Trajectory planning and control for grasping tumbling targets in zero gravity

Juan Salazar
Dept. of Electrical Engineering and Computer Science
MIT
Cambridge, United States
salazarj@mit.edu

***Abstract*—In the wake of advancing space exploration technologies, there is a need to address the problem of collecting space debris in the form of either obsolete spacecraft or other free-floating objects. Neglecting to do so endangers astronauts performing tasks outside of Earth-orbiting stations and could also jeopardize future missions to observe other bodies in the solar system. In this report, I discuss my implementation of a capture maneuver strategy that enables a simulated 7-DOF robotic arm to grasp a tumbling, free-floating target with a known state. My approach consists of using a trajectory planner that performs piecewise interpolation on desired end-effector poses and a differential inverse kinematics (IK) pseudo-inverse controller that enables the end-effector to move along the generated trajectories. I will present the system performing successful capture maneuvers under varying target conditions and will also benchmark the system against noise on the pose estimate. While my framework produces successful grasps under different conditions, the experiments that I will present here reveal a number of potential improvements that would enhance its robustness against noise, performance in extreme target scenarios, and other aspects.**

***Index Terms*—trajectory planning, grasping, differential inverse kinematics**

I. Introduction

Space manipulation systems (SMS) are designed for and applied in many critical activities within the area of space exploration. Their use allows astronauts stationed at the International Space Station (ISS) to remotely perform tasks that would otherwise be too dangerous or impossible, enables autonomous in-space construction and servicing, and even enables autonomous spacecraft to stabilize tumbling objects (such as other spacecraft or debris) [1]. In this report, I will demonstrate a simulated space debris collection system that uses a Kuka iiwa robotic arm to successfully grasp and stabilize a tumbling cylindrical object with a known initial state estimate. My system consists of 1) a differential IK (inverse kinematics) controller that enables gripper motion along a trajectory and 2) a trajectory planner that dictates where and when the iiwa's end manipulator should move and grasp the object by performing piecewise interpolation on desired gripper poses. In my implementation, the robot will be effectively welded to a ground plane, which means that I will specifically be approximating an SMS that is attached to a far larger free-floating body (such as the Canadarm currently servicing the ISS) that is practically unaffected by the arm's reaction forces. Finally, I will test the system's performance when the object's measured state is subject to noise in order to evaluate this system's compatibility to real-world scenarios.

A. Related Work

In application, SMSes are currently most prevalent on the ISS and are primarily used to assist (or even replace) astronauts with space hardware handling, inspection, and transport. Nevertheless, there is a wealth of ongoing study into equipping spacecraft with SMSes in order to perform capture maneuvers on other spacecraft or debris. One work attempts to solve the problem of generating free-floating manipulator trajectories that intercept the target while leaving the attitude of the manipulator's base unaffected, which is a case that I explicitly mentioned will not be addressed here but still serves as an initial source of inspiration for my trajectory generation algorithm [2]. Because my implementation is limited to a fixed-based manipulator, I also see this strategy as a potential route to further matching it to real-world applications.

More closely related to this project is the work done by Hirouki Nagamatsu et al, in which the the capture strategy and control method for a tumbling satellite are highlighted [3]. While their approach also considered a free-floating base, it served as a great source of inspiration and intuition for the design and improvement of my own strategy.

Although the two works mentioned so far (and several others I encountered in a literature review) consider the free-floating base case, I opted to neglect this case in order to focus on my direct application of the course material. For the purpose of implementing a differential IK controller and a trajectory planner for grasping a free-floating object, I saw that fixing the robot base would still lead to a successful outcome that could still be extended to handle more challenging scenarios.

