Direct Collocation for Dynamic Behaviors with Nonprehensile Contacts: Application to Flipping Burgers

Shishir Kolathaya[1], William Guffey[2], Ryan W. Sinnet[2], and Aaron D. Ames[3]

***Abstract*—To realize robotic systems in real-world settings, e.g., in restaurants, it will be necessary to achieve dynamic manipulation of non-trivial objects. In this context, this paper discusses methodologies used to realize trajectories in a robotic arm platform; specifically, applied to flipping burgers as an example of nonprehensile object manipulation. Flipping of burgers involves a series of tasks—going to the burger location, scooping, picking up and flipping. Since the goal is to obtain these trajectories in a reasonably fast manner, we employ direct collocation based multi-segmented trajectory optimization. We will first describe the problem setup, and then describe the constraints, decision variables employed, and then, to conclude, we will demonstrate these behaviors in a 6-DOF robot experimentally.**

***Index Terms*—Trajectory Optimization, Dynamic Motion Planning, Nonprehensile Contacts, Manipulation Planning.**

I. Introduction

A standard medium sized burger patty requires 90 seconds to have its one side cooked completely. The burger is then flipped over and cooked for 60 more seconds. This process of picking and flipping a burger by using a standard size spatula appears to be a straightforward task for us humans, but there are a series of challenges that need to be addressed if we wish to translate this in a robotic arm platform. Some of the major challenges are given as follows:

- The trajectory needs to be **optimal**. For example, for a simple task such as transportation of a burger, the robot has to take minimum time, torque, power or energy.
- The trajectory is **multi-segmented**, i.e., given the burger location, the robot has to a) go to the location b) slide underneath the burger that is hot, soft and greasy c) pick the burger off the grill while maintaining friction constraints, and finally d) either flip the burger so that the top side is facing the hot grill, or place the burger on a side table.
- The trajectory has to satisfy **dynamic constraints**, i.e., no-slipping conditions of the burger with the spatula.
- The trajectory has to incorporate **space limits**, i.e., operate within the boundaries of the walls and grill.
- Optimization needs to be **fast**, typically less than 20 seconds per trajectory. Sometimes up to $4-5$ burgers are placed on the grill at a time, and the optimizer should be able to generate trajectories for all these 5 burgers in 90 seconds.

Fig. 1: Figure showing a typical burger cooking apparatus. The grill is where the burger is cooked, and the table is where the freshly cooked burger is placed. We are using this setup for deploying the techniques described in the paper as it cooks burgers in real-world restaurants. A video demonstration is shown in this footnote[1].

The problem of flipping burgers can be classified as a *nonprehensile object manipulation* problem [1]. Existing literature has established techniques for prehensile contacts [2], like pick-and-place [3], surgery [4], and grasping [5], but very few techniques for nonprehensile contacts (see [6]). Nonprehensile contacts do not involve grasping, but use friction, gravity and other external forces for manipulation. The goal is to

Manuscript received: February, 24, 2018; Revised May, 23, 2018; Accepted June, 20, 2018.

This paper was recommended for publication by Editor Han Ding upon evaluation of the Associate Editor and Reviewers' comments. This work is supported by NSF award #: 1526519 and Miso Robotics.

[1]Shishir Kolathaya is an INSPIRE Faculty Fellow at the Indian Institute of Science, Bengaluru, India. He did this work as a Postdoctoral Scholar in the Department of Mechanical and Civil Engineering at the California Institute of Technology, Pasadena, California, USA shishirk@iisc.ac.in.

[2]William Guffey and Ryan. W. Sinnet (CTO/Co-Founder) are with Miso Robotics, Pasadena, California, USA {wguffey,rsinnet}@misorobotics.com.

[3]Aaron D. Ames is Bren Professor of Mechanical and Civil Engineering at the California Institute of Technology, Pasadena, California ames@caltech.edu.

Digital Object Identifier (DOI): see top of this page.

[1]https://vimeo.com/234073915. Here the flipping is achieved by obtaining an optimal end-effector trajectory and then using inverse kinematics to obtain the desired joint angle trajectories for the robot.

manipulate burgers with contact only on one side, while satisfying friction constraints, and also operate within some space limits (task space limits). The problems associated with nonprehensile contacts are well documented in [6], where the object has multiple degrees of freedom, and the number of inputs are limited. Sequential quadratic programming based solutions were proposed to obtain optimal trajectories in [1], hierarchical methods were proposed to handle multi-contact manipulation in [7]. Our focus in this paper is from an application viewpoint, and specifically with obtaining optimal trajectories for a 6-DOF robot for nonprehensile frictional contacts in a reasonably fast manner.

Since the main objective of this paper is flipping burgers, we studied and implemented the methods developed in [8] in detail, which demonstrated flipping of pancakes by obtaining optimal trajectories for the pose (SE(3)) of the end-effector. We also made an extension to this method by formulating this into a nonlinear program (NLP)[1]. This type of end-effector space optimization, even though fast (< 1 second), does not necessarily guarantee feasibility in a non-trivial arm configuration (like a 6-DOF arm with rotary joints Fig. 1). The inverse kinematics solver, which computes the desired joint angles given the desired end-effector pose, fails frequently for a sizeable number of test locations on the grill. This motivates the approach taken in this paper.

