WipingBot: Optimization-based Surface Wiping in Drake

Nishanth Kumar*
MIT CSAIL
Cambridge, MA
njk@csail.mit.edu

Willie McClinton*
MIT CSAIL
Cambridge, MA
wbm3@csail.mit.edu

***Abstract*—Wiping surfaces with persistent contact is an important capability to enable robots to solve a wide variety of practically-useful cleaning tasks. In this work, we implement and evaluate a robotic system in Drake that wipes dirt from arbitrary locations on a table or on plates located on the table. We document and discuss the iterative development process we took to produce our overall system, as well as its constituent parts. We also evaluate the system on several progressively-large problem instances and benchmark its performance with several different formulations of the optimization problem used by the planner on problems involving randomly-generated dirt and plate locations. A video providing an overview of our system, as well as various qualitative results, is available here.**

I. Introduction

The task of wiping a surface is fundamentally both a practical and interesting manipulation task. Wiping is a critical component of performing various cleaning tasks, such as cleaning windows, floors, dusting or washing dishes, which are among the most-significant desiderata for robots from humans [19]. Moreover, the task of successfully wiping a surface in a realistic setting requires a number of key robotic capabilities, such as maintaining persistent, forceful contact with a surface, reasoning about the order in which to wipe dirt from particular locations, perception to detect dirty regions, and many more. Thus, wiping a surface is a worthy and important task of study.

In this work, we focus on developing a robotic system that is capable of wiping up a dirty kitchen table in a simplified household setting. Specifically, our system is capable of wiping dirt from potentially numerous locations on a table in front of it, as well as from plates that might be placed arbitrarily on this table. We describe the iterative process we used to create the components of this system, namely a setup for simulating the desired scene, as well as dirt and the process of wiping it, low-level position and hybrid force-position controllers that accomplish various parts of a wiping trajectory, several optimization-based formulations for generating trajectories to wipe dirt efficiently, and a high-level control module that sequences the low-level controllers to accomplish a particular trajectory. We present and discuss quantitative experiments comparing the performance of the various optimization approaches across trajectory length, optimization time and percentage of dirt wiped. Finally, we present and discuss several avenues for future extensions of this work.

* denotes equal contribution
Paper written for Final Project of 6.843: Robotic Manipulation, Fall 2021

II. Related Work

The task of wiping a surface with a sponge fits nicely into the framework of hybrid position-force control, which has a rich history of literature [17, 9, 14]. Several previous works have applied such controllers to similar tasks of chamfering and polishing parts [10] and grinding [12]. Our work builds upon such approaches by leveraging hybrid force-position control for wiping either a table or plate on the table with a sponge. Unlike these approaches however, we aim to enable a robot arm to wipe dirt from a series of previously-unknown locations and leverage optimization techniques to yield a trajectory for our task.

There have been a number of works that attempt to plan trajectories for wiping dirt from regions on a surface [16, 8, 20, 2, 11]. These works generally attempt to find some shortest-length trajectory to clean a surface using probabilistic planning approaches. For instance, [16, 8] formulate the problem of wiping a surface as a Markov Decision Process (MDP) and specify rewards for cleaning various amounts of dust from the surface. More closely related to us is [11], who formulate the problem in terms of having a surface discretized into particular cells, that are either dirty or clean. They apply a sampling-based procedure that also uses some gradient descent to optimize the trajectories after sampling. In contrast to these works, we formulate the problem of finding a trajectory explicitly as an optimization problem and use off-the-shelf non-convex optimization tools to produce such trajectories. Moreover, we propose and compare several different optimization formulations for the problem.

A different line of related work has attempted to perform similar manipulation tasks by leveraging machine learning [4, 15, 5]. These works generally leverage Learning from Demonstrations (LfD) [1], Reinforcement Learning (RL) [22] or some combination of both to learn reactive policies that map pure sensor data to robot actions that accomplish the task. Similar to some of these approaches [15, 13], we output actions for a hybrid force-position controller. Unlike these approaches, our method utilizes no learning whatsoever. Consequently, we are unable to act based solely on pixels or other sensor data. Conversely however, our approach requires no demonstration

