CatchingBot: A Robotic System for Catching Objects in Flight

Bartłomiej Cieślar
EECS
Massachusetts Institute of Technology
Cambridge, MA, USA
bcieslar@mit.edu

Michael Zeng
EECS
Massachusetts Institute of Technology
Cambridge, MA, USA
michzeng@mit.edu

Russell Louis Tedrake
CSAIL
Massachusetts Institute of Technology
Cambridge, MA, USA
russt-office@mit.edu

***Abstract*—Throwing and catching presents a quick and efficient way to move an object from point A to point B, which can be highly useful in robotic applications such as pick-and-place. However, catching an object requires great finesse and precision; as such, throwing and catching is rarely used in practice. We propose a method to catch arbitrary objects using just a two-finger gripper by predicting the object's trajectory using ICP and RANSAC, and using trajectory optimization to plan the robot's catching motion. We evaluated our method by catching a tennis ball, a banana and a pill bottle thrown from distances between 2.33m to 3.75m in a simulated environment. With a total catching success rate of over 85%, we believe our work demonstrates the feasibility of trajectory optimization for dynamic robot behaviors like throwing and catching. In the future, we would like to increase the system's robustness for more varying throw trajectories and executing the method in the real world, where limitations on computation speed and non-simulated aspects such as camera noise may present new challenges.**

***Index Terms*—ICP, RANSAC, trajectory optimization**

I. Introduction

While most robots today perform slow and relatively static behaviors, dynamic object handling (i.e. throwing and/or catching) offers many benefits - it's faster, longer-range, and simply visually-impressive. To humans, it also comes very naturally, whether tossing laundry into a laundry bin across the room, or playing a sport like basketball that is entertaining for others to watch. However, catching flying objects is a challenging task. The objects may only be in the air for a few hundred milliseconds, leaving very little time to perceive, plan, execute, and re-plan the robot's motion, especially if estimates of the object's trajectory are noisy. In addition, the object may be traveling very fast by the time it is within catching range, and the shape, size, and orientation of the object may also necessitate specific catching strategies. Our goal was also to perform the catch using very general hardware–a standard 7-DoF robot arm and 2-finger gripper (imagine catching a banana using chopsticks). These challenges, however, make this a very fun and technically-interesting problem to try to solve.

Therefore, in this project, we propose a three-stage method for a 7-DoF robotic manipulator with a 2-finger gripper to catch objects in flight based on feedback from RGB-D cameras observing the scene using purely algorithmic approaches:

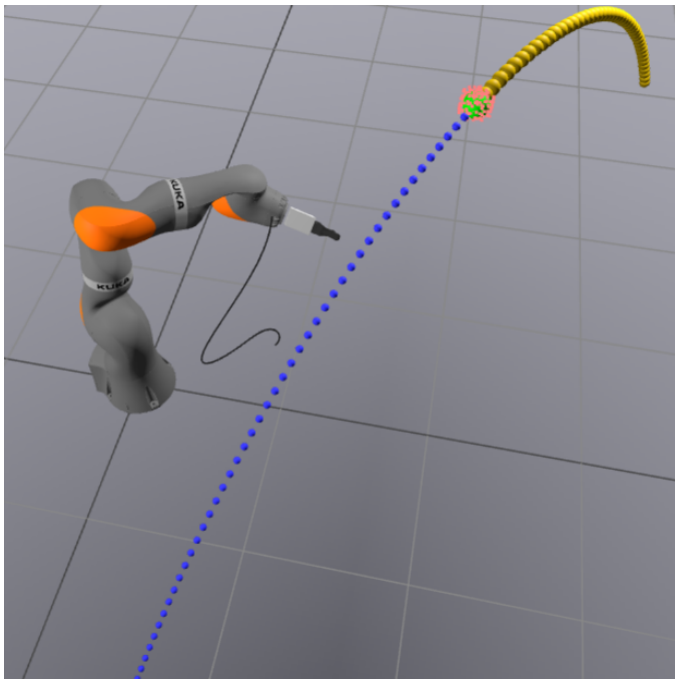

Fig. 1: Overall setup. The red dots around the ball are the captured point cloud. The yellow dotted line represents the trajectory predicted by Iterative Closest Point. The blue dotted line shows the trajectory predicted by RANSAC. The black line shows the robot's trajectory constructed by the motion planner.

