My 4YP

Robert Shi

May 7, 2020

1 Cover and Title Page

Motion Planning for A Driveby Pick-And-Place Problem.

2 Acknowledgements

I would like to express my deep gratitude to my project supervisor, Dr. Ioannis Havoutis, for his guidance and patience throughout the course of this project. I am truly grateful for the opportunity to learn about robotics and gain hands-on experience as an undergraduate student.

I would also like to express my appreciation to Dr. Wolfgang Merkt for setting up the driveby problem in EXOTica and for his valuable advice and assistance.

Finally, I would like to thank Mark Finean and Charlie Street for their time and support in helping me understand and operate the Toyota Human Support Robot.

3 Abstract

To do later

Contents

========== Everything above this point does not count towards the 50 page limit. Format restrictions: minimum 11pt font, 20mm margins, 8mm line spacing.==========

4 Introduction (2-3 pages)

4.1 Purpose

Robocup@Home is an international robotics competition that aims to pose a standard problem in order to further development of robotic technologies in the area of "service and assistance with high relevance for future personal domestic applications"[robocup reference]. Specifically, the Domestic Standard Platform League (DSPL) of Robocup@Home aims to "assist humans in a domestic environment, paying special attention to elderly people and people suffering of illness or disability." [robocup pdf reference]. Focus areas of the competition include "Human-Robot-Interaction and Cooperation" as well as "object manipulation"[robocup reference]. A common activity for an autonomous robot to perform is object retrieval or placement, more commonly known as a pick-and-place problem in robotics. This fourth year project centers on implementing a drive-by pick-and-place algorithm on a Toyota Human Support Robot (HSR), which is the designated robot for the Robocup@Home DSPL.

4.2 Problem Description

The goal of this project is for the HSR to perform a driveby pick-and-place action. This requires the HSR to continuously drive by a table, following a predefined omnibase trajectory, and pick up a cylindrical object on top of the table. The HSR then brings the object to the target placing location and places it. It is assumed that the base trajectory is given prior to planning and the free configuration space is known. [insert image of driveby pick and place]

4.3 Robot Description

The HSR is one of the "Toyota Partner Robots" which aims to "provide life support and to assist with independent living in the home for handicapped people"[hsr doc]. The HSR is subdivided into 4 segments: the base, the torso, the arm, and the head.

4.3.1 Base

The HSR has an omnidirectional wheeled base with two rear drive wheels and two passive casters for locomotion. Figure 2. The base is cylindrical in shape and houses a bumper sensor and laser

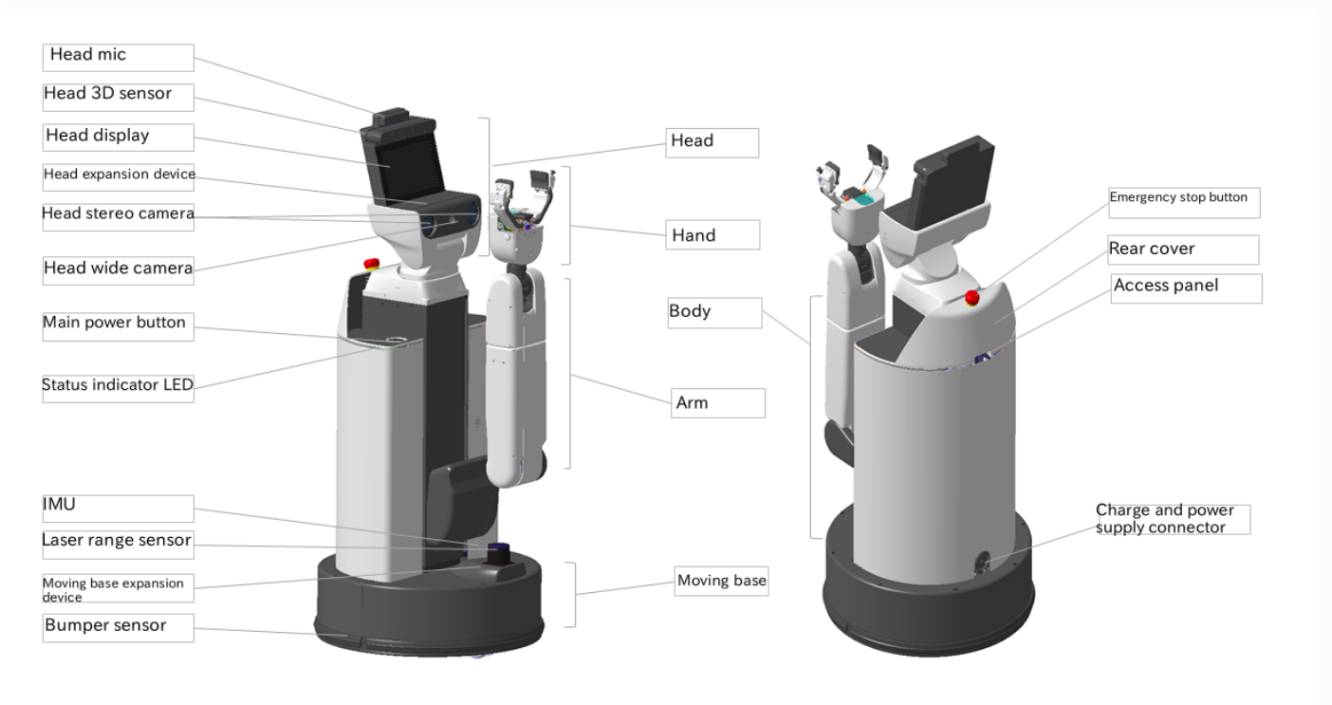


Figure 1: [Insert Reference to HSR Docs]. Overview of HSR Segments and Components.

range sensor to aid in obstacle detection. A magnetic sensor allows users to mark world boundaries using strips of magnetic tape.

[The omnibase of the HSR uses differential steering to turn and move forwards and backwards.][Need to go through: A mobile platform with a dual-wheel caster-drive mechanism for holonomic and omnidirectional mobile robots reference. similar to differential drive but some key differences. They somehow claim holonomicity. paper is in japanese though.] A continuous base roll joint in the HSR omnibase segment allows the body of the HSR to rotate freely of the base Figure 3. This feature greatly aids in the execution of a driveby pick-and-place trajectory.

4.3.2 Body and Head

The HSR has a height range between 1.0m and 1.3m due to a telescoping body. The head contains a wide angle camera, stereoscopic camera, and a three-dimensional ASUS Xtion depth sensor for visual input and object recognition. In addition, there is a microphone and screen display for human input. The head can pan and tilt.

