2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Trajectory Optimization for Distributed Manipulation by Shaping a Physical Field

Adam Uchytil and Jií Zemánek

***Abstract*— Trajectory optimization is used to solve various planning tasks. In this paper we present a optimization-based method that solves a planning problem for multiple independent objects manipulated by a spatially continuous physical field. The field is generated and controlled (shaped) in real time by an array of actuators. In the paper we first formulate a trajectory optimization problem and a related initialization scheme, and then we demonstrate the proposed method using an experimental platform for distributed magnetic manipulation. The demonstrated task is that of planar reconfiguration of an ensemble of multiple objects, which significantly benefits from the inherent parallelism of the manipulation enabled by the array of actuators shaping the physical field. We show that the system can rearrange up to eight objects simultaneously while avoiding collisions.**

I. Introduction

Trajectory optimization has been successfully used to solve planning problems in many areas, including spacecraft [1], multiple UAVs [2], humanoid robot locomotion [3], robotic manipulators [4]. In this paper, we bring trajectory optimization to the domain of *distributed manipulation by shaping the physical fields*. In distributed manipulation, the individual actuators are distributed in space, each actuator having only a local effect, while together, they constitute a physical field. This field then exerts a force on single or multiple objects, as opposed to e. g. robotic arm, which only exerts a force on a single object. Typically, a control system is employed, whose goal is to shape the field to reach the desired motion of the manipulated objects. Different physical fields have been used to achieve distributed manipulation by a physical field: MEMS [5], dielectrophoresis [6], acoustophoresis [7], electrosmosis [8] and magnetophoresis, which we present later in this paper.

When feedback control is employed, one needs to search for actuator commands that exert desired forces on manipulated objects based on their positions (and possibly velocities). Different methods have been employed to solve this task: optimization, more precisely constrained least-squares problem [6], [7], [9], pseudoinverse [10], [11], and matrix inversion [12]. In this work, we assume the task of finding actuator commands for desired forces and object positions is solved, allowing us to focus on planning the motion of the individual objects. We search for trajectories that are collision-free with respect to individual objects while also being feasible in the sense of the physical field, i. e. appropriate actuator commands can be found for the planned forces.

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS22/166/OHK3/3T/13.

The authors are with Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague. {uchytada, jiri.zemanek}@fel.cvut.cz

An example planning problem for distributed manipulating by shaping a physical field is solved in [9], specifically a motion planning problem for multiple nanowires subject to the electrophoretic force generated by an array of electrodes. Authors employ a sampling-based algorithm to plan an optimal path. A constrained least square problem is then solved to find appropriate electrode voltages to track the planned path. In [13], the authors developed a new sampling-based planner in which trajectory optimization, more precisely *boundary value problem* (BVP), is solved to connect two trees.

We go in a different direction and solve the entire planning problem using optimization, i. e. trajectory optimization. We demonstrate our planning results using our experimental platform for distributed magnetic manipulation dubbed Magman. Magman is a platform that can manipulate multiple steel balls using a magnetic field generated by an array of solenoids.

Furthermore, we chose to demonstrate our planning results on the problem of tabletop rearrangement [14]. This class of problems involves geometrically similar objects that are supposed to be rearranged by single or multiple manipulators. Here, the distributed manipulation by shaping a physical field shows undeniable advantages as opposed to using manipulator(s). It allows for parallel manipulation without considering possible collisions between manipulators but only between the manipulated objects. Further possible application of our work could be planning trajectories for microparticles manipulated by dielectrophoretic force [6].

Contribution

The contribution of our work lies in the employment of trajectory optimization for planning the motion of multiple objects manipulated by an array of actuators that constitute a physical field, together with the demonstration of the planned trajectories using an experimental platform for distributed magnetic manipulation.

Mathematical Notation

We use $\mathcal{I}_N$ to denote the index set $\mathcal{I}_N = \{1, 2, \ldots, N\}$. When presented with a parameter (variable) $p_{j,\text{description}}$ it is to be interpreted as the parameter (variable) $p_{\text{description}}$ of the jth object. Operator $\lfloor x \rfloor$ rounds the number x to the closest integer less or equal to x.

II. General Formulation

A. Problem Definition

We consider the problem of simultaneously manipulating M objects from known initial to desired positions. The dynamics of the jth object is given by the state-equation

$$\dot{\boldsymbol{x}}_j(t) = \boldsymbol{f}_j(\boldsymbol{x}_j(t), \boldsymbol{u}_j(t)), \tag{1}$$

where $\boldsymbol{x}_j(t) \in \mathbb{R}^{2D}$ is the vector of the object positions and velocities, while $\boldsymbol{u}_j(t) \in \mathbb{R}^D$ is the force exerted on the object, while D is the dimension of the manipulation problem (either 2D or 3D). We may consider the states to be bounded, i. e.

$$\underline{\boldsymbol{x}}_j \leq \boldsymbol{x}_j \leq \overline{\boldsymbol{x}}_j, \tag{2}$$

for some lower bound $\underline{\boldsymbol{x}}$ and upper bound $\overline{\boldsymbol{x}}$. Such state constraints can perhaps represent the position limits of the workspace.