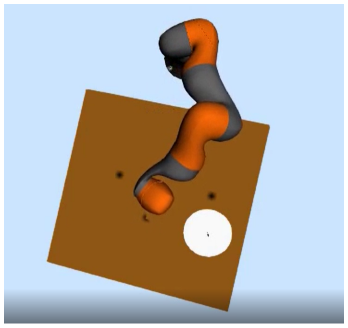

Fig. 1. A top-down view of our simulation setup. The black spots on the table represent dirt that can be cleaned when the robot's end-effector (a sponge) comes into contact with it. The white circle is a plate that contains some particle-based dirt

data or interaction with the environment and is able to produce trajectories of arbitrary size that generalize to wiping arbitrary dirt locations on the table and even from an arbitrary number of plates placed on the table. Although we did not pursue it in this work, we believe that experimentally comparing and contrasting our approach against these techniques could be useful in more precisely characterizing the advantages and disadvantages of these two approaches.

III. System Description

Our overall system consists of 4 interconnected pieces: the simulation setup for the environment, low-level closed-loop controllers that enable us to command the robot to achieve specific position or hybrid force-position profiles, an optimization problem setup and solver that generates trajectories for these controllers, and a high-level controller that makes calls to the low-level controllers in order to achieve the desired trajectories. All these components were implemented in Drake [24]. Below, we describe both the iterative process behind the implementation of these pieces, as well as the final versions of the pieces themselves, in more detail. An image of our final simulation setup is shown in Figure 1.

A. Simulation Setup

1) Simulating the Robot Arm: Initially, we planned for our simulation to consist of a 7 degree-of-freedom KUKA LBR iiwa arm with a Schunk WSG gripper that would be initialized to be holding a sponge to be used for wiping. However, since we wanted to make sure no relative motion between the gripper and sponge were possible, we opted to remove the gripper and instead rigidly weld the sponge to the end of the iiwa. Moreover, since simulating deformable objects is challenging, we chose to use a simple, rigid `Box` primitive to represent the sponge. While these choices render our setup somewhat unrealistic, they were useful simplifying assumptions. Moreover, we believe that our approach and setup can be easily extended to include both the gripper and a deformable sponge on a real robot in the future.

2) Simulating Dirt: Our initial implementation used a particle-based simulation for dirt. Specifically, we represented dirt pieces by small, colored `Box` primitives. While this was simple to use and setup, it had numerous disadvantages. Firstly, having many dirt particles in the scene made simulation rather slow, since it forced Drake's simulation engine to track the dynamics of numerous particles simultaneously. Secondly, this representation of dirt was unrealistic because there was no simple way to make the dirt disappear from the surface after wiping. Thus, we opted to simulate dirt by leveraging Drake's `MeshPainter` feature. This feature enables us to update the texture of one surface whenever it comes into contact with another surface. We leveraged this to represent dirt as colored patches on a table surface that are removed when the sponge applies force over them. However, since we were only able to use the `MeshPainter` between two surfaces in a scene (and we chose to leverage this for dirt on tables), we had to resort to our naive particle-based representation for dirt on plates.

3) Simulating Plates on a Table: We chose to simulate a table with potentially many plates as surfaces for our robot to wipe dirt from. Initially, we used table and plate assets from the PyBullet [3] library. While these were quite realistic, they presented several problems. Most importantly, we discovered that custom assets are not compatible with Drake's `MeshPainter` feature that we decided to use to simulate dirt. Moreover, the plates in particular were too bowl-shaped for the sponge to fit inside and wipe dirt without moving the plates themselves. As a result, we took a significantly simpler approach to simulating these and simply used a large `Box` primitive to represent the table with small, flat `Cylinder` primitives used to represent plates.

B. Controller Design

In order to accomplish our task of wiping dirt from the surfaces of both tables and plates, we chose to implement two controllers to interface with the robot arm. First, we implemented a position controller to move the arm to points in free-space, specifically to various points above the table or plate surfaces. Next, we also implemented a hybrid force-position controller to move the arm laterally against either the table or plate surface while exerting a certain force, thereby accomplishing wiping. At any given point in time, only one controller is actively commanding the robot. While the choice to use these two different controllers is rather natural for our setup, we note that it is unfortunately impractical on the real robot since the KUKA LBR iiwa must be rebooted before switching between controllers. Moreover, each of our controllers are closed-loop and designed to return a 'done' signal once they have achieved an input desired state.

