Agile Robotic Systems for Racket Sports

Kin Man Lee
College of Computing
Georgia Institute of Technology
klee863@gatech.edu

1 Introduction

Sports such as lawn tennis and table tennis are environments that demand high-speed athletic behaviors, requiring expertise in various areas such as computer vision, optimal control, motion planning, and machine learning to create high-performance agile robotic systems in these domains. As the popularity of live sports continues to grow, with an estimated 57.5 million people watching live sports at least once per month in the U.S. alone[1], the domain of sports robotics has the potential to attract public attention and further promote STEM and robotics. However, one of the most significant challenges in developing these systems is the need to meet real-time requirements, which are essential for successfully completing the task at hand, such as striking a ball with the precision and timing for a return.

This paper serves as a technical report with system development details that were not included in [1] and [2], providing additional insight on the design choices that were made. This paper further documents efforts towards improvements in these systems that are not yet shown in an official publication. In Section 2, a mobile, agile manipulator system for playing wheelchair tennis named ESTHER is discussed. Further details about the stroke planning that were not included in the original paper are added here. In Section 3, a robotic system for table tennis tennis is discussed, along with details on the current developments on a VR table tennis simulation.

2 Robotic System for Wheelchair Tennis

ESTHER (Experimental Sport Tennis wHEelchair Robot) [1] is an autonomous mobile manipulator system designed for playing regulation wheelchair tennis, named after one of the greatest wheelchair tennis players of all time, Esther Vergeer. The core components of the system consists of (1) a decentralized vision-perception system with six cameras surveying the court, (2) an electrified wheelchair serving as the mobile base and additionally houses the system electronics, and (3) a high-speed seven degree-of-freedom (DOF) Barrett WAM arm with a tennis racket end-effector to strike the ball. The remaining subsections on ESTHER are focused on the technical details of my contributions towards the motion planning of the WAM along with a brief discussion on improvements that could be made.

2.1 Ball Interception

The initial stage of the arm motion planning was to solve ball interception through MoveIt [3], a motion planning framework for the Robot Operating System (ROS) [4]. The proposed method was a virtual hitting plane approach, where the ball interception point is the intersection of a predetermined virtual plane near the arm and the predicted rollout of the ball trajectory. As the interception point is in Cartesian task space, inverse kinematics (IK) was utilized to transform the coordinate into the robot joint space for a joint configuration where the racket can intercept the ball. Experiments involved a variety of numerical IK solvers (KDL [5], IKFast [6], Trac-IK [7]) and trajectory planning libraries (OMPL [8], Pilz trajectory planner [9]) to generate a trajectory that could satisfy the real-time constraints of the task. A combination of Trac-IK and Pilz trajectory planner produced the best

[1] Statistic acquired from https://www.statista.com/statistics/1127341/live-sport-viewership/

results with $\approx 80\%$ success rate of interception on the real robot in a lab setting. Despite showing reasonable performance in the ball interception task, MoveIt-based planning was ultimately not sufficiently fast, nor strong enough to generate a robot stroke from the WAM that could return a ball over the net on a regulation tennis court.

2.2 Trajectory Profiler

Due to the deficiencies of MoveIt, a custom trajectory profiler was designed and implemented instead to generate ground stroke trajectories with sufficient force and accurate timing to return a ball. The virtual hitting plane for locating the interception point is retained, providing the intersection height as a parameter to the profiler. The trajectory profiler leverages a trapezoidal velocity profile to produce a joint trajectory that maximizes the velocities of each joint at the hit time given an initial, hit, and final joint configuration. The start time of the trajectory is shifted according to the estimated time of interception, and self-collision checking is performed at each timestep. To satisfy the strict timing requirements of the swing, the trajectory profiler is implemented completely in C++ following the ROS ActionClient and ActionServer paradigm to allow preemption of existing requests. By doing so, rapid replanning and execution of trajectories is possible, enabling trajectories based on new ball prediction estimates to replace existing trajectories mid-execution. The trajectory profiler resides in an ActionServer that actively listens to requests made from an ActionClient that supplies the three required joint configurations. Trajectory generation with this implementation typically requires less than 10 milliseconds, where self-collision checks incurs the most cost as trajectory generation executes in the order of microseconds without the safety checks.