Trajectory Prediction. The first stage of the pipeline predicts the trajectory of the flying object from the output of the RGB-D cameras. First, we use the Iterative Closest Point (ICP) method to find the object's position and orientation at certain key-frames from point clouds. Then, we use the RANSAC algorithm [1] to extract the ballistic trajectory of the moving object.

Grasp Selection. The second stage of the pipeline uses random sampling with heuristics to select a grasp time and a grasp pose. Darboux frames are first used as initial guesses for grasp poses. Then, the best grasp pose is chosen by evaluating all grasp pose samples on three proposed heuristics: "strength",

"catchability", "comfort", as well as ensuring non-collision.

Motion Planning. The third and final stage of the pipeline uses Trajectory Optimization to find the optimal path for the manipulator to reach the selected grasp pose, constraining the path to match the object's timing and velocity in order to perform a successful catch.

In order to evaluate our setup, we perform randomization on the object and its trajectory. We do make the assumption that the gripper is much faster than the real life version, and that the thrown object is not rotating. However, mild rotation can be accounted for somewhat easily in the kinematic optimization, with the other two subsystems natively supporting such environments.

From our results, we find that the system succeeds in catching the object 85% of the time across all testing scenarios, indicating that it is indeed possible to implement a system that catches a ball using a simple algorithmic approach and a two-finger gripper.

Project Inspiration. It is also worth mentioning that this project was greatly inspired by YouTube Legends Shane Wighton's Moving Basketball Hoop [2] and Mark Rober's Viral Moving Dartboard [3].

II. Related Work

A. Related Work on Dynamic Object Handling

The vast majority of past works we have seen related to catching objects use complex, high-DoF and high-speed manipulators and focus on catching balls, utilizing strategies similar to how a baseball glove catches a baseball [4] [5] [6] [7]. The method in [8] utilizes a 7 DoF arm and Allegro gripper also to catch objects, and uses learning-based methods and probabilistic models to simultaneously predict object trajectory and control the robot's position. The methods in [5] and [6] use only inverse kinematics and positional control to move the gripper to the catch position. Meanwhile, our method uses an optimization-based approach, which allows it to catch objects with just a 2-finger gripper, and using relatively slower hardware.

B. Related Work on Algorithms used in Our Method

Trajectory Optimization. We also investigated a number of related works that present methods that are similar or build upon the concepts we learned in class. Previous work discusses techniques for improving both speed and performance of trajectory-planning [9]. Firstly, they sample multiple initial predictions for the optimizer to better performance on local minimas. They also use a repeated method of lower-dimensional optimization and interpolation into higher dimensions to decrease the time taken by the optimization while improving the size of the search space (according to the paper). Both are possible extensions to our project, although we hypothesize that random sampling may not help much because the constraints on trajectory optimization are relatively simple to begin with.

Grasp Selection. Our grasp selection method is largely based on the method in [10], which samples points from the object point cloud and uses the local curvature of the object to determine a likely grasp. We found this method to work well in class work and adapted it using our own heuristics to work well for catching.

Trajectory Prediction. The RANSAC algorithm [1] is used to both stabilize the predicted trajectory and also account for outliers in the output from ICP. The original algorithm iteratively samples a trajectory based on two data-points (which uniquely define a ballistic trajectory) and chooses the one that has the most inlier points that are within some error of the predicted trajectory. For our method, for the trajectories that have the same number of inliers we also choose the one that has the smallest mean squared distance to the trajectory for all the inliers. This helps us make sure that we stabilize the trajectory prediction for small errors in ICP.

III. Approach

Our approach to the problem is divided into three stages that run continuously in time. In the first stage, we take the point cloud of the scene by performing the ICP algorithm, and passing the data into RANSAC, we estimate the trajectory of the object. Next, we choose a convenient grasp for the object to be caught in along with the point at the trajectory when it should be done. Finally, we iteratively perform kinematic trajectory optimization to find the best grasp for the object.