II. Approach to Performing a Grasp Maneuver

In this section I describe the simulation environment and steps involved in my target capture strategy implementation, which includes the differential IK controller, the trajectory planner, the different target cases I considered, and an incomplete optimization-based IK formulation that could improve the system's performance by considering the limitations of the iiwa7 arm. Note that Fig. 1 serves as a reference diagram

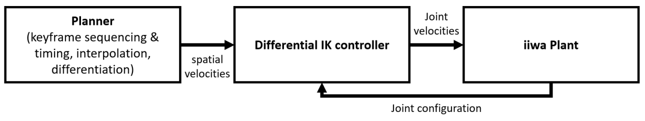
Planner
(keyframe sequencing & timing, interpolation, differentiation)
spatial velocities
Differential IK controller
Joint velocities
iiwa Plant
Joint configuration

Fig. 1. Overview diagram of the system, including the trajectory planner, differential IK controller, and the iiwa robot arm plant.

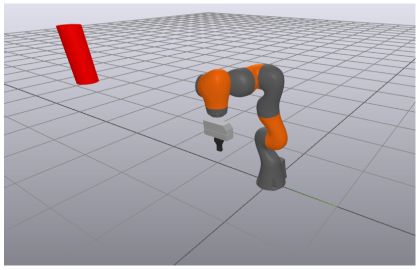

Fig. 2. Screenshot capture of the test environment. The target is the free-floating cylinder in red, Kuka iiwa7 at the center of the grid.

that illustrates the different system components and their interconnections.

A. Test Environment

In my simulation environment, the only two entities present are the robot arm and a uniform red cylinder, which serves as the target. An environment scenario - which we will later categorize as "stationary", "rotating", "translating", or "tumbling" - is defined by the target's initial pose $X^{T_{init}}$ and spatial velocity $V^{T_{init}}$. A screen capture of the simulation environment is presented in Fig. 2.

B. Differential Inverse Kinematics

The forward kinematics of manipulator with joint positions q establishes the following relationship between the end-effector (gripper) pose X^G and q.

$$X^G = f^G_{kin}(q). \tag{1}$$

Applying differentiation yields the following relationship between the gripper pose and the joint positions, which can then be expressed as a relationship between the gripper spatial velocity V^G and the joint velocities v.

$$dX^G = \frac{\partial f^G_{kin}(q)}{\partial q} dq = J^G(q)dq. \tag{2}$$

$$V^G = J^G(q)v. \tag{3}$$

where $J^G(q)$ is known as the gripper frame Jacobian. It follows that in order for the gripper frame to move with a spatial velocity V^G_d, we need to solve for for the joint velocities that are upheld by (3). To do this, we take the pseudo-inverse $[J^G(q)]^+$ to arrive the solution for the joint velocities. This solution provides the joint velocities that produce a gripper spatial velocity that is as close as possible to V^G_d for singular and non-singular joint configurations [4].

$$v = [J^G(q)]^+ V^G_d. \tag{4}$$

Given access to the Jacobian, the current joint configuration encoded in q, and the desired spatial velocity it is possible to command the gripper to move at the requested velocity (as closely as possible). Next, I will describe the planner and the trajectories it generates that ultimately become velocity commands for the pseudo-inverse controller to use an inputs.

C. Planning the Gripper Trajectory

For the problem of planning trajectories, I decided to separate four cases for the target's initial conditions. In the first case, the target is stationary (zero angular and linear velocity). In the second, the target is rotating (non-zero angular velocity, zero linear velocity). In the third, the target is translating (zero angular velocity, non-zero linear velocity). The fourth case is simply a combination of the second and third cases, which I defined for ease of reference as the "tumbling" case. For the sake of simplicity in the demonstrations, non-zero angular velocities are restricted to the x and y axes in the target's body frame and non-zero linear velocities are restricted to the y axis in the target's body frame. Now, I will describe my approach in each of the four target cases. Please note that these steps are all handled by the trajectory planner.

1) Stationary Target: This is the simplest case, however it is also the one that is most involved since it requires us to provide the fundamental planning infrastructure. Most debris capture scenarios are broken into different phases separated most notably by the initial, pre-grasp, grasp, and post-grasp keyframes [1]. To each of these keyframes we assign a desired gripper pose, which yields the keyframe poses $X^{G_{init}}$, $X^{G_{pregrasp}}$, $X^{G_{grasp}}$, and $X^{G_{postgrasp}}$ (all expressed in the world frame). Given a desired ${}^{G_{grasp}}X^T$, and ${}^{G_{grasp}}X^{G_{pregrasp}}$, we can compute each of the keyframe poses. Then, we can assign times (since the start of the simulation) at which the gripper is meant to be at each of these poses, which we will use for interpolation.