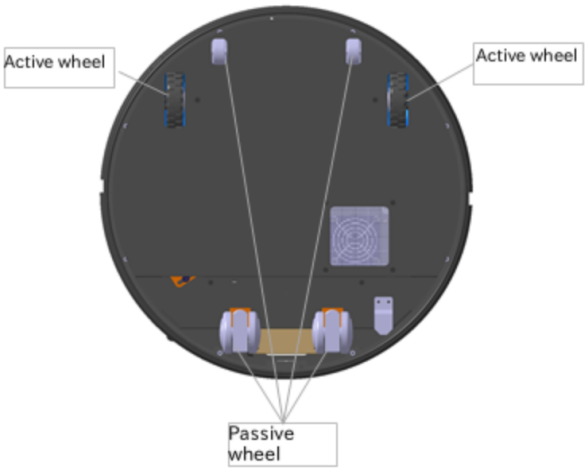


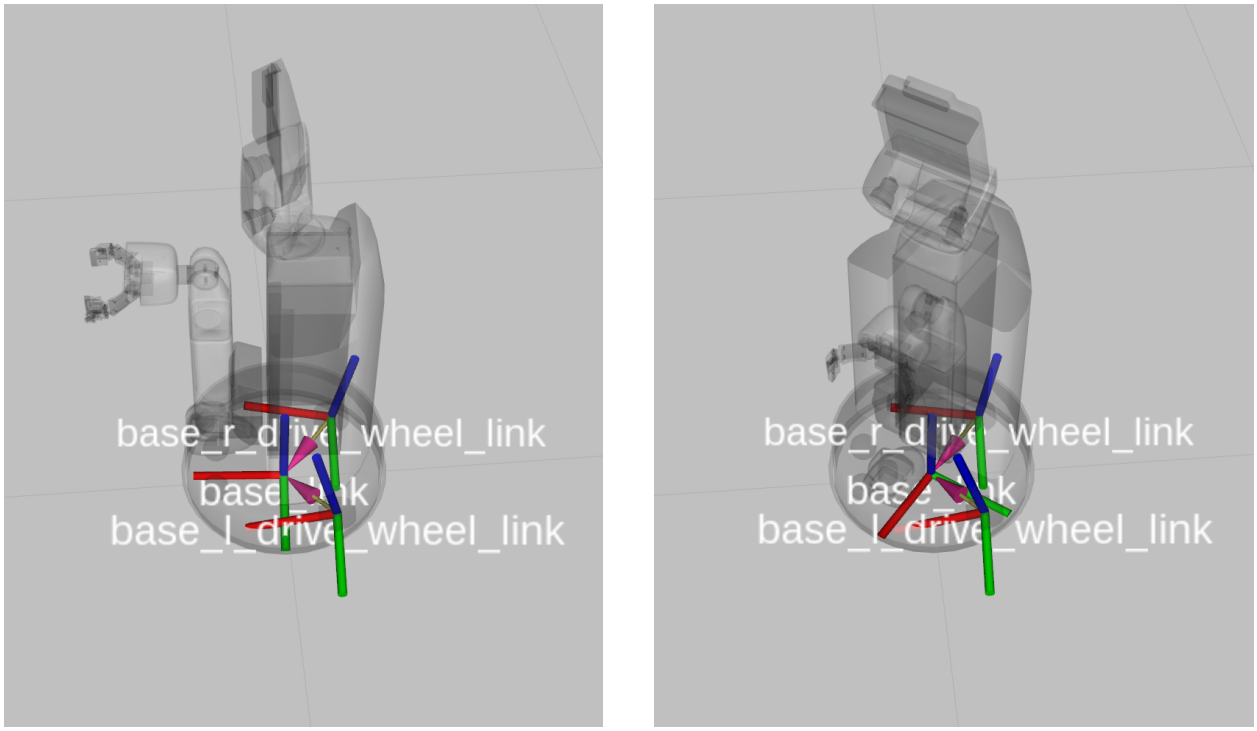
Figure 2: Omnibase of HSR

4.3.3 Arm and Hand

The arm is attached to the body by a lifting shoulder joint that vertically telescopes the head and arm. The arm, including the lift joint, has 5 degrees of freedom and a length of 0.60 meters. The end effector is a two-finger rubber gripper (hand) with a camera, force sensor, and suction pad. The gripper con open to a maximum of 13.5cm, apply a maximum of 40N of force, and open and close within 0.4 seconds.

4.3.4 Development

The HSR has software built on top of the Robot Operating System (ROS), which is an open-source framework used to write robot software. [ROS reference]. The HSR software includes a Python Interface and ROS interface. Toyota introduces an interactive shell based on iPython as an option for users to interface with the HSR without the steep learning curve associated with ROS. This shell interface allows for whole body motion, as well as individual subsystems by specifiying goal points, goal poses, and relative movement. Further, the HSR has a graphical web-interface for usage by non-technical operators that allows for basic teleoperation.



(a) (b)

Figure 3: The base roll link allows the HSR to rotate its torso without moving the omnibase. Note that the orientation of base_link changes between (a) and (b), while the drive wheels remain stationary.

5 Planning Algorithms

5.1 Overview

Navigation for autonomous mobile robots relies on two primary systems: a robot state estimator and a motion planner. The base HSR has implementations of both systems. For the purpose of accomplishing a driveby pick-and-place task, modifications will be made to the motion planning algorithm. The motion planning algorithm utilized in this project concatenates paths planned using Rapidly-Exploring Random trees and a non-linear programming solver called Approximate Inference Control (AICO).

5.2 Rapidly-Exploring Random Trees

Rapidly-Exploring Random Tree algorithm (RRT) and its variants are widely used for autonomous path planning for mobile robots. Proposed by Steven LaValle in 1998, RRT is a sampling-based

[ht]

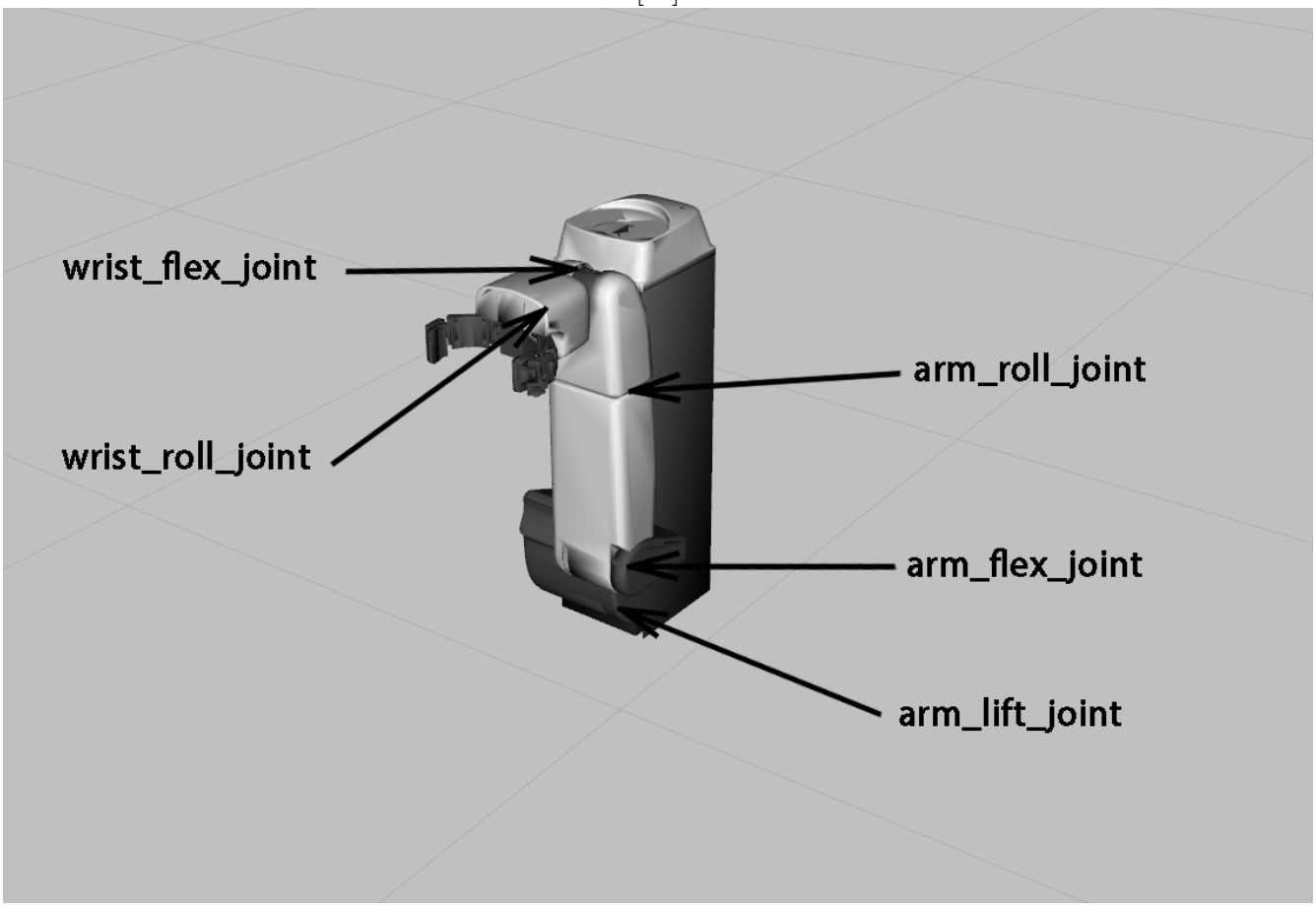


Figure 4: HSR arm joints

planning method that is probabilistically complete and naturally accomodates non-holonomic constraints. RRT builds on the Probabilistic Roadmap planning method, which has the desirable property of being probabilistically complete but lacks the ease of extension to non-holonomic systems.

