Flipping Pancakes via Trajectory Optimization

Charles Dawson

***Abstract*—Unlike many breakfast-related tasks, pancake preparation presents a particular challenge for robotic manipulation. While a box of cereal (for example) can be grasped using traditional prehensile manipulation techniques, pancakes must be flipped using dynamic non-prehensile manipulation strategies, either by using a spatula or by flicking the pan so that the pancake follows a ballistic flipping trajectory. In this report, we present the results of applying non-convex trajectory optimization with implicit contact modeling to find dynamically feasible pancake-manipulation strategies that rely only on the frictionless contact between the pan and pancake to move the pancake. We are able to successfully generate trajectories for both flipping the pancake and repositioning it by sliding. Furthermore, we find that these multi-contact trajectories can be effectively stabilized using a simple partial feedback-linearization strategy, due to the fully actuated substructure of the pan-pancake system (the pancake is unactuated, but the pan is fully actuated, allowing feedback linearization of the pan alone).**

I. Introduction

In a typical year, International House of Pancakes (IHOP) franchises serve more than 700 million pancakes [1], and at least half of US households made pancakes in 2016 [2]. Due to the proliferation of non-mix pancake recipes and the recent surge in home baking, the true scale of pancake production is likely even greater. However, despite the potential impact on the breakfast routines of millions, relatively little work has been done towards automating the crucial flipping operation required by most pancake preparation processes [5]. In particular, the visually-impressive technique of flipping a pancake without a spatula, by flicking the pan so that the pancake follows a ballistic trajectory before landing upside down in the pan, has received relatively little attention (while the spatula-assisted flipping task has received some attention from the food service industry [4]).

A likely explanation for this gap in the state of the art is the difficulty posed by the pancake flipping problem, which can be seen as an example of a dynamic non-prehensile manipulation task. Currently, most robot manipulation relies on fully-actuated strategies, where the robot first rigidly couples the manipulated object to the end-effector using an enveloping (or prehensile) grasp before moving the object to some target location and releasing. In contrast, a non-prehensile manipulation strategy establishes only a partial coupling between the end-effector and manipulated object before applying techniques from underactuated planning and control to influence the object through its dynamics [8]. For example, a fully-actuated manipulation strategy for moving a block across a table might use a parallel-jaw gripper to grasp the block securely, pick it up, and move it directly to the target location; throughout the manipulation, the block remains rigidly coupled to the end-effector, and dynamics may be canceled using feedback linearization. In contrast, a non-prehensile strategy might simply use the end-effector to push the block across the table. While using this strategy, the robot has only partial control of the pusher-slider dynamics and must plan its motion to be dynamically feasible instead of simply canceling the inherent dynamics of the problem.

A significant challenge of non-prehensile manipulation (as compared to fully-actuated manipulation) is the prevalence of hybrid contact dynamics. In the simple pusher-slider example of moving a block across a table, the motion of the block can be characterized by a transition through multiple modes (e.g. sticking, sliding, turning left, turning right, etc.) with qualitatively different dynamics. To effectively plan motions subject to such hybrid dynamics, past approaches have broken the planning problem into two stages by first searching for a sequence of modes that moves the object between its start and goal configurations, then planning and stabilizing trajectories accomplishing that set sequence [11, 4]. A similar approach does not commit to a single mode schedule ahead of time, but decides between alternative (predetermined) mode schedules at execution time [3]. These approaches have shown success in planning and executing non-prehensile manipulation tasks, but they require extensive task-specific fine-tuning; for example, Woodruff and Lynch successfully demonstrated non-prehensile manipulation of a block (flipping, sliding, and tossing it to a goal position), but their approach required manually selecting a mode sequence and hand-designing nominal state trajectories to stabilize between mode transitions [11]. As a result, mode sequence-based hybrid controllers cannot scale easily to more complex manipulation tasks, since it quickly becomes intractable to search over all possible mode sequences. To address this scaling issue, contact-implicit trajectory optimization techniques have been developed to search for locally-optimal trajectories that obey hybrid contact dynamics without explicitly specifying mode sequences in advance [6].

In this work, we apply the contact-implicit trajectory optimization strategy developed by Posa, Cantu, and Tedrake to the pancake-flipping problem, demonstrating both flipping and repositioning-through-sliding behaviors discovered via local trajectory optimization. Furthermore, we demonstrate that these trajectories can be stabilized using relatively simple state-tracking controllers, suggesting that a sufficiently robust feedback controller can effectively track simple trajectories even when those trajectories involve complicated hybrid dynamics.