A. Environment

We use the drake simulator to perform our experiments. In order to capture a clean object point cloud, we have a system of 100x75 pixel depth cameras in an 8x4 UV-sphere (a grid in polar coordinates), 1m from the object's origin. The cameras for capturing the instantaneous pose of the object when it's flying are 400x300 pixel depth cameras and are arranged in a 4x5 UV-sphere, 7m from the environment's origin. The robot we use, KUKA iiwa with the WSG gripper, is placed at the origin of the world frame. The individual parts of the algorithm are connected to each other, as shown in Fig 2.

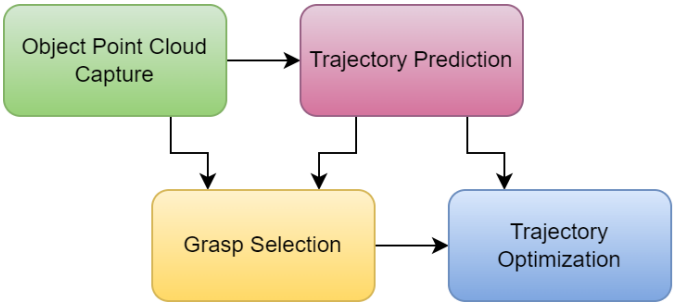

Fig. 2: Systems connections in drake. The trajectory optimization depends on grasp selection and trajectory prediction, the grasp selection needs the object's point cloud to sample antipodal grasps and the trajectory prediction for a suitable point along the trajectory, and trajectory prediction needs the object's point cloud for ICP.

B. Trajectory Prediction

Trajectory prediction is divided into two steps: object pose prediction and trajectory estimation (as shown on Fig 3 First, we perform the standard Interative Closest Point (ICP) algorithm to find the pose of the object based on its reference point cloud and the point cloud captured from the scene. We filter the scene point cloud based on the object label images from the camera. However, we only do so due to the limitation of the geometry query engine where one cannot query the object distance for multiple points at the same time. Otherwise, the points captured from the robot and the gripper could be keyed out using a signed distance function from the robot. It is also important to note that we initialize the ICP with the pose predicted during the last trajectory estimation in order to speed up the algorithm and make it more stable for continuous symmetry groups.

Second, we perform the standard Random Sample Consensus (RANSAC) algorithm. The algorithm takes 20 samples of potential trajectories based on random 2 poses from the ones predicted by ICP. Then, it first chooses the one with the most poses laying on or within some margin of error of the trajectory, and then based on the mean squared distance of the poses close to the trajectory from the trajectory. It is important to note that, for the sake of reducing computational complexity, we keep a window of only 30 past ICP predictions.

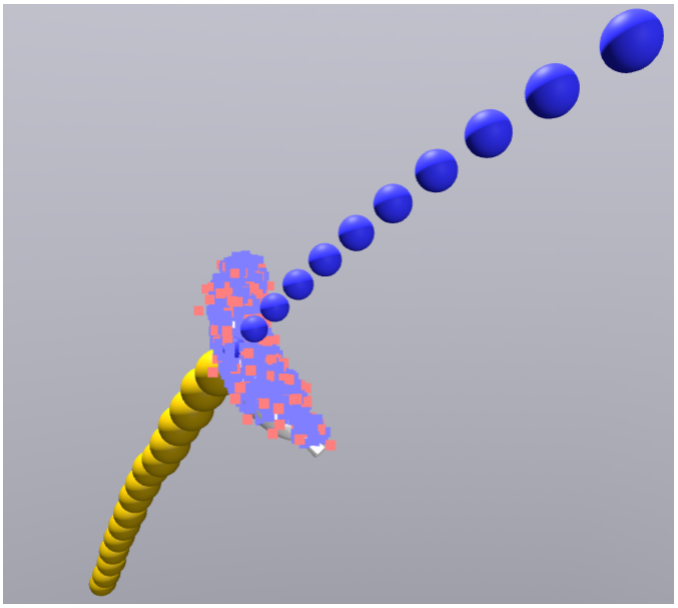

Fig. 3: Here we can see the setup for trajectory prediction. The blue dots are the ICP prediction of where the object is. The red dots are the reference point cloud captured in that time snapshot. The yellow dots are the previous predictions of ICP. The blue dotted line is the RANSAC prediction.