Sampling based planners are designed to adeptly search high dimensional configuration spaces without the high computation power required for dynamic programming [Bellman, 1957]. Due to the probabilistic completeness that arises from the process of randomly sampling configurations, sampling based planners can also better avoid the issue of becoming trapped in a local minima as in the case of gradient-descent approaches.

Figure 5: [image of prm algorithm and graph]

5.2.1 PRM

A PRM is initialized with the starting state of the robot and a desired goal state. Robot configurations are randomly drawn from the set of free configurations. This implies that a map of the environment with obstacle and free regions is known. These random configurations are represented as nodes on a graph. Neighboring nodes with an unobstructed path and a distance below a given threshold are connected. This represents a valid path between two configurations. An issue arises here as a euclidean distance metric gives a computational speed advantage, but assumes holonomicity. Alternatively, [PRM paper] proposes using a local planner to calculate swept-area/volume as a distance metric, which would allow the method to extend to non-holonomic robots, but the authors note that this method is time consuming. Given sufficient samples, the nodes would fill the entire free configuration space. A unidirectional search of the network using an alogrithm like Dijkstra's shortest path would give the shortest valid path in the PRM. Given infinite random samples, the PRM method would search the entire free configuration space, meaning that it is asymptotically optimal [C18 notes]. Further, since the random sampling is not affected by the start and end states, the roadmap can be saved and queried for other start and end states, reducing computation time for subsequent planning calls.

The main issue with PRM is that for every new sample, a k-nearest-neighbor query is performed to identify neighobring nodes within a threshold distance. This computation becomes increasingly

expensive as the number of nodes in the PRM increases.

5.2.2 Vanilla RRT

Figure 6: [image of rrt graph and algorithm details]

As mentioned previously, RRT builds on the PRM method. Instead of creating a cyclic graph in the case of the PRM, RRT forms an acyclic, directed tree rooted at the starting configuration node. Like PRM, RRT is also a sampling-based planner, although when extending the graph, a randomly generated configuration is not directly added to the tree. Instead, as the name suggests, Rapidly Exploring Random Trees take a step in a random direction to explore the free configuration space. A step of a given length is taken towards the random configuration from the closest node of the tree, given that the step does not result in a collision with an obstacle. This gives a slight advantage over PRMs since only a single nearest neighbor query is required. Once a branch of the tree steps within a threshold distance of the goal, a path from the start configuration to the goal configurations is found. Since the graph is rooted, directed, and acyclical, the solution is easily found with a simple tree traversal.

Unlike PRM, the tree for RRT is rooted at the starting node, so this single-query algorithm must be rerun for differing start and end states. RRT maintains probabilistic completeness for holonomic systems due to the random sampling of the free configuration space, given that states are uniformly sampled. LaValle explains the probabilistic completeness property in terms of a

Voronoi diagram of RRT vertices. Due to random sampling, exploration is biased towards large Voronoi regions, converging to a complete, uniform (or other sampling scheme) search of the free configuration space as large Voronoi regions are iteratively reduced. [image of voronoi] RRT also more readily adapts to non-holonomic constraints compared to PRMs. This is due to the directed nature of the tree. The closest node to a random configuration can be determined by a metric that accounts for a constraint such as rotation. The path from the closest node to the new node can be found by inverse kinematics under non-holonomic constraints.

5.2.3 Bidirectional RRT Variants

RRT has numerous variants, notably Optimal RRT (RRT*) which reorganized the tree structure as new nodes are generated. Two variants relevant to this project are RRT-Connect and Constrained BIDirectional RRT (CBiRRT2). A time series RRT Connect algorithm is implemented in [Time Series Planning ***] which is detailed in [Section Reference]. The HSR utilizes a planner that builds upon CBiRRT2. Both RRT-Connect and CBiRRT2 are bi-directional RRT variants, meaning that trees are extended from both the start and goal nodes by swapping the start and goal trees each loop. RRT-Connect incorporates a Connect heuristic, which is a greedy function that takes the place of the Extend step in the original RRT algorithm. Instead of taking a single step towards a target configuration, q, the Connect function iteratively extends the tree towards q until either an obstacle is encountered or the configuration is reached. LaValle and Kuffner note that q can be a random configuration or the nearest neighbor of the other tree. This allows for tuning the algorithm to prioritize either exploration or connection of the two trees.

The advantage of a bi-directional RRT over the original RRT is the expansion of the tree rooted at the goal node. As the goal-rooted tree expands, the number of possible points of connection increases, encompassing a larger portion of the free configuration space. This increases the rate of finding a feasile path.

So far, the problems described have been confined to planning in ¡=3 dimensions which is intuitively understandable by huamns, with vague suggestion of application to higher dimensions. This is satisfactory to describe the planning for the omnidirectional base of the HSR, which has motion in x, y, and θ. However in whole body planning for the HSR with 8-dof (3-dof base + 5-dof arm), higher dimension constraint manifolds represent the possible configurations of the robot that satisfy given constraints. The goal remains the same: to find a path from start to end states in

[ht] 

Figure 7: [RRT Connect Algorithm and diagram.]

task space that lies completely on constraint manifolds in configuration space.

5.2.4 CBiRRT

CBiRRT, the first published iteration of the algorithm, uses a method similar to the Connect-Heuristic that attempts to iteratively extend both trees towards a randomly sampled configuration. Unlike RRT-Connect, in higher dimensions, the constraint manifold may be of a lower dimension than the entire configuration space. This means rejection sampling, which is the uniform sampling the entire configuration space until a viable configuration is selected, does not work. Instead, a ConstrainedConfig step is introduced to project nodes onto the nearest constraint manifold, to ensure that configurations satisfy the given constraints. [reference CBiRRT]

5.2.5 CBiRRT2

CBiRRT2 introduces spacial expressions called Task Space Regions (TSRs) which are used to efficiently search the configuration space, especially for end-effector manipulation with pose constraints. TSRs encode a frame in configuration space and a set of boundaries in the coordinate of the frame. TSRs are designed for sampling-based planners because they are easy to sample from and distance from a given configuration to a TSR can be quickly computed. Further, TSR's can be chained together to encapsulate multiple pose constraints on an end effector. In the CBiRRT2

algorithm, trees are grown in each iteration using one of two modes, explore and sample, determined by a a set probability. The explore mode attempts to connect to a random configuration like in CBiRRT. The sample mode is a new feature that differs from CBiRRT. In sample mode, the algorithm samples from a set of TRS chains and adds a new goal node to the goal tree. One of the driving principles behind CBiRRT2, embodied by the sample mode, is the idea that a task can be accomplished in more than one way. By specifying a TSR for a goal, the CBiRRT algorithm attempts to both identify possible goal configurations that would fulfil a task, as well as solve for a path to a goal. The process of increasing the number of goal states essentially creates a pair of "seeded forests" that increase rate of discovering a motion plan. [reference CBiRRT2]

5.2.6 HSR CBiRRT2 Variant

The HSR variant of CBiRRT2 simply varies the weighting of either arm motion or base motion. [cite HSR paper] note that prioritizing arm movement results in accurate end effector positioning, while prioritizing base movement permits the application of a larger force on an object. The HSR motion planner incorporates time optimal path parameterization (TOPP)[HSR paper reference] to optimize the motion of the HSR in time under hardware constraints such as speed and torque limits. TOPP allows the HSR to reach goal poses at specified times as long as the time-from-start is greater than the optimal duration, by scaling the trajectory in time.

5.2.7 Commentary

The purpose of this project is to solve a driveby pick-and-place problem. The HSR can already pick-and-place an object with its 'built-in' motion planner. This can be done by specifying a desired end-effector position to be reached in order to grasp the target object. The placement target can be specified by a second end effector target pose. The HSR can create TSRs of both target poses and two calls to the CBiRRT2 variant planner would result in a satisfactory solution to the simple pick-and-place problem. However, the driveby component of the problem introduces a time-varying aspect that is not easily satisfied by the HSR planner.