The forces are the result of shaping the underlying force field, which we represent by

$$\boldsymbol{u}_j = \boldsymbol{F}_j(\boldsymbol{x}_j, \boldsymbol{v}), \tag{3}$$

where $\boldsymbol{F}_j \colon \mathbb{R}^{2D} \times \mathbb{R}^A \to \mathbb{R}^D$ is the force model of the jth object, $\boldsymbol{v} \in \mathbb{R}^A$ is the vector of global actuator commands and A is the number of actuators. The force depends on $\boldsymbol{x}_j$ because it depends on the position (and possibly the velocity) of the manipulated object. We present a one-dimensional example of such a force model in Fig. 1, where different objects (red dots) are subject to different forces based on their positions and the excitation of actuators, which is proportional to the supplied actuator commands. The force field is formed by summing up the contributions of individual actuators. Furthermore, we consider the actuator commands bounded

$$\underline{\boldsymbol{v}} \leq \boldsymbol{v} \leq \overline{\boldsymbol{v}}. \tag{4}$$

We assume the manipulated objects are spherical, so we do not consider their orientation and moments. However, the formulation can be extended to cover nonspherical objects.

Formally, we seek for every manipulated object to obtain collision-free state trajectory $\boldsymbol{x}_j \colon [0, T] \to \mathbb{R}^{2D}$, such that $\boldsymbol{x}_j(T) \in \mathcal{X}_{j,\text{goal}}$, and force trajectory $\boldsymbol{u}_j \colon [0, T] \to \mathbb{R}^D$, so that the force model (3) can be satisfied. The set $\mathcal{X}_{j,\text{goal}}$ is the goal state region of the jth manipulated object. We further assume that this set can represented by a collection of inequalities and equalities imposed upon $\boldsymbol{x}_j$. We call the $T > 0$ the control horizon, which can be provided beforehand or not.

B. Optimization Formulation

Our approach to solving the presented problem is to formulate it as a single optimization problem. First, we introduce the concatenated states and forces,

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_M \end{bmatrix} \quad \text{and} \quad \boldsymbol{u} = \begin{bmatrix} \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_M \end{bmatrix}. \tag{5}$$

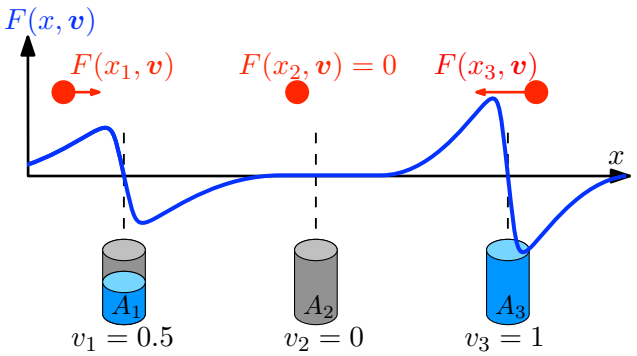


Fig. 1: Example of the force model. For simplicity, the force model is the same for all manipulated objects. Actuators (A_1, A_2, A_3) generate the physical field that exerts different forces on individual objects based on their positions (x_1, x_2, x_3) and actuator commands $\boldsymbol{v} = (v_1, v_2, v_3)$. Actuators in this example exert only an attractive force field

Second, we concatenate the state equations and the individual contributions of the force models,

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \boldsymbol{f}_1(\boldsymbol{x}_1, \boldsymbol{u}_1) \\ \vdots \\ \boldsymbol{f}_M(\boldsymbol{x}_M, \boldsymbol{u}_M) \end{bmatrix}, \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{v}) = \begin{bmatrix} \boldsymbol{F}_1(\boldsymbol{x}_1, \boldsymbol{v}) \\ \vdots \\ \boldsymbol{F}_M(\boldsymbol{x}_M, \boldsymbol{v}) \end{bmatrix}. \tag{6}$$

For now, we represent the collision avoidance by a general inequality constraint

$$\boldsymbol{C}(\boldsymbol{x}) \geq \boldsymbol{0}. \tag{7}$$

As we desire to build our problem formulation into an optimization problem, we need to define a cost to be minimized. Typical optimal control cost form is given by the functional

$$J\left(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)\right) = \phi(\boldsymbol{x}(T)) + \int_0^T L(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \boldsymbol{v}(t))\, \mathrm{d}\tau, \tag{8}$$

where function ϕ is the final state cost and weighs the state at the control horizon's end, while L is the additive cost that weighs both states and control inputs along the entire trajectory. If control horizon T is not given beforehand, the problem is then free-final time and T is considered an optimization variable. In that case, ϕ can also depend on T.

