Dynamic Regrasping by In-Hand Orienting of grasped objects using Non-Dexterous Robotic Grippers

Avishai Sintov and Amir Shapiro

Department of Mechanical Engineering, Ben-Gurion University of the Negev

Abstract

Almost any task on an object requires regrasping of the object prior to performing an intended task by varying between grasp configurations. The human hand uses many methods to perform regrasping manipulations such as in-hand sliding, finger gaiting, juggling, picking and placing, etc. The most complex and inspiring approach is the in-hand orienting dynamic regrasping where an object is released into mid air and regrasped in a different orientation. This manipulation is useful in industrial robotics for rapid manufacturing and reducing the number of robotic arms in production lines. In this work, we present a novel approach for performing in-hand orienting regrasping using computed torque control. To maintain an efficient and low-cost approach, the regrasping is performed using a non-dexterous two-jaw gripper and by utilizing the robotic arm's dynamics. We focus on the motion planning for the motion and propose a novel stochastic algorithm for performing an optimal manipulation satisfying the kinematic and dynamic constraints. The algorithm optimizes the initial pose of the arm and the control gains. Statistical analysis shows the probability for the algorithm to find a solution if such exists. Simulations on a KUKA arm and demonstration on a planar 3R arm validate the feasibility of the proposed regrasping approach and planning algorithm.

Keywords: `Dynamic Regrasping, Motion planning, Kinodynamic constraints, Trajectory optimization.`
2010 MSC: 00-01, 99-00

1. Introduction

When the human hand holds an object with a grasp not compatible with the task to be done, it performs *Regrasping*. In other words, the hand executes a set of manipulations backed by eye feedback to alternate the grasp configuration with respect to the intended task (Figure 1). The regrasping manipulations of humans have encouraged many robotic researchers to take inspiration for different regrasping motions and mimic them using dexterous robotic arms and hands. The ability of robots to perform regrasping tasks enhances their capabilities and dexterity. The main impact of regrasping manipulations is in industrial applications. For instance, current production lines utilize numerous robotic arms, each with an end-effector especially designed for grasping a specific part and to perform one designated task. This makes the end-effectors inflexible in handling variations in component shape or in task. Using an efficient regrasping method, the same arm can perform multiple operations on the same part and by that decrease the number of robotic arms in the plant. However, current regrasping methodologies work only with highly redundant (and hence expensive) hand architectures, and require agile sensory feedback, thus, not fitted for industrial applications.

In this paper, we present a novel motion planning approach to perform a rapid and optimal dynamic regrasp manipulation with a robotic arm and a non-dexterous simple gripper. Such approach would enable industrial practitioners to use existing robotic arms in the plant and low-cost grippers for performing multiple tasks on the same part. By that, the number of required robotic arms in the plant would be reduced, as well as the gripper design and manufacturing time, and at the end reduce the final product cost. In terms of robotic manipulation, the problem introduced in this paper and the proposed algorithm will increase the dexterity of any robotic arm and will expand its library of motions. Our long-term goal is to build a library of basic dynamic regrasping manipulations that will

*Corresponding author: Avishai Sintov, Department of Mechanical Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel, E-mail sintova@post.bgu.ac.il, Tel +972-54-5562555.

Figure 1: Snapshots of a human hand performing in-hand orienting dynamic regrasping on a TV remote control.

serve as building blocks for higher task executions to be performed by autonomous robots.

Previous work on regrasping used quasi-static motions and high dexterity manipulations. The alternative of dynamic regrasping using a single robot arm with a non-dexterous end-effector offers an attractive solution in terms of cost as well as task completion time. In this research, we focus on a simple robotic system that will perform transition between two or more grasps of an object. As mentioned, we focus on non-dexterous two-jaw grippers. That is, the gripper would only have a minimal number of primitive degrees of freedom (DOF) to only fixture the object after the manipulation is finished, e.g., a jaw gripper, a clamp, etc. Hence, we rely on the dynamics of the arm rather than on a highly dexterous hand or, as defined by [1], we use extrinsic dexterity. It should be noted that in this work we do not deal with regrasp synthesis of where on the object's surface to grasp while maintaining a force-closure grasp, as done by [2] and [3]. The idea of the regrasping problem in our context is grasping with a very simple gripper and regrasping by means of changing the relative position and orientation between the object and gripper.

The dynamic regrasping strategy for which we propose a motion planning algorithm in this paper is termed *in-hand Orienting*. The object is released into mid-air after opening the gripper. The robotic arm will then move the gripper in tandem with the falling object, but without disturbing the object. When the gripper reaches a new desired orientation relative to the object, its jaws will capture the object by closing around it. It should be mentioned that the capturing is done with matched velocities between the object and gripper for soft catch to prevent undesired shocks to the object. We use a novel Stochastic KInodynamic Planning (SKIP) algorithm for planning and optimizing the motion. A preliminary version of the algorithm was first introduced in [4] for object throwing where polynomial trajectories were optimized. In contrast, in this work we use a modified version of the algorithm to optimize the initial state of releasing the object and provide a profound analysis. Optimizing the state of releasing the object is essential to enable a feasible motion and can reduce joint torques and motion time. The algorithm parameterizes the proposed motion of the arm and object, including the control gains and the initial state of the system from where to release the object. To our knowledge, no optimal control or trajectory optimization methods can find a solution while choosing the best initial state of the system. Other methods such as sampling-based planners [5] plan the motion from a given initial state. The parameterized motion is the input of the SKIP algorithm, which outputs the optimal solution (based on a given cost function) satisfying the kinodynamic constraints of the problem. The algorithm is based on the use of the Computed Torque (CT) control and its response on the arm. The CT control enables good prediction of the motion which is essential for the implementation of the planning algorithm. Due to the fact that the CT control is model based, we perform model identification for approximating the dynamic parameters of the robotic arm. The proposed planning and control scheme is presented in Figure 2.

In this work we focus on the motion planning problem for performing the proposed regrasping strategy. Nevertheless, we choose a control method that will enable the planning approach. However, we acknowledge that additional work must be done before regrasping of this type can be implemented in industrial applications. In particular, we do not discuss the visual feedback and estimation problem necessary for a robust manipulation. We assume full knowledge of the object pose at all times and leave the tracking problem beyond the scope of this paper. However, we will provide further discussion about future requirements regarding this in the conclusions section. It is also important to note that the regrasping manipulation is done in a priori known environment. This is a reasonable assumption for industrial applications where the robots are isolated and monitored. Thus, off-line planning is feasible.

The paper is organized as follows. In Section 2 we present related work on regrasping and dynamic manipulations. Section 3 is the formal regrasping problem definition. In Section 4 we present the CT control. The parameterization, constraint formulation, and SKIP algorithm are presented in Section 5. In Section 6 we present performance analysis and complexity of the algorithm. Simulations and experiments are presented in Section 7. Finally, we conclude the algorithm and results in Section 8.

2. Related work

This paper discusses a dynamic manipulation approach for performing the regrasping motion of a human hand for future industrial implementations such as assembly tasks. Let us first describe the two regrasping approaches taken by industrial practitioners. The first approach is the common *pick and place* method that designates a special work area near each robotic arm, where the grasped workpiece can be dropped in a controlled manner, then picked up again at a new grasp configuration [6, 7, 8]. Alternatively, some high-end industrial practitioners resort to unthreading the regrasping task by placing several (hugely expensive) robot arms along a long production or assembly line, each picking a moving workpiece at a new grasp configuration [9, 10]. While both regrasping methods work nicely in practice, they consume valuable production time, occupy a substantial work area, and are highly expensive when multiple robot arms are being used.

In the robotics literature, there are four known approaches for regrasping (excluding picking and placing); First, the use of the gripper's DOF to move between contact points while maintaining a force-closure grasp during the entire process [2, 11, 12, 13, 14, 15, 16]. This approach is also called *quasi-static finger gaiting* in the robotics literature. However, quasi-static finger gaiting is quite wasteful, as it requires sufficiently many DOFs (requiring highly redundant finger linkages) to manipulate the grasped object between two grasp configurations while maintaining force closure grasps. Most of the finger gaiting algorithms use at least another extra finger that can be lifted at each step [17, 18, 19, 20]. Such motion results in a slow quasi-static manipulation, maintaining a state of constant contact with the object. This has an analogy to gait, where advancement is done only with one leg lifted at a time (when dynamic forces are not taken into consideration) so that the other legs can maintain balance. Such a process can take a valuable amount of time.

Alternatively, it has been recently suggested that a dual arm robot may be more suitable for object regrasping operations [21, 22]. While dual arms is a promising approach, it has two significant drawbacks. First, regrasping an object with two cooperating arms requires highly dexterous manipulation capabilities from both arms. Second, a dual-arm system is costly and occupies a fairly large work volume. The third approach is sliding the fingertips on the object's surface without losing contact, to reposition them at new contact points [23, 24]. This approach has not been widely investigated and has low feasibility due to the risks of scratches and dents on the object. Moreover, the approach also needs sufficient DOFs to develop trajectories on the object's surface.

The fourth approach is much faster and more efficient, however more complex, as it uses dynamical manipulations to switch between grasp configurations. In dynamic manipulation, the robots motion is quick and there is no need for constant contact. The end-effector loses contact (fully or partly) with the object after releasing it in the air and regains contact by catching it at the final contact points. Such a method has advantages in fast operation. However, most work done in this field used a multi-fingered highly dexterous hand for performing regrasping. [25] proposed a regrasping strategy based on visual feedback of the manipulated object, this with a multi-fingered hand. The work in [26] introduced a regrasping method using a 3-finger hand with no external sensing for feedback. A work that combines the notion of dynamic and sliding regrasping was presented in [27], where a robotic arm's motion is planned to create in-hand sliding motion of a pinched object. Our first work on non-dexterous dynamic regrasping was published in [28, 29]. There, we introduced the swing-up regrasping problem to alternate the relative angle between the object and hand around a point pinched by a simple jaw gripper using the dynamics of the hand and controlling the pinching force.

There is related work to dynamic regrasping in the field of dynamic manipulations [30, 31]. Dynamic manipulation of an object enables the change of its position and orientation by tossing, pushing, or hitting it. Lynch et al. [32, 33] developed a low degree of freedom robot to perform complex manipulations using rolling, sliding, and free-flight, this with no grasping (non-prehensile manipulation). Tabata and Aiyama [34] used a one degree of freedom manipulator to toss an object to a goal position. The work in [35] was done to mimic the manipulation done with a pizza peel. The research states that the pizza peel is controlled with only two DOF and utilizes vision control. Much work has also been done in pushing manipulations where the object is pushed on a desired trajectory to re-position or re-orient it [36, 37, 38]. Moreover, some performed manipulation of the grasped objects by dynamic manipulations between fingertips [39, 40].

3. Problem Definition

In this section we describe the system and its constraints, lay out our assumptions, and formulate the problem and its goal. Note that the problem is defined for the spatial problem but can be easily degraded to the planar problem without loss of generality.

3.1. Given system

Let $\mathcal{Q} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^n$ be the configuration space and the set of all possible torque or force inputs, respectively, of an n DOF robotic arm. The robotic arm is equipped with a simple non-dexterous jaw-gripper end-effector. We deal with objects such that their bounding sphere is of the same scale as the gripper. That is, the gripper is able to orient freely around the object. The world coordinate frame (CF) $\mathcal{O}$ is located at the base of the manipulator. The dynamic motion equations of the manipulator in the configuration space can be written as

$$M(\mathbf{\Phi})\ddot{\mathbf{\Phi}} + C(\mathbf{\Phi}, \dot{\mathbf{\Phi}})\dot{\mathbf{\Phi}} + G(\mathbf{\Phi}) + \Gamma = \mathbf{u} \tag{1}$$

where $\mathbf{\Phi}(t) = (\phi_1(t), \cdots, \phi_n(t))^T \in \mathcal{Q}$ is the configuration vector of the arm at time t, M is the $n \times n$ inertia matrix, C the $n \times n$ matrix of centrifugal and Coriolis acceleration terms, G is the $n \times 1$ vector of gravitational effects, and $\mathbf{u}(t) \in \mathcal{U}$ is a vector of control input torques to the joints. Γ is the $n \times 1$ vector of joints' frictional torques given by

$$\Gamma_i = f_{c_i}\text{sgn}(\dot{\phi}_i) + f_{v_i}\dot{\phi}_i, \;\; i = 1, ..., n \tag{2}$$

where f_{c_i} and f_{v_i} are the coulomb and viscous coefficients of friction, respectively, of joint i.

Next, we define the notion of the *Pose* related to an object's position and orientation.