II. Background

When a mode sequence can be specified ahead of time, a trajectory optimization problem can be formulated using direct

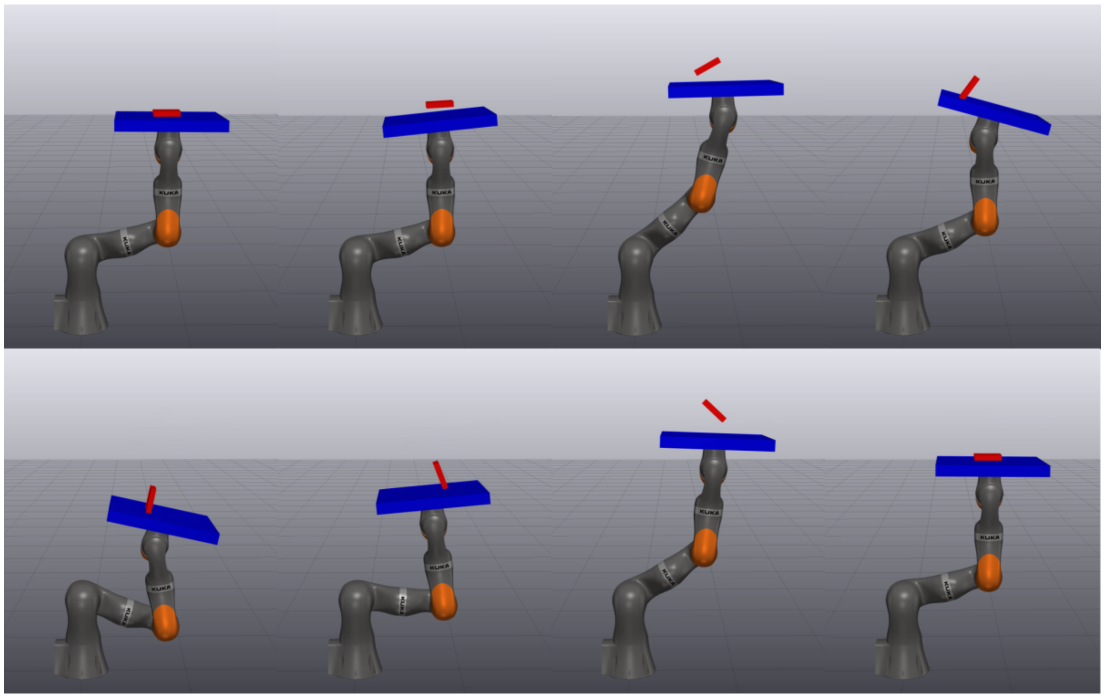

Fig. 1. A simulated KUKA robot arm executing a pancake-flipping trajectory found via trajectory optimization, where the pancake starts at configuration $\begin{bmatrix} x_{pancake} & z_{pancake} & \theta_{pancake} \end{bmatrix} = \begin{bmatrix} 0 & 0.15 & \pi \end{bmatrix}$.

transcription or direct collocation to explicitly constrain the trajectory to obey the dynamics of each mode in the specified sequence [4]. This formulation allows the duration of each mode to be set as a decision variable in the optimization, but nonetheless requires both a predefined sequence of modes and an explicit encoding of the instantaneous change in state that occurs at each state transition (e.g. an explicit description of the change in velocity after an inelastic collision [10]). Kolathaya et al. use an explicit mode sequence within a direct collocation framework to find locally optimal trajectories accomplishing a burger-flipping task [4].

An alternative approach, proposed by Posa, Cantu, and Tedrake, instead allows the mode sequence to be set *implicitly* in the case of control-affine rigid-body dynamics with contact [6]. In this formulation, the trajectory optimization is formulated using direct transcription, where the rigid-body dynamics are modified to include contact forces:

$$\begin{aligned} \text{find} \quad & q, \dot{q}, \ddot{q}, u, \lambda, h && (1) \\ \text{s.t.} \quad & M(q)\ddot{q} + C(q,\dot{q}) + \tau_G(q) = B(q)u + J(q)^T\lambda && (2) \\ & \phi(q) \geq 0 && (3) \\ & \lambda \geq 0 && (4) \\ & \phi(q)^T \lambda = 0 && (5) \end{aligned}$$