We propose using the so-called minimal control effort formulation, which typically means minimizing the squared action of actuators, e. g. minimizing the squared forces and torques acting upon a manipulator to prolong its longevity or minimizing the power consumption of a UAV to extend its flight time. In our case, the action of actuators is proportional to the actuator command $\boldsymbol{v}$, which yields the formulation

$$\phi(\boldsymbol{x}(T)) = 0, \quad L(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t)) = \boldsymbol{v}^{\mathrm{T}}(t)\boldsymbol{v}(t). \tag{9}$$

Now we got everything we need to formalize the planning

task as an optimization problem

$$\begin{aligned}
\underset{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),\boldsymbol{v}(\cdot)}{\text{minimize}} \quad & \int_0^T \boldsymbol{v}^{\mathrm{T}}(\tau)\boldsymbol{v}(\tau)\,\mathrm{d}\tau \\
\text{subject to} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t)), && \forall t \in [0,T] \\
& \boldsymbol{u}(t) = \boldsymbol{F}(\boldsymbol{x}(t),\boldsymbol{v}(t)), && \forall t \in [0,T] \\
& \underline{\boldsymbol{x}} \le \boldsymbol{x}(t) \le \overline{\boldsymbol{x}}, && \forall t \in [0,T] \\
& \boldsymbol{C}(\boldsymbol{x}(t)) \ge \boldsymbol{0}, && \forall t \in [0,T] \\
& \underline{\boldsymbol{v}} \le \boldsymbol{v}(t) \le \overline{\boldsymbol{v}}, && \forall t \in [0,T] \\
& \boldsymbol{x}(0) = \boldsymbol{x}_{\text{initial}}, \\
& \boldsymbol{x}(T) \in \mathcal{X}_{\text{goal}}.
\end{aligned} \tag{10}$$

The $\underline{\boldsymbol{x}}$ and $\overline{\boldsymbol{x}}$ in (10) are concatenated state bounds, $\boldsymbol{x}_{\text{initial}}$ is composed of the initial states of the individual objects and analogously $\mathcal{X}_{\text{goal}}$ are concatenated goal state regions.

The problem (10) is a typical continuous-time constrained optimal control problem with state $\boldsymbol{x}(\cdot)$ and control input $\boldsymbol{v}(\cdot)$. We need to apply some type of discretization in time to (10) to obtain a standard mathematical program, which can be, in turn, fed to a solver. To this end, we can employ a set of methods collectively called direct collocation [15], where in its simplest form, the sought trajectory is represented as values at an N knot points equally spaced in time. We start by discretizing the continuous-time dynamics of the manipulated objects in time. We denote the value of continuous-time variable $\boldsymbol{z}(t)$ at the time nh, where h is a timestep and $n \in \{1, 2, \ldots, N\}$, as $\boldsymbol{z}[n]$. It holds true that

$$\boldsymbol{x}[n+1] = \boldsymbol{x}[n] + \int_{nh}^{(n+1)h} \boldsymbol{f}(\boldsymbol{x}(\tau),\boldsymbol{u}(\tau))\,\mathrm{d}\tau. \tag{11}$$

We approximate the integral in (11) using the Runge-Kutta integration scheme of the 4th order (RK4), which we denote as

$$\boldsymbol{x}[n+1] = \boldsymbol{f}_{\mathrm{d}}(\boldsymbol{x}[n],\boldsymbol{u}[n]). \tag{12}$$

Moreover, we approximate the cost as the sum of the values at the knot points, i. e.

$$J = \sum_{n=1}^{N-1} \boldsymbol{v}^{\mathrm{T}}[n]\boldsymbol{v}[n]. \tag{13}$$

When provided a control horizon T, we can choose the number of knot points N, which allows us to match the timestep h of the RK4 method with the sample time of the tracking loop by letting $N = \lfloor T/h \rfloor + 1$, where $h = T_s$ is the tracking loop sample time. When T is not provided beforehand, and it is thus considered an optimization variable, the number knot of points N becomes a parameter, and the timestep h stretches or shrinks based on the value of T, as it satisfies $h = T/(N-1)$.

Combining the results above, we arrive at the planning problem in the form of a mathematical program:

$$\begin{aligned}
\underset{\boldsymbol{x}[\cdot],\boldsymbol{u}[\cdot],\boldsymbol{v}[\cdot]}{\text{minimize}} \quad & \sum_{n=1}^{N-1} \boldsymbol{v}^{\mathrm{T}}[n]\boldsymbol{v}[n] \\
\text{subject to} \quad & \boldsymbol{x}[n+1] = \boldsymbol{f}_{\mathrm{d}}(\boldsymbol{x}[n],\boldsymbol{u}[n]), && \forall n \in \mathcal{I}_{N-1}, \\
& \boldsymbol{u}[n] = \boldsymbol{F}(\boldsymbol{x}[n],\boldsymbol{v}[n]), && \forall n \in \mathcal{I}_{N-1}, \\
& \underline{\boldsymbol{x}} \le \boldsymbol{x}[n] \le \overline{\boldsymbol{x}}, && \forall n \in \mathcal{I}_{N-1}, \\
& \boldsymbol{C}(\boldsymbol{x}[n]) \ge 0, && \forall n \in \mathcal{I}_{N-1}, \\
& \boldsymbol{0} \le \boldsymbol{v}[n] \le \boldsymbol{1}, && \forall n \in \mathcal{I}_{N-1}, \\
& \boldsymbol{x}[1] = \boldsymbol{x}_{\text{initial}}, \\
& \boldsymbol{x}[N] \in \mathcal{X}_{\text{goal}}.
\end{aligned} \tag{14}$$