In order to represent the trajectories between each of the keyframes, I used a first-order hold piecewise polynomial for interpolating between keyframe positions and spherical linear interpolation for the orientations. The use of spherical linear interpolation is particularly important in order to exploit the power of representating rotations with quaternions, which provide a non-degenerate mapping of all rotations in 3D [4]. Additionally, I specified a command trajectory for the gripper to fully actuate (and grasp the target) between the occurrences of keyframe poses $X^{G_{grasp}}$ and $X^{G_{postgrasp}}$, since grasping and staying in place means that $X^{G_{grasp}}$ is equivalent to $X^{G_{postgrasp}}$.

Finally, given the interpolated pose trajectories we can approximate their derivatives to get the spatial velocity trajectories. Using the Jacobian pseudo-inverse solution specified in (4), we can now command the robot joint velocities for each spatial velocity in the pose trajectory derivative to execute the full maneuver.

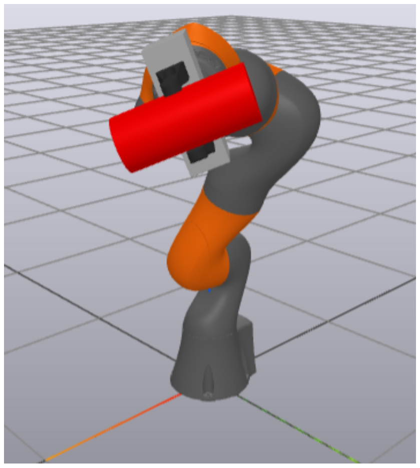

Fig. 3. Screenshot capture of a "proper" grasp.

2) Rotating Target: Given the infrastructure needed to performing capture on a stationary target, adapting it for a rotating target is not a monumental task. Recall that the target is now rotating about a fixed axis, but not translating. This means that at the keyframe "pregrasp", the gripper needs to rotate to intercept the target's at the correct orientation. To achieve this, I modified the planner to substitute the original spatial velocity commands from the stationary target trajectory with a new set of commands that keep the original position trajectory (to enable the linear approach to the target from the "pregrasp" pose) but feed forward a constant gripper angular velocity command needed to intercept the target. In particular I computed $w_{intercept}$, the angular velocity (about the gripper's body y-axis) that allows the gripper close its fingers on the rotating cylinder in a proper orientation at the time of the "postgrasp" keyframe.

$$w_{intercept} = \frac{\theta_0 + w_T \Delta t_{approach}}{\Delta t_{approach}} \tag{5}$$

where w_T is the target angular velocity in the target body frame, θ_0 and $\Delta t_{approach}$ are defined as follows:

$$\theta_0 = w_T t_{pregrasp} \tag{6}$$

$$\Delta t_{approach} = t_{grasp} - t_{pregrasp} \tag{7}$$

With this formulation, the gripper frame is able to intercept the target's frame in the sense of orientation in order to achieve a "proper" (antipodal) grasp, which is illustrated in Fig. 3. In order to neglect the time it takes for the gripper to close its fingers, I command it to fully close fairly quickly (within half a second). This is important for avoiding grasping the target at an "improper" grasp.