where q is the trajectory in state space, u is the control input, M, C, τ_G, and B are the matrices in the standard control-affine manipulator equations, λ represent the contact forces between bodies, J is the Jacobian of the contact points with respect to state, and $\phi(q)$ represents the "contact guards," typically equal to the signed distance between the rigid bodies. Eq. (2) enforces standard control-affine rigid-body dynamics, (3) enforces non-penetration between rigid bodies, (4) prevents contact forces from being negative (i.e. prevents the bodies from sticking together), and (5) is a so-called complementarity constraint that ensures that contact forces λ are non-zero only when the bodies are in collision (when the corresponding element of $\phi(q)$ is zero). If friction is considered, additional slack variables and complementarity constraints can be added to enforce the friction cone constraints. Additionally, we include the duration of each timestep h as a decision variable, and add additional constraints (not shown) to ensure that timesteps do not shrink to zero duration. Optimization problem (1) can be formulated as a nonlinear program (NLP) and solved using sequential quadratic programming (SQP, [6]).

III. Problem Statement

We model the pancake and pan as flat blocks, where the pancake has half the mass of the pan. For simplicity, we constrain the pan and pancake to lie in the x-z plane, and we focus our attention on the well-buttered regime where friction is negligible. We assume that the pan is fully actuated but that no control inputs are applied directly to the pancake. Furthermore, we make the simplifying whole-wheat assumption in considering the pancake and the pan as rigid bodies, and we model collisions as inelastic.

Following [6] in constructing a contact-implicit model, we employ a floating-base coordinate system and track the full 2D pose of each body in our configuration:

$$q = \left[x_{pan}, z_{pan}, \theta_{pan}, x_{pancake}, z_{pancake}, \theta_{pancake}\right]^T$$

as shown in Fig. 2. We assume non-dimensional masses of 1 and 0.5 for the pan and pancake, respectively.

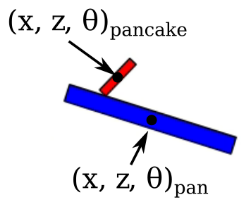

Fig. 2. The floating-based coordinate system used for contact-implicit trajectory optimization.

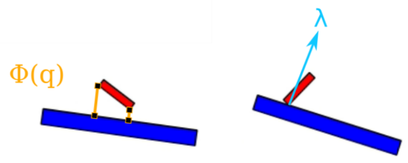

Fig. 3. An illustration of the signed distance function $\phi(q)$, used as a contact guard in our optimization problem, and the contact force λ.

IV. Non-convex Trajectory Optimization

In this section, we formalize and explain the additional constraints added to the basic contact-implicit formulation in (1) to yield the full optimization problem (6) used to solve the pancake flipping problem. First, we employ the variable-timestep approach described in [6] where we discretize time into a fixed number ($T = 60$) of timesteps, then allow the duration of each timestep h_i to vary on $[h_{min}, h_{max}] = [0.002, 0.05]$ while constraining the duration of neighboring pairs of timesteps to be constant (such that $h_1 + h_2 = h_3 + h_4 = h_5 + h_6 = \ldots$). These constraints, (13) and (14) in (6), allow the optimizer the freedom to use smaller timesteps as appropriate to model impulsive collisions, but prevents the timesteps from vanishing altogether.

Within this discrete-time framework, we enforce contact-implicit rigid body dynamics via the backwards Euler method constraints (7), (8), and (9). Note that, as in (2) above, (7) combines the standard second-order rigid-body dynamics equations with an additional $J(q)^T\lambda$ term, where λ are decision variables representing the contact forces at each corner of the pancake and $J(q)^T$ is the Jacobian of the contact points with respect to the joint state of the pan-pancake system, expressed in the pan frame, which serves to map contact forces into generalized forces on the configuration variables q. Note that in the well-buttered (zero-friction) regime, the contact forces can act only in the direction normal to the pan.

To ensure that the contact forces λ are non-zero only when the pancake is in contact with the pan, we track the signed distance $\phi(q)$ between each corner of the pancake and the top surface of the pan (as illustrated in Fig. 3) and enforce two additional constraints on this quantity. First, we ensure that the pancake does not penetrate the pan by constraining $\phi(q) \geq 0$; second, we ensure that contact forces must be zero whenever the pancake is not touching the pan via a complementarity constraint $\phi(q)^T\lambda = 0$. Additionally, since it is unrealistic for the contact force to be negative, we also constrain $\lambda \geq 0$. Adding these constraints to problem (6) yields (10), (11), and (12).