In general, the problem (14) is a so-called *nonlinear program* (NLP). The gradient-based solvers for these types of problems provide only locally optimal solutions as the problems are usually non-convex. Nevertheless, we will call the possibly suboptimal solutions to (14) optimal and denote with an asterisk, e. g. $\boldsymbol{x}^*, \boldsymbol{u}^*$.

C. Solving the NLP

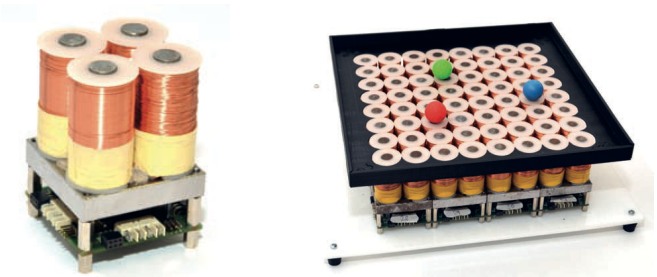
Standard tools, such as the framework for algorithmic differentiation CasADi [16] together with IPOPT solver, can be used to model and solve the presented NLP. More importantly, the formulated NLP is generally non-convex. The quality of the solution to these kinds of problems heavily depends on the quality of the provided initial guess. At the heart of the problem, we are facing a manipulation task. Therefore a collision-free path, i. e. a set of collision-free configurations of the individual objects between the initial and goal positions, could provide such an initial guess. This path can be planned using a sampling-based planner such as RRT [17]. Therefore, we propose the following simple initialization scheme:

Algorithm 1: NLP Initialization Scheme

Input: $\boldsymbol{x}_{\text{initial}}, \mathcal{X}_{\text{goal}}, N$
Output: Initial guess of $\boldsymbol{x}[\cdot], \boldsymbol{u}[\cdot], \boldsymbol{v}[\cdot]$

```
Plan collision-free path q_j : [0,1] → R^D from x_{j,initial}
  to X_{j,goal} for each object index j ∈ {1, 2, ..., M};
for n ← 1 to N do
    for j ← 1 to M do
        Initialize x_j[n] using the value of
          q_j((n − 1)/(N − 1));
        if n < N then
            u_j[n] ← 0;
    if n < N then
        v[n] ← 0;
```

The paths are collision-free in the sense that they satisfy constraint (7). We assume the planned path is parametrized on the interval $[0, 1]$, and this can be, for example, done by fitting the found waypoints using smoothing splines. To simplify the path planning, we do not consider any dynamic



(a) Single module (b) Complete platform

Fig. 2: Magman experimental platform

constraints. We initialize both the forces and actuator commands to zero and leave this part of the planning for the optimization itself.

III. Experimental Setup

A. The Experimental Platform Magman

Our designs are experimentally evaluated on distributed magnetic manipulation platform dubbed Magman. The platform is composed of modules (Fig. 2a), each containing four coils in a square array and electronics that allow for command-based control of a current flowing through the coils. Our configuration comprises 4×4 such modules. Therefore 64 coils in total (Fig. 2b), but theoretically arbitrary grid-based configuration could be built.

The platform can simultaneously and independently manipulate multiple steel balls based on desired forces using feedback control. The feedback is made up of the positions of the individual balls, which are provided using a camera overlooking the platform and an image processing algorithm that detects the balls in the picture. The control system of the platform is decomposable into an inner and outer loop. The *inner loop* computes appropriate coil currents from ball positions and desired forces. Solving a *quadratic program* (QP) is used to find the coil currents for desired forces. This QP is decomposed and solved in a distributed matter, as presented in [18]. The purpose of the *outer loop* is to ensure the tracking of a given trajectory. It computes correcting forces for deviation from the nominal trajectory. These forces are then supplied to the inner loop. The entire control loop, including the image processing algorithm, runs in real-time on NVIDIA Jetson AGX Xavier.