Definition 1. *A generalization vector* $\mathbf{p}$ *combining Cartesian position and spatial orientation of an object is called a Pose such that* $\mathbf{p} \in \mathcal{T}$ *where* $\mathcal{T} \subseteq \mathbb{R}^3 \times \mathbb{S}^3$ *is the robots task space.*

According to Definition 1, we denote $\mathbf{p_g}(t) \in \mathcal{T}$ to be the pose of the gripper at time t. That is, $\mathbf{p_g} = (x_g \; y_g \; z_g \; \varphi_g \; \theta_g \; \psi_g)^T$ is a generalization vector of the gripper's position, $(x_g, y_g, z_g) \in \mathbb{R}^3$, and its orientation $(\varphi_g, \theta_g, \psi_g) \in \mathbb{S}^3$ with respect to reference frame $\mathcal{O}$. Angles φ_g, θ_g, and ψ_g are the roll, pitch, and yaw angles (Euler angles) of the gripper, respectively. In addition, we define $\mathcal{X}_g$ as a fixed CF at the center of the gripper.

Based on arm limitations, we impose several constraints. The allowed subset of control input torques is given by $\mathcal{U}_{al} \subset \mathcal{U}$. Furthermore, the arms' joints are limited within a certain range of motion due to mechanical bounds, operational demands, or obstacles in the workspace such as a table or a wall. Meaning, the configuration space of the arm is constrained to be within $\mathcal{Q}_{free} \subseteq \mathcal{Q}$ through the whole motion. Constraints can also be formulated in the task space as $\mathcal{T}_{free} \subseteq \mathcal{T}$. However, for simplicity they are included in the configuration space constraints using inverse kinematics as will be defined later.

Given object $\mathcal{B}$ with mass m and CF $\mathcal{X}_o$ fixed at its center of mass. We denote the object's configuration in the task space by $\mathbf{p_o} = (x_o \; y_o \; z_o \; \varphi_o \; \theta_o \; \psi_o)^T \in \mathcal{T}$. Hence, we describe its configuration by its position (x_o, y_o, z_o) and orientation $(\varphi_o, \theta_o, \psi_o)$ in Euler angles relative to $\mathcal{O}$. After neglecting air resistance, the equation of motion of the object, while in free-flight, is given by

$$\ddot{\mathbf{p}}_\mathbf{o} = \mathbf{g} \tag{3}$$

where $\mathbf{g} = \begin{pmatrix} 0 & 0 & -g & 0 & 0 & 0 \end{pmatrix}^T$ is the gravitational vector operating on the object. We note that it is assumed that the state (pose and pose velocity) of object $\mathcal{B}$ is fully known. That is, feedback of $\mathbf{p_o}(t)$ is acquired at all times as well as $\dot{\mathbf{p}}_{\mathbf{o}}(t)$. This feedback can be acquired by cameras or motion capture systems. However, sensing and estimation are beyond the scope of the paper.

Next, we define the relative pose between the gripper and object.

Definition 2. *The relative pose between the gripper and object $\mathcal{B}$ is defined to be*

$$\overline{\mathbf{p}}(t) = \mathbf{p}_{\mathbf{g}}(t) - \mathbf{p}_{\mathbf{o}}(t) \tag{4}$$

and the relative pose velocity is

$$\dot{\overline{\mathbf{p}}}(t) = \dot{\mathbf{p}}_{\mathbf{g}}(t) - \dot{\mathbf{p}}_{\mathbf{o}}(t) \tag{5}$$

where $\dot{\mathbf{p}}_{\mathbf{g}}$ and $\dot{\mathbf{p}}_{\mathbf{o}}$ are the pose velocities of the gripper and object, respectively.

At time $t = 0$, since the gripper is stationary and holds the object, they are located at the same position and at rest. Moreover, we define the initial grasp posture to be the zero orientation between the gripper and object. Therefore, they have equal poses $\mathbf{p}_{\mathbf{g}}(0) = \mathbf{p}_{\mathbf{o}}(0)$ and zero velocities $\dot{\mathbf{p}}_{\mathbf{g}}(0) = \dot{\mathbf{p}}_{\mathbf{o}}(0) = 0$, or according to the terminology in Definition 2, their initial relative pose is $\overline{\mathbf{p}}(0) = 0$ with initial relative pose velocity $\dot{\overline{\mathbf{p}}}(0) = 0$.

3.2. Manipulation Goal

Object $\mathcal{B}$ is held by the gripper of system (1) with relative pose $\overline{\mathbf{p}}(0) = 0$, i.e., CFs' $\mathcal{X}_g$ and $\mathcal{X}_o$ are coincident. Let $\hat{\omega}_o = (\omega_{o_x}\ \omega_{o_y}\ \omega_{o_z})^T$ be a unit vector in $\mathcal{X}_o$ and let $\alpha \in \mathbb{S}$ be the desired rotation angle about $\hat{\omega}_o$ (Figure 3). It is required for the regrasping manipulation to grasp the object at the final time t_f such that $\mathcal{X}_g$ is rotated relative to $\mathcal{X}_o$ about $\hat{\omega}_o$ with angle α. Following, we develop a formal definition. First, we define a rotation matrix for a spatial object relative to coordinate frame $\mathcal{O}$.

Definition 3. *Given: an arbitrary object $\mathcal{C}$ rotated relative to coordinate frame $\mathcal{O}$ by the Euler angles φ, θ, and ψ about the x, y, and z axes of frame $\mathcal{O}$, respectively. Let matrix $R_{\mathcal{C}} \in SO(3)$ be the map $R_{\mathcal{C}} : \mathbb{R}^3 \to \mathbb{R}^3$ from object $\mathcal{C}$'s CF to CF $\mathcal{O}$ such that*

$$R_{\mathcal{C}} = R_z(\psi)R_y(\theta)R_x(\varphi) \tag{6}$$

where $R_i,\ i = x, y, z$ is a rotation matrix about the i-axis of $\mathcal{O}$.

By neglecting air resistance we can say that a falling arbitrary object $\mathcal{C}$ with no initial angular velocity will maintain its orientation and therefore has a constant rotation matrix $R_{\mathcal{C}}$. The following proposition defines the required relative pose based on the desired rotation angle α about the unit vector $\hat{\omega}_o$.

Proposition 1. *Given coincident coordinate frames $\mathcal{X}_o$ and $\mathcal{X}_g$ at time $t = 0$, necessarily means that $\overline{\mathbf{p}}(0) = 0$. If at time t_f, $\mathcal{X}_g$ is rotated about $\hat{\omega}_o \in \mathcal{X}_o$ with angle α, there exists a vector $\zeta = (\zeta_x\ \zeta_y\ \zeta_z\ \zeta_\varphi\ \zeta_\theta\ \zeta_\psi)^T \in \mathcal{T}$ such that $\overline{\mathbf{p}}(t_f) = \zeta$.*

Proof. The following is a constructive proof in which we develop the specific formula for the relative position and orientation ζ between the object the gripper. In case of relative translation between $\mathcal{X}_g$ and $\mathcal{X}_o$, the Cartesian coordinates in $\overline{\mathbf{p}}(t_f)$ are equal to

$$x_g(t_f) - x_o(t_f) = \zeta_x,\ \ y_g(t_f) - y_o(t_f) = \zeta_y,\ \text{ and } z_g(t_f) - z_o(t_f) = \zeta_z, \tag{7}$$

where ζ_x, ζ_y, and ζ_z are the desired translation constants. The rotation of $\mathcal{X}_g$ relative to $\mathcal{X}_o$ about $\hat{\omega}_o$ with angle α is given by the Rodrigues rotation formula [41]:

$$Q = I + W \sin\alpha + W^2(1 - \cos\alpha) \tag{8}$$

where I is a 3×3 identity matrix and W is the skew-symmetric matrix representation. Hence, the rotation matrix mapping the rotated $\mathcal{X}_g$ to CF $\mathcal{O}$ is given by $R = R_{\mathcal{B}}Q$. From the rotation matrix R we can extract the desired roll, pitch, and yaw angles of the gripper to satisfy the α rotation angle demand by

$$\begin{pmatrix} \varphi_g(t_f) \\ \theta_g(t_f) \\ \psi_g(t_f) \end{pmatrix} = \begin{pmatrix} \tan^{-1}(r_{32}/r_{33}) \\ \tan^{-1}\left(-r_{31}/\sqrt{r_{32}^2 + r_{33}^2}\right) \\ \tan^{-1}(r_{21}/r_{11}) \end{pmatrix} \tag{9}$$

where r_{ij} is a component of matrix R at row i and column j. The use of function $atan2$ is preferred in this case to acquire appropriate signs. According to (9) and Definition 2, the rotational coordinates in $\overline{\mathbf{p}}(t_f)$ are

$$\varphi_g(t_f) - \varphi_o = \zeta_\varphi,\ \theta_g(t_f) - \theta_o = \zeta_\theta,\ \psi_g(t_f) - \psi_o = \zeta_\psi\ . \tag{10}$$

Thus, there exists a vector ζ that is the required relative pose to satisfy the rotation about $\hat{\omega}_o$ with angle α at time t_f and is given according to (7) and (10) by

$$\overline{\mathbf{p}}(t_f) = \mathbf{p_g}(t_f) - \mathbf{p_o}(t_f) = \zeta = (\zeta_x\ \zeta_y\ \zeta_z\ \zeta_\varphi\ \zeta_\theta\ \zeta_\psi)^T\ . \tag{11}$$

□

The desired relative translation ζ_y in the y direction cannot be unequal to zero in a symmetric gripper but is parameterized to preserve the generality of the formulation. The other two, ζ_x and ζ_z, can be used for longitude regrasping, that is, regrasp along the object.

The regrasping manipulation should therefore bring the object to the desired relative pose ζ with zero relative pose velocity. Formally, the regrasping problem is to perform an optimal manipulation motion such that at some finite time $t_f = (0, \infty)$ the following conditions

$$\|\overline{\mathbf{p}} - \zeta\| \leq \epsilon, \quad \|\dot{\overline{\mathbf{p}}}\| < \epsilon_1 \tag{12}$$

will be satisfied while satisfying $\mathbf{\Phi}(t) \in \mathcal{Q}_{free}$ and $\mathbf{u}(t) \in \mathcal{U}_{al}$ and where $\epsilon, \epsilon_1 > 0$ are predefined convergence tolerances. An optimal manipulation is the one forming a feasible motion by satisfying the constraints such that the relative pose goal is reached and a given cost function $C(\mathbf{\Phi}, \mathbf{u})$ is minimized.

4. Computed Torque Control

In this work, we have outlined the in-hand spinning approach for the regrasping problem. In such an approach, the gripper lets go of the object but follows its position while changing its relative orientation. That is, the object remains within the gripper thorough the free-fall but with no direct contact. The gripper tracks the position of the object and therefore gripper-object collision is prevented. Their relative orientation changes during the fall until the desired relative pose is reached and the gripper regains contact. After releasing the object, the problem is basically an interception problem where the gripper follows the motion of the free-falling object. The problem is approached in this work with the use of the Computed Torque (CT) Control [42], which is presented in this section. The CT control is a model-based control scheme that can be a disadvantage when the arm's dynamic model is not fully known. We address this problem by using a model identification method. Nevertheless, we choose to use the CT control due to its great advantages for further motion planning where the system's behavior can be predicted, i.e., an analytical expression of the controlled system's trajectory can be easily acquired. The motion planning problem of the regrasping will be addressed later after reviewing the CT control.