Finally, to avoid trajectories involving nonphysical control inputs, we add the bounding box constraint (15) to constrain the minimum and maximum allowable control effort, and to ensure that the trajectory ends at the center of the pan, we constrain the final configuration of the system to be $q_{end} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0.15 & 0 \end{bmatrix}^T$, as in constraint (17) in problem (6) (the 0.15 offset in $z_{pancake}$ accounts for the thickness of the pan). We vary q_{start} in constraint (16) to perform qualitatively different actions (flipping and repositioning through sliding). A key advantage of this contact-implicit trajectory optimization approach is that it does not require an explicit mode sequence in order to plan motions involving contact. As we will see in Section VI, this flexibility means that we can discover new behaviors simply by changing the starting configuration of the pancake. When the pancake starts "right side up," the optimizer finds a solution that simply tilts the pan to slide the pancake into the center, but when the pancake starts upside down relative to its desired configuration, the optimizer finds a solution that includes a ballistic phase to flip the pancake. In all experiments, we seed the optimization with a simple linear interpolation between the start and goal configurations, so all complex behaviors are emergent properties of the optimization, not artifacts of our initial guess.

We do not add a cost to our optimization formulation, as we are interested only in finding feasible trajectories, but additional costs (e.g. a quadratic penalty on control effort) can be added easily.

$$\begin{aligned}
\text{find} \quad & q, \dot{q}, \ddot{q}, u, \lambda, h && (6)\\
\text{s.t.} \quad & M(q_{i+1})\ddot{q}_i + C(q_{i+1}, \dot{q}_{i+1}) + \tau_G(q_{i+1}) = \\
& \qquad B(q_{i+1})u_i + J(q_{i+1})^T\lambda_i && (7)\\
& q_{i+1} = q_i + h_i\dot{q}_{i+1} && (8)\\
& \dot{q}_{i+1} = \dot{q}_i + h_i\ddot{q}_i && (9)\\
& \phi(q) \geq 0 && (10)\\
& \lambda \geq 0 && (11)\\
& \phi(q)^T\lambda = 0 && (12)\\
& h_i + h_{i+1} = h_{i+2} + h_{i+3} && (13)\\
& h_{min} \leq h \leq h_{max} && (14)\\
& u_{min} \leq u \leq u_{max} && (15)\\
& q(0) = q_{start} && (16)\\
& q(T) = q_{end} && (17)
\end{aligned}$$

V. Stabilization

Once we have solved the trajectory optimization problem to find a feasible trajectory that moves the system between the desired start and end configurations, we need to design a controller capable of tracking that trajectory. In general, stabilizing a trajectory through contact is a difficult challenge, as projecting linearized dynamics onto the contact manifold often yields an uncontrollable linear system. Some techniques for stabilizing multi-contact trajectories solve a constrained-LQR problem by working in the null space of the contact manifold and projecting the resulting controller into the higher-dimensional state space [7], but this method can encounter difficulties when a system "chatters" between contact modes.

To avoid these difficulties, instead of pursuing a constrained-LQR approach for stabilizing the trajectories found by solving (6), we can exploit the special structure of our model to find a simpler controller to track the desired trajectories. Recall that our system can be split into two parts: a fully-actuated pan and a completely unactuated pancake. As the pancake is less massive than the pan itself, we can think of the contact forces exerted on the pan by the pancake as a disturbance applied to the pan. Thus, we can reduce the problem of stabilizing a multi-contact trajectory to the much simpler problem of tracking a trajectory for the fully-actuated pan while being robust to the disturbances resulting from contact. To solve this tracking problem, we construct a partial feedback-linearization controller that applies feedback linearization to the pan alone, then uses PID state feedback to track the nominal pan trajectory. As we will see in Section VI, this simple controller is capable of remarkable tracking performance in the pancake-manipulation context (although at the cost of requiring additional PID gain tuning).