C. Grasp Selection

The goal of the Grasp Selection algorithm is to select a catching pose ${X^G}_{\text{grasp}}$ for the gripper and a corresponding time T for the catch to occur.

The algorithm contains two parts: first, we sample points from the object point cloud and compute their Darboux frames. This is a well-established method [10] that utilizes the Eigen-decomposition of the covariance matrix of all surface normal vectors in the neighborhood of the sampled point. The eigenvectors of the covariance matrix can be used to construct a rotation matrix, representing the orientation of the Darboux frame. We then offset the pose of the gripper away from the object in the direction of the surface normal a fixed distance, so the gripper isn't in contact with the object.

Second, we define three heuristics to evaluate each grasp sample in order to pick the best one. We call these heuristics "strength", "catchability", and "comfort". "Strength" measures the distance between the centroid of the object's point cloud and the y-axis ray of the gripper frame, and attempts to ensure the grasp "points toward" the object's center of mass. Using the vectors denoted in Fig. 4, This is calculated as:

$$\text{"Strength"} = \|\vec{v_c} - proj_{\vec{v_y}}\vec{v_c}\|_2$$

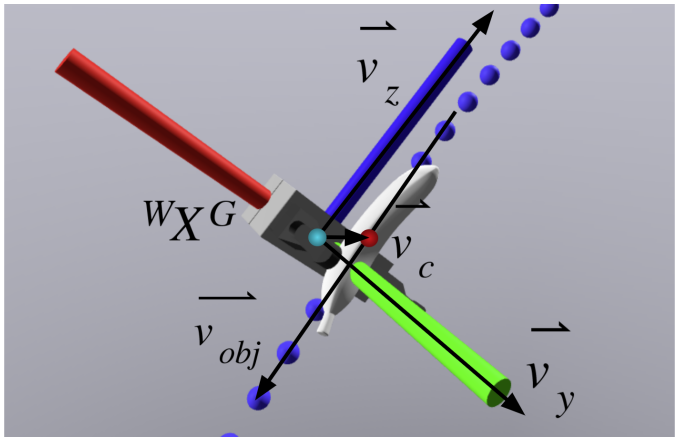

Fig. 4: Vectors used in Grasp Heuristics Calculations

"Catchability" measures the alignment of the gripper with the velocity of the object, allowing the object to fly in between the fingers. This heuristic is weighted most heavily and is calculated as the magnitude of the dot product between the z-axis of the gripper frame and the object velocity at the catch:

$$\text{"Catchability"} = \|\vec{v_{obj}} \cdot \vec{v_z}\|_2$$

"Comfort" measures the ease with which the robot can reach the grasp pose. This is calculated as the dot product between the y-axis ray of the frame and the vector drawn from vertical axis of the world origin (where the robot is located) to the grasp pose (denoted by ${}^W X^G$):

$$\text{"Comfort"} = \vec{v_y} \cdot \begin{pmatrix} {}^W X^G.translation()[0] \\ {}^W X^G.translation()[1] \\ 0 \end{pmatrix}$$

This method does have some limitations; most randomly sampled grasps do not perform well on these three heuristics,

therefore requiring a large number of samples to produce a good result. Additionally, using this grasping strategy places high thresholds on the heuristics to ensure a successful grasp; certain object geometries or rotations produce no feasible grasps at all. Aside from this edge case, the grasp sampling process should produce grasps as seen in Fig. 5.