4.1. The computed torque control law

Given the dynamics of the robotic arm in (1), the CT control scheme applied to the system is expressed as follows:

$$\mathbf{u} = \tilde{M}\left(K_p(\check{\mathbf{\Phi}} - \mathbf{\Phi}) + K_d(\dot{\check{\mathbf{\Phi}}} - \dot{\mathbf{\Phi}}) + \ddot{\check{\mathbf{\Phi}}}\right) + \tilde{C}\dot{\mathbf{\Phi}} + \tilde{G} + \tilde{\Gamma} \tag{13}$$

where $\tilde{M}, \tilde{C}, \tilde{G}, \tilde{\Gamma}$ indicate estimated values of M, C, G, Γ, respectively; $\check{\mathbf{\Phi}}, \dot{\check{\mathbf{\Phi}}}, \ddot{\check{\mathbf{\Phi}}}$ are the desired configuration, velocity, and acceleration of the manipulator's joints; and $K_p = diag(k_{p_1}\ \cdots\ k_{p_n})$ and $K_d = diag(k_{d_1}\ \cdots\ k_{d_n})$ are proportional and derivative diagonal gain matrices, respectively. Applying the control scheme of (13) to the dynamic system of (1) and assuming that the robot's model is fully known, i.e., $(\tilde{\cdot}) = (\cdot)$, the closed loop system is reduced to the following linear form

$$\ddot{\mathbf{e}} + K_d\dot{\mathbf{e}} + K_p\mathbf{e} = 0 \tag{14}$$

where $\mathbf{e} = \check{\mathbf{\Phi}} - \mathbf{\Phi}$ denotes the joint angles error. Equation (14) is a set of second order linear differential equations with constant coefficients. The general solution of the system in (14) is of the form

$$e_i(t) = c_1 e^{-\xi_i \omega_{n_i} t} + c_2 t e^{-\xi_i \omega_{n_i} t}, \ i = 1, ..., n, \tag{15}$$

where ω_{n_i} is the *natural frequency* and ξ_i is the *damping ratio* of the controlled joint i. The closed loop system solution is a damped harmonic oscillator where

$$k_{p_i} = \omega_{n_i}^2 \ \Rightarrow \ \omega_{n_i} = \sqrt{k_{p_i}} \tag{16}$$

and

$$k_{d_i} = 2\xi_i \omega_{n_i} \ \Rightarrow \ \xi_i = \frac{k_{d_i}}{2\sqrt{k_{p_i}}} \ . \tag{17}$$

To prevent overshooting the desired state and to acquire a rapid response, we demand critical damping of the system. That is, we demand $\xi_i = 1, \ \forall i = 1, ..., n$. Therefore, the following constraint for the gains must be satisfied

$$k_{d_i} = 2\sqrt{k_{p_i}} \ \ , i = 1, ..., n \ , \tag{18}$$

where $\xi_i \omega_{n_i} = \sqrt{k_{p_i}}$ and the initial conditions are $e_i(0) = e_{0_i}$ and $\dot{e}_i(0) = \dot{e}_{0_i}$. Therefore, using (15),(18) and the initial conditions, $e_i(0) = e_{0_i}$ and $\dot{e}_i(0) = \dot{e}_{0_i}$, the time response error of joint i is

$$e_i(t) = e_{0_i} e^{-\sqrt{k_{p_i}} t} \left(1 + \sqrt{k_{p_i}} t\right) + \dot{e}_{0_i} t e^{-\sqrt{k_{p_i}} t} \ . \tag{19}$$

The CT controller in (13) is model based and the analysis done for approximating the error response and convergence time is based on full knowledge of the model. That is, knowledge of the dynamic parameters of the system (such as moment of inertia, mass, and center of mass position) are essential for accurate motion. Therefore, we apply a model identification scheme presented in [43] to estimate the dynamic parameters of the manipulator performing the regrasping.

4.2. Interception problem

Using the above CT control scheme, we would like to formulate the free-fall phase from the time of release ($t = 0$) to the time of interception ($t = t_f$). Through the whole phase, feedback of the object's state, $\mathbf{p_o}(t)$ and $\dot{\mathbf{p}}_\mathbf{o}(t)$, is constantly acquired via visual feedback. The state feedback is used to calculate the desired pose $\check{\mathbf{p}}_\mathbf{g}(t)$ and pose velocity $\dot{\check{\mathbf{p}}}_\mathbf{g}(t)$ of the gripper based on (12) according to

$$\check{\mathbf{p}}_\mathbf{g}(t) = \mathbf{p_o}(t) + \zeta \ \text{ and } \ \dot{\check{\mathbf{p}}}_\mathbf{g}(t) = \dot{\mathbf{p}}_\mathbf{o}(t). \tag{20}$$

Vector ζ is calculated according to Proposition 1 based on the desired rotation angle α about the unit vector $\hat{\omega}_o$. Constant measure of the object's orientation and update of ζ can be done in real time during the fall. However, by neglecting air resistance, we can say that the object with no angular velocity has constant ζ and real time update is not necessary. We calculate the desired joint configuration $\check{\mathbf{\Phi}}$ corresponding to $\check{\mathbf{p}}_\mathbf{g}$ using

$$\check{\mathbf{\Phi}}(t) = \Theta^{-1}(\check{\mathbf{p}}_\mathbf{g}(t)) \tag{21}$$

where $\Theta : \mathcal{Q} \to \mathcal{T}$ is the direct kinematics map of the manipulator from the joint's configuration to the gripper pose configuration. Similarly, the desired joint velocity and accelerations are calculated by

$$\dot{\check{\mathbf{\Phi}}}(t) = J^{-1}\dot{\check{\mathbf{p}}}_\mathbf{g}(t) \ \text{ and } \ \ddot{\check{\mathbf{\Phi}}}(t) = J^{-1}(\ddot{\check{\mathbf{p}}}_\mathbf{g} - \dot{J}\dot{\check{\mathbf{\Phi}}}_d), \tag{22}$$

respectively, where J is the Jacobian matrix of the manipulator. The angles, velocity, and acceleration given by (21)-(22) are the desired angle's trajectory inputs to the CT controller in (13).

Since there is no initial relative velocity between the gripper and object, the initial error $\dot{e}_{0_i} = 0, \ i = 1, \ldots, n$ and therefore the angle error response of (19) is now

$$e_i(t) = e_{0_i} e^{-\sqrt{k_{p_i}} t} \left(1 + \sqrt{k_{p_i}} t\right) , \; i = 1, ..., n \; . \tag{23}$$

From the definitions of the joint angle error and response of (23), we can extract the analytical trajectory of the controlled joints

$$\mathbf{\Phi}(t) = \check{\mathbf{\Phi}}(t) - \mathbf{e}(t). \tag{24}$$

where $\mathbf{e} = (e_1 \;\cdots\; e_n)^T$ and the desired joint angles $\check{\Phi}_1, ..., \check{\Phi}_n$ at time t are calculated in (21) using real-time visual feedback of the object's state. The initial error vector $\mathbf{e_0} = (e_{0_1} \;\cdots\; e_{0_n})^T$ is given according to (20)-(21) by

$$\mathbf{e_0} = \Theta^{-1}(\mathbf{p_o}(0) + \zeta) - \mathbf{\Phi}(0) \; . \tag{25}$$

Next, we seek to approximate the time in which the gripper and object reach the desired relative state. That is, the time at which the goal condition of (12) is satisfied. That is, the relative pose and velocity are close enough to ζ within a predefined tolerance. Therefore, we approximate the time t_f it takes for the state error of the gripper and object to reach within tolerance ϵ of the final value. From the step response of a second order system with natural frequency ω_{n_i} and damping ratio ξ_i [44], the desired relative state is reached approximately at time $\tilde{t}_f$ that satisfies the condition

$$e^{-\xi\omega_n \tilde{t}_f} < \epsilon \; . \tag{26}$$

Therefore, the approximated settling time of joint i for the error to converge within tolerance ϵ is

$$\tilde{t}_{s_i} = -\frac{\log(\epsilon)}{\xi_i \omega_{n_i}} = -\frac{\log(\epsilon)}{\sqrt{k_{p_i}}} \; . \tag{27}$$

The total approximated settling time of the system will be the maximum of the n values

$$\tilde{t}_f = \max_i \{\tilde{t}_{s_i}\} \; . \tag{28}$$

5. Regrasp planning algorithm

The use of the CT controller allows obtaining an analytic solution (24) of the joint trajectory from release to interception at the desired relative pose, with dependence of the object's state acquired using visual feedback. However, after release, the object is at free-fall and its state could easily be approximated with (3). The motion planning to perform the regrasping could be solved using optimal control or stochastic methods such as the RRT [5, 45]. However, these methods are unfavorable due to the following reasons. The initial pose of the arm is the configuration from where the object would be released. The in-hand regrasping cannot be performed from all initial configurations due to the kinodynamic constraints. For example, setting the initial pose in some area close to the workspace boundary of the robot may not enable enough motion to complete the task. Moreover, some initial configurations result in better solutions than others in terms of the cost function. For a minimal torque cost function, having the gripper start far from the base will probably have a higher cost than starting from a closer pose. The motion also depends on the control gains which have a direct impact on motion time and accelerations, and thus on the cost function. Hence, the optimization problem in the context of dynamic regrasping should optimize the initial pose of the arm and the control gains. To our knowledge, no optimal control or trajectory optimization methods can find a solution while choosing the best initial state of the system. Therefore, in the following section, we parameterize the free variables of the problem: gripper initial pose and control gains, and present the algorithm for planning and optimization of the motion.

5.1. Motion parameterization

The free-fall motion of the object depends on its release pose. That is, releasing the object from initial pose $\mathbf{p_0}$ and with zero velocity defines its uncontrollable trajectory solely under the influence of gravity. Therefore, the definition of motion in this phase includes the initial pose at time $t = 0$. Moreover, the motion of the arm controlled with the

CT scheme from that time instant is given by (24). The estimated time for the gripper and the object to reach the desired relative state, i.e., satisfaction of (12), is given by $\tilde{t}_f$ in (28). Notice that $\tilde{t}_f = \tilde{t}_f(k_{p_1}, \ldots, k_{p_n})$; that is, the proportional gains of the controller defines the time of interception t_f with the object.

With the understanding of the parameters that define the free-fall and interception, we define the parameters vector $\sigma \in \Omega$ where $\Omega \subseteq \mathbb{R}^d$ and $d = 6 + n$. The form of the parameters vector σ is defined as follows:

$$\sigma = \begin{pmatrix} \mathbf{p_0}^T & k_{p_1} & \cdots & k_{p_n} \end{pmatrix}^T . \tag{29}$$

while assuming $\dot{\mathbf{p}}_\mathbf{0} = \mathbf{0}$. Parameterization vector σ defines the motion of both the arm and object after release. Determination of σ defines the approximated trajectory and motion time of the arm according to (24) and (28) to intercept the flying object and finally regrasp it at the desired relative pose.

After parameterization, the time responses of the arms' joints $\mathbf{\Phi}(t)$ given in (24) can be written as a function of time and σ, i.e., $\Phi(t) = \mathbf{f_\Phi}(t, \sigma)$, by replacing the initial position and control gains notations with the σ notation. Similarly, the CT controller from (13) has the form

$$\mathbf{u}(t) = \mathbf{f_u}(t, \sigma) = \tilde{M}(\mathbf{\Phi})\mathbf{a_f}(t, \sigma) + \tilde{C}(\mathbf{\Phi}, \dot{\mathbf{\Phi}})\dot{\mathbf{\Phi}} + \tilde{G}(\mathbf{\Phi}) + \tilde{\Gamma} \tag{30}$$

where

$$\mathbf{a_f}(t, \sigma) = K_p(\check{\mathbf{\Phi}} - \mathbf{\Phi}) + K_d(\dot{\check{\mathbf{\Phi}}} - \dot{\mathbf{\Phi}}) + \ddot{\check{\mathbf{\Phi}}} , \tag{31}$$

$\check{\mathbf{\Phi}}$, $\dot{\check{\mathbf{\Phi}}}$ and $\ddot{\check{\mathbf{\Phi}}}$ are calculated according to (20)-(22) while measuring the object's state along its free-fall. The term in (30) expresses the control law with the equations of motion. In the next subsection we integrate this in the set of kinodynamic constraints of the problem.

5.2. Motion constraints

In this subsection we formulate the constraints of the system's motion in terms of the free parameters of the problem, i.e., according to parameterization vector σ. We formulate the configuration space constraints defined by $\mathcal{Q}_{free}$ and the torque constraints imposed by $\mathcal{U}_{al}$.

As mentioned in section 3, due to joint limitations, it is possible that not all the configuration space $\mathcal{Q}$ of the arm is accessible. Therefore, we can formulate the free space $\mathcal{Q}_{free}$ explicitly as a set $\mathbf{\Lambda} \in \mathbb{R}^a$ of a constraints

$$\Lambda_i(\mathbf{f_\Phi}(t, \sigma)) \leq 0, \ \forall i = 1, ..., a. \tag{32}$$

As mentioned, the configuration constraints incorporate pose constraints such as obstacles and arm boundaries in the workspace. The same could be done to explicitly express the set of allowed torques $\mathcal{U}_{al}$. Usually, the allowed torques would be bounded by

$$u_{min_i} \leq u_i(t) \leq u_{max_i}, \ \forall i = 1, ..., n, \tag{33}$$

where u_{max_i} and u_{min_i} are the upper and lower torque bounds of joint i. Therefore, in terms of t and σ, the torque constraints are written as

$$\left\{ \begin{array}{l} f_{u_i}(t, \sigma) - u_{max_i} \leq 0 \\ -f_{u_i}(t, \sigma) + u_{min_i} \leq 0 \end{array} \right. , \ \forall i = 1, ..., n \tag{34}$$

and are equivalent to (33) if both are satisfied. One constraint that is not included here is an object-gripper constraint to prevent collision through-out the manipulation. Such constraint is not essential as the CT control and its critical damping ensure non-collision. However, such constraint can be easily included.