1) Position Controller: We chose to implement our position controller by formulating and solving an optimization problem via Inverse Kinematics (IK). Specifically, we chose to use the following constrained optimization problem to perform IK:

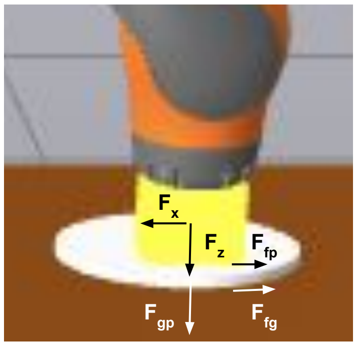

Fig. 2. A free body diagram for the case when our end-effector is wiping dirt from a plate. Here, F_x is the force in the x-direction that the end-effector is exerting, F_z is the downward force from the end-effector, F_{fp} is the force of friction on the end-effector by the plate, F_{fg} is the force of friction on the plate from the ground, and F_{gp} is the weight of the plate

$$\begin{aligned} \min_{q} \quad & (q - q_{nom})^2 \\ \text{s.t.} \quad & |x_{sol} - x_{des}| \le \epsilon_x \\ & |y_{sol} - y_{des}| \le \epsilon_y \\ & |z_{sol} - z_{des}| \le \epsilon_z \\ & |\theta_{sol} - \theta_{des}| \le \epsilon_\theta \\ & |\phi_{sol} - \phi_{des}| \le \epsilon_\phi \\ & |\psi_{sol} - \psi_{des}| \le \epsilon_\psi \end{aligned} \tag{1}$$

Where q_{nom} is some nominal joint position (we chose one where the arm is positioned just above the table surface), $(x_{sol}, y_{sol}, z_{sol}, \theta_{sol}, \phi_{sol}, \psi_{sol})$ are the x, y, z, roll, pitch, and yaw of the end-effector in the world frame that are the result of computing the forward kinematics of the arm given the joint angles q, $(x_{des}, y_{des}, z_{des}, \theta_{des}, \phi_{des}, \psi_{des})$ are the desired x, y, z, roll, pitch and yaw in the world frame, and $(\epsilon_x, \epsilon_y, \epsilon_z, \epsilon_\theta, \epsilon_\phi, \epsilon_\psi)$ are user-specified tolerances.

We used SNOPT [7] initialized with the current pose to solve this IK optimization problem. Additionally, given some desired final end-effector pose X_{des} and some current pose X_{curr}, we generated several intermediate poses to pass to the IK problem and thereby encourage the solver to not return a solution that exceeds the joint velocity limits.

While this formulation worked well enough for position control for our purposes, we noticed that we had to specify rather large tolerances along the x and y axes for the solver to return any solution. This is certainly undesirable, and something that we believe can be significantly improved in the future, especially since we are only moving through free-space and not providing any complicated collision constraints. In particular, we believe formulating the optimization problem and controller to use differential IK could have yielded more accurate solutions.

2) Hybrid Force-Position Controller: Our objective with this controller was to command the robot end-effector to exert some constant force in the z-axis while moving laterally to some desired position in the x and y axes. In order to command a constant force in the z-axis, we must put the robot in torque-control mode. However, this makes it difficult to accomplish motion in the x and y axes, since it is not straightforward to derive what torques, or differential torques, we should exert on the end-effector in order to reach some desired x and y positions. Thus, inspired by the examples of hybrid force-position control for dragging a book from the 6.843 course notes [23], we chose to use a Proportional Differential (PD) controller to output control signals for our robot based on input measured position and velocity of the end-effector in cartesian space.

Our first attempt at implementing the overall hybrid force-position controller was to simply command this force in the z-axis, implement PD controllers that attempted to drive the x and y positions to the desired positions and the x and y velocities to 0, and command 0 torque for the roll, pitch and yaw axes. We then computed the torque to be applied at each of the robot's joints by multiplying the inverse of the robot's Jacobian with respect to its joints with the control signal output by our PD controller. Unfortunately, this implementation did not achieve the desired motion because the arm would eventually tip over along the pitch or yaw axes, as shown in this video!

