in the search space.

This enables the use of so called "constrained global optimization", which is a broad topic that has been investigated in, e.g., [7] and [8]. The goal of constrained global optimization is to find the solution, $\mathbf{x}$, that returns the highest objective score, $f(\mathbf{x})$, where the solutions to check are limited by some bounds $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$. This can be formulated as follows:

$$\mathbf{x}_{opt} = \underset{\mathbf{x}\in\mathbb{R}^n | \mathbf{x}_{min} \leqq \mathbf{x} \leqq \mathbf{x}_{max}}{\operatorname{argmax}} f(\mathbf{x}) \tag{1}$$

A limitation of global optimization is that it does not directly address the effects of uncertainties in the system. However, it is possible to design $f(\mathbf{x})$, such that it captures the effects of the uncertainties, as done in robust optimization [9]. The goal of robust optimization is to find a solution with at high objective score, that is also robust to uncertainties in the evaluation process. It is difficult to make a general definition of robustness since the desired robustness vary for different general problem types. A common approach though is to use Wald's maximin model [10]. This method simply maximizes the worst case performance of the solution, and can be formulated as follows:

$$\mathbf{x}_{robust} = \underset{\mathbf{x}\in\mathbb{R}^n | \mathbf{x}_{min} \leqq \mathbf{x} \leqq \mathbf{x}_{max}}{\operatorname{argmax}} \; \underset{\mathbf{u}\in\mathbb{R}^m | \mathbf{u}_{min} \leqq \mathbf{u} \leqq \mathbf{u}_{max}}{\min} g(\mathbf{x}, \mathbf{u}) \tag{2}$$

where $g(\mathbf{x}, \mathbf{u})$ is the objective score for the solution $\mathbf{x}$, given the system uncertainties $\mathbf{u}$.

However, this model was considered to be too strict since a single failure in several hundreds of runs might be acceptable and also because the computer simulation might produce a few unrealistic results which could drag a potentially good solution down. Therefore a more outlier resistant version of Wald's model is used, which will be further discussed in Section 3.

In order to optimize the use cases, we have tested five different optimization algorithms, which are all illustrated in Fig. 2. The algorithms are chosen to represent three classes, namely point-based, region-based and global optimization methods, since we believe these classes perform differently with regard to finding robust solutions.

The first method, Coordinate Descent (CD), is a point-based method that iteratively improves one dimension at a time. The second method, Conjugate Gradient Descent (CGD), is also a point-based method which iteratively improves the solution in the approximated gradient direction. The third method, BOBYQA, is a region-based method that