We will now briefly present the mathematical model of Magman (more details in [19]) and how the individual parts of the model correspond to the previously introduced general formulation. We start with the dynamics of the manipulated objects, that is, steel balls. The steel balls traverse the platform by rolling. We model these planar dynamics as that of a frictionless point mass, resulting in the following state-space model corresponding to (1):

$$\boldsymbol{f}_j(\boldsymbol{x}_j, \boldsymbol{u}_j) = \boldsymbol{A}\boldsymbol{x}_j + \boldsymbol{B}_j\boldsymbol{u}_j, \tag{15}$$

where $\boldsymbol{x}_j = \begin{bmatrix} x_j & \dot{x}_j & y_j & \dot{y}_j \end{bmatrix}^{\mathrm{T}}$ is the state, which is composed of ball's center position (x_j, y_j) and the ball's center velocity $(\dot{x}_j, \dot{y}_j)$, while $\boldsymbol{u}_j = \begin{bmatrix} F_{j,x} & F_{j,y} \end{bmatrix}^{\mathrm{T}}$ is the force acting on the ball. Moreover, the matrices in (15) are given as

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{B}_j = \begin{bmatrix} 0 & 0 \\ 1/m_{j,\mathrm{e}} & 0 \\ 0 & 0 \\ 0 & 1/m_{j,\mathrm{e}} \end{bmatrix}, \tag{16}$$

where $m_{j,\mathrm{e}} = 7/5m_j$ is the ball's effective mass, which takes into account the ball's inertia and the ball's mass m_j.

Our platform is rectangular. Thus we constrain the ball state by its bounds. Furthermore, we do not consider any bounds on the ball's velocity. Therefore the bounds in (2) result in

$$\underline{\boldsymbol{x}}_j = \begin{bmatrix} \underline{x} & -\infty & \underline{y} & -\infty \end{bmatrix}^{\mathrm{T}}, \quad \overline{\boldsymbol{x}}_j = \begin{bmatrix} \overline{x} & \infty & \overline{y} & \infty \end{bmatrix}^{\mathrm{T}}, \tag{17}$$

where $(\underline{x}, \underline{y})$ is the position of the platform's lower left corner and $(\overline{x}, \overline{y})$ of the platform's upper right corner. Moreover, the inequalities in equation (2) in the case of $\pm\infty$ must be interpreted as strict, which we, for simplicity of the notation, leave out.

The individual coils are modeled as magnetic monopoles and there is A of them in total. The force exerted by the platform on a single ball is given as the sum of the contributions of individual coils

$$F_a = \sum_{k=1}^{A} G_{a,k}(x, y)v_k, \quad a \in \{x, y\}. \tag{18}$$

For simplicity of the notation, we now omit the ball index j. The function $G_{k,a}(x, y)$ in (18) is derived from the field of the magnetic monopole, and it dictates how the force scales with the position relative to the coil. It equals

$$G_{a,k}(x, y) = \frac{ca}{\left[(x - X_k)^2 + (y - Y_k)^2 + b^2\right]^3}, \quad a \in \{x, y\}, \tag{19}$$

where (X_k, Y_k) is the position of the kth coil and constants c, b are dependent on the parameters of the used steel ball. Moreover, the factor v_k in (18) captures how the force scales with current flowing through the kth coil. We refer to $\boldsymbol{v} = \begin{bmatrix} v_1, & \ldots, & v_A \end{bmatrix}^{\mathrm{T}}$ as the vector of coil currents, and it plays the role of the actuator command introduced in (3). The force $\boldsymbol{F}_j(\boldsymbol{x}_j, \boldsymbol{v})$ exerted by the magnetic field on the jth ball in vector form can be expressed as

$$\boldsymbol{F}_j(\boldsymbol{x}_j, \boldsymbol{v}) = \boldsymbol{G}_j^{\mathrm{T}}(\boldsymbol{x}_j)\boldsymbol{v}, \tag{20}$$

where

$$\boldsymbol{G}_j(\boldsymbol{x}_j) = \begin{bmatrix} G_{x,1}(x_j, y_j) & \ldots & G_{x,A}(x_j, y_j) \\ G_{y,1}(x_j, y_j) & \ldots & G_{y,A}(x_j, y_j) \end{bmatrix}^{\mathrm{T}}. \tag{21}$$

Furthermore, the coil currents are limited by the module construction as follows

$$\boldsymbol{0} \leq \boldsymbol{v} \leq \boldsymbol{1}, \tag{22}$$

this corresponds to acuator command bounds from (4).

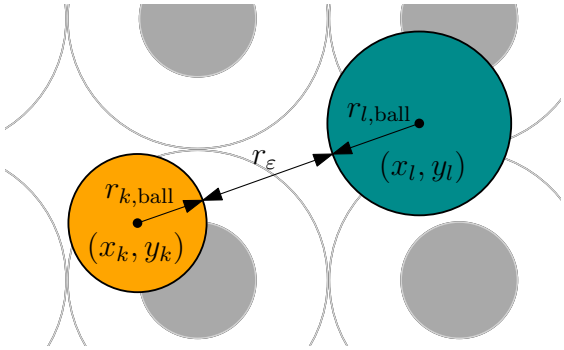


Fig. 3: Illustration of collision avoidance constraint. Centers of individual balls must be at least 2 radii away ($r_{k,\text{ball}} + r_{l,\text{ball}}$) to prevent collisions. Furthermore, we increase the required distance by the parameter $r_\varepsilon > 0$ to improve robustness