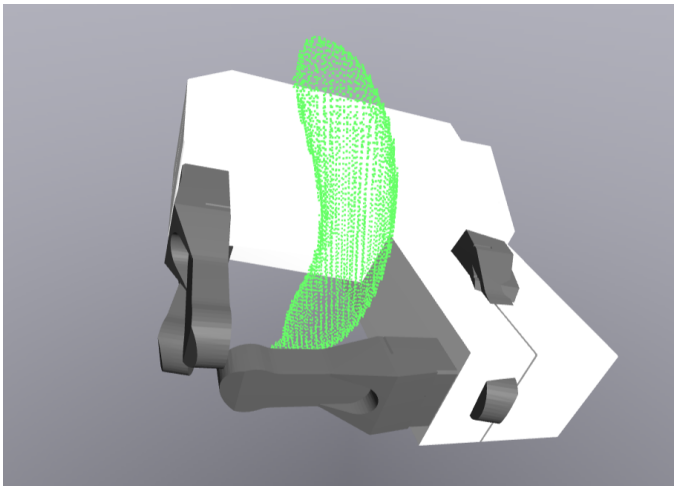

Fig. 5: Sampled Grasps that Pass a Threshold on all Three Grasp Heuristics

D. Trajectory Optimization

Our system uses kinematic trajectory optimization to plan a path for the robot. The decision variables of the optimization are q_α, the evenly-spaced (in time) control points of a 7-dimension B-spline (in joint configuration space), with time-rescaling so that the duration of the B-spline matches the time to catch. We optimize the deviation from each joint configuration $q(t)$ in the path to a set of "comfortable" joint angles q_0. With the following variable definitions:

- q_{start}: the current joint positions of the robot.
- T: time (in seconds) at which catch should occur (chosen by the Grasp Selection algorithm described above).
- $t_{pre-grasp}$: Pre-grasp time; computed as $T - 0.03$.
- $X^{G_{grasp}}$: gripper pose in "grasp" position (chosen by the Grasp Selection algorithm described above).
- $X^{G_{pre-grasp}}$: gripper pose in "pre-grasp" position.
- $V^{O_{grasp}}$: object spatial velocity when gripper is in "grasp" position.
- $V^{O_{pre-grasp}}$: object spatial velocity when gripper is in "pre-grasp" position.
- J_{kin}: Jacobian matrix mapping from joint positions and velocities to spatial poses and velocities.

The optimization is expressed mathematically like so:

$$
\begin{aligned}
\min_{q_\alpha} \quad & (q - q_0) \\
\text{subject to:} \quad & q(0) = q_{\text{start}}, \\
& \dot{q}(0) = \dot{q}_{\text{start}}, \\
& X^{G_{\text{pre-grasp}}} = J_{\text{kin}}(q(t_{\text{pre-grasp}})), \\
& \frac{V^{O_{\text{pre-grasp}}}}{\|V^{O_{\text{pre-grasp}}}\|} = \frac{J_{\text{kin}}(\dot{q}(t_{\text{pre-grasp}}))}{\|J_{\text{kin}}\dot{q}(t_{\text{pre-grasp}})\|}, \\
& X^{G_{\text{grasp}}} = J_{\text{kin}}(q(T)), \\
& \frac{V^{O_{\text{grasp}}}}{\|V^{O_{\text{grasp}}}\|} = \frac{J_{\text{kin}}(\dot{q}(T))}{\|J_{\text{kin}}\dot{q}(T)\|}, \\
& \forall t,\ |\dot{q}(t)| \leq v_{\text{max}}, \\
& \forall t,\ q_{\text{min}} \geq q(t) \leq q_{\text{max}}.
\end{aligned}
$$

One example of the resulting trajectory (in spatial coordinates) can be seen in Fig. 6 as the black line (note that the trajectory does not intersect the predicted trajectory of the object (the blue dots) because of the offset between the gripper's origin and actual "catching point").

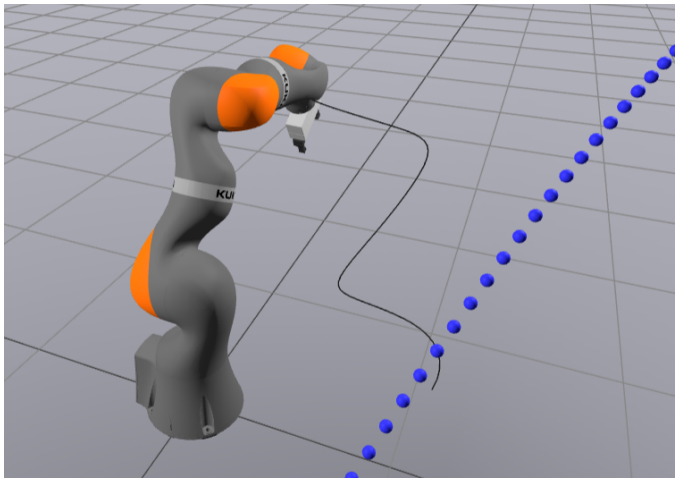

Fig. 6: The solution to the trajectory optimization problem in the form of the gripper's trajectory. The blue dotted line is the predicted trajectory of the object. The black line is the solved for trajectory of the gripper.