Main contribution. The main contribution of this paper is the generation of optimal joint-space trajectories via a well known technique called *direct collocation*. We will be using the toolkit called FROST [9], which was successfully used to realize walking and running behaviors in bipedal robots [10], [11]. The goal is to obtain joint angle trajectories to achieve two main tasks in the robot: 1. *Translate burger behavior* and 2. *Pickup and flip burger behavior*. *Translate burger behavior* involves moving the Cartesian position of the spatula (end-effector) from one point to another while maintaining friction constraints to prevent the burger from slipping. Similarly, for *pickup and flip burger behavior*, the robot has to pick the burger using its spatula and flip in such a way that the under side of burger faces up after the completion of flip.

Related work on direct collocation. Direct collocation based trajectory optimization [12] has been successfully realized in a wide variety of robotic systems like bipeds [11], [13]. Similar in flavor to our problem statement is the throwing of balls [14], which not only focuses on obtaining optimal trajectories but also on the shape of the end-effector. Our problem statement requires searching for multi-segmented trajectories for a 6-DOF robot. [13] also used direct methods along with linear complementarity constraints in order to handle multiple types of contact conditions. The key difference with our methods is the Lagrangian formulation of the dynamics and the use of *defect* variables that resulted in faster optimization (see [11]).

Section II consists of the robot model, Section III provides a brief review on direct collocation. Section IV consists of a detailed description of the constraints utilized to realize various types of behaviors in the 6-DOF robot arm. Finally, Section V describes the experimental results, and gives concluding remarks.

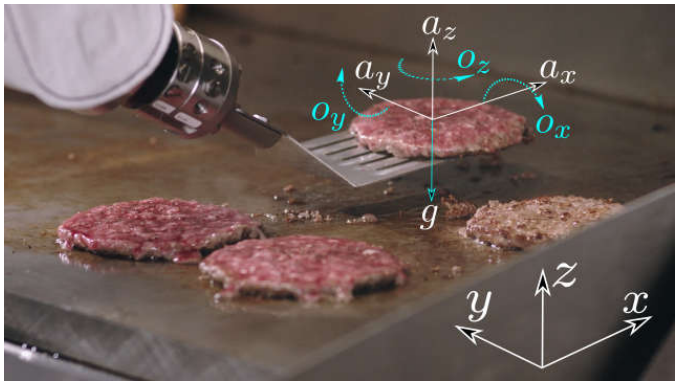

Fig. 2: Figure showing the spatula holding a freshly cooked burger. Accelerations and orientations along x,y,z directions are shown.

II. Model

This section describes the basic model of a typical burger cooking platform. Fig. 1 shows a robot along with the cooking setup consisting of a grill and a table. The robot considered is of the form given by Fig. 1 that allows the end-effector to be fitted with a spatula of appropriate size (based upon the diameter and weight of the burger). See Fig. 2 for an example spatula that is holding a freshly cooked burger. We will briefly describe the robot model first and then the remaining components, namely the spatula and the burger.

A. Robot model

We consider a n-DOF robot manipulator ($n = 6$), and the corresponding configuration space $\mathbb{Q}$. The configuration $q = (q_1, q_2, q_3, q_4, q_5, q_6) \in \mathbb{Q}$ consists of n joint angles. Therefore, we denote the state $x := (q, \dot{q}) \in T\mathbb{Q}$. We will denote the torque input u, which is of dimension m ($= 6$). Given the states, and the inputs, the Euler-Lagrangian dynamics are given by:

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = Bu, \tag{1}$$

where $D(q) \in \mathbb{R}^{n\times n}$ is the positive definite inertia matrix, $C(q, \dot{q}) \in \mathbb{R}^{n\times n}$ is the Coriolis-centrifugal matrix, and $G(q) \in \mathbb{R}^n$ is the gravity vector, $B \in \mathbb{R}^{n\times m}$ is the one-on-one mapping of the torques to the joints. Equation (1) can be represented in state space form in the following fashion:

$$\dot{x} = f(x) + g(x)u, \tag{2}$$

where f and g can be appropriately obtained [15, eqn. (13)].

B. End-effector or spatula model

Let p_x, p_y, p_z be the positions, v_x, v_y, v_z be the velocities, and a_x, a_y, a_z be the accelerations of the spatula center along x, y, z directions respectively. We can put these variables in vector form as

$$p = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}, \quad v = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}, \quad a = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}. \tag{3}$$

The orientation of the spatula is also similarly defined: $o = [o_x, o_y, o_z]^T$. Note that the positions are functions of the configuration q, velocities are functions of the state $(q, \dot{q})$, and the accelerations are functions of the state and angular accelerations $(q, \dot{q}, \ddot{q})$. Having defined the positions, velocities, accelerations and orientations of the robot end-effector, we have the first set of constraints imposed due to hardware and workspace limitations of the robot.

***Constraint** 1: Position p, velocity v, acceleration a, and also the orientation o have lower and upper bounds i.e., there exist $\delta_{\square,\min}$, $\delta_{\square,\max} > 0$, where $\square \in \{p, v, a, o\}$, such that*

$$\begin{aligned} \delta_{p,\min} < p < \delta_{p,\max}, \qquad & \delta_{v,\min} < v < \delta_{v,\max} \\ \delta_{a,\min} < a < \delta_{a,\max}, \qquad & \delta_{o,\min} < o < \delta_{o,\max}. \end{aligned} \tag{4}$$

The center of the burger may not necessarily be aligned with the center of the spatula. This is overcome by imposing friction constraints at multiple points on the spatula. For the purposes of analysis, we will assume that the centers are perfectly aligned. We can now describe the burger model and also the interactive forces between the burger and the spatula.