The set of inequalities given in (32) and (34) could be written as

$$\Psi_i(t, \sigma) \leq 0, \ i = 1, ..., b. \tag{35}$$

$\mathbf{\Psi}(t, \sigma)$ is a set of b functions:

$$\Psi(t, \sigma) = \begin{pmatrix} \mathbf{\Lambda}(\mathbf{f_\Phi}(t, \sigma)) \\ \mathbf{f_u}(t, \sigma) - \mathbf{u}_{max} \\ -\mathbf{f_u}(t, \sigma) + \mathbf{u}_{min} \end{pmatrix}_{b \times 1} \tag{36}$$

where by summing the number of constraints in (32) and (34) the total number of constraints is $b = a + 2n$. Inequality

(35) defines a feasible region of the dynamic system in terms of time and the desired trajectory parameter σ. That is, we obtained a set of constraints that defines a time-varying domain in Ω. Next, we present the time-varying constraint problem and the search algorithm to find an optimal trajectory satisfying the constraints.

5.3. Time-varying constraint (TVC) problem

In the previous section we obtained a set of inequalities depending on t and the parameter's vector σ. Recall that the components in σ are independent of time. Therefore, we would like to find an optimal vector $\sigma^* \in \Omega$ that satisfies the constraints from release to goal time $t_f(\sigma^*)$, and minimizes the cost function $C(\sigma)$. Such an optimal vector will sufficiently define the motion of the system under the constraints. Let $\Sigma \subset \Omega$ be the allowed region for σ defined by the pose boundaries of the arm and maximal possible control gains. Next, we define the notion of a feasible set.

Definition 4. *A set $\Omega_f \subset \Omega$ is a feasible set if $\Omega_f \subseteq \Sigma$ and every $\sigma \in \Omega_f$ satisfies inequality (35) for all time $t \in [0, t_f(\sigma)]$.*

We now define a feasible vector.

Definition 5. *A motion parameterization vector $\sigma \in \Omega$ is said to be feasible if $\sigma \in \Omega_f$.*

The above two definitions conclude that a vector is feasible if

$$\sigma = \{\sigma \in \Omega_f | \sigma \in \Sigma, \Psi_i(t,\sigma) \leq 0, \ \forall \ t \in [0, t_f(\sigma)], i = 1, ..., b\}. \tag{37}$$

We now face the the problem of finding the feasibility set $\Omega_f \subseteq \Sigma$ where for all vectors within it, inequality (35) is being met at all times. Formally, the problem is as follows.

Problem 1. *(TVC) Given the set of constraints in (35) and the set Σ, find the feasibility set $\Omega_f \subseteq \Sigma$.*

Solving the above problem provides the feasible set Ω_f from which the optimal solution is to be chosen. Hence, we define the following minimization problem.

Problem 2. *Find the vector $\sigma^* \in \Omega_f$ where $\Omega_f \subseteq \Sigma$ such that*

$$\begin{aligned} \sigma^* = arg\min_{\sigma} \quad & C(\sigma) \\ subject\ to \quad & \sigma^* \in \Omega_f \end{aligned} \tag{38}$$

In other words, the above general problem is finding an optimal vector σ^* that is feasible and minimizes some cost function $C(\sigma)$. Figure 4 illustrates the two problems. The position and volume of $\Psi(t,\sigma)$ in Ω varies in time and therefore, the solution of the problem is in a domain formed by projecting the constraints for time 0 to t_f on Ω. The intersection formed by these projections, if one exists, is the feasibility domain Ω_f and from there the optimal solution σ^* should be chosen.

The above problem seeks a feasibility set over a period of time. It is important to note that we seek a feasibility set if such exists. In the next subsection we present an algorithm for finding the feasibility set and choosing an optimal solution from it. The probability to find a solution if one exists is also presented.

5.4. Feasibility search algorithm

A search algorithm is now presented to find the set Ω_f of feasible vectors. The domain formed by the set of constraints in inequality (35) is non-linear, non-convex, and may not be continuous. Therefore, an analytical solution of the reachable set is only possible in rare and simple cases. We present a numerical search algorithm to find a set of vectors satisfying the above constraints. Further, we can choose one vector from the set that best minimizes the cost function. We begin by presenting a simple definition for normalizing the time interval.

Lemma 1. *A vector σ is feasible in $t \in [t_1, t_2]$ if the constraint $\Psi_i(\lambda t_1 + (1-\lambda)t_2, \sigma) \leq 0$ is satisfied for all $0 \leq \lambda \leq 1$ and for all $i = 1, ..., z$.*

Proof. For each time instant $t \in [t_1, t_2]$ there exists $0 \leq \lambda \leq 1$ such that $t = \lambda t_1 + (1-\lambda)t_2$. Therefore, if a vector σ satisfies $\Psi_i(t,\sigma) \leq 0$ for all $i = 1, ..., z$ and $t_1 \leq t \leq t_2$, it must also satisfy $\Psi_i(\lambda t_1 + (1-\lambda)t_2, \sigma) \leq 0$ for all $i = 1, ..., z$ and $0 \leq \lambda \leq 1$. □

Lemma 1 is utilized as a criterion for determining whether a vector σ is feasible. Numerically, we check the constraint for times $t = \lambda t_1 + (1-\lambda)t_2$ with $\lambda = \{0, \Delta\lambda_1, \Delta\lambda_1 + \Delta\lambda_2, ..., 1\}$. The value of the step $\Delta\lambda_j$ will be further defined. Figure 5 illustrates an abstraction of the feasibility problem and the line in time defined by Lemma 1. From here on we assume $t_1 = 0,\ t_2 = t_{f_i}$, and therefore $t = \lambda t_{f_i}$.

The search algorithm is presented as the *Search_domain*$(\cdot)$ function in Algorithm 1. The algorithm's input is the allowed set Σ in Ω, the set of constraints of equation (35), and the number of random points N. The basis of the algorithm's operation is selecting a set of N random points within Σ and checking each for its feasibility. The determination of N would be discussed later on. The first step is to sample N random points $\mathcal{P} = \{\sigma_1, ..., \sigma_N\}$ uniformly distributed in Σ. The allowed region formed by Σ is a hyper-rectangle in Ω and therefore we sample points in each axis of Ω within the boundaries defined by Σ. Such sampling provides a Poisson distribution over the volume of Σ [46].

The next step is going through all the N points in $\mathcal{P}$ and filtering out those that are not feasible. The final time $t_{f_i}(\sigma_i)$ is determined for each point σ_i checked. Hence, we check if the constraints are satisfied for time $t = \lambda t_{f_i}$ where $\lambda = \{0, \lambda_1, \lambda_2, ..., 1\} = \{0, \Delta\lambda_1, \Delta\lambda_1 + \Delta\lambda_2, ..., 1\}$. Meaning, vector $\mathbf{\Psi}(\lambda t_{f_i}, \sigma_i)$ is checked to be negative for all $0 \le \lambda \le 1$. Those that do not satisfy the constraints are eliminated and the feasible set $\Omega_\Sigma \subset \Sigma$ with size $m \le N$ is output. However, scanning the constraint $\mathbf{\Psi}(\lambda t_{f_i}, \sigma_i)$ for $\lambda = \{0, \Delta\lambda, 2\Delta\lambda, ..., 1\}$ where $\Delta\lambda$ is a constant value is

Algorithm 1 *Search_domain*$(\Sigma, \Psi, N, \epsilon_b)$

Input: The allowed set Σ, set of constraints Ψ, number of points to generate N, and tolerance ϵ_b.
Output: Set of feasible points Ω_Σ in Σ.

```
1: Generate the set $\mathcal{P} = \{\sigma_1, ..., \sigma_N\}$ of $N$ uniformly distributed random points within $\Sigma$.
2: Calculate $S_{max}$.  // using optimization prob. in (41).
3: Initiate empty set $\Omega_\Sigma$.
4: for $i = 1 \to N$ do
5:    if $\neg(Adaptive\_Check(\sigma_i, \Psi, S_{max}, \epsilon_b))$ then
6:       Add $\sigma_i$ to $\Omega_\Sigma$.
7:    end if
8: end for
9: return  $\Omega_\Sigma = \{\sigma_1, ..., \sigma_m\}$   // return a set of $m \le N$ feasible points.
```

rather risky. For a relatively large step size $\Delta\lambda$, a value in $\mathbf{\Psi}$ might ascend over zero and descend below again within the discretized step size. Moreover, too small step sizes could be unnecessary and the computational price would include very long runtime. Therefore, we present a simple adaptive step size algorithm to fine tune the time steps and diagnose or rule out such scenarios.

Definition 6. *The constraint value of a feasible point σ_i at time λt_{f_i} is defined to be*

$$\tilde{\Psi}_{i,\lambda} = \max_j \{\Psi_j(\lambda t_{f_i}, \sigma_i)\} \ , \tag{39}$$

where Ψ_j is the j^{th} component of the constraint vector $\mathbf{\Psi}$.

That is, the constraint value is the shortest distance at time λt_{f_i} from point σ_i to the boundary of the closest constraint. Notice that we refer to a distance with a positive value, but the value of $\tilde{\Psi}_{i,\lambda}$ is maintained negative for an indication that σ_i is a feasible point. Assume that the change rate of the constraint value with regard to λ is bounded by

$$\frac{\Delta\tilde{\Psi}_{i,\lambda}}{\Delta\lambda} \le S_{max}, \ \forall\ 0 \le \lambda \le 1 \ . \tag{40}$$

That is, the maximum slope of the constraint value $\tilde{\Psi}_{i,\lambda}$ is S_{max}. Under this assumption we can say that if at time λt_{f_i} the constraints are satisfied, $\tilde{\Psi}_{i,\lambda} < 0$, then the minimum time for the constraint to reach the zero line is $(\lambda + \Delta\lambda_{min})t_{f_i}$ where $\Delta\lambda_{min} = -\frac{\tilde{\Psi}_{i,\lambda}}{S_{max}}$. Therefore, as we get closer to a boundary of a constraint, we decrease $\Delta\lambda$ such that reaching above the zero line in that time frame is not possible. Figure 6a illustrates the selection of $\Delta\lambda$ as it gets smaller when approaching the zero line and larger when receding. However, in this adaptive approach, even though $\tilde{\Psi}_{i,\lambda}$ might pass the zero line, the algorithm will never do so as it will continue to decrease $\Delta\lambda$. Therefore, we bound such that the algorithm will stop checking the current σ_i (and remove it) if $0 < \tilde{\Psi}_{i,\lambda} < \epsilon_b$, where $\epsilon_b < 0$ is a value that will be defined further in the algorithm's analysis. This also serves as a safety distance, assuring the solution is far enough from the constraint's boundary. An example of such is shown in Figure 6b.

In order to calculate $\Delta\lambda_{min}$ we must first find S_{max}. Thus, we differentiate the constraint vector by λ to acquire its slope $\frac{\partial\Psi(\lambda t_f,\sigma)}{\partial\lambda}$, where t_f is the maximum possible goal time based on the maximum allowed control gains given in Σ. S_{max} is the maximum slope of all components over all time and can be computed by the following maximization problem

$$\begin{aligned} S_{max} = \max_{\lambda,\sigma,j} \quad & S_j(\lambda t_f, \sigma) \\ \text{subject to} \quad & 0 \le \lambda \le 1 \\ & \sigma \in \Sigma \end{aligned} \tag{41}$$

where S_j is the j^{th} component of the constraint's derivative $S(\lambda t_f, \sigma) = \frac{\partial\Psi(\lambda t_f,\sigma)}{\partial\lambda}$. This could be computed analytically using Kuhn-Tucker conditions [47] or numerically. The adaptive step size function *Adaptive_Check*$(\cdot)$ is presented in Algorithm 2.