VI. Results

In this section, we present the nominal trajectories found by solving the optimization problem (6) with various initial conditions, along with the results of stabilizing those trajectories using linear state feedback. We implemented our solution in the Drake framework, which uses the SNOPT SQP solver [9]. To visualize the trajectories on a KUKA robotic arm, we solved the inverse kinematics of the arm to find joint-space trajectories that matched the robot's end-effector pose to the planned pose of the pan. The code for formulating the optimization problem and stabilizing the resulting trajectories was written in Python, while the code for finding inverse kinematic solutions for executing these trajectories on a robotic arm was written in C++.

As mentioned in Section IV, because we do not need to provide an explicit mode schedule *a priori*, we can obtain qualitatively different emergent behaviors simply by varying the start configuration of the pancake. When the pancake starts at configuration $\begin{bmatrix} x_{pancake} & z_{pancake} & \theta_{pancake} \end{bmatrix} = \begin{bmatrix} 0 & 0.15 & \pi \end{bmatrix}$ (i.e. upside-down relative to its desired final configuration; the offset of 0.15 in the z direction accounts for the thickness of the pan), we find the feasible solution shown in Fig. 1 that tosses the pancake into the air, catches it on a side, allows it to slide for a moment, then tosses it once more to catch it "right side up." Remarkably, this relatively complex sequence of actions is found without specifying any mode-sequence *a priori*; we only provide a simple linear interpolation between the start and goal configurations as a seed to the optimizer. The optimization typically takes less than one minute to solve on an Intel i7-8565U CPU at $1.80\,$GHz.

TABLE I
PID GAINS USED FOR PARTIAL FEEDBACK-LINEARIZATION, APPLIED TO $\begin{bmatrix} x_{pan} & y_{pan} & \theta_{pan} \end{bmatrix}$

	Flipping Controller	**Sliding Controller**
$\mathbf{K}_P$	$\begin{bmatrix} 1 & 10 & 10 \end{bmatrix}$	$\begin{bmatrix} 10 & 20 & 500 \end{bmatrix}$
$\mathbf{K}_I$	$\begin{bmatrix} 1 & 50 & 1 \end{bmatrix}$	$\begin{bmatrix} 20 & 20 & 100 \end{bmatrix}$
$\mathbf{K}_D$	$\begin{bmatrix} 50 & 50 & 50 \end{bmatrix}$	$\begin{bmatrix} 1 & 20 & 10 \end{bmatrix}$

When we change the starting configuration to $\begin{bmatrix} x_{pancake} & z_{pancake} & \theta_{pancake} \end{bmatrix} = \begin{bmatrix} 0.75 & 0.15 & 0 \end{bmatrix}$, so that the pancake is right side up relative to the desired position but offset in the x direction, we see entirely different behavior. In this case, the optimization finds a feasible solution that does not involve a ballistic phase; instead, it discovers a simple sliding behavior that moves the pancake to the goal configuration, as shown in Fig. 6.

To evaluate the performance of the partial feedback-linearization controller, we used the PID gains shown in Table I, then used Drake's built in `InverseDynamicsController` to track a piecewise-linear reference trajectory for the pan constructed from the results of solving problem (6). The tracking performance of this controller while executing the flipping behavior is shown in Fig. 4, while its performance in executing the sliding behavior is shown in Fig. 5.

VII. Discussion

Although the general problem of stabilizing a multi-contact trajectory can require highly sophisticated control strategies, we found that due to the structure of our problem, which permits partial feedback-linearization, we were able to obtain good tracking performance from a simple state-feedback controller. Although these feedback linearization approaches often risk demanding inordinately high control inputs in order to cancel the inherent dynamics of the plant, we found that the demanded control effort rarely exceeds the nominal control effort found by solving our trajectory optimization problem. This close agreement is likely due to the fact that we are tracking a trajectory that is already dynamically feasible, so the controller does not need to affect a large change in the underlying dynamics in order to stabilize these trajectories. Furthermore, the performance of this simple controller is likely aided by the fact that the pancake has only half the mass of the pan, so the disturbances due to making and breaking contact with the pancake are small enough that they do not require large control efforts. The danger of this approach can be seen at the end of the plot of control effort in Fig. 4; when the nominal trajectory ends before the tracking controller has completed its motion, and the tracking controller is asked to track a constant trajectory at $q = q_{goal}$, which is not necessarily dynamically feasible and results in a large spike in control effort as the controller has to reject the impact from the pancake in this regime. A future refinement on