C. Burger model

For the burger model, we consider a disc of constant diameter d and of rigid mass m kg. The mass distribution is assumed to be uniform. It is also further assumed that the co-efficient of friction between the spatula and the burger is not lower than μ. Let l_w be the width and l_l be the length of the spatula. Given the robot model (1), and the associated position, velocity, accelerations of the spatula center, we have the following description of the normal force normalized by the mass of the burger (see Fig. 3):

$$\frac{F_n}{m}(a, o) = gc(o_x)c(o_y) + a_z c(o_x)c(o_y) - a_y s(o_x)s(o_y),$$

where $s(o_x) := \sin(o_x)$, $c(o_x) := \cos(o_x)$, and g is acceleration due to gravity. Given the normalized normal force $\frac{F_n}{m}$, the goal is to restrict the tangential forces F_x, F_y acting along the horizontal directions of the robot. These are given by

$$\begin{aligned} \frac{F_x}{m}(a, o) &= -a_y c(o_x)c(o_y) - gs(o_x)s(o_y) - a_z s(o_x)s(o_y) \\ \frac{F_y}{m}(a, o) &= -a_x c(o_x)c(o_y) + gs(o_x)s(o_y) + a_z s(o_x)s(o_y). \end{aligned}$$

In order to prevent slipping of the burger, the normal force acting on the burger must be positive, and must be at least as much as μ^{-1} times the tangential forces, i.e.,

$$\begin{aligned} F_n(a, o) &\geq 0 \\ |F_x(a, o)| &\leq \mu F_n(a, o) \\ |F_y(a, o)| &\leq \mu F_n(a, o). \end{aligned} \tag{5}$$

Having described the model, we will now describe the behaviors to be realized in the robot.

D. Behaviors

We will provide a brief description of the two behaviors *translate burger behavior* and *pickup and flip burger behavior* below.

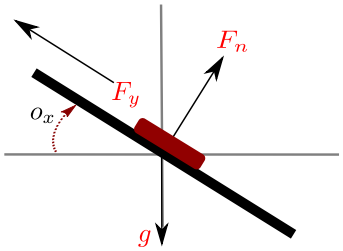

Fig. 3: Figure indicates the forces acting on the burger. The tangential force must be restricted by the friction forces.

1) Translate burger behavior: This behavior is required if the user has to transport the burger from point A to point B. This task is mainly used to move a freshly cooked burger from the grill to the table. Slipping constraints need to be satisfied in order to execute this task.

2) Pickup and flip burger behavior: This behavior can be further divided into five subbehaviors:

- **Go to burger location**: Given the burger location in x,y,z coordinates, the spatula is moved from its home position to *scoop-ready* position. In this subbehavior, the spatula is tilted in such a way that its tip is in contact with the grill, waiting for the next action.
- **Scoop**: The tip of the spatula is made to slide on the grill in such a way that the spatula slides under the burger.
- **Pickup**: The spatula is moved upwards along with the burger. This subbehavior is required in order to prepare for the flipping behavior.
- **Flip**: The spatula is turned by almost 180° in such a way that the burger lands on its other side on the grill. The take-off angular velocity from the spatula is crucial for a perfect landing of the burger.
- **Turn back**: Spatula is rotated back to 0°, waiting for a new task request.

See Fig. 4 for the pictorial representation of each of the subbehaviors of *pickup and flip burger behavior*. These behaviors are used as templates for the creation of optimal trajectories via trajectory optimization [11]. This is explained more in the next section.

III. Trajectory Optimization

Trajectory optimization involves designing an optimal trajectory via some measure of performance while satisfying a set of constraints. For the robotic system considered in this paper, the measure of performance can be either power consumption, torque inputs or even the time duration of the trajectory. The set of constraints includes an array of equalities and inequalities that includes limits, terminal conditions, and also friction conditions.

The problem of trajectory optimization is typically solved via transcription [16]. As mentioned previously, the end-effector space optimization yields feasible solutions reasonably fast, but with a risk: the inverse kinematics solver needed to convert back to joint-space could be infeasible. We will instead employ the joint-space optimization technique that will yield optimal joint angle, velocity and control trajectories, $x(t)$,

(a) Starting position of the robot. Burger is located on the grill.

(b) The spatula goes to the *scoop-ready* position.

(c) Scooping action is executed.

(d) With the burger on top, the spatula is lifted.

(e) Flipping action is executed.

(f) After flipping, spatula is rotated back, waiting for its next task.

Fig. 4: Figures showing each of the subbehaviors throughout the course of the flipping trajectory.

$u(t)$, as functions of time. This paper utilizes the following general optimization problem:

$$\begin{aligned}(x^*(t), u^*(t)) = \underset{x,u}{\text{argmin}} \quad & \text{Cost}(x,u) := \int_0^T \Phi(x(t), u(t))dt \\ \text{s.t.} \quad & \dot{x}(t) = f(x(t)) + g(x(t))u(t) \\ & \text{Other constraints},\end{aligned}$$

where Cost is the objective function that is desired to be minimized subject to a list of constraints. By a slight abuse of notations, here we are searching for x, u in the function space: $x : \mathbb{R}_{\geq 0} \to T\mathbb{Q}, u : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$. The function Φ is the cost at each time instant t. The process of transcription is applied to this optimization problem to obtain an NLP formulation.