CBiRRT2 and by extension, the HSR motion planning algoritm, address scleronomic holonomic constraints, which are defined as 'time-invariant constraints evaluated at a given configuration of the robot' [reference CBiRRT2]. In a practical application, it is necessary for the HSR to plan for motion in a dynamic environment with moving collision constraints such as humans in a hospital

environment. [Yang, Merkt, et.al. 2018] propose a method to plan a pick-and-place problem in time-configuration space to address such time-varying constraints.

5.3 Planning in Time-Configuration Space for Efficient Pick-and-Place in Non-Static Environments with Temporal Constraints

5.3.1 description and overview of method

[time indexed PPP*** reference] identify two classes of constraints that are considered in the course of a pick-and-place problem. Type 1 constraints are differentiable and involve inequality and equality constraints, which means approaching certain desired values. These are the constraints that the driveby component of the problem fall into, such as end-effector positioning. Type 2 constraints are binary and include constraints such as collisions. Problems with a special case of static type 2 constraints can be solved with the sampling-based planners described in [insert section reference]. Another modification to the RRT algorithm is introduced later in [section reference] to accomodate for the time-indexed nature of dynamic environments.

[time indexed PPP*** reference] decompose a time-varying pick-and-place problem into 3 segments, depending on the classes of constraints active. These segements are: reaching, grasping, and placing. The reaching and placing steps serve the purpose of bringing the robot to the target object and then to the desired object location. During the reaching and placing motions, only type 2 constraints are active, primarily collision avoidance. [time indexed PPP*** reference] note that the planning for the placing segment involves accounting for the altered geometry of the end effector grasping the target object. The time varying case differs from the solution for the simple pick-and-place problem described in [reference to commentary section] in that the HSR must start and end each segment at a designated time in order for the planned robot trajectory to coincide with the trajectory of collision obstacles. During the grasping phase, the end-effector follows the trajectory of the target object until the end-effector can close and grasp the target object. In the case of the driveby pick-and-place task, the HSR must continuously drive along a given base trajectory while keeping the gripper centered on the target soda can until the gripper can fully grasp the soda can. The time frame of the grasping phase is determined by the hardware-defined time for the gripper to close around the target object. For the HSR, the gripper closes within 0.4 seconds. The start and end configurations of the grasping phase determine the goal and start

state of the reaching and placing segments. The 3 stages are concatenated to produce a complete pick-and-place trajectory.

5.3.2 Time-Configuration Space RRT-Connect

Reach planning in the presence on time varying obstacle trajectories searches for a path in time configuration space. This space includes a time parameter in addition to the entire configuration space of the robot. The time parameter is constrained by velocity and temporal constraints. The velocity constraint ensures that joint velocity limits are not exceeded. The temporal constraint ensures causality/time-monotonicity, so the robot does not travel backwards in time. Similar to the case in CBiRRT2, the sample space may be of a higher dimension that the constraint manifold, so rejection sampling does not work efficiently. This issue is further exaggerated by the introduction of the dimension of time. [time indexed PPP*** reference] introduces a CorrectTime function that works by adjusting the time parameter after a valid sample is chosen from the free configuration space. The time parameter is adjusted based on a custom distance function that only returns a real value if both time constraints are satisfied. [Reference figure 3 of time indexed PPP*** reference].

5.3.3 AICO

The grasping is solved for by a non-linear programming solver such as SNOPT and AICO. As SNOPT is a proprietary, licensed software, AICO is used for this project.

5.3.4 Approximate Inference Control

Approximate Inference Control (AICO) [Toussaint reference] builds on Iterative Linear Quadratic Gaussian (iLQG) which is a method of sequential quadratic programming [Todorov & Li first iLQG paper Reference]. iLQG optimizes a control sequence by linearizing the system dynamics around each system state and control imput in the sequence. This local linearization casts the non-linear system dynamics problem into a sequence of LQG problems, where each iteration steps throught the sequence to further optimize the entire trajectory. The iLQG alogritm utilizes a forward-backward algorithm that computes a trajectory with a forward pass by applying a sequence of control inputs in an open-loop manner. A backwards pass computes a second-order approximation of the cost-to-go of the trajectory, which is minimized to find a locally optimal trajectory and a linear quadratic regulator that can handle small perturbations around the selected trajectory.

Adjustments are made to the sequence of control inputs in the subsequent forward pass until the trajectory converges. The iLQG method minimizes a cost function to determine a locally optimal trajectory. On the other hand, AICO takes a probabilistic inference approach by conditioning a trajectory on constraint conditions. Toussaint demonstrates that for the LQG case, the maximum likelihood trajectory approaches the solution found by cost minimization. To Do Later: Need to read through AICO paper and get a better understanding before summarizing.

5.3.5 Commentary

[Toussaint] notes that it is difficult to include collision avoidance in AICO and describes that in practice the trajectory optimization would be based on a physical simulator. [Yang, Merkt, et.al.] explain that the solver gets easily trapped in local minima in static environments under collision constraints. They elect to solve grasping trajectories until a collision free trajectory is found. [ppp*** reference] implemented their pick-and-place framework in the Extensible Optimization Toolset (EXOTica) framework [Exotica reference]. EXOTica is "a framework of software tools designed for development and evaluation of motion synthesis algorithms within ROS" and includes an implementations of motion solvers such as AICO as a benchmark for new motion planning algorithms.[Exotica reference] This project uses EXOTica to plan the grasping phase of the driveby pick-and-place motion. Further details on the EXOTica framework are provided below.

6 Method/Approach

6.1 Overview

This project closely follows the method proposed in [time indexed PPP*** reference]. One significant difference is that the time-configuration planning method is designed for picking and placing a moving object. A driveby pick-and-place scenario also has the target object moving relative to the robot, but in the driveby case, the HSR must approximately follow a predefined trajectory which complicates the end-effector tracking of the object during the grasping phase. [insert figures of robot grasping object and figure of hsr driving by object] This issue is largely due to limited reachability of the HSR arm. [may go further in depth here] To address this issue, the grasping phase is alloted an extended duration of time to encompass the HSR's approach and grasping of the object. The new, extended grasping phase plans the sub-reaching, sub-grasping, and sub-leaving steps, which are similar to the reaching, grasping, and placing steps in [*** reference].[Insert figure to visualize the new method, with sub-divided grasping phase] reaching=¿ extended grasping phase(sub-reaching=¿ sub-grasping=¿ sub-leaving)=¿placing

This method differs from the Time-Configuration RRT-Connect in that the grasping phase is extended to incorporate a sub-reaching and a sub-leaving stage while maintaining the reaching and placing phases.

The HSR CBiRRT2 variant is used to reach the start of the extended grapsing phase and leave from the end pose of the extended grasping phase to place the object. This is largely for convenience. The HSR RRT planning algorithm is built-in on the robot and the world is static for the purpose of this project. The time varying component of the problem is the position of the target object, a soda can in this instance, relative to the HSR as it drives-by, following a trajectory. The extended grasping step can be planned using AICO and activating specific constraints during the sub-grasping phase time window.

The HSR CBiRRT2 motion planning algorithm is already implemented on the HSR and TOPP allows for control over the timing of planned trajectories. This covers the reaching and placing steps of the motion plan. The key step for this project is the extended grasping phase.

6.2 Scenario Setup

To simplify the problem described in [section reference], a bare-bones world containing only an Ikea Nyboda Table and a soda can was created in Gazebo. Gazebo is an open-source robotics simulator commonly used for ROS-based robot simulations. The objects in the simplified scenario were chosen for convenience. The Ikea Nyboda Table was chosen as it was the model of table present in the testing space for the HSR. A soda can was chosen as the target pick-and-place object since it was a default, tutorial object for the HSR Gazebo package. Further, the radius of the soda can is larger than items expected to be used in real world testing, which produces a scenario with a smaller margin for error in terms of end-effector positioning. ¡insert figure¿. In this example, the HSR follows a given base trajectory that is parallel to the table and attempts to grasp the bottle.