We prevent the collisions between individual balls by imposing the constraint

$$C_{kl} \geq 0, \ \forall k, l \in \mathcal{P}_M, \tag{23}$$

where

$$C_{kl} = (x_k - x_l)^2 + (y_k - y_l)^2 - (r_{k,\text{ball}} + r_{l,\text{ball}} + r_\varepsilon)^2, \tag{24}$$

is the difference the between the squared mutual distance of the kth and the lth balls and the square sum of their radii $r_{\text{ball},k}$, $r_{\text{ball},l}$ together with safety distance r_ε, while

$$\mathcal{P}_M = \{(k, l) \in \mathcal{I}_M \times \mathcal{I}_M \mid k > l\}. \tag{25}$$

is the set of all ball pair indexes. We illustrate the situation in Fig. 3. The constraint must be satisfied for each pair of balls on the platform, and we present the set of ball indexes for which we enforce the constraint in (25). It also holds that

$$|\mathcal{P}_M| = \frac{M(M-1)}{2} \sim \mathcal{O}(M^2). \tag{26}$$

Therefore, the number of collision-avoidance constraints scales quadratically with the number of manipulated balls.

B. Trajectory Tracking

Now we describe tracking the planned trajectories. The feedback has the form of the individual ball's positions and the control system needs to approximate the velocities from measured positions to get the entire state. Velocity estimation is generated by a filtered derivative system, given by the transfer function

$$H(z) = \frac{z - 1}{Tz + T_s - T}, \tag{27}$$

where T is time filter time constant and $T_s = 0.01\,\text{s}$ is the control loop sample time. We then estimate the state $\hat{\boldsymbol{x}}_j(z)$ of the jth ball from its measured position $(x_j(z), y_j(z))$ as

$$\hat{\boldsymbol{x}}_j(z) = \begin{bmatrix} x_j(z) & H(z)x_j(z) & y_j(z) & H(z)y_j(z) \end{bmatrix}^\mathrm{T}. \tag{28}$$

For trajectory tracking itself, we designed an LQR controller using the empirically obtained weight matrices:

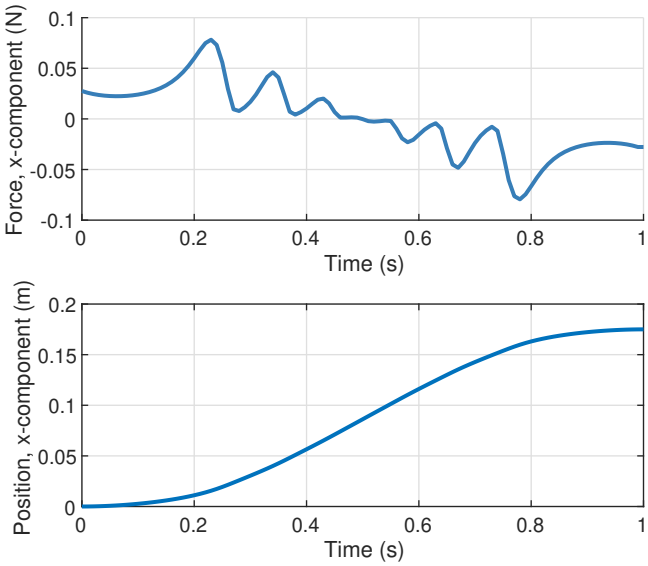


Fig. 4: Planned single ball motion

$\boldsymbol{Q} = \text{diag}\left(\begin{bmatrix} 2 & 3 & 2 & 3 \end{bmatrix}^\mathrm{T}\right)$ and $\boldsymbol{R} = \text{diag}\left(\begin{bmatrix} 2 & 2 \end{bmatrix}^\mathrm{T}\right)$. The resulting control law is

$$\boldsymbol{u}_j[n] = \underbrace{\boldsymbol{u}_j^*[n]}_{\text{Feedforward}} - \underbrace{\boldsymbol{K}(\hat{\boldsymbol{x}}_j[n] - \boldsymbol{x}_j^*[n])}_{\text{Feedback}}, \tag{29}$$

where $\boldsymbol{K}$ is the LQR Full-State-Feedback matrix, $\boldsymbol{x}_j^*$ and $\boldsymbol{u}_j^*$ are the planned state and force trajectories, while $\boldsymbol{u}_j$ is the outputted force, passed to the inner control loop, which computes the coil currents by solving a constrained least-squares problem. Notice that the control law (29) comprises the feedforward force trajectory and feedback correcting force.

IV. Experiments

Now we experimentally showcase the trajectories planned we planned using formulation (14) for the Magman platform.