Our next attempt involved implementing PD controllers on the roll, pitch and yaw values in an attempt to keep the arm upright. Specifically, we found the roll, pitch and yaw angles that corresponded to the end-effector being upright and then used a PD controller to drive the measured roll, pitch and yaw angles to these desired angles and also drive the angular velocities to 0. However, to our surprise, we found that this too caused the arm to tip over along the pitch or yaw axes (watch video) even after attempting to tune the gains. Upon closer inspection, we realized that this is because of the circular property of angles: π radians is the same as $-\pi$ radians, but simply taking their difference yields 2π, which leads to a larger-than-required proportional term.

To combat this issue of the arm tipping over, we implemented a simple D controller on the roll, pitch and yaw values instead of a PD controller. Specifically, we ensured the end-effector was initialized upright and implemented D controllers to drive any measured velocities in the roll, pitch or yaw directions to 0. After tuning the gains for the PD controllers on x and y and the D controllers on roll, pitch and yaw.

One lingering minor issue that we noticed was that the arm tended to bounce-off surfaces after coming into contact with them. We mitigated this by adding a differential term to the z-axis output. Specifically, we set the force along the z-axis to be the desired force minus some constant times the current velocity of the end-effector along the z-axis. This worked well and led to smooth trajectories that maintained contact with the surface shown here.

Now that we had a controller that was able to stay upright, we needed to decide on a z-force to exert such that we can wipe dirt from both plates or the table. We performed a simple force analysis shown in Figure 2. In particular, we want to exert a force F_z that is greater than 0 (in order to actually wipe the dirt), but also that is insufficient to cause the plate to move with respect to the table surface when wiping dirt from a plate. For this case, we can derive the following inequalities:

$$\begin{aligned} F_x &> \mu_{sp} F_z \\ F_x &< \mu_{pt}(F_z + m_p g) \end{aligned} \tag{2}$$

Where μ_{sp} is the coefficient of static friction between the sponge end-effector and the plate and μ_{pt} is the coefficient of static friction between the plate and the table. After some algebra, we can derive that:

$$\frac{F_x}{\mu_{pt}} - m_p g < F_z < \frac{F_x}{\mu_{sp}} \tag{3}$$

Given this, the known values for μ_{pt} and μ_{sp} and assuming $g = 10m/s^2$, we have that:

$$\frac{F_x}{0.1} - 10m_p < F_z < \frac{F_x}{0.1} \tag{4}$$

Unfortunately, F_x is not known ahead of time. Moreover, F_z is not simply static, but rather depends on the current z-velocity of the end-effector. We measured it empirically by having our controller execute a few trajectories and found that it ranged from 0 to roughly 1.5. In order to increase the set of feasible F_z values, we set the mass of the plates to be fairly high (5 kg) and tuned the desired force along the z-axis, F_z^{des}, as well as the coefficient for the differential term in our z-force controller to be within the bounds from equation (4) assuming that $F_z = 1.5$, which worked well enough for our purposes (**ToDo: Link to video**).

Our final controller has the following form:

$$\begin{aligned} u &= [F_x, F_y, F_z, F_\theta, F_\phi, F_\psi] \\ F_x &= K_{px}(x - x_{des}) + K_{dx}(\dot{x}) \\ F_y &= K_{py}(y - y_{des}) + K_{dy}(\dot{y}) \\ F_z &= F_z^{des} - K_{dz}(\dot{z}) \\ F_\theta &= K_{d\theta}(\dot{\theta}) \\ F_\phi &= K_{d\phi}(\dot{\phi}) \\ F_\psi &= K_{d\psi}(\dot{\psi}) \end{aligned} \tag{5}$$

Where u is the output control signal from the controller, K_p represents a proportional gain, K_d represents a differential gain, θ, ϕ, ψ represent roll, pitch and yaw respectively, $\dot{x}$ represents cartesian velocity along the x-axis, $\dot{\theta}$ represents the angular velocity about the roll axis, and F_z^{des} is the desired force along the z-axis calculated above.

C. Optimization Formulation of Trajectory for Wiping

Given the controllers from the previous section that can navigate to various waypoints for dirt locations, we must now produce these waypoints given dirt locations. We considered a number of different approaches to generating these way points.

1) Naive Approach: The most straightforward approach is to simply set waypoints to be exactly the dirt locations. However, this can be extremely sub-optimal. Consider Figure 3: the naive approach (labelled "Approach baseline") zig-zags through the different dirt locations, leading to unnecessarily-long trajectories.