6.3 Constraints On Grasping

The grasping method in [ppp*** reference] identifies valid grasping trajectories by generating new trajectories using AICO until a collision-free trajectory is found. In those cases, the grasping phases have a relatively short time frame. For instance, the HSR can close its gripper within 0.4 seconds, giving a maximum grasping duration of 0.4 seconds. In the case of the extended grasping phase for the driveby problem, the duration for the HSR is around 15 seconds. Although computation time depends on the number of waypoints in a trajectory, the distances traversed are much larger in the case of the driveby problem. This means that checking for collisions after computing trajectories is no longer efficient, since it is highly likely that the arm or end effector would collide with the table in this test scenario. To accomodate for this issue, constraints are introduced to reduce the likelihood of colliding with the table. As mentioned above, EXOTica is used to plan this grasping phase.

6.4 EXOTica [reference]

As a benchmark framework, EXOTica introduces abstractions that separate a problem into 3 components so multiple motion solvers can be used to compare efficiency.[ref] The components are a planning scene, a planning problem, and a motion solver.[ref]

The planning scene contains information about the robot and its environment. In the test

example, the planning scene contains the position and geometry of the Nyboda table and soda can as well as information about the HSR. This environment is read from a file with a .scene file format. The HSR dynamics and kinematic properties are read from URDF and SRDF files. (Unified Robot Description Format and Semantic Robot Description Format)[probably some reference to the file formats]. The base trajectory for the driveby is also specified here, since it defines the robot's base position in the scene. This input base trajectory acts as a base position and orientation constraint.

The motion solver is set as AICO, since the purpose of this project is to use the motion solver implementation instead of benchmarking a new solver. It is important to note that AICO does not guarentee that constraints are satisfied. Instead, it optimizes the trajectory according to a quadratic cost function, in which each constraint is weighted according to its cost.

The planning problem defines the type of problem being solved and the tasks to be completed. The problem definition used for the grasping phase is an Unconstrained Time-Indexed Problem, since it defines a 'defines a problem minimizing the quadratic cost over a trajectory using a kinematic model of the system', which can be solved by AICO. The planning problem uses task maps, which are functions that map the configuration space to task space. EXOTica includes several common taskmaps that are utilized in this project to define constraints on the HSR during the grasping phase.

6.4.1 Taskmaps

Task maps are active at various points of the extended grasping phase depending on the sub-phase. A task map is considered active when its associated cost is greater than zero. Throughout the entire extended grasping phase, 2 task maps are constantly active:

- A Joint Limit task map penalizes joint limit violations to promote HSR arm trajectories that fall within viable configurations.
- A Base Position task map penalizes deviations from the driveby base trajectory specified in the planning scene.

3 task maps are activated during the sub-grasping phase. Despite the notion that the task maps are active at a certain time, AICO attempts to optimizes the trajectory such that the constraint imposed by the task is satisfied in the window of time that the task map is active.

- An End Effector Position Task map causes the HSR to reach the location of the target soda can object. This task map is activated in the window of the sub-grasping phase. The sub-grasping window is defined by

$$grasp\ window = t_{grasp\ start} : t_{grasp\ start} + grasp\ duration$$

where $t_{grasp\ start}$ is the point in time where the HSR is sufficiently close to reach the soda can and *grasp duration* is the maximum time necessary to close the HSR gripper, which is 0.4 seconds.

- An End Effector Axis Alignment task map aligns the HSR gripper with the axis of the cylindrical soda can so the gripper stays around the soda can while the fingers of the gripper close. This Axis Alignment task begins at $t_{grasp\ start}$ and remains active to keep the soda can upright for the remainder of the extended grasping phase. [Insert figure]

- A Point to Plane task map named 'Lift Off Table' keeps the end effector a set distance above the plane of the table to avoid the arm and gripper colliding with the table. This task map is active momentarily before *grasp window* and momentarily after *grasp window* to position the gripper above the table when transioning from sub-reach to sub-grasp and from sub-grasp to sub-leave, such that the gripper clears the end of the table when lowering the starting arm configuration. This task map results in a swooping motion of the gripper during the sub-graping phase. This Lift Off Table task map takes the place of collision checking the trajectory by creating a trajectory that likely does not collide.

6.4.2 Commentary on Task Map Weights

[create a diagram of extended grasping phase and when constraints are active.]

6.5 HSR Controller

The AICO solver outputs a trajectory with position waypoints equally spaced in time. The trajectory is specified through a joint trajectory controller in ROS and sent to the HSR. According to the ROS wiki documentation for the joint trajectory controller, a trajectory composed solely of position waypoints is linearly interpolated and results in a trajectory with discontinuous velocities at the waypoints.

Taking the difference in positions and dividing by the time step produces an array of desired velocities at each waypoint. This results in a cubic spline that ensures continuity at the velocity level. [Include a bit about simple action client and how trajectories get passed to HSR]

6.6 Procedure

6.6.1 Hardware and Simulation Specifications

Driveby pick-and-place motions are run on an HSR simulated in Gazebo. [Computer specs like in ***].

6.6.2 Grasping Phase

The driveby base trajectory was defined in the planning scene as a 1.4 meter path parallel to the edge of the table with a duration of 10 seconds and a distance from the edge of the table of 0.35 meters. This trajectory was defined by two waypoints and results in a linear base trajectory that runs parallel to the Nyboda table. [Insert figure of top-down view of setup in notes]. Since the task maps passed to the AICO solver do not depend on time, the 10 second duration can be scaled in time by increasing or reducing the time step between trajectory waypoints of the solved trajectory.

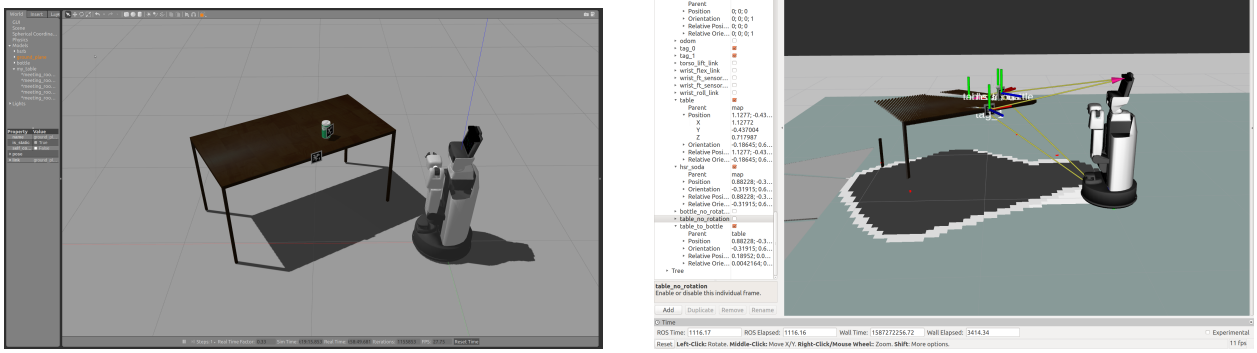
Due to limitations in the reachability of the HSR arm, the target object was placed near the edge of the table to mimimize the liklihoold of collision with the table. [Section Reference] further details the relationship between table height and bottle distance from table edge for the HSR.

6.6.3 Object Detection

Visual markers are used as a fiducial to accurately acquire the position of the table and soda can. The HSR includes AR marker detection which uses the depth sensor as well as the steroscopic camera to determine the position of a defined object. However AprilTags were used instead since AprilTags were found to permit tag detection from a longer range compared to AR markers. [AprilTag2 Reference. Maybe add to literature section].