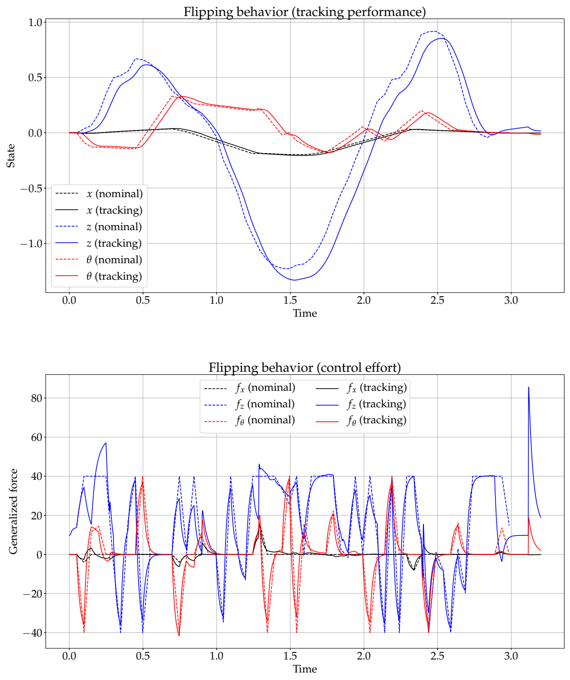

Fig. 4. Above: the performance of a simple partial feedback-linearization controller in tracking the flipping reference trajectory found using contact-implicit trajectory optimization. Below: the control effort demanded by this controller while tracking the reference trajectory. Note that the demanded effort rarely exceeds the nominal effort found by solving problem (6), although the demanded effort spikes near the end of the trajectory due to a later-than-expected impact from the pancake (occurring after the nominal trajectory has finished but before the tracking controller has caught up).

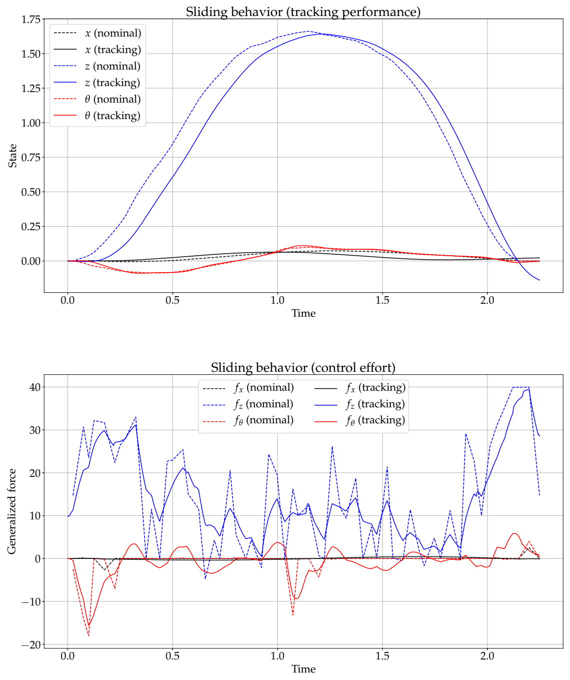

Fig. 5. Above: the performance of a simple partial feedback-linearization controller in tracking the sliding reference trajectory found using contact-implicit trajectory optimization. Below: the control effort demanded by this controller while tracking the reference trajectory.

this controller might compensate for the lag between the tracking controller and the nominal trajectory, adjusting the set-point of the controller to the point on the nominal trajectory closest to the current state of the pan rather than moving the set-point forward in time exactly in sync with the nominal trajectory. Alternatively, we might explore augmenting this partial feedback-linearization approach with a time-varying LQR instead of a state-tracking PID controller.

The non-convex trajectory optimization framework used here provides a remarkable degree of flexibility in solving multi-contact underactuated control problems. Without specifying a mode sequence *a priori*, we were able to find dynamically feasible trajectories that accomplish not only pancake flipping, but pancake repositioning-via-sliding as well. Although the standard caveats of non-convex optimization apply to this approach (we make no claim to completeness or guaranteed convergence, and this approach is sensitivity to initial guess trajectory, etc.), we hope that these results will nevertheless help boost productivity in the breakfast-preparation sphere.

A video summarizing our results can be found online at https://youtu.be/q2j53m65DKU, and the source code for this project is available at https://github.com/dawsonc/pancake_flipper.

Acknowledgments

We would like to thank the 6.832 course staff for their support on this project. The feedback survey keyword is "underactuated."