2) Constraints-based Optimization Formulation: In order to find more optimal trajectories than the naive approach, we decided to formulate the waypoint generation problem as a constrained optimization problem. One natural formulation that attempts to fix the problems with the naive approach is shown below:

$$\begin{aligned} &\min_{x_i, y_i} \sum_{i=1}^{n} (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 \\ &\text{s.t. for each } d_x, d_y \in D \\ &\min_i ((x_i - d_x)^2 + (y_i - d_y))^2) < \epsilon \\ &x_0, y_0 = x_{\text{start}}, y_{\text{start}} \\ &x_{n-1}, y_{n-1} = x_{\text{end}}, y_{\text{end}} \end{aligned} \tag{6}$$

Where n is a user-specified constant representing the number of waypoints in the trajectory to be generated, D is the set of all x,y positions of dirt locations on the surface of the table in the world frame, ϵ is a user-defined tolerance, and $x_{\text{start}}, y_{\text{start}}$ represent some user-defined starting location for the trajectory, while $x_{\text{end}}, y_{\text{end}}$ represent some user-defined ending location for the trajectory. Intuitively, this formulation forces the solver to produce a trajectory that goes to all the dirt-locations while also minimizing the total L2-length of the trajectory. Because this problem gets significantly harder as ϵ is made smaller, we implemented two versions of this formulation: "Approach strict_constraint" where we set $\epsilon = 0.0001$, and "Approach threshold_constraint" where $\epsilon = 0.005$.

3) Objective-based Formulation: An alternative formulation from the above is to take the constraint that the produced trajectory must be ϵ-close to the dirt locations and then move it into the objective, thereby making it more of a 'soft' constraint. This is shown below:

$$\begin{aligned} &\min_{x_i, y_i} \min_i ((x_i - d_x)^2 + (y_i - d_y)^2) + \\ &\sum_{i=1}^{n} (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 \\ &\text{s.t. for each } d_x, d_y \in D \\ &x_0, y_0 = x_{\text{start}}, y_{\text{start}} \\ &x_{n-1}, y_{n-1} = x_{\text{end}}, y_{\text{end}} \end{aligned} \tag{7}$$

Where the variables are the same as those listed in the previous sub-section. This formulation is termed "Approach min_distance_cost" in Figure 3.

To solve these constrained optimization problems we use SNOPT again [7], since these problems are non-convex. Our initial guess is critical to finding ad solution and we found that randomly sampling points on our table as the initial guess

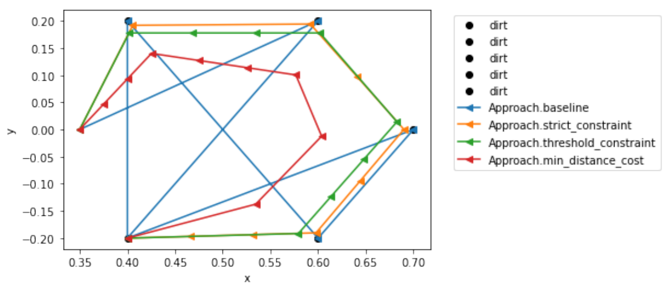

Fig. 3. An example of generate trajectories for five dirt locations [(0.6, 0.2), (0.4, -0.2), (0.7, 0.0), (0.6, -0.2), (0.4, 0.2)] using our three constraint optimization formulations compared to our naive baseline of wiping each dirt location in order given.

for our start and end points, while randomly sampling dirt locations for our waypoints works for most problems.

D. Overall Control Strategy

With the above components implemented, we can now perform our wiping actions. For problems involving plates, we do this by first navigating to a position above our dirt location and descending till touching the surface via pure position control, then switching to hybrid-position force control to perform our wiping motion maintaining constraint force with proper orientation, and finally switching back to pure position control to ascend and repeat the process for the next dirt location. For problems involving only wiping dirt from tables, we simply navigate to the starting location of the trajectory output by our optimizer using pure position control, and then navigate to each of the waypoints using our hybrid force-position controller. Note that the ability to switch between these controllers only exists in simulation, in order to switch between position and force control on the iiwa the robot would need to be turned fully on and off.