Essentially, we discretize based on evenly spaced time t_i, where $i = 0, 1, 2...N$ is defined as the grid index. Let $x_i = (q_i, \dot{q}_i)$ and $\dot{x}_i$ be the state and its derivative at node i. Similarly let u_i be the control law that is applied at t_i. If we choose to minimize torque, we have the following objective function[2]:

$$\text{Cost}(\mathrm{u}_0, \mathrm{u}_1, \dots) := \sum_{\mathrm{i}=0}^{\mathrm{N}} \mathrm{u}_\mathrm{i}^2, \tag{6}$$

along with the following dynamics constraint:

[2]Note that the experimental robot platform uses its own PD based position control laws for tracking. After obtaining the optimal joint angle trajectories, these are seeded as desired positions and velocities to the robot. Torque minimization might seem counter-intuitive, but this procedure results in lower differences between a model based and a PD based control law on the trajectory. For example, like in [17], linear feedback laws yield lower ultimate bounds for trajectories that require lower torque inputs.

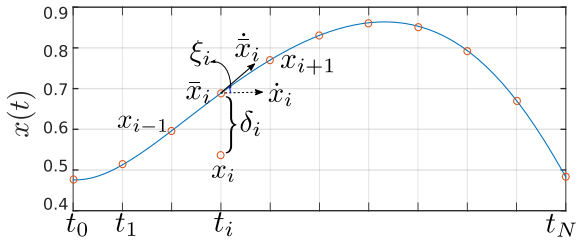

Fig. 5: Figure showing the defect constraints. δ_i corresponds to (8) and ξ_i corresponds to (9). We chose the number of nodes to be 21, $N = 20$.

***Constraint** 2: For each t_i (each node), we have the following dynamics based constraint, written as a function of states and accelerations:*

$$D(q_i)\ddot{q}_i + C(q_i, \dot{q}_i)\dot{q}_i + G(q_i) = Bu_i, \tag{7}$$

which is used as an implicit constraint in the optimization scheme. This type of constraint is preferred over (2), since the evaluation of f,g involves the computation of D^{-1}.

***Constraint** 3: We use the following collocation scheme:*

$$\bar{x}_i = \frac{x_{i-1} + x_{i+1}}{2} + \triangle t_i \frac{\dot{x}_{i-1} - \dot{x}_{i+1}}{8}$$
$$x_{i+1} - x_{i-1} - \frac{1}{6}\triangle t_i(\dot{x}_{i-1} + 4\dot{\bar{x}}_i + \dot{x}_{i+1}) = 0$$
$$\delta_i = x_i - \bar{x}_i = 0 \tag{8}$$
$$\xi_i = \dot{x}_i - \dot{\bar{x}}_i = 0, \tag{9}$$

where $\triangle t_i = t_{i+1} - t_{i-1}$. δ_i, ξ_i are the defect variables (see Fig. 5).

In summary, the nonlinear dynamics are treated as an equality constraint with the use of implicit Runge-Kutta methods and defect variables [12]. Given the objective function (6) and the Constraints 2-3, the general optimization can be transformed to the following direct collocation based optimization problem:

$$\mathcal{Z}^* = \underset{\mathcal{Z}}{\text{argmin}}\ \mathrm{J}(\mathcal{Z}) \tag{10}$$
$$\text{s.t. } \mathcal{Z}^{\min} \leq \mathcal{Z} \leq \mathcal{Z}^{\max} \tag{11}$$
$$\mathbf{C}^{\min} \leq \mathbf{C}(\mathcal{Z}) \leq \mathbf{C}^{\max}, \tag{12}$$

where $\mathscr{Z} = \{(q_i, \dot{q}_i, \ddot{q}_i, u_i) : 0 \leq i \leq N\}$ is the set of decision variables, $\mathrm{J}(\mathcal{Z}) = \mathrm{Cost}(\mathrm{u_o}, \mathrm{u_1}, \dots)$ is the objective function, and $\mathbf{C}(\mathcal{Z})$ is a collection of constraints. $\mathcal{Z}^{\square}$, $\mathbf{C}^{\square}$ with $\square \in \{\min, \max\}$ are the limits. By utilizing this NLP along with additional constraints, we are able to generate the desired behaviors within seconds, whereas previous methods would require minutes or even hours to converge to a feasible solution [18]. We will describe additional constraints next.

IV. Additional Constraints

We will describe the additional constraints used to obtain translate and flipping trajectories in the robot. Position, velocity, acceleration and orientation limits of the spatula were described previously (Constraint 1). We will describe the remaining constraints below.

***Constraint** 4: Joint angle, velocity and acceleration limits: similar to Constraint 1, we have lower and upper limits on the joint states and accelerations. Therefore, we have $q^{\min}$,$q^{\max}$,$\dot{q}^{\min}$,$\dot{q}^{\max}$ and $\ddot{q}^{\min}$,$\ddot{q}^{\max}$ such that*

$$q^{\min} \leq q_i \leq q^{\max}$$
$$\dot{q}^{\min} \leq \dot{q}_i \leq \dot{q}^{\max}$$
$$\ddot{q}^{\min} \leq \ddot{q}_i \leq \ddot{q}^{\max}, \quad i = 0, 1, 2, \dots, N. \tag{13}$$

Hereafter we will assume that $i \in \{0, 1, 2, \dots, N\}$ unless otherwise mentioned.