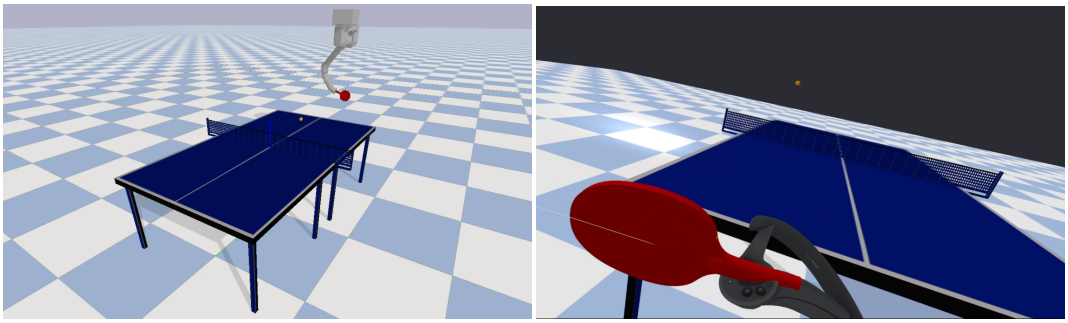
2.3 Discussion and Future Work

The current iteration of the trajectory profiler is fairly fast and robust. It is also designed to be swing-type agnostic and can handle generating a variety of motion as long as it is provided the three required initial, hit, and final joint configurations. The main improvement to be made is to the implementation of a mechanism that allows returning the ball at a desired velocity. Currently, the profiler maximizes the joint velocities at hit time and offers no precision on where the ball is returned. Another improvement that could be made is on the self-collision checking, which is performed by MoveIt by providing the joint configuration at each timestep as an input. A custom collision checker could instead be developed to eliminate the runtime bottleneck of the profiler. Although there are improvements that could be made to the profiler, focusing on the wheelchair control and vision-perception subsystems in the short term would provide much more performance benefits at this point in time.

For potential future work in the long term that are swing related, a better method for providing demonstrations to the WAM on the wheelchair is an avenue for investigation. With kinesthetic teaching, it is tedious and difficult to provide near-optimal demonstrations with the speed and power required to return a tennis ball. As the wheelchair control aspects begins to get better, approaching the entire system as a whole-body control problem and conducting a full literature review on this topic can help provide insights on a strategy to couple the arm and wheelchair control for increased fluidity in movements to strike the ball.

3 Robotic System for Table Tennis

In [2], a collaborative table tennis system was developed to enable the study of latent factors on human behavior that can affect the overall performance of mixed human-robot teams. My core contribution to this system is a safety module that prevents the robot from colliding with a human collaborator through camera-based human pose tracking. A virtual robot workspace is created such that any detected body part entering this workspace immediately terminates all robot motion. The implementation of the safety module follows a similar ActionServer–ActionClient paradigm described in Section 2.2, where a thread is dedicated to the server during execution to ensure the safety module is always active to handle stop requests to the robot.



(a) Table Tennis PyBullet Environment (b) First-person view of demonstrator in VR

Figure 1: PyBullet Environment

Although the collaborative system was designed to be safe, it has the drawback of being overly conservative in order to prioritize collision avoidance. This limitation prohibits the study of tighter human-robot interactions as the current implementation effectively places a virtual cage on the robot. One approach to allow complete freedom of interaction without safety risks is through simulation with virtual reality (VR). By replicating the real-world environment in VR, both the human and the robot have the freedom to perform any action they desire without risks of collisions. Another benefit that VR offers is the ability to quickly collect high quality human data because of the precise tracking capabilities of the headset and controllers. This can be leveraged to collect human demonstration data for training imitation learning algorithms for robot strokes in table tennis. In the remaining subsections on table tennis, I will discuss my work towards building the VR simulation, and methods to tackle the correspondence problem [10, 11] when converting human swing trajectories into robot joint trajectories from data collected in the simulation.