Algorithm 2 *Adaptive_Check*$(\sigma_i, \Psi, S_{max}, \epsilon_b)$.

```
Input: $\sigma_i$, the set of constraints $\Psi$, slope $S_{max}$, and the tolerance $\epsilon_b$.
Output: Boolean: 1 if $\sigma_i$ is feasible, 0 if not feasible.
 1: Set $\lambda = 0$.
 2: Calculate $t_{f_i}$ according to the control gains in $\sigma_i$ using (28).
 3: while ( $\lambda \le 1$ ) do
 4:    Calculate $\tilde{\Psi}_{i,\lambda} = \max_j \{\Psi_j(\lambda t_{f_i}, \sigma_i)\}$.
 5:    if $\neg$( $\tilde{\Psi}_{i,\lambda} > \epsilon_b$ ) then
 6:       Return 0.    // failed point.
 7:    else
 8:       Calculate $\Delta\lambda = -\frac{\tilde{\Psi}_{i,\lambda}}{S_{max}}$.
 9:       $\lambda = \lambda + \Delta\lambda$.
10:    end if
11: end while
12: Return 1.    // feasible point.
```

The adaptive step size with the maximum slope S_{max} provides efficient sweep of the generated points by reducing the number of checks for each point in $\mathcal{P}$. However, for high-dimensional systems such adaptive search by its own can take a long time due to complex equations of motion of the system. Therefore, it is important to check first for the kinematic constraints and then for the torque constraints. In that way, only candidate trajectories that passed the kinematic constraints will be checked for the torque constraints. Experiments have shown a 90% runtime reduction compared to a non-separated constraint check.

Algorithms 1 and 2 search for a feasible set of points in Σ. Though the search will find a solution in a given probability, as will be discussed later on, domain Σ is relatively large and the solutions found would be sparse. Hence, we treat the first search of N points within Σ as a preliminary search followed by refinement and a more focused search. The initial set Ω_0 found outlines a coarse structure of the feasible region Ω_f and the purpose of the focused search is to broaden the number of solutions within that structure and optimize the solution. Therefore, the idea is for each of the m points found in Ω_0, to search in its area bounded by a small hyper-rectangle denoted as Σ_i as seen in Figure 7. The size of the hyper-rectangles is chosen heuristically to achieve a good cover by calculating the standard deviation $\varsigma \in \mathbb{R}^d$ of Ω_0 along each axis. Function *Enclosed_box*(σ_i, ς) returns the allowed search domain Σ_i around point $\sigma_i \in \Omega_0$ such that it is a hyper-rectangle with edge lengths $\varsigma_j,\ j = 1, ..., d$.

Algorithm 3 presents the whole scheme of using the *Search_domain* function for initial and advanced search in Σ. The first step (Line 1) of the algorithm is to determine the number of random points N such that the probability to find a solution is more than a user defined probability $1 - P_{max}$. The calculation of N based on the choice of P_{max} will be presented later in the algorithm's analysis. Next, in Line 2 the initial feasible set Ω_0 is found by calling function

Search_domain to search Σ with N random points. Next, the standard deviation of the m points in Ω_0 is calculated followed by a search around each point in Ω_0 using a for-loop. In each iteration of searching around $\sigma_i \in \Omega_0$, the corresponding search domain Σ_i is calculated and searched with N/m points yielding a total of $2N$ random points checked. For each iteration, a set Ω_i of m_i feasible points is outputted. Finally, the algorithm outputs the unified set Ω_f of the feasible sets found $\Omega_0, \Omega_1, ..., \Omega_m$ with a total of $m_f = \sum_{i=0}^{m} m_i$ feasible solutions. In the experimental section we present the contribution of Algorithm 3 to the overall performance. The algorithm dramatically increased the number of feasible solutions but also increased the runtime.

Algorithm 3 $Feasibility\_search(\Sigma, \Psi, P_{max}, \epsilon_b)$

```
Input: The allowed set $\Sigma$, set of constraints $\Psi$, probability $P_{max}$, and tolerance $\epsilon_b$.
Output: Set of feasible points $\Omega_f$.
1: Calculate number of random points $N$ according to (56), such that the probability to find a solution in $\Sigma$ is more than $1 - P_{max}$.
2: $\Omega_0 = Search\_domain(\Sigma, \Psi, N, \epsilon_b)$.
3: $m = length(\Omega_0)$.   $//\Omega_0 = \{\sigma_1, ..., \sigma_m\}$.
4: $\varsigma \leftarrow$ Standard deviation of $\Omega_0$.
5: for $i = 1 \rightarrow m$ do
6:    Set $\Sigma_i = Enclosed\_box(\sigma_i, \varsigma)$.
7:    $\Omega_i = Search\_domain(\Sigma_i, \Psi, \frac{N}{m}, \epsilon_b)$.
8: end for
9: Return $\Omega_f = \bigcup_{i=0}^{m} \Omega_i$.
```

5.5. Optimal solution

The final step of the algorithm is selecting the optimal solution among the set of feasible points $\Omega_f = \{\sigma_1, ..., \sigma_{m_f}\}$ found in Algorithm 3. Given the cost function $C(\sigma)$, the optimal solution σ^* is found according to a simple naive search. That is, we search for point $\sigma_k = \sigma^* \in \Omega_f$ that best minimizes $C(\sigma)$ according to

$$\sigma^* = \arg\min_{\sigma_i} C(\sigma_i), \ i = 1, ..., M_f, \tag{42}$$

where $M_f = length(\Omega_f)$. It is also possible to use a Gradient Descent (GD) method [48] to refine the solution and find a local minimum in the neighborhood of σ^*.

5.6. Overall algorithm

The full in-hand regrasping planning algorithm is presented in Algorithm 4. The algorithm receives input of the system's constraints in the form of inequalities (35), the desired probability P_{max} (presented in the next section), and the convergence tolerance ϵ_b. The algorithm computes the optimal parameterization vector σ^*. Based on the structure of σ^* defined in (29), the algorithm outputs the optimal initial pose $\mathbf{p_0}$ of the gripper and object before release, and the gains for the CT controller.

Algorithm 4 In-hand regrasping planning algorithm

```
Input: Set of constraints $\Psi$, probability $P_{max}$, and tolerance $\epsilon_b$.
Output: Initial gripper pose $\mathbf{p_0}$ and the control gains matrix $K_p$.
1: Define allowed subset $\Sigma \in \Omega$.
2: $\Omega_f = Feasibility\_search(\Sigma, \Psi, P_{max}, \epsilon_b)$.
3: Find optimal solution $\sigma^* \in \Omega_f$ using (42).
4: Extract initial pose $\mathbf{p_0}$ from $\sigma^*$.
5: Extract control gains matrix $Kp$ from $\sigma^*$.
6: Output $\mathbf{p_0}$ and $K_p$.
```

6. Analysis

In this section statistical and complexity analysis is performed for the proposed planning algorithm.

6.1. Statistical analysis

A statistical analysis is performed in this subsection to calculate the probability to find a solution if it exists. First, we define an operator that measures the volume of a set.

Definition 7. *Let a map $\mathcal{L}_V : \Upsilon \to \mathbb{R}_{>0}$ such that $\Upsilon \subset \mathbb{R}^d$. The value of $\mathcal{L}_V \circ \Upsilon$ is the hyper-volume measure of Υ according to the Lebesgue measure [49].*

Let us assume Ω_f is known and let $0 \le \rho \le 1$ be the relative portion of Σ that is feasible; that is, the ratio between the hyper-volumes of Ω_f and Σ:

$$\rho = \frac{\mathcal{L}_V \circ \Omega_f}{\mathcal{L}_V \circ \Sigma} \ . \tag{43}$$

For simplification and practical usage, we will refer to Σ as a hyper-rectangle in Ω (Figure 7). Let X_j be a random variable that takes the value 1 if $\sigma_j \in \Sigma$ falls within Ω_f and 0 if not. Thus, X_j is defined

$$X_j = \begin{cases} 1, & \sigma_j \in \Omega_f \\ 0, & \sigma_j \notin \Omega_f \end{cases} \ . \tag{44}$$

Because X_j can take only two values $\{0, 1\}$ and the probability of a random $\sigma_j \in \Sigma$ to fall in Ω_f is ρ, then $X_j \sim$ *Bernoulli*(ρ). Therefore, the probability for a random point σ_j to have a compatible random variable $X_j = y$ is

$$P(X_j = y) = \rho^y (1-\rho)^{1-y} \text{ for } y \in \{0, 1\} \ . \tag{45}$$

If a set of N points $\mathcal{P} = \{\sigma_1, ..., \sigma_N\} \in \Sigma$ are independent, identically distributed random variables, and their compatible random variables $X_1, ... X_N$ are all Bernoulli distributed with success probability ρ, then the sum of the random variables $Y = \sum X_i \sim$ *Binomial*(N, ρ). Therefore, the probability to have k points in Ω_f for N random points ($k = 0, 1, ..., N$) is

$$P(Y = k) = \begin{pmatrix} N \\ k \end{pmatrix} \rho^k (1-\rho)^{N-k} \tag{46}$$

where $\begin{pmatrix} N \\ k \end{pmatrix}$ is the binomial coefficient representing the number of combinations distributing k successes in N random points. The Poisson distribution can be used as an approximation of the Binomial distribution if the number of random points N goes to infinity ($N \to \infty$) and the Bernoulli probability is sufficiently small ($\rho \to 0$). Therefore, $Y \sim$ *Poisson*$(\overline{\lambda})$ where $\overline{\lambda} = N\rho$. That is,

$$P(Y = k) \simeq \frac{\overline{\lambda}^k}{k!} e^{-\overline{\lambda}} \ . \tag{47}$$

We determine that if a solution exists, the highest allowed probability that the solution will not be found ($k = 0$) out of N random points is defined to be

$$P(Y = 0|\rho) = e^{-N\rho} \ . \tag{48}$$

In the next Theorem we show the probability to find a solution in Σ if one exists.

Theorem 1. *The probability to find a solution in Σ, if one exists, approaches 1 as the number of generated random points approaches infinity.*

Proof. The probability not to find a solution in Σ is given in (48). Therefore, the probability to find a solution is

$$P_s(N) = P(Y > 0|\rho) = 1 - e^{-N\rho} \ . \tag{49}$$

For a given constant and finite ρ, if we increase N to infinity, then $e^{-N\rho} \to 0$ and the limit for the probability is

$$\lim_{N \to \infty} P_s(N) = 1 \ . \tag{50}$$

That is, as N approaches infinity, the probability to find a solution in Σ approaches 1. $\square$

However, taking an infinite number of points is not feasible. Hence, we provide a formula for finding the number of random points to be taken given the highest allowed probability P_{max} not to find a solution. Based on (48), the number N of random points in Σ must be chosen such that

$$e^{-N\rho} \leq P_{max} \ . \tag{51}$$

This condition states that given ρ, the number of random points to be chosen such that if a solution exists, the probability not to find it is lower than P_{max}. That is, the probability to find a solution is higher than $1 - P_{max}$.

The last issue that must be addressed is how to determine ρ such that (51) could be used to calculate N given P_{max}. The determination of ρ is a resolution of how close we allow the desired parameters vector to be from the constraint's boundaries. That is, how small do we allow the feasibility set to be in order to be considered to exist. If the volume of the feasibility set Ω_f is very small, the points within it must be very close to the constraint's boundary. This is an undesired situation. Such a case must be bounded so that if Ω_f is too small, we would determine that a solution does not exist. For that matter, we propose a heuristic approach of approximating a characteristic measure of the constraints. We define a characteristic measure that is the average minimum distance from the center of the feasible region to the constraint's boundaries over time. The centroid of the subspace formed by the constraints in (35) at time $\lambda_i t_g$ is calculated by

$$\sigma_{b_i} = arg \min_{\sigma} \left\{ \sum_{j=1}^{z} a_{j,i} \Psi_j(\lambda_i t_g, \sigma) \right\} \ , \tag{52}$$

Ψ_j is the j^{th} component of the constraint vector $\mathbf{\Psi}$ and $\lambda_i = 0, \overline{\Delta\lambda}, ..., \beta\overline{\Delta\lambda}$, where $\overline{\Delta\lambda}$ is some constant step size and $\beta = 1/\overline{\Delta\lambda}$. Because each constraint in $\mathbf{\Psi}$ has a different gradient, which affects the distance-value ratio, it is multiplied by its relative weight $a_{j,i}$ given by

$$a_{j,i} = \frac{\sum_{k=1}^{z, k\neq j} \Psi_k(\lambda_i t_g, \sigma)}{\sum_{k=1}^{z} \Psi_k(\lambda_i t_g, \sigma)} \ . \tag{53}$$

Hence, the average distance over time (with step size $\overline{\Delta\lambda} t_g$) is given by

$$r = \frac{1}{\beta} \sum_{i=1}^{\beta} \left(\frac{1}{z} \sum_{j=1}^{z} \Psi_j(\lambda_i t_g, \sigma_{b_i}) \right) \tag{54}$$

and is a characteristic measure of the constraint's size. Heuristically, the average is computed in β time steps and increasing β will increase the accuracy. Then, the minimum allowed volume Ω_{min} is defined as

$$\mathcal{L}_V \circ \Omega_{min} = (\eta r)^d \ , \tag{55}$$

where $0 < \eta \leq 1$ is the allowed fraction of r and is user-defined. That is, we define a hyper-cube enclosed within Ω_f with edge length ηr and compute its volume to be the minimal allowed one. Thus, ρ is chosen as $\rho_{min} = (\mathcal{L}_V \circ \Omega_{min})/(\mathcal{L}_V \circ \Sigma)$ or practically $\rho_{min} = (\eta r)^d / V_c$, where V_c is the volume of hyper-rectangle Σ calculated by multiplying its edge lengths. The minimal number of random points N for a given P_{max} can be calculated according to (51) by

$$N \geq -\frac{1}{\rho} \log(P_{max}) \ . \tag{56}$$

Moreover, the value of ϵ_b defined in section 5.4 to be the minimum allowed distance from the constraint boundary is chosen using the characteristic measure calculated in (54) to be $\epsilon_b = \eta r$.