***Constraint** 5: Torque limits: we have constant vectors $u^{\min}$ and $u^{\max}$ such that*

$$u^{\min} \leq u_i \leq u^{\max}. \tag{14}$$

A. Desired trajectory polynomials

The desired trajectories for each subbehavior are Beziér polynomials [11, eqn. (11)]:

$$P_j(t) = \sum_{k=0}^{6} t^k (1-t)^{6-k} \alpha_{kj} \frac{6!}{(6-k)!}, \ t \in [0, T], \tag{15}$$

where $j = 1, 2, \dots, 6$ and α_{0j}, α_{1j}, $\dots$ are the coefficients of the polynomial for joint j. The time t varies from 0 to T. It is important to note that the parameters α_{kj} and the total time T are also NLP variables that can be optimally obtained along with the rest of the NLP variables.

***Constraint** 6: For $T_{\min}$,$T_{\max} > 0$, we have the terminal constraints on time as*

$$T_{\min} \leq T \leq T_{\max}, \quad t_0 = 0, \quad t_N = T. \tag{16}$$

For 6 joint angles, we have six polynomials $P_1(t)$, $P_2(t)$, $P_3(t)$, $\dots$, and the corresponding output vector form

$$y(q, t) = \left[q_1 - P_1(t), q_2 - P_2(t), \dots, q_6 - P_6(t)\right]^T, \tag{17}$$

where the goal is to drive this output vector $y \to 0$. We therefore impose the following constraints (see [11, C5, C6, C10, C11] for the motivation for these, wherein they enforce convergence of the outputs):

***Constraint** 7: For each t_i and for some desired rate $\varepsilon > 0$*

$$\ddot{y}(q_i, \dot{q}_i, \ddot{q}_i, t_i) + 2\varepsilon \dot{y}(q_i, \dot{q}_i, t_i) + \varepsilon^2 y(q_i, t_i) = 0$$
$$y(q_0, 0) = 0, \qquad \dot{y}(q_0, \dot{q}_0, 0) = 0. \tag{18}$$

B. Friction cone constraints

The friction constraints used in the NLP formulation are slightly different than the inequalities specified in (5). Given the orientation of the spatula described as a function of the configuration q: $o(q) = [o_x(q), o_y(q), o_z(q)]^T$, we can define rotation matrices $R_x(q)$, $R_y(q)$, $R_z(q)$ that define the rotations around x,y,z axes respectively. With these rotation matrices, we have the total orientation of the spatula (end-effector) as $R(q) = R_z(q) \times R_y(q) \times R_x(q)$. The orientation of the normal vector of the spatula (axis perpendicular to the spatula) is

described as $F_n(q) := R(q) \times \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$. Similarly, we denote the normalized acceleration F_a of the burger as

$$F_a(q,\dot{q},\ddot{q}) = \frac{1}{\sqrt{a_x^2+a_y^2+a_z^2}} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, \tag{19}$$

where the arguments for a_x,a_y,a_z are suppressed for convenience. Having defined the normal vector F_n and the normalized acceleration vector F_a, we have the following constraint.

***Constraint** 8: Given the coefficient of friction μ, we have the inner product of these two vectors satisfying the following:*

$$\langle F_n(q_i), F_a(q_i,\dot{q}_i,\ddot{q}_i)\rangle \geq \cos(\mu), \tag{20}$$

which ensures that the burger does not slip.

C. Subbehavior based constraints

We have the following subbehavior based constraints:

1) Constraints for translate burger behavior: Given the current configuration of the robot, with the burger on the spatula, the goal is to attain a desired position and orientation for the spatula while maintaining the burger on the spatula. Therefore, for this type of subbehavior, we have the friction cone constraints (Constraint 8) satisfied along with the following additional constraints.

***Constraint** 9: Let q_s be the starting configuration of the robot, and let p_N, o_N be the desired final position and orientation of the end effector respectively. We have*

$$\begin{aligned} q_0 = q_s, \quad \dot{q}_0 = 0, \quad \dot{q}_N = 0, \\ p(q_N) = p_N, \quad o(q_N) = o_N. \end{aligned} \tag{21}$$

We will now describe the constraints for each subbehavior in the *pickup and flip burger behavior*.

2) Constraints for go to burger location: For this subbehavior, given the location of the burger $p^b = [p_x^b, p_y^b, p_z^b]^T$ and the starting configuration q_s, the spatula is tilted in the positive pitch direction (o_y) by around $o_y^* = 0.314$ rad (in order to get ready for the scoop subbehavior), and its tip is placed very close to the edge of the burger ($p_x^b - \frac{1}{2}d$). Accordingly, we have the following constraints.

***Constraint** 10: Go to burger constraints:*

$$\begin{aligned} p(q_N) &= \begin{bmatrix} p_x^b - \frac{1}{2}d - \frac{1}{2}l_l\cos(o_y^*) \\ p_y^b \\ p_z^b + \frac{1}{2}l_l\sin(o_y^*) \end{bmatrix}, & \dot{p}(q_N) &= 0 \\ o(q_N) &= \begin{bmatrix} 0 \\ o_y^* \\ 0 \end{bmatrix}, & \dot{o}(q_N) &= 0 \\ q_0 &= q_s, & \dot{q}_0 &= 0. \end{aligned} \tag{22}$$

3) Constraints for scoop: For the scoop action, the constraints imposed for the end points of the previous subbehavior are now imposed for the starting points of this behavior. The end points of this behavior are in turn constrained such that the spatula is perfectly lying flat below the burger.