3) Translating Target: Once again given the fundamental planner and the target's initial conditions, handling this case simply required predicting the target's future position $p_{intercept}$ after a predefined mount of time, $t_{intercept}$.

$$p_{intercept} = p_{T,init} + v_{T,init} t_{intercept} \tag{8}$$

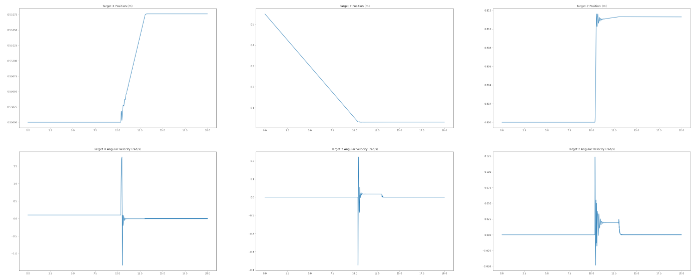

Fig. 4. Target trajectory plots from the tumbling scenario. Initial target angular velocity is 0.1 rad/s and initial linear velocity is 0.05 m/s. For ease of reading, please observe the plots submitted as separate PNGs with the report.

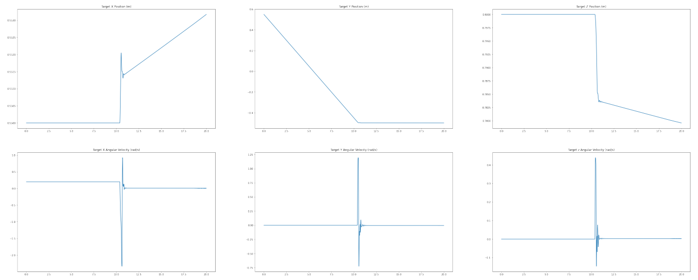

Fig. 5. Target trajectory plots from the tumbling scenario. Initial target angular velocity is 0.2 rad/s and initial linear velocity is 0.1 m/s. For ease of reading, please observe the plots submitted as separate PNGs with the report.

Given the future target pose $X^T_{intercept}$ (its orientation remains the same), the planner treats this pose as the target's new "stationary pose" as if the target had simply spawned at that pose and remained there. Due to the fact that target is actually still moving at the time of the gripper's approach, I also included an intentional delay that slows the approach enough to prevent the fingers colliding with the target before they surround it.

4) Tumbling Target: Now that the stationary, rotating, and translating cases are handled, executing a grasp maneuver on a tumbling target is a matter of combining the modifications in each case into a single planner. We will look at the results from this case later in this report.

III. Results

All simulation, planning, and control routines were written using Drake's Python toolbox. Below I will demonstrate the results from running the system on the "tumbling" case and on the same case with noise added to the initial target pose estimate as well to the initial target angular velocity estimate. In all of these scenarios, the predefined intercept time $t_{intercept}$ was set to 10 seconds. In addition, video links for each of the scenarios summarized below are provided at the end of this section.

A. Tumbling Target

The tumbling scenario was tested under various initial target angular and linear velocities. In this section, I present two successful capture maneuvers performed on a tumbling target.

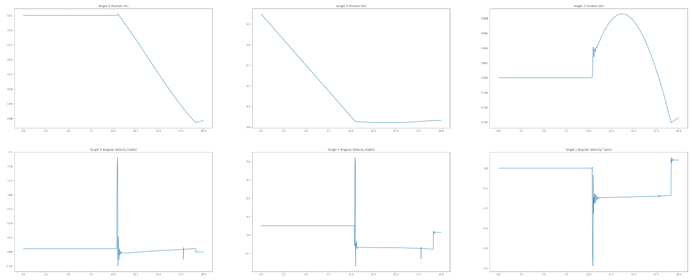

Fig. 6. Target trajectory plots from a successful capture with noisy target pose estimate. Actual initial target angular velocity is 0.05 rad/s and initial linear velocity is 0.05 m/s. For ease of reading, please observe the plots submitted as separate PNGs with the report.

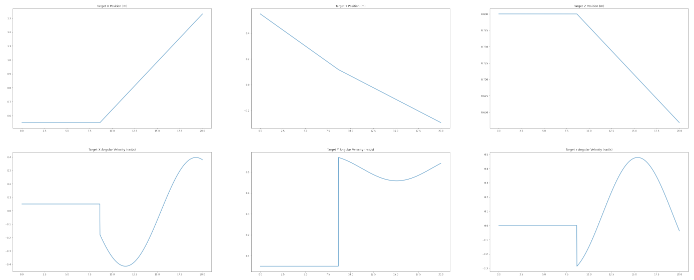

Fig. 7. Target trajectory plots from an unsuccessful capture with noisy target pose estimate. Actual initial target angular velocity is 0.05 rad/s and initial linear velocity is 0.05 m/s. For ease of reading, please observe the plots submitted as separate PNGs with the report.