In the following Theorem we conclude the probability to find a solution if such exists.

Theorem 2. *If a solution for the time-varying problem exists in Σ and N satisfies* (56)*, the probability for the algorithm to find a solution is at least $1 - P_{max}$.*

Proof. The probability not to find a solution if one exists is given in (48). If the algorithm selects N random points in the allowed set Σ such that (56) is satisfied with given probability P_{max} and a minimum allowed volume ratio ρ_{min}, then substituting it in (48) yields

$$P(Y = 0|\rho_{min}) = P_{max} \ , \tag{57}$$

which is the probability not to find a solution. Hence, the probability to find a solution is $1 - P_{max}$. □

The choice of N is done ensuring finding a solution if such exists above a known probability $1-P_{max}$. The following corollary extends Theorem 2 to calculate the probability to find a solution given a maximal calculation runtime T_c and assuming the time for computing one point is known to be τ_c. τ_c is varies from one computer to another.

Corollary 1. *Given a desired calculation time T_c and the computer cycle time τ_c to calculate one point, the probability to find a solution if one exists is*

$$P_s = 1 - e^{-\frac{T_c}{\tau_c}\rho} \ . \tag{58}$$

Proof. The maximal number of random points that can be generated is $N_c = \left\lfloor \frac{T_c}{\tau_c} \right\rfloor$. Therefore, given ρ and according to (48), the probability not to find a solution is

$$P(Y = 0|\rho) = e^{-N_c\rho} = e^{-\frac{T_c}{\tau_c}\rho} \ . \tag{59}$$

Therefore, the probability to find a solution is $1 - e^{-\frac{T_c}{\tau_c}\rho}$. □

6.2. Complexity

The complexity of the proposed algorithm is presented next.

Theorem 3. *Searching for a solution using N random points, the upper bound time complexity of the proposed algorithm is in the order of $O(\frac{S_{max}}{\epsilon_b}N)$.*

Proof. We generate N random points in Σ and therefore, the N points are to be checked for feasibility, given the maximum constraint slope S_{max} and the maximum allowed boundary constraint distance ϵ_b. In the worst case, the constraint value $\tilde{\Psi}_{i,\lambda}$ approximately equals ϵ_b for all $0 \leq \lambda \leq 1$. Moreover, in the worst case, during the δ_S iterations, no points will be filtered-out except in the last iteration. Therefore, each time step size will be $\Delta\lambda = \frac{\epsilon_b}{S_{max}}$ and the number of time steps each σ_i will be checked for satisfying the constraints is $\frac{1}{\Delta\lambda} = \frac{S_{max}}{\epsilon_b}$. Hereafter, N random points will be checked, in the worst case, $\frac{S_{max}}{\epsilon_b}$ times. Thus, the upper bound time complexity of the algorithm is in the order of $O(\frac{S_{max}}{\epsilon_b}N)$. □

7. Simulations and Experiments

In this section we present simulations and experiments of the proposed methods. Clearly illustrating a spatial motion on paper is not an easy task. Therefore, for easy illustration of the regrasp manipulation, we first present a non-optimal simple motion using the CT control method. Afterwards, we test the proposed algorithm and simulate the output optimal solution. Both simulations are conducted on a KUKA LWR iiwa 14 robotic arm. Further, we present a regrasp demonstration conducted with a planar 3R manipulator and an object sliding on an inclined air hockey frictionless table. Video of the simulations and demonstrations can be seen in Extension 1 enclosed to this paper.

7.1. Simple regrasp simulation

In this subsection we present a simple regrasp manipulation according to the method presented in Section 4. We modeled the 7R KUKA LWR iiwa 14 robotic arm in Matlab and SimMechanics[1], and can be seen in Figure 8. For simplicity of the inverse kinematics problem, the 3^{rd} joint is considered rigid and not used during the motion. Hence, the arm's DOF are as seen in Figure 8. For easier illustration, we set the goal and the initial poses of the gripper and object such that the manipulation would be planar. Hence, the chosen manipulation is arbitrary and was not computed to be optimal using the SKIP algorithm.

The goal of the following simulation is to regrasp a ball in a different orientation. At the end of the manipulation, the gripper should be rotated about the $\hat{\omega}_o = (0\ 0\ 1)^T$ axis of the ball's CF with angle $\alpha = -45^o$. The initial pose of the ball and gripper are equivalent and chosen to be $\mathbf{p_g}(0) = \mathbf{p_o}(0) = (0.6\ 0\ 0.3\ 90^o\ 90^o\ 0^o)^T$, yielding a relative pose of $\overline{\mathbf{p}}(0) = 0$. Assuming no rotation of the ball during free-fall, the desired relative pose according to Proposition 1 at the final time t_f would be $\zeta = (0\ 0\ 0\ -180^o\ -45^o\ -180^o)^T$. We chose a control gains matrix to be $K_p = diag(1600\ 1600\ 1600\ 400\ 1600\ 400)$ yielding an approximate final time of $\tilde{t}_f = 0.23s$. Figure 9 presents snapshots of the regrasping motion under these terms. The change in the rotation angle $\alpha(t)$ about the desired vector $\hat{\omega}_o$ is seen in Figure 10. As anticipated, approximately at time $0.23s$ the rotation angle reaches -45^o with zero velocity and the gripper closes on the ball.

Table 1: Torque and angle ranges for the KUKA LWR iiwa 14.

Joint	1	2	3	4	5	6
Ang.	±170	±120	±120	±170	±120	±175
Torq.	320	320	176	110	40	40

7.2. Regrasp planning algorithm experiments

Here we present experiments of the planning algorithm presented in Section 5. The KUKA arm described in Subsection 7.1 is also used here. The KUKA's joint range and maximum torques as stated by the manufacturer are shown in Table 1. Hence, we have formulated the constraints in (35) such that the angles and torques would be in these ranges, and the gripper would only be above the table, i.e., with positive z coordinate in CF $\mathcal{O}$. By solving the maximization problem in (41), the maximum slope is $S_{max} = 3.18$. The allowed region Σ for generating random parameterization points was defined such that the initial pose would be within the arm's workspace and the control gains have an upper boundary. This forms an hyper-rectangle in Ω defined by

$$\begin{pmatrix}0 & -L & -L & -\pi & -\pi & -\pi\end{pmatrix}^T \le \mathbf{p_o} \le \le \begin{pmatrix}L & L & L & \pi & \pi & \pi\end{pmatrix}^T \tag{60}$$

$$0 \le K_{p_i} \le 2000,\ i = 1, ..., 6 \tag{61}$$

where $L = \sum_{i=1}^{6} L_i$.

First, we conducted experimental runs while requesting rotation of $\alpha = -30^o$ about $\hat{\omega}_o = (0\ 0\ 1)^T$ with different values for N. The initial relative pose between the gripper and ball is as seen in Figure 8. In Figure 11 the average number of feasible solutions was found as a function of N. In blue, the number of points found in the initial run and in yellow, the number of feasible points found in the advanced search. The contribution of the advanced search can be clearly seen for significantly increasing the feasible number of points found. It can be seen that in this case, about 2% of the $2N$ generated points were found feasible. The average computation time of this experiment can be seen in Figure 12. It should be noted that the high runtime is due to mathematical operations on 6×6 matrices of the robot's kinematics and dynamics. Lower DOF systems take significantly less time. For example, in a three DOF system, the computation time is around 500 milliseconds.

Next, we present a time optimal solution for rotation of the gripper about the $\hat{\omega}_o = (0\ 1\ 1)^T/\sqrt{2}$ with angle $\alpha = -80^o$ relative to the ball. By selecting $\eta = 0.1$ (10% of the average distance r), we obtain the relative portion $\rho = 1.38 \cdot 10^{-3}$ and the minimum allowed distance from the constraint boundary $\epsilon_b = -1.01 \cdot 10^{-7}$. Choosing a $P_{max} = 0.6$, i.e., 60% of not finding a solution, yields the generation of $N = 3,697$ random points. We chose such a high probability not to find a solution but to acquire lower computation time. If we fail to find solutions, we could reduce the probability. Nevertheless, generating the N points yielded 4 solutions in the first run with a total of 432 solutions after the advanced search. The whole computation time took 9.6 minutes due to the high-dimensionality of the arm's dynamics. The output time for an optimal solution yielded an initial pose of

$$\mathbf{p_o} = (0.4\ \ 0.06\ \ 0.35\ \ -29.55^o\ \ 61.01^o\ \ -57.64^o)^T, \tag{62}$$

[1]Matlab and SimMechanics are registered trademarks of Mathworks, Inc.

control gains

$$K_p = diag(1,097 \;\; 742 \;\; 3,278 \;\; 1,223 \;\; 1,167 \;\; 2,751), \tag{63}$$

and has an approximated goal time of $\tilde{t}_f = 0.23s$. Snapshots of the time optimal regrasp motion can be seen in Figure 13. The response of the relative rotation angle about $\hat{\omega}_o$ is shown in Figure 14. The joint angle and torque responses are in Figures 15 and 16, respectively. They are shown to be within their limits as defined in Table 1. In Figure 17 the xyz coordinates of the gripper and object are illustrated in solid and dashed lines, respectively. The gripper is shown to follow the falling ball's position with a small error up to 1.6 millimeters after regrasp along the axes. This error is due to the regrasp based on the convergence of the angle α and not the position. Increasing the control gains would lead to zero position error. However, this would lead to exceeding the limits of the torques as joints 2 and 4 are on the boundaries. The aim is to rotate the relative angle and not translate; therefore, this solution is feasible and optimal to the demand.

7.3. Regrasp experiments

An experimental setup was designed and built to provide a demonstration of the proposed regrasping method. A three degrees of freedom robotic arm was assembled with four MX-106 Dynamixel actuators (two are coupled at the joint of the first link to double the torque) for its joints and two MX-28 actuators for the two jaws' gripper. The actuators were found unable to provide the torques specified by the producer. Therefore, the arm was mounted parallel to a 5^o inclined air hockey table with a $2000W$ air compressor. Thus, the falling disk-shaped object could slide with minimum friction. A motion capture system by OptiTrack including six peripherals cameras was set to track the moving objects position and orientation. The experimental setup can be seen in Figure 18. Since this is a custom-built arm, its dynamic properties are not known and were approximated according to the model identification method described in Section 4.

Figure 19: Snapshots of an experiment regrasping a disk to $\alpha = 25^o$.

Since this is a planar setup, the pose of the gripper is three-dimensional. For a time optimal solution, the initial pose of the gripper was set to $\mathbf{p_g}(0) = (x_g \; y_g \; \theta_g)^T = (0.22 \; 0.22 \; 90^o)^T$ and the CT control gains matrix was $K_p = diag(20 \; 18 \; 23)$. The relative pose at time $t = 0$ was initialized to zero and the goal relative angle was set to $\alpha = 25^o$ about the axis perpendicular to the table. Snapshots of this motion are shown in Figure 19. The relative angle response and the positions of the gripper and object can be seen in Figures 20 and 21, respectively. It should be noted that due to noisy signal readings from the cameras and dynamixels, a simple mean filter was applied for noise reduction. Accurate angle regrasp can be seen along with a relatively good position tracking of up to $9mm$ error during the fall. However, at the regrasp the position error was about $0.5mm$, which is a very good accuracy. The results prove the feasibility of the proposed regrasping approach.

AlphaExp1Regrasp.png

Figure 20: Response of the rotation angle $\alpha(t)$ during the experimental regrasp to $\alpha = 25^o$.

PositionExp1Regrasp.png

Figure 21: Position response of the gripper and object during the experimental regrasp to $\alpha = 25^o$.