***Constraint** 11: Scoop behavior constraints:*

$$\begin{aligned} p(q_0) &= \begin{bmatrix} p_x^b - \frac{1}{2}d - \frac{1}{2}l_l\cos(o_y^*) \\ p_y^b \\ p_z^b + \frac{1}{2}l_l\sin(o_y^*) \end{bmatrix}, & o(q_0) &= \begin{bmatrix} 0 \\ o_y^* \\ 0 \end{bmatrix} \\ p(q_N) &= p^b, & o(q_N) &= 0. \end{aligned} \tag{23}$$

The front edge of the spatula should be touching the grill:

***Constraint** 12: Given the grill height h_g, we have*

$$p_z(q_i) - \frac{1}{2}l_l\sin(o_y(q_i)) = h_g. \tag{24}$$

4) Constraints for pickup: For the pickup action, the starting points in turn satisfy the end point constraints of the previous subbehavior (which will be omitted for this and also the remaining subbehaviors to avoid repetition), and the end points will be such that the burger is lifted around 2cm forward and 5 cm above p^b.

***Constraint** 13: Pickup constraints:*

$$p(q_N) = \begin{bmatrix} p_x^b + 0.02 \\ p_y^b \\ p_z^b + 0.05 \end{bmatrix}, \qquad o(q_N) = \begin{bmatrix} 0 \\ -0.05 \\ 0 \end{bmatrix}. \tag{25}$$

5) Constraints for flipping: For the flipping action, the trajectory is designed in such a way that the friction cone constraints are satisfied for 60% of the total time. The spatula is also forced to point downwards and stay clear of the grill when the trajectory reaches its end. In addition, we also ensure that the spatula has minimal yaw angle throughout the trajectory, in order to prevent twisting of the arm. Therefore, we have the following constraints.

***Constraint** 14: Burger flipping constraints:*

$$\begin{aligned} \langle F_n(q_i), F_a(q_i,\dot{q}_i,\ddot{q}_i)\rangle \geq \cos(\mu), \;\; 0 \leq i \leq 0.6N \\ [-0.5,-0.5,-1]^T \leq F_n(q_N) \leq [0,0.5,-0.8]^T \\ [0.6,-1,-1]^T \leq R(q_i)\times[1,0,0]^T \leq [1,1,1]^T \\ h_g + 0.1 \leq p_z(q_N) \leq h_g + 0.4. \end{aligned} \tag{26}$$

6) Constraints for turning back: For the turn back action, the spatula that is pointing downwards is rotated back. In addition, we typically enforce the robot to its home configuration q^{home} so that it can be ready for its new task.

***Constraint** 15: Turning back constraints:*

$$o(q_N) = 0, \qquad q_N = q^{\text{home}}. \tag{27}$$

Final NLP formulation. Due to the addition of the constraints and also other parameters like α_{kj}'s and T, we have the new set of augmented decision variables $\{\alpha, T, \mathscr{Z}\}$. The trajectory optimization problem in its NLP formulation extending (10) is now described as

$$\begin{aligned} \{\alpha^*, T^*, \mathscr{Z}^*\} = \underset{\alpha,T,\mathscr{Z}}{\text{argmin}} \quad & \text{J}(\alpha,T,\mathscr{Z}) \\ \text{s.t.} \quad & \mathscr{Z}^{\min} \leq \mathscr{Z} \leq \mathscr{Z}^{\max} \\ & \alpha^{\min} \leq \alpha \leq \alpha^{\max} \\ & T^{\min} \leq T \leq T^{\max} \\ & \mathbf{C}^{\min} \leq \mathbf{C}(\alpha,T,\mathscr{Z}) \leq \mathbf{C}^{\max}, \end{aligned} \tag{28}$$

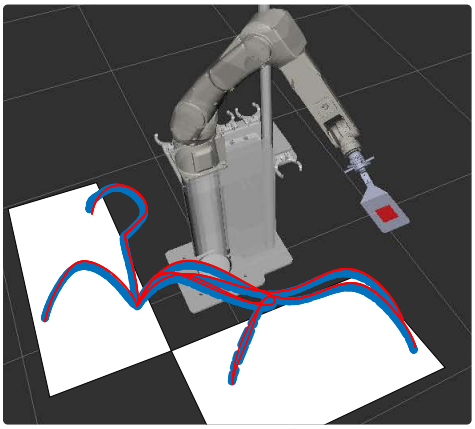

Fig. 6: Comparison between the actual (blue) and desired (red) end-effector positions during the translation of a burger across six points.

where $\mathbf{C}$ consists of the list of constraints. For each subbehavior $\alpha \in \mathbb{R}^{42}$, $T \in \mathbb{R}_{\geq 0}$ and $\mathcal{Z} \in \mathbb{R}^{504}$. So the total number of decision variables is 547. This NLP is then solved using IPOPT with the linear solver `ma57` [19].

D. End-point constraints

We need to ensure that the trajectories switch smoothly from one subbehavior to another. An alternative option would have been to include the decision variables of all the subbehaviors in one optimization problem. Due to constraints on optimization time, we generate the optimal trajectories sequentially and feed to the robot one after the other. Accordingly, we need to ensure that starting states for the current subbehavior match with that of the previous subbehavior.