Once the frames of both the table and soda can were determined, a series of transforms using KDLFrames wrapped in EXOTica produced the absolute position and orientation of the table as well as the position of the bottle in the frame of the table.[references] Since the position of the soda can with respect to the table's frame is known, the planning scene utilized in the test example can

be used to compute the extended grasping trajectory. The absolute position of the table allows the HSR motion planner to reach the start of the extended grasping phase, which is defined in the frame of the table. [Insert Image of tf's in rviz]



(a) Gazebo Visualization

(b) Rviz visualization

Figure 8: Will clean up image. (a) Gazebo visualization of world. (b) Soda can and starting pose of HSR in the extended grasping phase in the frame of the table

7 Results and Discussion

7.1 Trajectory Analysis

AICO outputs a trajectory that is smooth in position 9(b), but there are no guarentees of smoothness in velocity or acceleration. For the extended grasping phase, AICO produced a trajectory with 100 waypoints. Initial trials with a timestep of 0.1 seconds for a 10 second extended grasping phase duration, found that the HSR could not reach waypoints in the alloted time and returned an error. Through incrementing the timestep by 0.01 seconds, the first timestep that did not result in an error was 0.15 seconds that resulted in a 15 second trajectory. Figure 9 depicts the ideal trajectory output by the AICO solver, sent to the HSR through the joint trajectory controller.

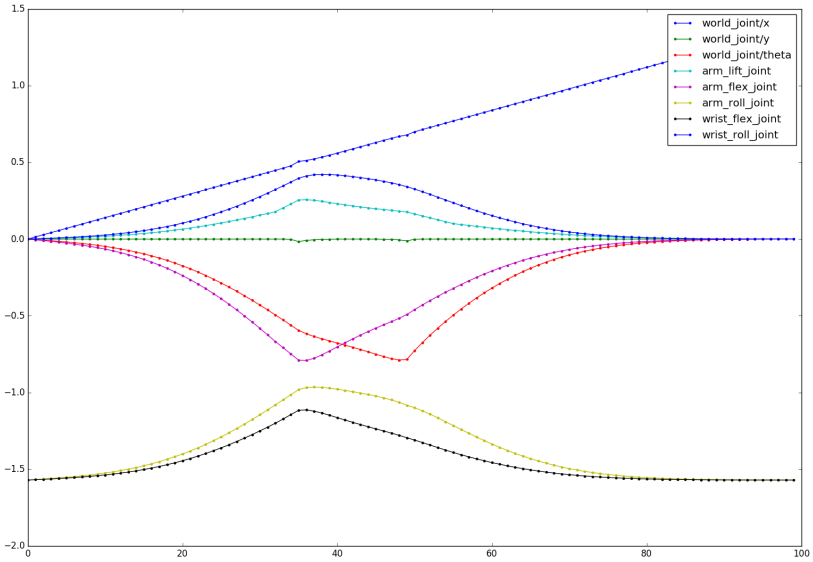
7.2 Analysis of Plots

It is important to note that trajectory feedback measurements differ from the ideal trajectory due to noise in the robot system and also due to alterations made by the joint trajectory controller. For planning the path of the base, AICO gives a trajectory with x,y coordinate positions and an angle θ that represents the yaw of the HSR. The HSR base trajectory controller takes the x,y, θ values and converts the values into goals for the drive wheels and the base roll joint, described in [hsr section reference]. As a result, Figure 9 base joint trajectories and labels differ from the represenation in Figure 10. Because base joint velocity, acceleration, and efforts are defined for the physical HSR joints, the latter labels are used for analyzing the trajectory.

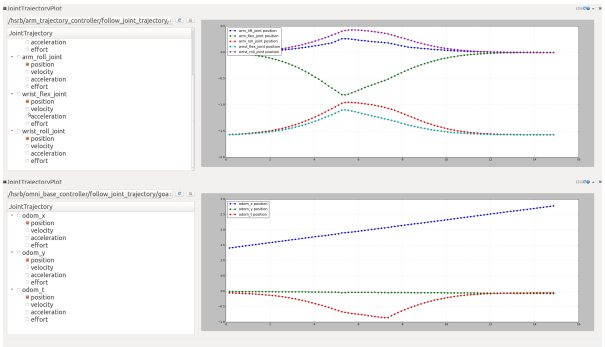
7.2.1 Joint Positions

At the position level, the measured arm joint positions in 10(a) match the ideal arm joint positions in 9(a) and 9(b). Figure 11 depicts that the error in position is small in comparison to the position values. It is notable that the arm flex joint and arm lift joint deviate the most from the expected values. However, considering that the 2 links are at the base of the HSR arm kinematic tree and are responsible for moving the entire weight of the arm, relatively larger deviations are acceptable.

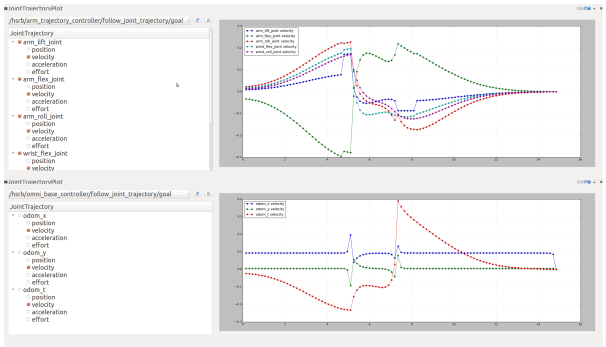
Further, a visual check of the trajectory in Rviz and Gazebo confirms that the error in joint positions did not significantly impact the trajectory, since the HSR successfully grasped the soda can during the extended grapsing phase and reached the end pose.



(a) AICO trajectory output

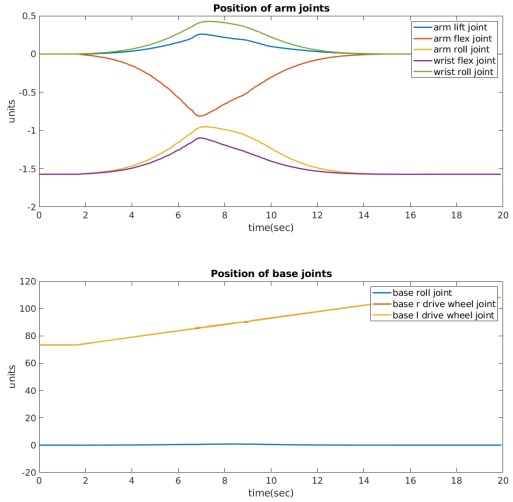


(b) Waypoint Positions

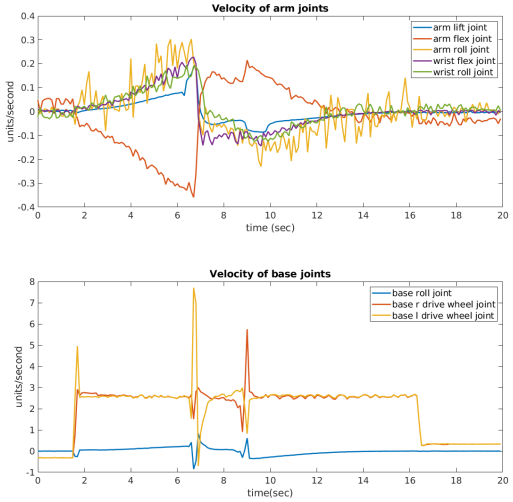


(c) Waypoint Velocities

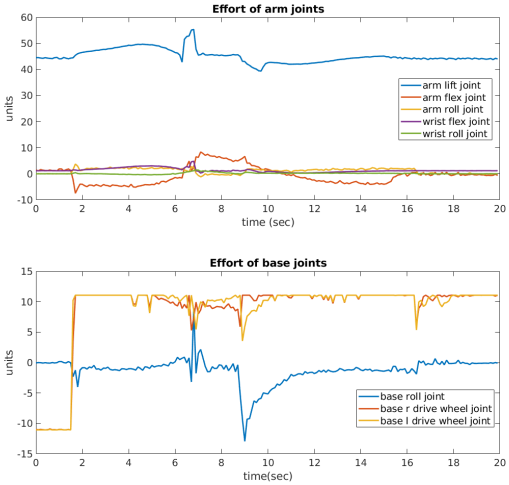
Figure 9: Ideal position and velocity values at each waypoint. (a) shows the joint position values output by the AICO solver as a sanity check for (b). (c) shows a plot of joint velocities at each waypoint. Values plotted from (b) and (c) were received from messages published by /follow_joint_trajectory/goal topics for the HSR arm and base. This topic publishes joint_trajectory messages at 10hz, which represent the ideal waypoints for the HSR to follow. Connections between points should be ignored since they are an artifact of plotting with RQT