The complete pipeline takes in a series of dirt locations, a start and end position, and whether it will be wiping plates or table dirt spots. Then sets up the optimization problem to generate a path for the wiper to take along the table and passes that to our controller which, depending on whether it is approaching, descending, ascending, or wiping, uses the pure position or the hybrid-postion force controller to execute these commands on the iiwa.

IV. Experiments

We performed multiple experiments to test our overall system, and particularly, to compare the different optimization formulations. Our experiments consist of wiping: (1) a single dirt location on a table (2) multiple dirt locations on a table (3) a single plate on a table (4) multiple plates on a table. Each of the plate and dirt locations are randomly generated in a reachable location on the table, and given to the controller in a list in the order of generation. Also, for each experiment we measured the percentage of dirt wiped, the total distance traveled, and the time it took to solve for the trajectory. Since the trajectory generated for dirt wiping and plate wiping are the same, we only present results for dirt on table location optimization [1].

We ran each of our optimization approaches on one (single dirt) and five (multi dirt) locations. The results are displayed below in Table 1.

Table 1: Evaluation of Optimization Approaches

Task Name	% Wiped	$\sum(\Delta x)^2$	Solver Time
Baseline			
Single Dirt	100	0.186	N/A
Multi Dirt	100	0.792	N/A
Strict			
Single Dirt	100	0.132	0.089
Multi Dirt	100	0.412	4.02
Threshold			
Single Dirt	100	0.020	0.164
Multi Dirt	100	0.212	25.7
Objective Based			
Single Dirt	100	0.005	0.057
Multi Dirt	60	0.025	0.305

These results indicate that taking an optimization approach over a naive wiping approach can significantly reduce the total distance traveled while maintaining the ability to successfully clean the table. In particular, for the single dirt case, all of our optimization formulations are able to wipe 100% of the dirt from the table with a trajectory that is in some cases significantly shorter than the baseline trajectory. Interestingly, though the objective-based formulation finds the smallest length trajectory for the single dirt location case, it fails to wipe all the dirt from the multi dirt location case. This makes sense because there is no hard constraint that all the dirt must be wiped for this formulation. Additionally, for all of our formulations, the solve time increases significantly when going from the single to multi dirt case, although this increase is not as dramatic for the objective-based formulation than for the constraint-based formulation. This suggests that there exists a tradeoff between the goals of decreasing solve time, wiping all the dirt from the surface and minimizing the length of the trajectory needed to wipe: for problems with many dirt locations, we might have to sacrifice percentage of dirt wiped or trajectory length in order to solve the problem in a reasonable about of time.

In order to further investigate how the solve time and percentage of dirt wiped for our different formulations scale with the number of dirt locations, we performed experiments with progressively more dirt locations. The results are displayed in Tables 2 and 3 below.

[1]Note also that for these experiments, we did not run our full system to observe the percentage of dirt wiped since simulating the full trajectories for each of our cases would be prohibitively slow. Rather, we measured the percentage of dirt wiped by writing a procedure to check the distance between the various waypoints of the trajectory and the dirt locations.

Table 2: Strict Constraint Formulation Ablation

# of Dirt	% Wiped	$\sum(\Delta x)^2$	Solver Time
1	100	0.132	0.089
2	100	0.008	1.34
3	100	0.101	2.97
4	100	0.162	3.23
5	100	0.213	7.61
6	100	0.412	4.02
7	100	0.568	11.62
8	100	0.728	13.3
9	100	0.749	18.3

Table 3: Objective Based Formulation Ablation

# of Dirt	% Wiped	$\sum(\Delta x)^2$	Solver Time
1	100	0.005	0.057
2	50	0.004	0.116
3	33	0.014	0.144
4	0	0.036	0.223
5	60	0.025	0.305
6	17	0.119	0.301
7	43	0.085	0.290
8	37	0.173	0.415
9	33	0.085	0.338

As can be seen above, as the number of dirt locations grows so does the time it takes the solver to find a solution. As observed in Table 1, the objective-based formulation seems to scale more easily than the constraint-based formulation. However, this scaling comes at the cost of successfully wiping all the dirt: while the constraint-based solution always manages to wipe 100% of the dirt, the objective-based formulation is inconsistent. In the future, it would be interesting to study to what extent we can improve the performance of this objective-based formulation by weighting the two components of the objective differently.