In the first scenario, in which the target has an initial angular velocity of 0.1 rad/s and a linear velocity of 0.05 m/s, the system was able to predict the target's position and orientation and intercept it. The target's angular velocity profiles shown in Fig. 4 show that it was successfully captured and stabilized around 10 seconds into the simulation. In the body y-axis and body z-axis angular velocity plots there appears to be momentary spikes and non-zero plateaus, which correspond to the gripper rotating the captured target in its fingers for a brief period.

Similarly, the second scenario displayed a successful capture maneuver for a target with initial angular velocity of 0.2 rad/s and a linear velocity of 0.1 m/s. The angular velocity profiles in Fig. 5 also demonstrate that the target was stabilized a brief period after 10 seconds.

B. Noisy Initial Target Pose Estimate

Given a tumbling target, I conducted experiments with random noise on the initial target pose estimate. The noise was sampled from a zero-mean normal distribution with standard deviation 0.001 and was added to the actual initial target position to generate a 'noisy' position estimate. Fig. 6 and Fig. 7 illustrate the target position and angular velocity profiles for two unique noise samples, where the first experiment was successful and the second unsuccessful. The successful capture demonstrates a target angular velocity profile indicative of a stabilized cylinder, with the caveat that after capture the z-axis angular velocity was non-zero due to the target rotating a bit even when pinched in between the gripper fingers. The unsuccessful capture demonstrates fluctuating angular velocities for much time after the intended capture time. In this case, the target was grasped for a brief moment before slipping through the fingers and being ejected at some undetermined spatial velocity.

C. Noisy Target Angular Velocity

Given a tumbling target, I conducted experiments with random noise on the measured target angular velocity. As in the pose estimate experiment, the noise was sampled from a zero-mean normal distribution with standard deviation 0.001. The resulting target trajectory from one experiment is illustrated in Fig. 8, in which the target is successfully grasped despite

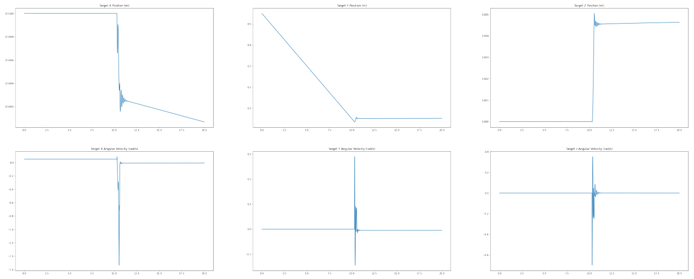

Fig. 8. Target trajectory plots from the tumbling scenario with noisy target angular velocity estimate. Actual initial target angular velocity is 0.05 rad/s and initial linear velocity is 0.05 m/s. For ease of reading, please observe the plots submitted as separate PNGs with the report.

the perturbation. Like in the previous trial, this experiment was repeated several times to gauge the frequency of failure in the maneuver. It was clear that small deviations in the target angular velocity estimate proved to have no affect on the success, whereas larger deviations would cause target trajectories similar to that seen in Fig 7.

D. Videos

Below are clickable hyperlinks to videos of the results summarized above.