The constraints to the optimization are explained in more detail below:

Start joint position and velocity. The B-spline starts at the point where the robot currently is. We make sure that that start point has the current position and joint velocity of the robot.

Grasp gripper pose. The B-spline does not necessarily end at the grasp pose, but we make sure that the pose of the gripper at the time of grasping matches the one selected by the grasp selection system and, in consequence, the pose of the object we are catching.

Grasp gripper velocity. Ideally, the gripper's end velocity should move at the object's velocity at the catch, however, we found it infeasible to accelerate the robot up to such speeds (the object can be moving up to 7m/s at the catch point) in

the roughly half-second duration of a catch. Therefore, we try to make the robot arm move at 30% velocity of the object, but matching in direction.

Pre-grasp gripper pose and velocity We also encountered issues where often the gripper would end up hitting the object just before trying to grasp it. Thus, to avoid that, we make sure that we start matching the trajectory of the object 30ms earlier than it arrives at the gripper's location. Just like with the grasp-time gripper pose and velocity, we only try to match 30% of the velocity of the object (but ensure the directions match). For this constraint we also increase the solution tolerances for the pose of the gripper by 10 times.

The final element of the trajectory optimization is initialization and solving. We use the non-linear optimizer SNOPT to solve the optimization. However, due to the large number of constraints, SNOPT also requires a good initialization to succeed. Therefore, we first solve (also using SNOPT) the inverse kinematics (IK) problem at the gripper's catch pose, and linearly interpolate joint positions between the start positions and the catch positions solved using IK. We feed this as an initial guess to SNOPT. Then, we solve the optimization iteratively, starting with large error tolerances and gradually narrowing down until failure. Finally, every time step of the program, when the trajectory prediction of the object might vary slightly, we use the previously optimized trajectory as an initial guess to solve an updated trajectory. With these techniques in place, the trajectory optimization always finds a solution (or a very close solution, if not).

IV. Results and Discussion

A. Testing Procedure

To evaluate the abilities of our system, we simply measure the rate of catching success. We decide a catch to be successful when the distance between the object origin and gripper origin in gripper frame's z-axis is less than 2.5 cm at the end of the simulation time.

To test the system, we chose three small objects that demonstrate variety in shape but all fit in the WSG grippers: a tennis ball, a banana, and a pill bottle. Our system has some known limitations; namely, the trajectory optimization is sensitive to the location from which the the object is thrown as well as the orientation of the object. Therefore, we predetermined and fixed two different throwing trajectories and orientations for each object in our test setup. The throw trajectories were from 3.75m away and 2.33m away from the robot, with initial velocities of 7.87m/s and 5.68m/s respectively. The orientations of the throwing object were determined qualitatively, by roughly aligning the longest dimension of the object with the object's velocity at the catch pose. We then performed 48 trials for each combination of object and throwing distance/orientation, with each trial having a different randomization for RANSAC and grasp sampling. We also performed nine additional trials for each combination of object and throwing distance, with recorded video, to perform qualitative analysis of the results. These test videos can be found here. In total, 342 trials were conducted (57 for each combination of object and throwing distance/orientation).

The results of the trials are shown in Table I:

Ball 2.33m	Banana 2.33m	Pill Bottle 2.33m	Ball 3.75m	Banana 3.75m	Pill Bottle 3.75m
98.3%	86.0%	91.2%	87.7%	63.2%	86.0%

TABLE I: Catch Success Rate

B. Quantitative results

From the quantitative data, we observe that the reliability is quite high for shorter distance throws, and a little bit worse at longer distance throws. The ball is the easiest object to catch; it's roundness sometimes corrects for positional error by rolling off the gripper fingers into a successful catch (which we observed in video data). The banana fails to be caught most often due to its long shape, which makes it easy to collide with the fingers if the angle or velocity of the gripper is slightly off.