A. Single Ball Experiment

For ease of demonstration, we chose a simple goal for our first experiment, i. e. moving a single 10 mm ball across the platform while stopping at the end. We chose the time length of the trajectory to be 1 s. Fig. 4 shows the planned trajectory and force. Notice that the planned force looks akin to bang-bang control, i. e. switching between two extremal values. However, instead of two extremal values, the ball is actuated by short impulses of force seen as peaks in Fig. 4. The positions where these impulses are exerted are close to the positions where maximal, respectively, minimal force values are attainable. Such behavior is expected, as, in those positions, the most force for the least amount of current can be achieved. Moreover, Fig. 5 shows the evaluation of the tracking of the planned trajectory on the actual platform.

B. Rearrangement Experiment

In the next experiment, we attempted to simultaneously rearrange 10 mm steel balls positioned at the platform's border.

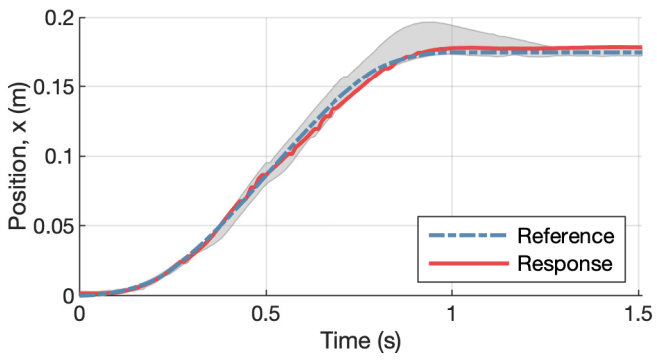


Fig. 5: Tracking of the single ball trajectory on the actual platform. The red line portrays one particular experiment, and the gray envelope shows variance across multiple experiments

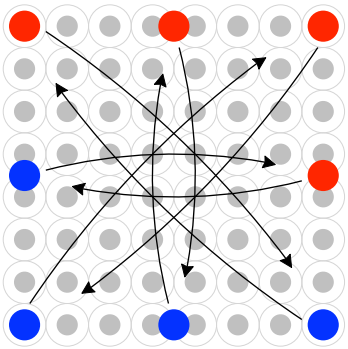


Fig. 6: Visualization of the rearrangement experiment. We want to simultaneously move all balls to the positions of the corresponding opposite side balls. This all while avoiding collisions between individual balls

The visualization of the experiment is available in Fig. 6. The experiment was motivated by the problems of planar object rearrangement. Here, we can present the inherent benefit of distributed manipulation by shaping a physical field, i. e. parallel manipulation. Suppose the aim is to rearrange multiple objects simultaneously. In that case, we only need to handle possible collisions between individual objects as opposed to using multiple manipulators, where one must also consider the possible collisions between manipulators.

We chose the control horizon of the experiment to be $T = 2\,\mathrm{s}$. The balls have zero velocity in their initial positions and we desire to stop them at their final positions. Therefore, the final state is also composed of zero velocities. The resulting planned trajectories and the executed trajectories from the experiment are presented in Fig. 7. We initialized the optimization using the scheme in Alg. 1 by a collision-free path planned using RRT. The RRT initialized trajectory optimization took $235.4\,\mathrm{s}$ to compute on Intel Core i5-1038NG7 CPU and resulted in the cost function value of 16.55. We have also attempted to initialize the trajectory optimization by path created by linear interpolation between the initial and goal positions. This path resulted in shorter optimization computational time of $190.64\,\mathrm{s}$ but a significantly

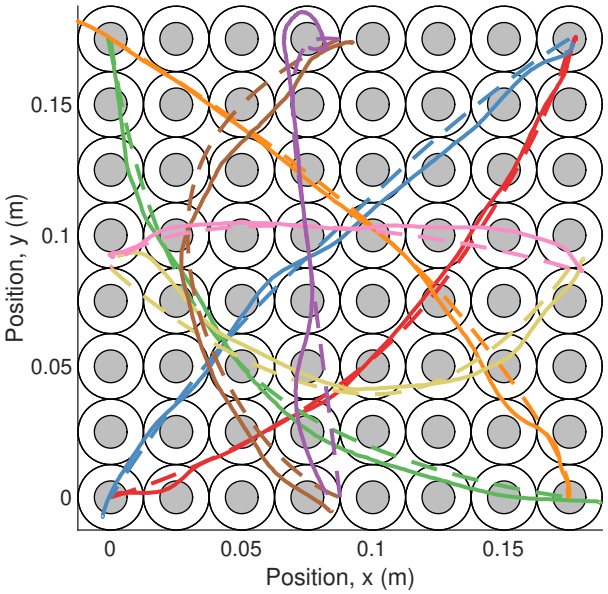


Fig. 7: Planned and executed trajectories from the rearrangement experiment. Solid lines represented the executed trajectories, while dashed the planned ones. Each color represents one ball. Video available at https://youtu.be/QWUwkquIDQM

higher cost value of 20.80.