3.1 Simulation Environment

The simulation environment was set up in PyBullet [12] to mirror the same real-world table tennis environment that was shown in [2] and runs at 240 Hz. The simulation consists of a regulation-sized table tennis table with an overhanging Barrett WAM at one end of the table. Balls can be launched from the opposite end to mimic the behavior of a ball launcher. The ball dynamics includes a force on gravity, bounce restitution, and contact friction with the table and racket. An image of the PyBullet environment is shown in Fig. 1a. PyBullet has native virtual reality support that enables the use of any VR headset compatible with the OpenVR interface, with the caveat that the underlying Bullet physics engine must be running on the Windows ecosystem. The simulator has been tested exclusively on the Valve Index headset and controllers thus far, but should function similarly with new button mappings to the controller if a different set of hardware were to be used.

While in VR, a user can move the racket through controller movements as the racket is constrained to the same position and orientation of the controller. Fig. 1b shows the first-person view of the user in VR with a racket. To improve immersion in the VR environment, haptic feedback was triggered in the controller whenever contact was detected between the ball and racket. Since the PyBullet API only supports the reading of controller and headset data and cannot send commands, haptic feedback functionality was implemented directly on the physics server.

3.2 Data Collection

To record data from the simulation, the trigger on the Index controller is held down to start the logging process and also launch a ball at the same time. Releasing the trigger will end data logging of the current sample and save the racket trajectory to a new JSON file. Only samples with a ball hit are saved. In each sample, the Cartesian coordinates of the racket positions ($\mathbf{x}$), linear velocities ($\dot{\mathbf{x}}$), orientations (ω), and angular velocities ($\dot{\omega}$) in the world frame are extracted based on the controller position. The racket data, along with the timestamps of each timestep in the demonstrated swing

trajectory is recorded. Once a ball hit is detected, the hit timestamp is also recorded to allow simple retrieval of the racket data at hit time. Three different types of swing data were collected to assess the robustness of the mapped swings at performing the table tennis task. The motions described as below:

- *Push*: This is the simpliest motion where the racket is "pushed" directly opposite the direction where the ball is traveling with the highest velocity. Hitting the ball with this motion is the easiest as the swing trajectory has high overlap with the ball's trajectory.
- *Lob*: This is an upward motion with the racket normal almost parallel to ground. On contact, the ball is lifted on a high arcing trajectory to the opponent court. Hitting the ball is harder with lob as there is less overlap of trajectories.
- *Top Spin*: This is an upward motion with the racket normal almost orthogonal to the ground, mimicking the backhand topspin. Hitting the ball with this motion is the most difficult as the swing timing needs to be exact since there is little overlap between the racket and ball trajectories.

3.3 Motion Mapping

The mapping of the racket trajectories to robot joint trajectories can be denoted with the state transformation in (1), where the dimensionality of the racket task space is $\mathbb{R}^{12}$ and the dimensionality of the robot joint state space is $\mathbb{R}^{14}$. Note that the dimensionality of the joint space is greater than the task space which indicates redundancy of solutions in the mapping.

$$\begin{bmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \\ \boldsymbol{\omega} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} \tag{1}$$

Two approaches were considered to perform this transformation:

Inverse kinematics: The full racket trajectories in task space are directly transformed into robot joint state trajectories by performing the selectively damped least squares (SDLS) inverse kinematics method [13] at each timestep of the trajectory. This method allows the use of motion in the nullspace to satisfy secondary targets – in this instance, satisfying the joint limits of the WAM. A seed state with the joint configuration $\mathbf{q}$ of the current timestep is provided to the subsequent timestep to guide the solution to a similar joint configuration. With this approach, joint velocities $\dot{\mathbf{q}}$ can be acquired from numerical differentiation since $\mathbf{q}$ is solved at each timestep.