8. Conclusions

In this paper we have presented a bioinspired approach for performing in-hand regrasping of an object. Similarly to using the motion of the whole arm and the simple opening and closing operation of the human hand, we use the dynamics of a robotic arm and a non-dexterous gripper. The object is released into mid-air while the gripper reconfigure its relative pose until catching it in the desired orientation. We believe that the approach, once implemented in industrial production lines, can enhance the use of robotic arms for multiple tasks on the same object. This will reduce the number of robotic arms in the factory, shorten the manufacturing time and reduce costs.

In this paper we proposed a motion planning paradigm for the regrasping. Interception between the gripper and the desired object in the desired relative pose is implemented using computed-torque control. The CT controller enables acquiring an analytical expression of the joint trajectories and an approximation of the regrasping time. With that, we have presented a stochastic motion planning algorithm that finds a set of regrasping solutions satisfying the kinodynamic constraints of the problem. An important contribution of this algorithm is its ability to optimize the initial state of the gripper and the control gains magnitude. A feasible set of solutions is found by generating random points within the allowed region of the free parameters and checking each one for feasibility. The feasibility analysis was done with an adaptive step size algorithm, decreasing the step size if the point approaches the constraint boundary. Moreover, an optimal solution is chosen from the set of feasible solutions.

Analysis has shown that the probability to find a solution if one exists approaches one as the number of random points increases to infinity. Nevertheless, it chooses the number of random points based on our choice of a probability to find a solution. Moreover, the statistical analysis enabled an automatic parameters selection for the algorithm to find a solution in a given probability or runtime. In the worst case, the algorithm has a time complexity in the order of $O(\frac{S_{max}}{\epsilon_b}N)$. It should be noticed that the algorithm plans the motion off-line prior to the motion. The planning time is in the order of a few minutes which is reasonable for off-line planning in industrial applications. Future work will involve runtime reduction. To reduce the runtime, parallel computation could be used as the check of each point is independently computed. Moreover, reducing region Σ could also be utilized to replace pose constraints that are currently integrated in Ψ. This could dramatically reduce the runtime of checking each point. Future work would also consider application for online planning where uncertainty in the environment such as dynamic obstacles is taken into account.

The proposed algorithm is a motion planning paradigm for the in-hand regrasping manipulation. We provided a demo of the manipulation for the planar case. However, to perform such manipulation in a spatial and practical industrial application, one must also address the tracking side of the problem. In particular, an agile vision and estimation system must be setup such that it will provide high frequency feedback to the controller. Examples of such potential systems are presented in [25, 50]. It is also possible to add a constraint to (35) based on the feedback frequency of a given system. This may enable a lower frequency tracking system by allowing increased interception time. We also note that the tracking system will have to operate from several directions to overcome possible occlusion of the object by the robot. We leave this issue for future work.

References

[1] N. Dafle, A. Rodriguez, R. Paolini, B. Tang, S. Srinivasa, M. Erdmann, M. Mason, I. Lundberg, H. Staab, T. Fuhlbrigge, Extrinsic dexterity: In-hand manipulation with external forces, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2014, pp. 1578–1585.

[2] M. A. Roa, R. Suarez, Regrasp planning in the grasp space using independent regions, in: Proceedings of the IEEE/RSJ international conference on Intelligent robots and systems, IROS'09, IEEE Press, Piscataway, NJ, USA, 2009, pp. 1823–1829.

[3] A. Fernandez, J. Gazeau, S. Zeghloul, S. Lahouar, Regrasping objects during manipulation tasks by combining genetic algorithms and finger gaiting, Meccanica 47 (4) (2012) 939–950.

[4] A. Sintov, A. Shapiro, A stochastic dynamic motion planning algorithm for object-throwing, in: Proceedings of the IEEE International Conference on Robotics and Automation, Seattle, WA, USA, 2015, pp. 2475–2480.

[5] S. LaValle, J. Kuffner, J.J., Randomized kinodynamic planning, in: Proceedings. IEEE International Conference on Robotics and Automation, Vol. 1, 1999, pp. 473–479.

[6] P. Tournassoud, T. Lozano-Perez, E. Mazer, Regrasping, in: Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 4, 1987, pp. 1924–1928.

[7] T. Lozano-Pérez, J. L. Jones, E. Mazer, P. A. O'Donnell, W. E. L. Grimson, P. Tournassoud, A. Lanusse, Handey: A robot system that recognizes, plans, and manipulates, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1987, pp. 843–849.

[8] Z. Xue, J. M. Zollner, R. Dillmann, Planning regrasp operations for a multifingered robotic hand., in: Proceedings of the IEEE Conference on Automation, Science and Engineering, 2008, pp. 778–783.

[9] H. Kim, S. Park, A strong cutting plane algorithm for the robotic assembly line balancing problem, International Journal of Production Research 33 (8) (1995) 2311–2323.

[10] G. Levitin, J. Rubinovitz, B. Shnits, A genetic algorithm for robotic assembly line balancing, European Journal of Operational Research 168 (3) (2006) 811 – 825.

[11] D. Rus, In-hand dexterous manipulation of piecewise-smooth 3-d objects, The International Journal of Robotics Research 18 (4) (1999) 355–381.

[12] B. Corves, T. Mannheim, M. Riedel, Re-grasping: Improving capability for multi-arm-robot-system by dynamic reconfiguration, in: Intelligent Robotics and Applications, Vol. 7101, Springer Berlin Heidelberg, 2011, pp. 132–141.

[13] G. A. D. P. Caurin, L. C. Felicio, Learning based regrasping applied to an antropomorphic robot hand, ABCM Symposium series in mechatronics (2006).

[14] A. Sudsang, T. Phoka, Regrasp planning for a 4-fingered hand manipulating a polygon, in: Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 2, 2003, pp. 2671 – 2676 vol.2.

[15] P. Vinayavekhin, S. Kudohf, K. Ikeuchi, Towards an automatic robot regrasping movement based on human demonstration using tangle topology, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2011, pp. 3332 –3339.

[16] M. Stuheli, G. Caurin, L. Pedro, R. Siegwart, Squeezed screw trajectories for smooth regrasping movements of robot fingers, Journal of the Brazilian Society of Mechanical Sciences and Engineering 35 (2) (2013) 83–92.

[17] P. Grosch, R. Suarez, R. Carloni, C. Melchiorri, Planning setpoints for contact force transitions in regrasp tasks of 3d objects, in: Proceedings of the 17th IFAC World Congress, Vol. 17, IFAC International Federation of Automatic Control, Seoul, Korea, 2008, pp. 6776–6781.

[18] Y. Hasegawa, M. Higashiura, T. Fukuda, Simplified generation algorithm of regrasping motion - performance comparison online-searching approach with EP-based approach, in: Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 2, 2003, pp. 1811 – 1816 vol.2.

[19] T. Phoka, A. Sudsang, Contact point clustering approach for 5-fingered regrasp planning, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 4174 –4179.

[20] D. Rapela, U. Rembold, B. Kuchen, Planning of regrasping operations for a dextrous hand in assembly tasks, Journal of Intelligent and Robotic Systems 33 (2002) 231–266.

[21] B. Balaguer, S. Carpin, Bimanual regrasping from unimanual machine learning, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2012, pp. 3264 –3270.

[22] K. Harada, T. Foissotte, T. Tsuji, K. Nagata, N. Yamanobe, A. Nakamura, Y. Kawai, Pick and place planning for dual-arm manipulators, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2012, pp. 2281 –2286.

[23] J. Chen, M. Zribi, Control of multifingered robot hands with rolling and sliding contacts, The International Journal of Advanced Manufacturing Technology 16 (2000) 71–77.

[24] A. Cole, P. Hsu, S. Sastry, Dynamic control of sliding by robot hands for regrasping, IEEE Transactions on Robotics and Automation 8 (1) (1992) 42 –52.

[25] N. Furukawa, A. Namiki, S. Taku, M. Ishikawa, Dynamic regrasping using a high-speed multifingered hand and a high-speed vision system, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2006, pp. 181 –187.

[26] K. Tahara, K. Maruta, A. Kawamura, M. Yamamoto, Externally sensorless dynamic regrasping and manipulation by a triple-fingered robotic hand with torsional fingertip joints, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2012, pp. 3252 –3257.

[27] J. Shi, J. Z. Woodruff, K. M. Lynch, Dynamic in-hand sliding manipulation, in: Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2015, pp. 870 – 877.

[28] A. Sintov, O. Tslil, A. Shapiro, Robotic swing-up regrasping manipulation based on impulse-momentum approach and cLQR control, IEEE Transactions on Robotics 32 (2016) 1079–1090.

[29] A. Sintov, A. Shapiro, Swing-up regrasping using energy control, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 2016, pp. 4888–4893.

[30] T. Senoo, A. Namiki, M. Ishikawa, High-speed throwing motion based on kinetic chain approach, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008, pp. 3206–3211.

[31] S. Srinivasa, M. Erdmann, M. Mason, Using projected dynamics to plan dynamic contact manipulation, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005, pp. 3618 – 3623.

[32] K. M. Lynch, Dynamic manipulation with a one joint robot, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1997, pp. 356–366.

[33] K. Lynch, N. Shiroma, H. Arai, K. Tanie, The roles of shape and motion in dynamic manipulation: the butterfly example, in: Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 3, 1998, pp. 1958 –1963 vol.3.

[34] T. Tabata, Y. Aiyama, Tossing manipulation by 1 degree-of-freedom manipulator, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 1, 2001, pp. 132 –137.

[35] M. Higashimori, K. Utsumi, Y. Omoto, M. Kaneko, Dynamic manipulation inspired by the handling of a pizza peel, IEEE Transactions on Robotics 25 (4) (2009) 829 –838.

[36] K. M. Lynch, The mechanics of fine manipulation by pushing, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1992, pp. 2269–2276.

[37] J. Bernheisel, K. Lynch, Stable transport of assemblies by pushing, IEEE Transactions on Robotics 22 (4) (2006) 740 –750.

[38] M. Kopicki, R. Stolkin, S. Zurek, T. Morwald, J. Wyatt, Predicting workpiece motions under pushing manipulations using the principle of minimum energy, in: Proceedings of the RSS workshop on Representations for Object Grasping and Manipulation in Single and Dual Arm Tasks, 2010, submitted.

[39] M. Yashima, H. Yamaguchi, Dynamic motion planning whole arm grasp systems based on switching contact modes, in: Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 3, 2002, pp. 2492 –2499.

[40] R. Garcia-Rodriguez, G. Diaz-Rodriguez, Grasping and dynamic manipulation by soft finger-tips without object information, in: Proceedings of the 9th IEEE International Conference on Control and Automation, 2011, pp. 766 –771.

[41] O. Rodrigues, Des lois gometriques qui regissent les dplacements d' un systme solide dans l' espace, et de la variation des coordonnes provenant de ces dplacement considres indpendent des causes qui peuvent les produire, Journal de Mathematiques 5 (1840) 380–440.

[42] P. Khosla, T. Kanade, Experimental evaluation of nonlinear feedback and feedforward control schemes for manipulators, The International Journal of Robotics Research 7 (1) (1988) 18 – 28.

[43] J. Swevers, W. Verdonck, J. De Schutter, Dynamic model identification for industrial robots, IEEE Control Systems Magazine 27 (5) (2007) 58–71.

[44] R. Dorf, R. Bishop, Modern Control Systems, Pearson Prentice Hall, 2011.

[45] A. Sintov, A. Shapiro, Time-based RRT algorithm for rendezvous planning of two dynamic systems, in: Proceedings of the IEEE International Conference on Robotics and Automation, Hong-Kong, China, 2014, pp. 6745–6750.

[46] S. M. Stigler, Poisson on the poisson distribution, Statistics & Probability Letters 1 (1) (1982) 33 – 35.

[47] H. W. Kuhn, A. W. Tucker, Nonlinear programming, in: J. Neyman (Ed.), Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley, CA, USA, 1950, pp. 481–492.

[48] A. Cauchy, Méthode générale pour la résolution des systémes d'équations simultanées, C. R. Acad. Sci. Paris 25 (1847) 536–538.

[49] H. Lebesgue, K. May, Measure and the Integral, The Mathesis Series, Holden-Day, 1966.

[50] A. Namiki, N. Itoi, Ball catching in kendama game by estimating grasp conditions based on a high-speed vision system and tactile sensors, in: Proceedings of the IEEE-RAS International Conference on Humanoid Robots, 2014, pp. 634–639.