V. Conclusion and Future Work

In this work, we have successfully built a system in Drake that uses position control to for navigate between objects on a table and uses hybrid force-position control for wiping. We also generate waypoints via different optimization approaches that range from a constraint-based approach where we constrain our waypoints to contain points within some epsilon distance of every dirt location to a objective based formulation where the strict constraint is move to a 'soft' constraint in the objective that can also be optimized. Our experiments showed that a constrained optimization based formulation can lead to more-efficient waypoint planing than using the dirt locations naively as waypoints, but does take time to solve the optimization problem. This time taken can become unreasonably large when optimizing for a trajectory parameterized by a large number of waypoints. We also empirically found the benefit using a simple D controller on row, pitch and yaw instead of a PD controller and adding a differential component to smooth the force applied on the z-axis.

We believe our project leads into several interesting avenues for future work. One natural extension is to incorporate vision into our system. In particular, instead of assuming we are given the locations of all the dirt patches, we could use a camera to identify them. We envision this could be done with fairly simple classical techniques as long as the contrast between the dirt and the rest of the table is significant. In order to detect dirt on more textured and variable surfaces, we might need to leverage modern deep-learning based approaches to computer vision, perhaps by fine-tuning an existing network to recognize dirt by generating a number of examples in simulation. Moreover, we currently assume that all dirt is removed once we exert force over it, but this is clearly unrealistic. Incorporating vision would allow us to detect when we have successfully wiped dirt and thereby modify our system to continuously apply larger forces and wipe the surface until the dirt is gone, similar to how humans perform wiping.

Another avenue for future work is enabling our approach to easily generalize to novel surfaces. Currently, our system is only capable of wiping the specific table and plates we used in simulation because of the non-trivial amount of manual tuning required for the hybrid force-position controller. It would be interesting to attempt to leverage techniques from System Identification [18] and/or Machine Learning (ML) to automatically compute the desired force along the z-axis and tune the gains for the hybrid force-position controller from a few examples or demonstrations with a new surface. These efforts could also enable us to transfer our system to a real robot more easily.

Finally, another avenue for future work is to extend our system to work with a mobile robot. This would enable us to not only wipe dirt from a nearby reachable surface, but also from multiple locations in a kitchen for example. This will likely require us to integrate our approach with higher-level planning or optimization techniques that reason about reachability of various locations and return paths through the environment that stop at various locations from which we can wipe dirt. We are particularly excited about integrating our current system with a larger system for Task and Motion Planning (TAMP) [6] to solve larger-scale cleaning tasks, such as those that are included in the BEHAVIOR dataset [21].

VI. Team Member Contributions

A. Nishanth

My main contributions to the project were (1) writing and tuning the hybrid force-position controller (including the force-balance analysis required to derive force to be exerted in the z-axis), (2) implementing the `MeshPainter` code in PyDrake and adapting it to the task of wiping surfaces instead of simply painting on them, (3) suggesting the idea of using optimization to yield waypoints that create a wiping trajectory. I also contributed to the writing of this report and the production of the presentation video.

B. Willie

My main contributions to the project were (1) creating the dirt particle simulations for plate wiping, (2) writing the pure position controller and intergrating it with Nishanth's hybrid force-position controller for our full control pipeline, (3) formulating the constraint optimization approaches in pydrake and using a non-linear solver to create a trajectory generator for our controller. Lastly, I also contributed to the writing and the production of our report and presentation.

Acknowledgments

We would like to thank Prof. Russ Tedrake for tirelessly answering conceptual and Drake-related questions, and for providing the `MeshPainter` code that updates the canvas to show wiping has been performed. We would also like to thank the Fall 2021 6.843 course TA's Rachel Holladay and Dani White for conceptual help with the project, particularly the proposal. We'd also like to thank Prof. Dylan Hadfield-Menell for suggesting the idea of tackling wiping as an optimization problem, and for helping formulate an initial version of the objectives and constraints. Finally, we thank Erwin Coumans and the pybullet [3] team for creating urdf files for frying pans and other dinnerware that we leveraged in an earlier iteration of this project.

References

[1] B. Argall, S. Chernova, M. M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics Auton. Syst.*, vol. 57, pp. 469–483, 2009.