Trajectory optimization: Three key frames in the racket trajectories inform the constraints to an optimal control problem of satisfying the table tennis task to hit and return the ball. The three key frames are the start, hit, and final frame of the racket trajectory and occur at time t_0, t_h, t_f, respectively. By constraining the full joint state $[\mathbf{q}\ \dot{\mathbf{q}}]^T$ at the key frames to match the demonstrated swing, model-based trajectory optimization can be utilized to generate joint trajectories that are feasible for the robot and potentially remove suboptimally within the human demonstrated swings. The robot joint state at the key frames is acquired in the same fashion as the inverse kinematics approach.

The trajectory optimization algorithm that was utilized to generate the joint trajectories is the Hermite-Simpson collocation method [14]. This is a form of direct collocation where the state trajectory is represented by a cubic-Hermite spline, the input trajectory with a first-order hold, and the dynamics are satisfied using Simpson's rule for numerical integration in the collocation constraints. The implementation was performed in Drake [15], a toolbox for model-based optimization and simulation. The physical properties of the WAM are imported into Drake through a URDF file to model the system dynamics. The control problem is formulated as follows to minimize unnecessary movement and control:

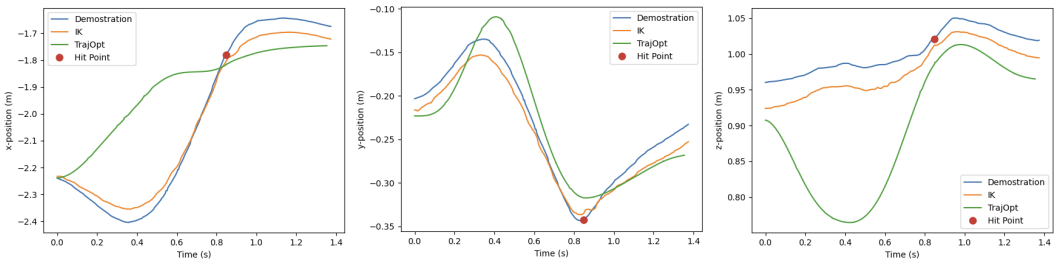


Figure 2: End-effector trajectories (x,y,z) of IK and TrajOpt compared to demonstrated trajectory of a *Push* motion. The hit point denotes the time at which the ball was hit.

$$\min_{\dot{\mathbf{q}},\mathbf{u}} \int_0^{t_f} \dot{\mathbf{q}}(t)^T Q \dot{\mathbf{q}}(t) + \mathbf{u}(t)^T R \mathbf{u}(t) dt \tag{2}$$

$$\dot{\mathbf{q}} = f(\mathbf{q}, \mathbf{u}) \tag{3}$$

$$\begin{bmatrix}\mathbf{q}_0 & \dot{\mathbf{q}}_0\end{bmatrix}^T = \begin{bmatrix}\mathbf{q}(t_0) & \dot{\mathbf{q}}(t_0)\end{bmatrix}^T \tag{4}$$

$$\begin{bmatrix}\mathbf{q}_h & \dot{\mathbf{q}}_h\end{bmatrix}^T = \begin{bmatrix}\mathbf{q}(t_h) & \dot{\mathbf{q}}(t_h)\end{bmatrix}^T \tag{5}$$

$$\begin{bmatrix}\mathbf{q}_f & \dot{\mathbf{q}}_f\end{bmatrix}^T = \begin{bmatrix}\mathbf{q}(t_f) & \dot{\mathbf{q}}(t_f)\end{bmatrix}^T \tag{6}$$