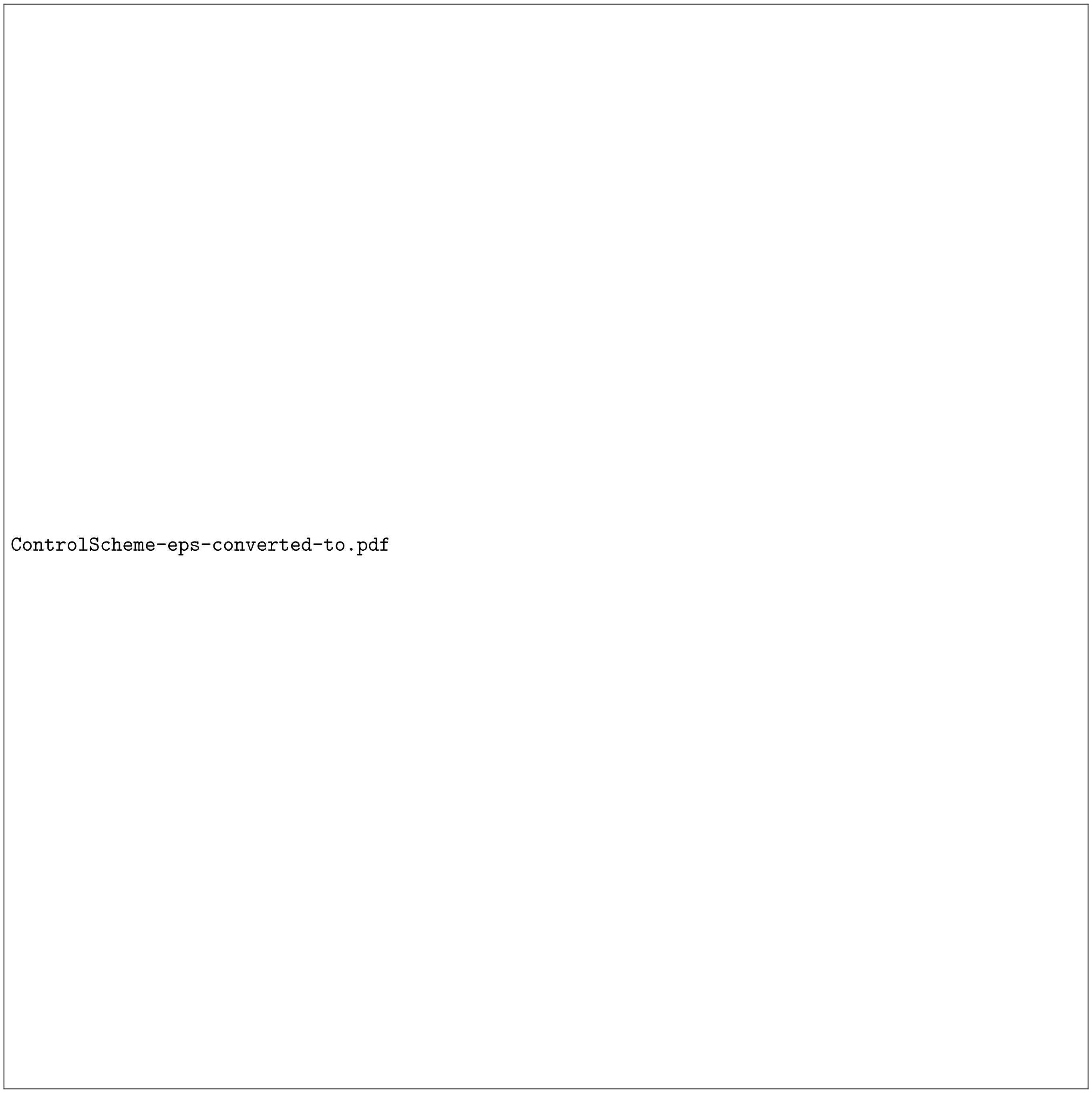

Figure 2: The proposed control scheme for the regrasping manipulation.

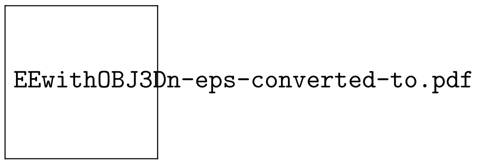

Figure 3: Illustration of the desired rotation about $\hat{\omega}_o$.

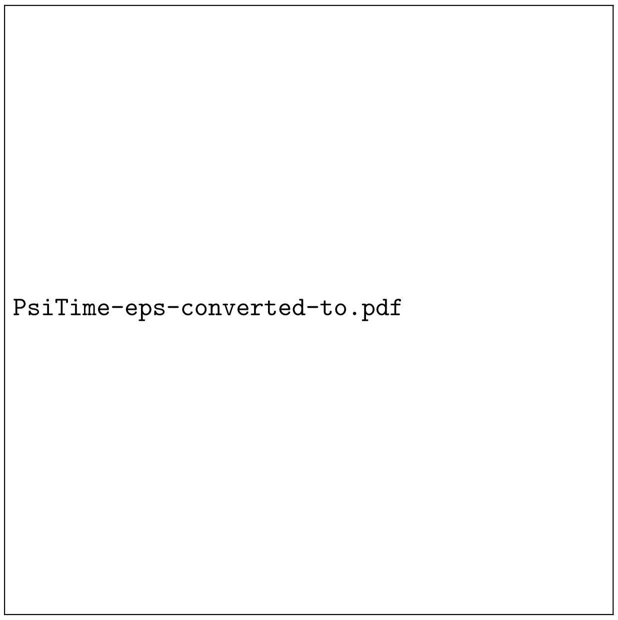

Figure 4: The TVC problem where $0 < t_a < t_b < t_f$.

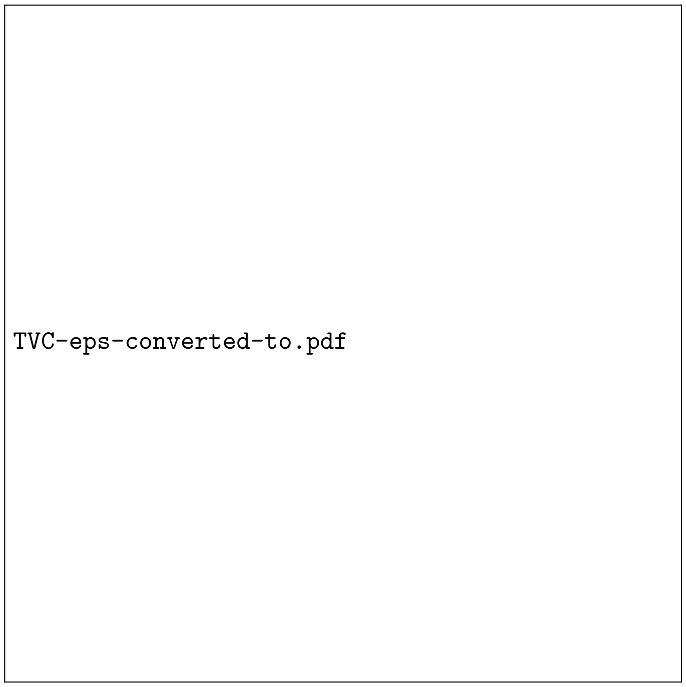

Figure 5: Vector σ^* satisfying Lemma 1 in an example where $\Omega \subseteq \mathbb{R}^2$. $\hat{\sigma}_1$ and $\hat{\sigma}_2$ are the axes Ω.

Smax-eps-converted-to.pdf

Figure 6: Adaptive step size algorithm.

rectangle-eps-converted-to.pdf

Figure 7: Search in small hyper-rectangles $\Sigma_i,\ i = 1, ..., m$ around feasible points found in the preliminary search.

KUKA-eps-converted-to.pdf

Figure 8: The KUKA iiwa arm with six joints and a jaw gripper holding a ball with zero relative pose.

SimpleRegrasp1.png

SimpleRegrasp2.png

SimpleRegrasp3.png

SimpleRegrasp4.png

SimpleRegrasp5.png

SimpleRegrasp6.png

SimpleRegrasp7.png

SimpleRegrasp9.png

AlphaSimpleRegrasp.png

Figure 10: Response of the rotation angle $\alpha(t)$ during simple regrasping.

SolVsN.png

Figure 11: Average number of solutions found with regard to different values of N.

TimeVsN.png

Figure 12: Average runtime as a function of N.

Figure 13: Snapshots of a time optimal ball regrasp to $\alpha = -80^o$.

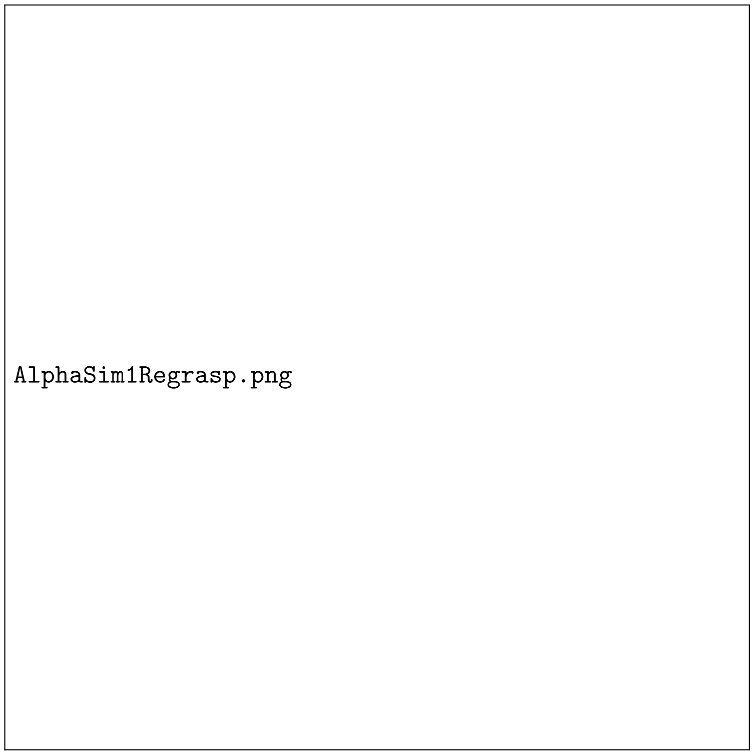

Figure 14: Response of the rotation angle $\alpha(t)$ during a time optimal regrasp to $\alpha = -80^o$.

AnglesSim1Regrasp.png

Figure 15: Joint angles of the KUKA arm during the time optimal regrasp to $\alpha = -80^o$. The dotted lines are the joint angle limits as declared by the KUKA manufacturer.

TorqueSim1Regrasp.png

Figure 16: Joint torques of the KUKA arm during the time optimal regrasp to $\alpha = -80^o$. The dotted lines are the joint torque limits as declared by the KUKA manufacturer.

xyzSim1Regrasp.png

Figure 17: Position of the gripper and object along the xyz-axes during the time optimal regrasp to $\alpha = -80^o$.

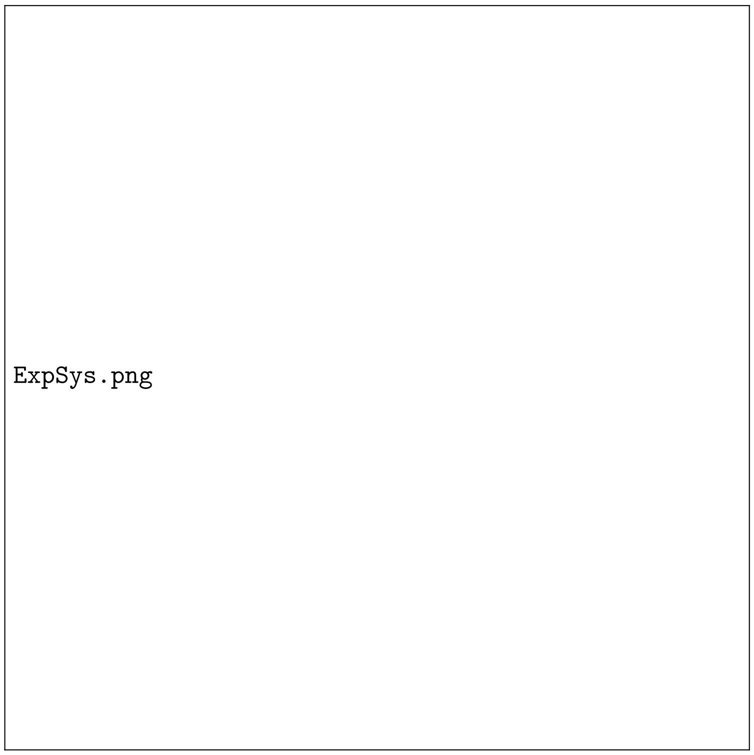

Figure 18: Experimental system: three DOF arm on an inclined air-hockey table.