(a) Position of Arm and Base Joints



(b) Velocity of Arm and Base Joints



(c) Effort of Arm and Base Joints

Figure 10: Graphs of (a) Position, (b) Velocity, (c) Effort of a trajectory executed by the HSR when position and velocities are specified. Trajectory starts at 1.5 seconds and ends at 16.5 seconds. Plots depict trajectory messages over the /hsrb/robot_state/joint_states, which publishes at a rate of 10 hz. This topic Connections between points should be ignored since they are an artifact of plotting with RQT

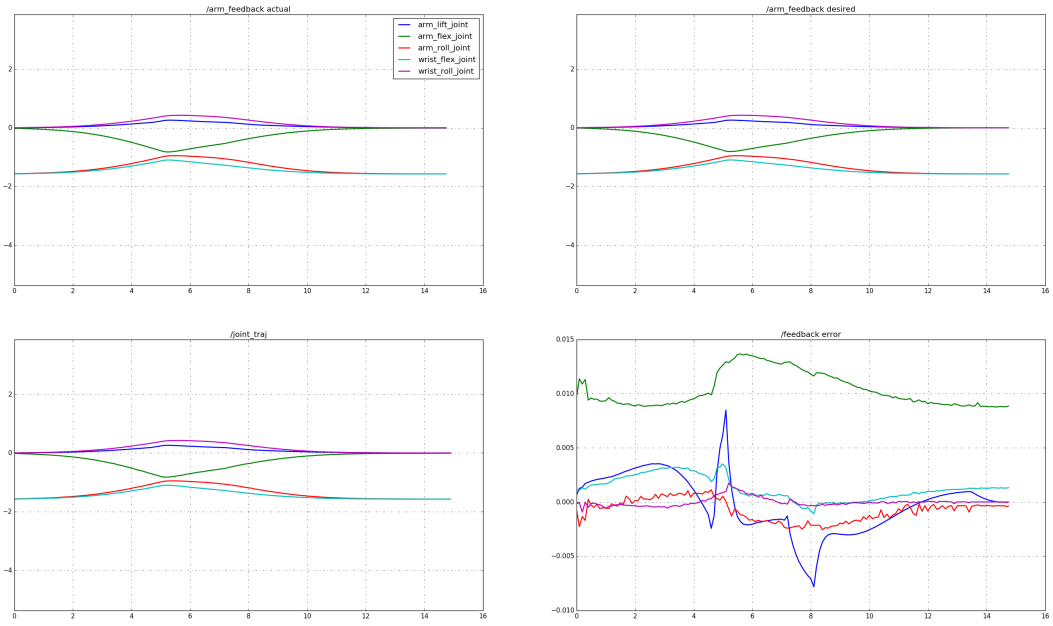


Figure 11: All trajectories were sampled from the simulated HSR at 10 hz. The horizontal axis has units of time in seconds and the vertical axis has units of position, meters in the case of arm lift joint and radians for all else. (top left): actual joint positions of the sim HSR arm. (top right): Expected joint positions. (bottom right): difference between position values in measured and expected joint positions. Actual and desired values were received from a control_msgs/FollowJointTrajectoryActionFeedback message, so the values have matching time stamps.

From 10(a) it appears that the HSR base follows the linear driveby trajectory very closely, with slight deviations at the start and end of the sub-grasping phase between 7 and 9 seconds in 10(a) indicated by small disturbances in base drive joint plots. These abrubt changes in position are due to task maps becoming active during the sub-grasping phase. The base roll joint plot does not accurately caputure the change in HSR body orientation, due to the large values in drive wheel joint position. [insert new plots without resets and separate roll joint from drive joints].

The HSR successfully accomplished the grasping phase of the driveby pick-and-place problem in simulation. Although the joint position limits for the simulated HSR are slightly greater than the values given for the actual robot, the joint positions in the trajectory simulation fall below both boundaries as a result of the joint limit task map. [insert chart of joint limits in hsr manual compared to limits in urdf file.]

7.2.2 Joint Velocities

[insert table of joint velocity limits] The joint velocity measurements fall within velocity limits for both the physical robot and simulated robot. [reference limit table].

One issue is the prevalence of periodic perturbations in the expected arm joint velocities in Figure 12 that are not present in the expected velocity plot in 9(c). The joint velocity values in 9(c) represent the velocity goal at each trajectory waypoint and are obtained from differentiating the joint position trajectory generated by the AICO solver. The perturbations in Figure 12 result from cubic interpolation performed by the HSR trajectory controller that ensures smoothness in velocity.

Another glaring issue is the significant noise in the measured velocities in Figure 12. The histogram plot in Figure 12 appears to be zero-mean and roughly gaussian. Thus, three possibilities were proposed to explain the disturbances.

- 1. The deviations are due to noise in measurement.
- 2. The deviations are due to noise in the simulated joint velocity.
- 3. The velocity is attempting to return to zero at each time step, resulting in large disturbances.

Figure 13 demonstrates the effect of setting the velocity at each waypoint to zero. The perturbations are much more ordered than in the case of Figure 12, and the expected joint velocity plot

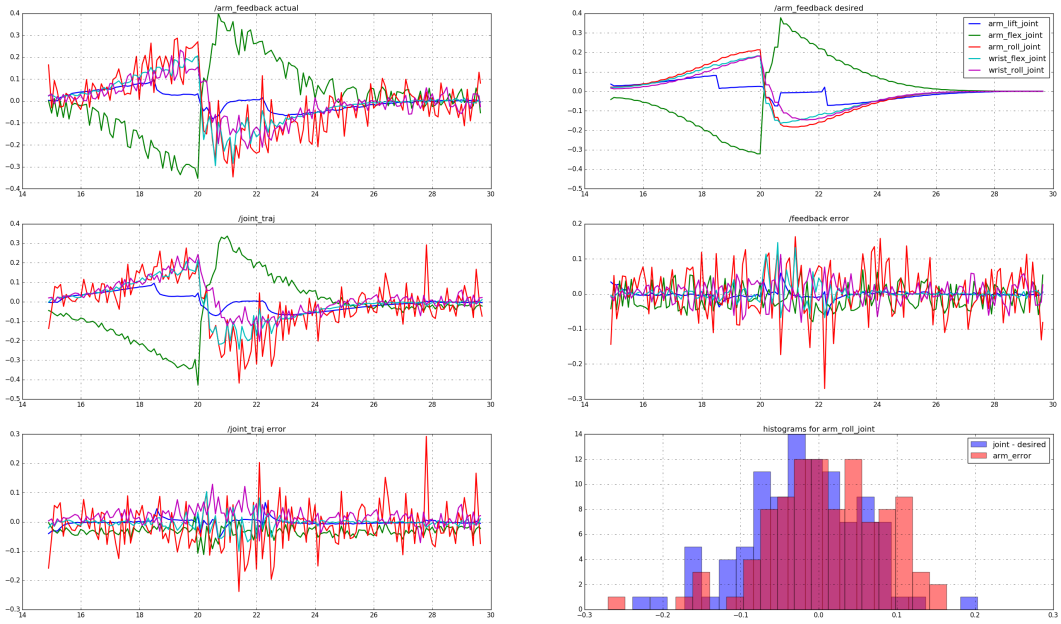


Figure 12: For non histogram plots, the horizontal axis has units of seconds and the vertical axis has units of velocity. (Top row): Measured and expected joint velocity values for the simulated HSR. Corresponding points have matching time stamps. (Middle left): Velocity measurements sampled from the simulated HSR trajectory. Time stamps do not exactly match those of the expected plot. (Middle Right): Plot of difference between time synchronized joint velocity measurements. (Bottom Left): Plot of difference between velocity measurements of /joint_traj and expected. (Bottom right): histogram of velocity error values for the arm roll joint.