$$\mathbf{q}_{min} \leq \mathbf{q} \leq \mathbf{q}_{max} \tag{7}$$

$$\mathbf{u}_{min} \leq \mathbf{u} \leq \mathbf{u}_{max} \tag{8}$$

where (2) is the running cost function, (3) represents the dynamical constraints of the system, (4)-(6) are constraints on the key points to ensure the generated trajectory satisties the requirements of the table tennis task, and (7), (8) are constraints for the joint and torque limits, respectively. The problem is set up as a mathematical program in Drake and a interior point nonlinear solver (IPOPT [16]) to find a solution.

3.4 Preliminary Results

To assess the performance of each method, the joint trajectories acquired from IK and collocation approaches were replayed on the simulation and the end-effector trajectory in task space was recorded. These position trajectories were then compared to the demonstrated swing data, which is considered the ground truth. A comparison plot of a *Push* motion on the x, y, z coordinates can be seen in Fig. 2. The inverse kinematics approach follows the demonstrated trajectory closely which is expected. IK was quite robust at creating smooth trajectories that were consistently able to return the balls over the net across all three swing types. On the other hand, the collocation method produced trajectories that can deviate greatly from the demonstration, however, the error at hit time is very low, indicating the hit time constraints are effective. Note that there is a large dip in the z-axis just prior to striking the ball. This was observed in all of the solutions from trajectory optimization and is due to the fact that gravity compensation is not performed in the optimization. The IPOPT solver takes around 5 seconds to solve for a trajectory on average, but could take up to a maximum time of around 80 seconds when it is successful.

The state and control trajectories of a *Push* motion that were directly acquired from the collocation method are shown in Fig. 3. As expected of the Hermite-Simpson collocation method, the state (joint angles and joint velocity) are continuous cubic piecewise polynomials where the control (desired joint torque) is a trajectory constructed from first-order piecewise polynomials. For each of the trajectories, the transition between the green and red segment is where the end-effector makes contact with the ball. Note that there is always a large change in each of the curves just before the hit point in order to satisfy the hit constraints that were set in (5).

In Fig. 4, image sequences of the simulation from all three types of swings are shown for the collocation method. The differences between how Drake and PyBullet handles system dynamics makes it

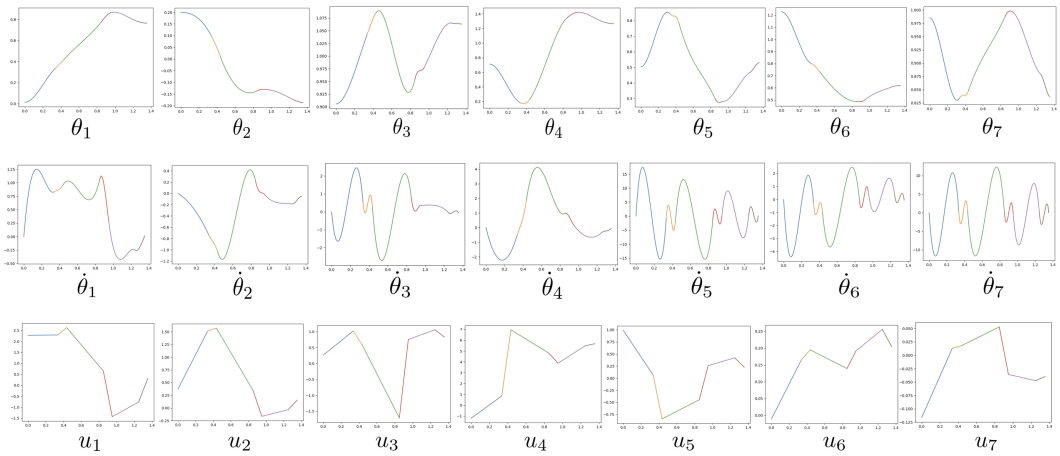
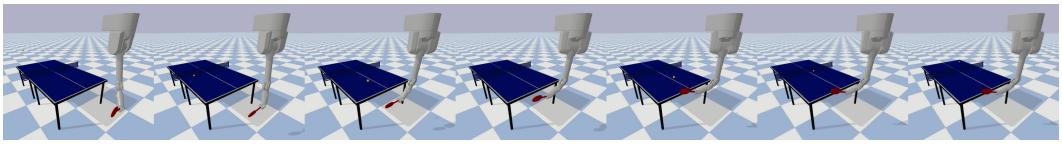
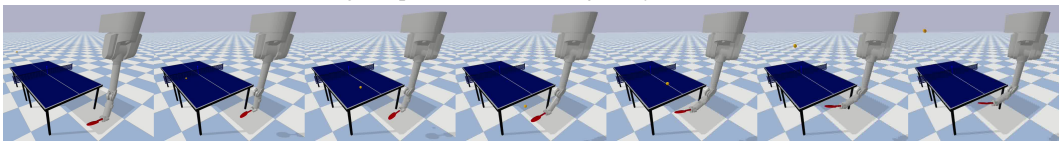