The rate of catch success for balls is even comparable or higher to similar published works. Of course, we must take many assumptions into account; our system uses a total of 20 depth cameras for trajectory prediction, resulting in much higher quality point-clouds and trajectory prediction. We also assume highly repeatable throws, and seamless transferability into the real-world, while, of course, there will be some sim-to-real gap. However, our system is the only one we have seen that uses a 2-finger gripper to perform catches, which adds a significant element of difficulty compared to other approaches.

C. Discussion of limitations

Analyzing the qualitative data from the recorded videos, we observe that the primary limitation of our current system is the trajectory optimization. In all the missed grasps, the gripper's pose and velocity at the catch appear to have significant error; specifically, the gripper is tilted up and is too high. This error is still within the allowed bounds of our optimization, but is enough to cause the catch to fail. Tightening the error tolerances any further causes the optimization to fail. We hypothesize that this error is likely due to the high speeds and short durations during which these catches occur; the robot comes very close but isn't able to meet the velocities and accelerations necessary to catch the object. This is further corroborated by the lower performance at higher throwing speed and distance. Therefore, no method of motion-planning may be able to achieve significantly higher success on the same hardware. In the future, we may test this hypothesis by performing even shorter distance and lower velocity throws and catches and seeing if this issue persists.

Other limitations of our method include that our trajectory prediction algorithm assumes the object will follow a parabolic trajectory, which is a reasonable assumption for heavy-enough objects, and that the object is either near-spherical or is not rotating in the air. In addition, the system does not yet run in real time.

Given the high success rates of our approach, we conclude that, with the limitation in variability of the throwing trajectory, constrained optimization can be a feasible approach to catching small-sized objects even when limited to a two-finger gripper. Further, we believe our results show that constrained optimization can be a powerful, expressive tool for many dynamic applications beyond catching, such as throwing.

V. Conclusion

Our work presents a method for catching objects using simple algorithms and a two-finger gripper. Moreover, we achieve a 85% success rate when trying to catch objects of various shapes and sizes. While we do make several simplifying assumptions, this work shows compelling evidence that purely algorithmic approaches to the task of catching objects with a robot arm are possible. In the future we would like to omit those unrealistic assumptions, such as the gripper being unrealistically fast, the cameras having perfect depth detection or that the object is not rotating. Another potential direction of future work would be to optimize the algorithms for speed, so that it's possible to perform the trajectory prediction and optimization steps more frequently.

References

[1] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[2] Stuff Made Here, "Moving hoop won't let you miss," 2020.

[3] Mark Rober, "Automatic Bullseye, MOVING Dartboard," Mar. 2017.

[4] P. Cigliano, V. Lippiello, F. Ruggiero, and B. Siciliano, "Robotic ball catching with an eye-in-hand single-camera system," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 5, pp. 1657–1671, 2015.

[5] M. Sato, A. Takahashi, and A. Namiki, "High-speed catching by multi-vision robot hand," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9131–9136, 2020.

[6] J. Kober, M. Glisson, and M. Mistry, "Playing catch and juggling with a humanoid robot," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pp. 875–881, 2012.

[7] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters, "Trajectory planning for optimal robot catching in real-time," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3719–3726, 2011.

[8] S. Kim, A. Shukla, and A. Billard, "Catching Objects in Flight," *IEEE Transactions on Robotics*, vol. 30, pp. 1049–1065, Oct. 2014. Conference Name: IEEE Transactions on Robotics.

[9] A. Tringali and S. Cocuzza, "Globally Optimal Inverse Kinematics Method for a Redundant Robot Manipulator with Linear and Nonlinear Constraints," *Robotics*, vol. 9, p. 61, Sept. 2020. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

[10] A. ten Pas, M. Gualtieri, K. Saenko, and R. P. Jr., "Grasp pose detection in point clouds," *CoRR*, vol. abs/1706.09911, 2017.