[2] R. D. Carvalho, H. Vidal, P. Vieira, and M. I. Ribeiro, "Complete coverage path planning and guidance for cleaning robots," *ISIE '97 Proceeding of the IEEE International Symposium on Industrial Electronics*, vol. 2, pp. 677–682 vol.2, 1997.

[3] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation in robotics, games and machine learning," 2017.

[4] S. Elliott, Z. Xu, and M. Cakmak, "Learning generalizable surface cleaning actions from demonstration," *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 993–999, 2017.

[5] W. Gao and R. Tedrake, "kpam 2.0: Feedback control for category-level robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 6, pp. 2962–2969, 2021.

[6] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 265–293, 2021. [Online]. Available: https://doi.org/10.1146/annurev-control-091420-084139

[7] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM Rev.*, vol. 47, no. 1, p. 99–131, jan 2005. [Online]. Available: https://doi.org/10.1137/S0036144504446096

[8] J. Hess, J. Sturm, and W. Burgard, "Learning the state transition model to efficiently clean surfaces with mobile manipulation robots," 2011.

[9] N. Hogan, "Impedance control: An approach to manipulation: Part i—theory," *Journal of Dynamic Systems Measurement and Control-transactions of The Asme*, vol. 107, pp. 1–7, 1985.

[10] M. Jinno, F. Ozaki, T. Yoshimi, K. Tatsuno, M. Takahashi, M. Kanda, Y. Tamada, and S. Nagataki, "Development of a force controlled robot for grinding, chamfering and polishing," *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1455–1460 vol.2, 1995.

[11] A. M. Kabir, K. N. Kaipa, J. A. Marvel, and S. K. Gupta, "Automated planning for robotic cleaning using multiple setups and oscillatory tool motions," *IEEE Transactions on Automation Science and Engineering*, vol. 14, pp. 1364–1377, 2017.

[12] K. Kashiwagi, K. Ono, E. Izumi, T. Kurenuma, and K. Yamada, "Force controlled robot for grinding," *EEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*, pp. 1001–1006 vol.2, 1990.

[13] P. Kormushev, D. N. Nenchev, S. Calinon, and D. G. Caldwell, "Upper-body kinesthetic teaching of a free-standing humanoid robot," *2011 IEEE International Conference on Robotics and Automation*, pp. 3970–3975, 2011.

[14] D. Leidner, C. W. Borst, A. Dietrich, M. Beetz, and A. O. Albu-Schäffer, "Classifying compliant manipulation tasks for automated planning in robotics," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1769–1776, 2015.

[15] R. Martín-Martín, M. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space. an action space for reinforcement learning in contact rich tasks," in *Proceedings of the International Conference of Intelligent Robots and Systems (IROS)*, 2019.

[16] D. M. Martínez, G. Alenyà, and C. Torras, "Planning robot manipulation to clean planar surfaces," *Eng. Appl. Artif. Intell.*, vol. 39, pp. 23–32, 2015.

[17] M. T. Mason, "Compliance and force control for computer controlled manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, pp. 418–432, 1981.

[18] K. J. Åström and P. Eykhoff, "System identification-a survey," *Automatica*, vol. 7, no. 2, p. 123–162, mar 1971. [Online]. Available: https://doi.org/10.1016/0005-1098(71)90059-8

[19] C. Ray, F. Mondada, and R. Siegwart, "What do people expect from robots?" in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 3816–3821.

[20] M. T. Ribes, "Optimization of floor cleaning coverage performance of a random path-planning mobile robot," 2007.

[21] S. Srivastava, C. Li, M. Lingelbach, R. Martín-Martín, F. Xia, K. E. Vainio, Z. Lian, C. Gokmen,

S. Buch, K. Liu, S. Savarese, H. Gweon, J. Wu, and L. Fei-Fei, “BEHAVIOR: Benchmark for everyday household activities in virtual, interactive, and ecological environments,” in *5th Annual Conference on Robot Learning*, 2021. [Online]. Available: https://openreview.net/forum?id=TavPBk4Zs9m

[22] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[23] R. Tedrake, “Drake: Model-based design and verification for robotics,” 2021. [Online]. Available: http://manipulation.csail.mit.edu/

[24] R. Tedrake and the Drake Development Team, “Robot manipulation: Perception, planning, and control (course notes for mit 6.843),” 2019. [Online]. Available: https://drake.mit.edu