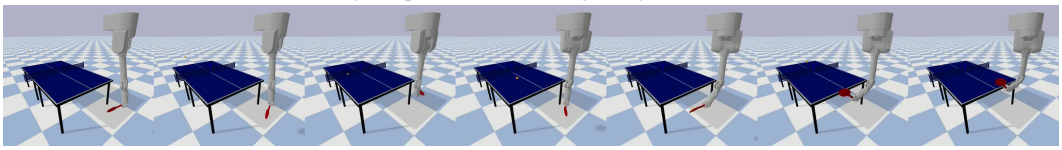
Figure 3: Generated trajectories of a push motion for all seven joints from Hermite-Simpson collocation.



(a) Image sequence of a *Push* trajectory in simulation



(b) Image sequence of a *Lob* trajectory in simulation



(c) Image sequence of a *Top Spin* trajectory in simulation

Figure 4: Collocation trajectories replayed in PyBullet simulation

difficult to directly apply torque control from the Drake solutions in PyBullet. Position control with a high-gain PD controller was used to control the robot instead. In all three of the motions, a swing-up motion can be observed just prior to hitting the ball due to the lack of gravity compensation. If the solver returns a solution, the robot always hits the ball, but a return to the opponent's court is not guaranteed.

3.5 Discussion and Future Work

VR PyBullet Simulation: The table tennis simulation is in a good state for quickly collecting human demonstrations of swings. Over a 4 minute period, 60 samples evenly split across the three swing types were collected. One major improvement that needs to be made is factoring in air drag and the Magnus effect in the ball dynamics for spin. For upcoming future work, ZED2 cameras can be incorporated into the system to acquire pose data of the VR user interacting with the simulation, similar to how pose data was acquired in [2]. Additional work that could utilize the simulator is to begin exploring learning-based controllers and tackling the sim-to-real problem to get these controllers working on the real robot.

Motion Mapping: Based on the preliminary results, a simple IK approach performed much better at mapping the racket motion than the collocation method. However, the IK approach does not factor in the systems dynamics. While the results look good in the simulation, real-world testing is needed to confirm that it performs similarly on the actual robot. As for the collocation method, the suboptimal trajectories that it generates can largely be attributed to the lack of gravity compensation. Although all of the solutions it provided had some extraneous swing-up motion involved, the executed trajectories were always able to hit the ball. One major problem with the current implementation is how frequently the solver fails to find a solution. Out of 60 demonstrations that were collected, the solver was unable to find a solution for 18 of the samples. Resolving these issues could greatly improve the performance of the collocation method, however, Drake has a very steep learning curve and is likely not worth the effort given how well the inverse kinematics approach already performs. The immediate next step is to test how well the inverse kinematics mapping performs on the hardware and assess whether or not it can provide real-time mapping.