The resulting mean of the trajectory tracking NRMSE (RMSE normalized by the platform dimension) over all balls is $1.49\,\%$. The increase in tracking error of the trajectories in the case of multiple balls can be caused by the inaccuracy of the mathematical model; more precisely, the model does not incorporate that the presence of the steel balls influences the underlying magnetic field, balls' magnetic hysteresis, and damping caused both by the friction and the normal component of the magnetic force. Based on these inacuracies, we further believe that reducing the order of the integration scheme by e. g. using forward Euler instead of RK4 could speed up the trajectory computation without much affecting the tracking error.

V. Conclusion

We presented a general formulation of trajectory optimization for distributed manipulation by shaping a physical field through an array of actuators. We applied the general formulation to the problem of independently manipulating several steel balls using our experimental platform for distributed magnetic manipulation and demonstrated the functionality through experiments aimed at planar rearrangement (repositioning) of several steel balls without the balls colliding. Although demonstrated with a particular physical platform, the presented approach is sufficiently general and can be for other physical fields shaped in real time by actuator arrays.

References

[1] A. Shirazi, J. Ceberio, and J. A. Lozano, "Spacecraft trajectory optimization: A review of models, objectives, approaches and solutions," *Progress in Aerospace Sciences*, vol. 102, pp. 76–98, Oct. 2018.

[2] J. Tordesillas and J. P. How, "MADER: Trajectory Planner in Multi-agent and Dynamic Environments," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 463–476, Feb. 2022.
[3] A. Herzog, N. Rotella, S. Schaal, and L. Righetti, "Trajectory generation for multi-contact momentum control," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov. 2015, pp. 874–880.
[4] A. Malik, T. Henderson, and R. J. Prazenica, "Trajectory Generation for a Multibody Robotic System using the Product of Exponentials Formulation," in *AIAA Scitech 2021 Forum*, ser. AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, Jan. 2021.
[5] K.-F. Bohringer, B. Donald, R. Mihailovich, and N. MacDonald, "A theory of manipulation and control for microfabricated actuator arrays," in *Proceedings IEEE Micro Electro Mechanical Systems An Investigation of Micro Structures, Sensors, Actuators, Machines and Robotic Systems*, Jan. 1994, pp. 102–107.
[6] J. Zemánek, T. Michálek, and Z. Hurák, "Phase-shift feedback control for dielectrophoretic micromanipulation," *Lab on a Chip*, vol. 18, no. 12, pp. 1793–1801, Jun. 2018.
[7] J. Matou, A. Kollarík, M. Gurtner, T. Michálek, and Z. Hurák, "Optimization-based Feedback Manipulation Through an Array of Ultrasonic Transducers," *IFAC-PapersOnLine*, vol. 52, no. 15, pp. 483–488, Jan. 2019.
[8] S. Chaudhary and B. Shapiro, "Arbitrary steering of multiple particles independently in an electro-osmotically driven microfluidic system," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 4, pp. 669–680, Jul. 2006.
[9] K. Yu, J. Yi, and J. Shan, "Simultaneous Multiple-Nanowire Motion Control, Planning, and Manipulation Under Electric Fields in Fluid Suspension," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 80–91, Jan. 2018.
[10] M. P. Kummer, J. J. Abbott, B. E. Kratochvil, R. Borer, A. Sengul, and B. J. Nelson, "OctoMag: An Electromagnetic System for 5-DOF Wireless Micromanipulation," *IEEE Transactions on Robotics*, vol. 26, no. 6, pp. 1006–1017, Dec. 2010.
[11] E. Diller, J. Giltinan, and M. Sitti, "Independent control of multiple magnetic microrobots in three dimensions," *The International Journal of Robotics Research*, vol. 32, no. 5, pp. 614–631, Apr. 2013.
[12] J. E. Luntz, W. Messner, and H. Choset, "Distributed Manipulation Using Discrete Actuator Arrays," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 553–583, Jul. 2001.
[13] X. Li and K. Yu, "Informed Sampling-Based Motion Planning for Manipulating Multiple Micro Agents using Global External Fields," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, Aug. 2020, pp. 888–893.
[14] S. D. Han, N. M. Stiffler, A. Krontiris, K. E. Bekris, and J. Yu, "Complexity Results and Fast Methods for Optimal Tabletop Rearrangement with Overhand Grasps," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1775–1795, Dec. 2018.
[15] C. Hargraves and S. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of Guidance, Control, and Dynamics*, vol. 10, no. 4, pp. 338–342, 1987.
[16] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, Mar. 2019.
[17] S. LaValle, "Rapidly-exploring random trees : a new tool for path planning," *Technical report 98-11*, 1998.
[18] M. Gutner, J. Zemánek, and Z. Hurák, "Alternating direction method of multipliers-based distributed control for distributed manipulation by shaping physical force fields," *The International Journal of Robotics Research*, Feb. 2023.
[19] J. Zemánek, "Distributed manipulation by controlling force fields through arrays of actuators," Doctoral thesis, Czech Technical University in Prague, 2018.