References

[1] Pancake Facts — Sunshine Restaurant Partners. http://www.ihopsrp.com/pancake.php.

[2] Packaged Facts: Still Room for Pancakes and Waffles at America's Breakfast Table. https://www.foodmanufacturing.com/consumer-trends/news/13163815/packaged-facts-still-room-for-pancakes-and-waffles-at-americas-breakfast-table, August 2016.

[3] Georg Bätz, Arhan Yaqub, Haiyan Wu, Kolja Kühnlenz, Dirk Wollherr, and Martin Buss. Dynamic manipulation: Nonprehensile ball catching. In *18th Mediterranean Conference on Control and Automation, MED'10*, pages 365–370, June 2010. doi: 10.1109/MED.2010.5547695.

[4] Shishir Kolathaya, William Guffey, Ryan W. Sinnet, and Aaron D. Ames. Direct Collocation for Dynamic Behaviors With Nonprehensile Contacts: Application to Flipping Burgers. *IEEE Robotics and Automation Letters*, 3(4):3677–3684, October 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2854910.

[5] Jessie Damuck Lau, Alex. BA's Best Buttermilk Pancakes Recipe. https://www.bonappetit.com/recipe/bas-best-buttermilk-pancakes.

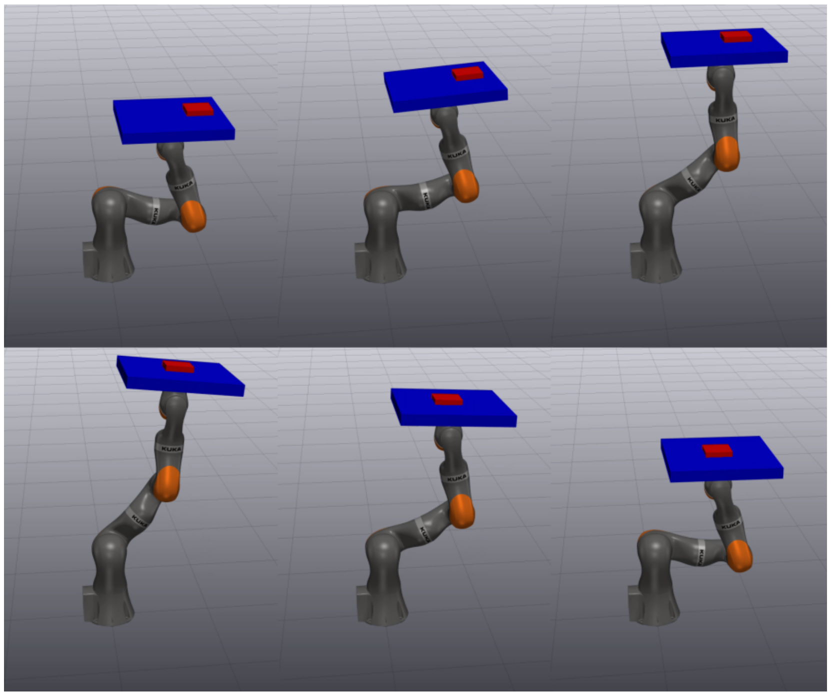

Fig. 6. A simulated KUKA robot arm executing a pancake-sliding trajectory found via trajectory optimization, where the pancake starts at configuration $\begin{bmatrix} x_{pancake} & z_{pancake} & \theta_{pancake} \end{bmatrix} = \begin{bmatrix} 0.75 & 0.15 & 0 \end{bmatrix}$.

[6] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, January 2014. ISSN 0278-3649. doi: 10.1177/0278364913506757.

[7] Michael Posa, Scott Kuindersma, and Russ Tedrake. Optimization and stabilization of trajectories for constrained dynamical systems. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1366–1373, May 2016. doi: 10.1109/ICRA.2016.7487270.

[8] Fabio Ruggiero, Vincenzo Lippiello, and Bruno Siciliano. Nonprehensile Dynamic Manipulation: A Survey. *IEEE Robotics and Automation Letters*, 3(3):1711–1718, July 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2801939.

[9] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019.

[10] Russ Tedrake. Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832). Accessed on 10/5/2020 from http://underactuated.mit.edu.

[11] J. Zachary Woodruff and Kevin M. Lynch. Planning and control for dynamic, nonprehensile, and hybrid manipulation tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4066–4073, May 2017. doi: 10.1109/ICRA.2017.7989467.