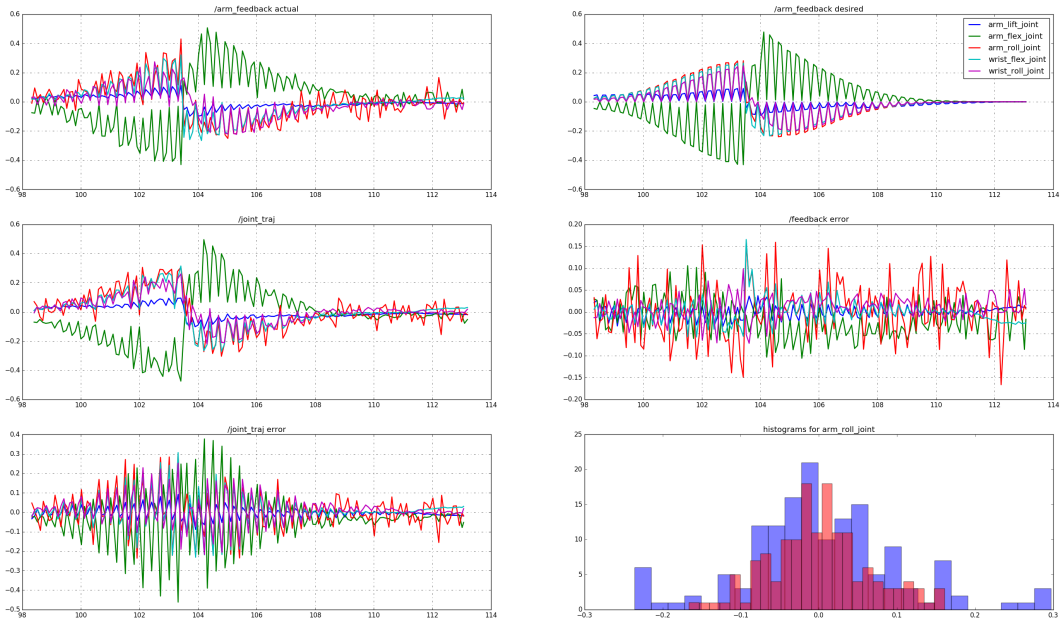


Figure 13: Plots in a similar order to those in Figure 12. Joint velocities were set to zero at each trajectory waypoint.

would show more extreme rapid changes in velocity. This rules out explanation 3.

Since the physics are simulated, explanations 1 and 2 become indistuniguishable unless noise is specifically added after a measurement is taken. Conditions relating to noise must be described in the HSR urdf. Checking the urdf file yields Gaussian noise tags for joints in the head and base segments of the HSR, but none specified for arm joints. However a damping parameter in the dynamics tag is specified for each arm joint. The damping parameter is set to 1.0 for all arm joints aside from the arm roll joint which has a damping parameter of 0.1. The dynamics tag specifies physical properties of joints and the damping parameter specifically controls the decay of velocity. The fact that the damping parameter of the arm roll joint is lower partially explain the higher amounts of noise in Figure 12.

7.2.3 Joint Acceleration

[Need to make new plots and adjust the trajectory to reduce the exceeding ofacceleration constraints] [Caveat= Velocity is noisy so acceleration calculation is somewhat unreliable.]

7.2.4 Joint Effort?

The plot of joint effort in 10(c) shows that the base joints are nearing the effort limits of 11. On the one hand, it is approaching the effort limit, but is applying maximum effort good and expected behavior?

8 Conclusion

subsectionResults [Recap of method] [Putting it all together: Insert section with whole motion plan.] subsectionExtensions

- Currently requires a predetermined trajectory. Can find one itself
- Currently, timing of grasps is hard coded since the HSR positions itself relative to the table and soda can. The relative positioning assumes open space surrounding the table.
- Include moving obstacles
- Use object detection/recognition instead of apriltags. HSR has some surface detection and can use YOLO with the nvidia jetson TK1.
- self identify optimal time step

- Project title: RoboCup@ Home: motion planning for mobile robot navigation. Include Robocup@ home competition description as a motivation for the project.
- Overview of the problem at hand: Driveby pick and place problem with the Toyota HSR Robot. Include brief description of the HSR and tech specs/features. General problem overview of picking up a cylindrical object from a table.
- Project description (these are from the project description on the 4yp website from early 2019. The project Ioannis shared with me at the start of the academic year revolved more specifically around the driveby pick-and-place problem and implementing the RRTConnect, AICO combination detailed in the Pick-and-Place planning in time-configuration space paper)
  - Easily Attainable: A focused literature survey on search-based and sampling-based planning, and optimal control. Plan paths for the HSR robot in a given arena. Avoid collisions and optimise for time taken to reach goal. Integrate sensory information and update internal representations of the environment. Guide a person to reach a goal location. (This can be done with the hsr interface/navigation stack)
  - Medium Complexity: Dynamically changing environments. Navigate around people and deal with dynamic obstacle changes. Extensive evaluation in simulation and first tests on the real robot platform. (Haven't done extensive evaluation, but there is the capacity to send obstacle trajectories to the planning scene for the AICO solver in exotica)
  - Advanced: Plan paths through variable configuration spaces, e.g. a path can be made accessible by pushing a chair aside. Evaluate on the real system. (Have not accomplished yet.)
- Main point: Solving a time-indexed, drive-by pick and place problem by optimising a whole body trajectory (to address reachability issues).

9 Literature Review 10-15 pages

- RRTConnect(2-3 pages)
- AICO solver (2-3 pages)

- Planning in Time-Configuration Space for Efficient Pick-and-Place in Non-Static Environments with Temporal Constraints (3-5 pages)
- maybe include: Apriltag visual fiducial system (1-2 pages)

10 Related Work

10.1 RRTConnect

11 Technical (15-20 pages)

- Preliminary work.
  - General overview of ROS framework.
  - Relevant hardware/features of the HSR: Sensors and joints, type of end effector. ROS-based software, navigation stack, trajectory controllers.
- Description of problem
  - wrs competition gazebo world with tables, shelf, and two people. Drive-by-pick up a mug/bottle on table and place on a shelf. The complicated part is the driveby pickup. Getting to the start pose of the pickup trajectory and getting to/placing the object can be done with the hsr interface. I have it mostly in place. Need to make it work in the wrs world which shouldn't take too much extra time.
  - Introduce bare bones empty world setup for testing with ikea nyboda table and soda can. Ikea Nyboda table is used since it is the model of table available in the area of testing (ori big meeting room). A soda can is used because it is a default gazebo tutorial asset for the hsr.
- Approaches
  - Whole body planning with the AICO solver.
  - assumptions and required information/sensor data:
    - planning scene with object (table and bottle) locations and collision geometry known /obtainable.

        * trajectory of base / start and end points of trajectory known/obtainable.
        * Duration of base trajectory does not violiate hsr velocity constraints.
        * HSR python interface to move to the starting position, using cameras and depth camera. Map server for wrs world.
        * There is the option to include the trajectory and geometry of obstacles for time-indexed planning.
    - constraints, costs, timing of constraints (grasping and axis-alignment) and problem engineering. Justification and explanation of hand-tuned costs.
    - analysis of output trajectories. position, velocity, acceleration. material from emails. Smoothness, noise, sampling rates and possible interpolation.
    - (if time permits/need more pages) demonstrate/quantify the height of table and distance from table edge of bottle the hsr can reach. (plot of table height and bottle distance vs success of grasping?)
    - Could/should I compare the output trajectory of the whole body AICO problem with the rrtconnect and snopt/ik. Noting the reachability issue could justify the deviation from the method outlined in the pick-and-place in time-configuration space paper. I could graph the trajectory output using scripts I wrote already.

- (if time permits and if there is space) Put it all together in the wrs world and demonstrate feasibility. Nearly there. Have april tags working in gazebo. Probably need to hard code the location of placement though. Will try to get it working in the wrs world.

12 Conclusion 2-3 pages

- Recap whole body planning for pick and place problem.
- Possible extensions/future directions. Point back to assumptions and required information input.

13 References 2-3 pages

14 Appendices 3-5 pages

- Appendices content counts towards page count.
- Code
- Images and maps of world/experimental setup.