***Constraint** 16: Let $x^- = (q^-, \dot{q}^-)$ be the ending state of the previous subbehavior. We have constraints on the starting state as*

$$q_0 = q^-, \quad \dot{q}_0 = \dot{q}^-, \tag{29}$$

which are included in the constraint set $\mathbf{C}$ in (28).

V. Results and Conclusions

We will describe the experimental results in this section. For verifying *translate burger behavior*, we created six key points, three on the grill and three on the tray side of the robot. The optimal trajectories were generated from each key point to the next key point. Fig. 6 shows the comparison between the simulated (desired) and experimental (actual) trajectories for each of the joints of the robot.

For verifying *pickup and flip*, we placed a burger patty on the grill and obtained its location via the camera (by employing machine learning techniques). With this information and also the current configuration of the robot, the optimization for the five subbehaviors: *go to burger location*, *scoop*, *pickup*, *flip*, and *rotate back* were executed one after the other. Note that the robot stores these trajectories as soon as they are available, and its internal scheduler has its own time set for the execution of these trajectories. Fig. 7 shows the comparison between simulated (desired) and experimental (actual) trajectories of the end-effector. Videos demonstrating both the main behaviors in the robot are provided in Table I.

Location $p = (p_x^b, p_y^b, p_z^b)$(m)	Success ratio (out of 10 trials)		
	scoop	*pickup*	*flip*
$(+0.64, -0.33, 0.073)$	7/10	7/10	7/10
$(+0.75, +0.00, 0, 073)$	10/10	9/10	9/10
$(+0.76, +0.09, 0.072)$	10/10	10/10	10/10
$(+0.64, -0.33, 0.072)$	10/10	10/10	10/10
$(+0.68, -0.31, 0.072)$	10/10	10/10	10/10
$(+0.69, -0.16, 0.072)$	10/10	10/10	10/10
$(+0.73, -0.13, 0.072)$	9/10	9/10	9/10
$(+0.73, -0.18, 0.072)$	10/10	10/10	10/10
$(+0.75, -0.45, 0.072)$	10/10	10/10	10/10
$(+0.84, +0.12, 0.072)$	10/10	10/10	10/10
Video list: Behaviors: https://youtu.be/cBhngkE0WLY Robustness: https://youtu.be/Gau0hgkZSbg			

TABLE I: Table showing the success ratio of *scoop*, *pickup* and *flip* subbehaviors for different locations on the grill.

Robustness. To analyze the success ratio of the flipping trajectories, we picked ten arbitrary locations, widely spread across the grill area. These locations are provided in the first column of Table I. Optimal trajectories were obtained for these ten locations and executed in the robot. We mainly focused on determining the success ratio of *scoop*, *pickup* and *flip* operations (contacts are made only during these three subbehaviors). The *scoop* operation is considered successful if at least half of the burger is over the spatula after the completion of this behavior. Similarly, *pickup* is considered successful if the spatula is successfully able to separate the burger from the grill, and finally, the *flip* operation is considered a success if the burger maintains contact for at least 60% of the entire duration, and then lands on its other side. In addition, the burger has to fall in the designated area within some acceptable margins (5cm in x and y directions). The success ratio for each subbehavior is shown in Table I. A robustness video for some of these trials is also given in Table I.

Burger flipping with the described method is very sensitive to location, including grill height. Even millimeter-level changes result in different outcomes (compare the first and fourth row in Table I, and also see the robustness video to view the perturbation analysis on the $x - y$ plane). There are also indirect factors that affect the success ratio. For example, too much grease makes the burgers slippery, and low grill temperature makes the burgers sticky. Success ratio is affected even further if the burgers are partially twisted, damaged or torn.

References

[1] K. M. Lynch and M. T. Mason, "Dynamic nonprehensile manipulation: Controllability, planning, and experiments," *The International Journal of Robotics Research*, vol. 18, no. 1, pp. 64–92, 1999.

[2] T. Watanabe, K. Yamazaki, and Y. Yokokohji, "Survey of robotic manipulation studies intending practical applications in real environments-object recognition, soft robot hand, and challenge program and benchmarking." *Advanced Robotics*, vol. 31, no. 19-20, pp. 1114–1132, 2017.

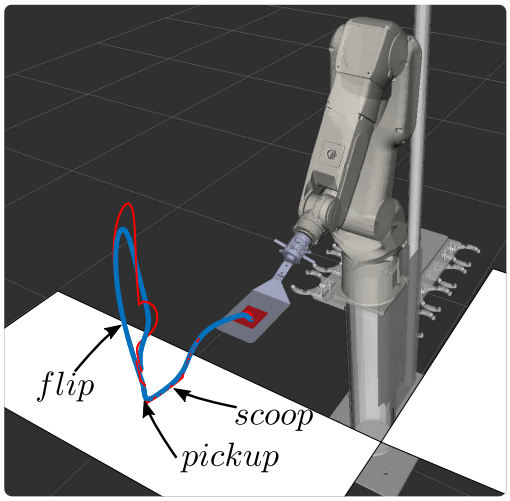

Fig. 7: Comparison between the actual (blue) and desired (red) end-effector positions during the *pickup and flip burger behavior*. The individual subbehaviors are also shown. Total execution time is < 3s. As shown by the end-effector position waveform, the maximum error is in the vertical position of the end-effector, which goes as high as 5 cm.

[3] A. Cowley, B. Cohen, W. Marshall, C. J. Taylor, and M. Likhachev, "Perception and motion planning for pick-and-place of dynamic objects," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 816–823.