1) Tumbling scenario, 0.1 rad/s and 0.05 m/s: https://youtu.be/auAhlANvBxI
2) Tumbling scenario, 0.2 rad/s and 0.1 m/s: https://youtu.be/hcL-fXBQbfo
3) Noisy initial target pose estimate (successful), 0.05 rad/s and 0.05 m/s: https://youtu.be/itoxFBvtz2I
4) Noisy initial target pose estimate (unsuccessful), 0.05 rad/s and 0.05 m/s: https://youtu.be/esyGf3kmGHg
5) Noisy initial target angular velocity estimate, 0.05 rad/s and 0.05 m/s: https://youtu.be/IQpxBokT9CE

E. Code

This is a clickable hyperlink to the GitHub repo where the code is stored (in a Python notebook): https://github.com/juansala/manipulation_project

IV. Discussion

Looking at only at the noise-free experiments, the trajectory planner and pseudo-inverse controller were enough to allow the robot to grasp and stabilize the target under certain conditions. Some of the experiments I conducted also revealed the limitations of this approach, although they were not included in this report. These failed experiments appeared to imply a limit for the max target rotational velocity as well as a max target linear velocity. I believe that these two limits are connected to the iiwa robot's limited maximum joint velocities and its allowed work space, respectively. For instance, in some cases the target linear velocity is high enough that the gripper capture pose computed by the planner is far outside of the robot's work space, causing the simulation to crash. Admittedly, these are well limitations of relying on the pseudo-inverse IK controller to command the joint velocities, since it imposes no constraints on those velocities and on the configuration space. In order to incorporate such constraints, an optimization-based approach to differential kinematics could be adopted. This generally appears in the form of a quadratic program with a cost on the square-error gripper spatial velocity and constraints on the joint velocities and joint configurations. Another apparent limitation is the system's sensitivity to random noise. To an extent, this was expected due to the single-procedure nature of the system; it only computes the trajectory to the target once at initialization. If this methodology was kept, then applying filtering techniques on real-world sensor data (e.g. depth camera, IR, etc.) such as Kalman filtering and sensor fusion would an immediate help. Of course, even with the noise the system could also be adapted to performance constant kinematic trajectory optimizations after every N sensor samples. Real-world systems continually receive and record sensor measurements and are designed to be adaptive beyond single-procedure methods like the one presented in this report.

V. Conclusion

Overall, the aim of this project was to design a system that can plan and execute a grasp maneuver on a tumbling target and to benchmark the robustness of this system against noisy state estimates of the target. In my experiments I demonstrated that my system is capable of achieving consistent capture maneuvers when free of random noise in the target state, and that it is capable of succeeding under noisy scenarios as well. More importantly, my experiments revealed the limitations in my system, which have provided me with several ideas for improvements. In terms of possible extensions to this project, I can identify multiple areas that could enhance the system's performance and compatibility to real-world scenarios. Some of these I already mentioned, including applying filtering techniques and implementing an optimization-based controller, however even with these additions the system would continue to be limited against real-world needs. In particular, there is a need to determine the allowable friction coefficients for the gripper fingers that can ensure that the captured target remains in stationary grasp. Conducting this study would also reveal the kinds of materials this robot would be capable of grasping, which would then help us gauge its effectiveness against real space debris. Regarding the workspace limit on the iiwa arm, the only way to directly overcome the workspace limitation would be either to use a different arm or to design a tool for the iiwa arm to use as an extended gripper. I would find it interesting to see how the iiwa could make use of a diverse toolset that it can manipulate with its fingers to perform captures on far away objects or perform other tasks.

References

[1] E. Papadopoulos, F. Aghili, O. Ma, R. Lampariello, ”Robotic Manipulation and Capture in Space: A Survey”, Frontiers in Robotics and AI, 2021, pp. 228, doi: 10.3389/frobt.2021.686723

[2] P. Piersigilli, I. Sharf, A.K. Misra, ”Reactionless capture of a satellite by a two degree-of-freedom manipulator”, Acta Astronautica, 2010, pp. 183-192, doi: 10.1016/j.actaastro.2009.05.015

[3] H. Nagamatsu, T. Kubota and I. Nakatani, ”Capture strategy for retrieval of a tumbling satellite by a space robotic manipulator,” Proceedings of IEEE International Conference on Robotics and Automation, 1996, pp. 70-75 vol.1, doi: 10.1109/ROBOT.1996.503575.

[4] Russ Tedrake. Manipulation: Perception, Planning, and Control (Course Notes for MIT 6.834). Downloaded on December 9, 2021 from https://manipulation.csail.mit.edu/