4 Conclusion

In summary, this paper has discussed my contributions to the development of agile systems for table tennis and lawn tennis. Specifically, the work presented here focuses on the wheelchair tennis robot, ESTHER, and the ball interception work leading up to the journal publication, as well as the trajectory profiler currrently used for WAM motion planning. Additionally, the ongoing work on the table tennis VR environment and stroke mapping has been highlighted. The immediate next steps for each project have been identified, along with potential long-term directions for future research. Notably, many of the methods developed for one system can be applied to the other, such as using motion mapping techniques for human demonstrations in table tennis and applying it to ESTHER's tennis swings on the court. As these systems continue to mature, they will enable more intricate studies on human factors and research on human-robot teaming strategies.

References

[1] Z. Zaidi, D. Martin, N. Belles, V. Zakharov, A. Krishna, K. M. Lee, P. Wagstaff, S. Naik, M. Sklar, S. Choi, Y. Kakehi, R. Patil, D. Mallemadugula, F. Pesce, P. Wilson, W. Hom, M. Diamond, B. Zhao, N. Moorman, R. Paleja, L. Chen, E. Seraj, and M. Gombolay. Athletic mobile manipulator system for robotic wheelchair tennis. *IEEE Robotics and Automation Letters*, 8(4):2245–2252, 2023. doi:10.1109/LRA.2023.3249401.

[2] K. M. Lee, A. Krishna, Z. Zaidi, R. Paleja, L. Chen, E. Hedlund-Botti, M. Schrum, and M. Gombolay. The effect of robot skill level and communication in rapid, proximate human-robot collaboration. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '23, page 261–270, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450399647. doi:10.1145/3568162.3577002. URL https://doi.org/10.1145/3568162.3577002.

[3] I. A. Sucan and S. Chitta. Moveit!, 2013.

[4] Stanford Artificial Intelligence Laboratory et al. Robotic operating system. URL https://www.ros.org.

[5] Orocos kinematics and dynamics. URL https://www.orocos.org/kdl.html. Accessed: 2023-05-01.

[6] R. Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010. URL http://www.programmingvision.com/rosen_diankov_thesis.pdf.

[7] P. Beeson and B. Ames. Trac-ik: An open-source library for improved solving of generic inverse kinematics. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 928–935, 2015. doi:10.1109/HUMANOIDS.2015.7363472.

[8] I. A. Şucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. doi:10.1109/MRA.2012.2205651. https://ompl.kavrakilab.org.

[9] Pilz industrial motion: Industrial trajectory generation for moveit! URL https://github.com/PilzDE/pilz_industrial_motion. Accessed: 2023-05-01.

[10] K. D. C. L. Nehaniv. Like me?- measures of correspondence and imitation. *Cybernetics and Systems*, 32:11 – 51, 2001.

[11] C. L. Nehaniv and K. Dautenhahn. *The Correspondence Problem*, page 41–61. MIT Press, Cambridge, MA, USA, 2002. ISBN 0262042037.

[12] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.

[13] S. R. Buss and J.-S. Kim. Selectively Damped Least Squares for Inverse Kinematics. *Journal of Graphics Tools*, 10(3):37–49, Jan. 2005. ISSN 1086-7651. doi:10.1080/2151237X.2005.10129202. URL https://www.tandfonline.com/doi/full/10.1080/2151237X.2005.10129202.

[14] C. Hargraves and S. Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics*, 10(4):338–342, July 1987. ISSN 0731-5090, 1533-3884. doi:10.2514/3.20223. URL https://arc.aiaa.org/doi/10.2514/3.20223.

[15] Model-based design and verification for robotics. URL https://drake.mit.edu/. Accessed: 2023-05-01.

[16] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar. 2006. ISSN 1436-4646. doi:10.1007/s10107-004-0559-y. URL https://doi.org/10.1007/s10107-004-0559-y.