[4] A. Murali, S. Sen, B. Kehoe, A. Garg, S. McFarland, S. Patil, W. D. Boyd, S. Lim, P. Abbeel, and K. Goldberg, "Learning by observation for surgical subtasks: Multilateral cutting of 3D viscoelastic and 2D orthotropic tissue phantoms," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1202–1209.

[5] K. Tahara, S. Arimoto, and M. Yoshida, "Dynamic object manipulation using a virtual frame by a triple soft-fingered robotic hand," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 4322–4327.

[6] K. M. Lynch, "Issues in nonprehensile manipulation," in *Robotics the Algorithmic perspective: The Third Workshop on the Algorithmic Foundations of Robotics*, 1998, p. 237–250.

[7] G. Lee, T. Lozano-Prez, and L. P. Kaelbling, "Hierarchical planning for multi-contact non-prehensile manipulation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 264–271.

[8] T. Tsuji, J. Ohkuma, and S. Sakaino, "Dynamic object manipulation considering contact condition of robot with tool," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 3, pp. 1972–1980, March 2016.

[9] A. Hereid and A. D. Ames, "FROST: Fast robot optimization and simulation toolkit," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC, Canada: IEEE/RSJ, Sep. 2017.

[10] W.-L. Ma, S. Kolathaya, E. R. Ambrose, C. M. Hubicki, and A. D. Ames, "Bipedal robotic running with DURUS-2D: Bridging the gap between theory and experiment," in *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '17. New York, NY, USA: ACM, 2017, pp. 265–274.

[11] A. Hereid, E. A. Cousineau, C. M. Hubicki, and A. D. Ames, "3D dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 5 2016, pp. 1447–1454.

[12] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of Guidance, Control, and Dynamics*, vol. 10, no. 4, pp. 338–342, 1987.

[13] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[14] O. T. Taylor, "Optimal shape and motion planning for dynamic planar manipulation," Ph.D. dissertation, Massachusetts Institute of Technology, 2017.

[15] S. Kolathaya and A. D. Ames, "Parameter to state stability of control Lyapunov functions for hybrid system models of robots," *Nonlinear Analysis: Hybrid Systems*, vol. 25, pp. 174 – 191, 2017.

[16] J. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd ed. Society for Industrial and Applied Mathematics, 2010.

[17] J. T. Wen and D. S. Bayard, "New class of control laws for robotic manipulators part 1. non–adaptive case," *International Journal of Control*, vol. 47, no. 5, pp. 1361–1385, 1988.

[18] H. Zhao, S. N. Yadukumar, and A. D. Ames, "Bipedal robotic running with partial hybrid zero dynamics and human-inspired optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 1821–1827.

[19] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar 2006.

Shishir Kolathaya is an INSPIRE Faculty Fellow in the Robert Bosch Center for Cyber Physical Systems at the Indian Institute of Science, Bengaluru, India. Previously, he was a Postdoctoral Scholar in the department of Mechanical and Civil Engineering at the California Institute of Technology. He received his Ph.D. degree in Mechanical Engineering (2016) from the Georgia Institute of Technology, his M.S. degree in Electrical Engineering (2012) from Texas A&M University, and his B. Tech. degree in Electrical & Electronics Engineering (2008) from the National Institute of Technology Karnataka, Surathkal. Shishir is interested in stability and control of nonlinear hybrid systems, especially in the domain of legged robots. His more recent work is also focused on fast trajectory optimization and real-time safety critical control for cyber physical systems.

William Guffey is an engineer at Miso Robotics, Pasadena, California where he works on robot behavior design and optimization. He received B.S. degrees in Mathematics and Physics (2018) from the University of North Carolina at Chapell Hill, North Carolina, USA. His interests include applied optimization, dynamic motion control, and unsupervised learning.

Ryan W. Sinnet is CTO and Co-founder of Miso Robotics, Pasadena, California. He received his Ph.D. (2015) and M.S. degree (2013) in Mechanical Engineering from Texas A&M University, College Station, and his B.S. degree (2007) in Electrical Engineering from Caltech, Pasadena. In his role at Miso Robotics, Ryan has leveraged recent advances in artificial intelligence and formal control methods to create Flippy—Miso's first commercial product—which is able to assist cooks by cooking hamburger patties on commercial grills. His research interests include robotics, nonlinear systems, and optimization as well as machine learning and high-performance computation methods.

Aaron D. Ames is the Bren Professor of Mechanical and Civil Engineering and Control and Dynamical Systems at Caltech. Prior to joining Caltech in 2017, he was an Associate Professor at Georgia Tech in the Woodruff School of Mechanical Engineering and the School of Electrical & Computer Engineering. He received a B.S. in Mechanical Engineering and a B.A. in Mathematics from the University of St. Thomas in 2001, and he received a M.A. in Mathematics and a Ph.D. in Electrical Engineering and Computer Sciences from UC Berkeley in 2006. He served as a Postdoctoral Scholar in Control and Dynamical Systems at Caltech from 2006 to 2008, and began his faculty career at Texas A&M University in 2008. At UC Berkeley, he was the recipient of the 2005 Leon O. Chua Award for achievement in nonlinear science and the 2006 Bernard Friedman Memorial Prize in Applied Mathematics, and he received the NSF CAREER award in 2010 and the 2015 Donald P. Eckman Award.