Fetching Simulator Final

By:
Trevor Green: trevor.s.green@colorado.edu
Megan McGinnis: memc1235@colorado.edu
Jonathan Conser: joco8546@colorado.edu
Lukas Bush: lubu0783@colorado.edu

Original Abstract:

Using python and Webots our team will develop a fetching simulator. This project utilizes various high level concepts such as joint space calculations, odometry, path planning, and robot sensing to simulate a dog and a person playing fetch in the backyard. The F&P Robotics P-Rob 3 arms will have 4 different objects to choose from to throw for the dog to fetch. However, the Aibo ERS-7 dog-like robot will only fetch the object that resembles a tennis ball then bring it back to the arm. Every other object will be ignored by the dog. This may seem like a simple task although, in order to have a dog like robot and humanoid robot perform detailed actions in a 3D world you must perform complex mathematics and programming.

Changes to the Project and Obstacles:

Our project experienced many different changes throughout its course. Our biggest change came when we discovered after working the Sony Aibo ERS7 for a week that the robot's jaw was not implemented with physics meaning that all the work we had made to make the robot walk, object detection, and attempts to pick up an object came to a halt. We could've either learned how to attach the ball using Supervisor, or switch to a robot that would be able to grab the ball. With very little understanding of the supervisor class and very little documentation or examples on the internet to help, the plan came to switch to the Tiago Steel which we had used in labs before and were familiar with.

Tiago Steel came with its own set of problems as well. When adding a camera for camera recognition, the camera was placed in a rather odd spot on top of the shoulder and unable to see anything at the robot's feet. This meant we had to position the robot in a certain position and not allow balls to be thrown too close to the robot. Otherwise, most of the time the camera was able to recognize objects and find the object with the correct ID. However, upon testing the inverse kinematics for Tiago Steel's path to the ball, we discovered that sometimes despite being able to see the ball and even putting a box around the ball on the camera, the object wouldn't be detected. This seemed to happen randomly and upon restarting the simulation, the ball would be correctly detected with no issues. Another issue that came up in regards to locating the correct ball using camera recognition was variance of the IDs of the balls. We discovered that upon reopening the project, sometimes the IDs would be different and we would have to change the ID the robot is looking for to account for the fact that all the balls now have different IDs. On

another note, depending on where the ball ends up, Tiago Steel will sometimes run into P-Rob 3 in its path to the ball. The inverse kinematics also came with many issues, as it required a whole new learning of the python library IKPY. With help from sample worlds, the IKPY documentation, Piazza posts, classmates, and the TA's, the inverse kinematics for the arm was almost completed but the IKPY library has some flaws and the limitations of our robot as well made it rather difficult to make a fully functional inverse kinematics for the arm.

Revised Abstract:

Using python, C, and Webots our team developed a fetching simulator. This project utilizes various high level concepts such as joint space calculations, manual/randomized motor controls, odometry, path planning, and robot sensing to simulate a robot fetching the correct predetermined yellow ball out of 8 total balls thrown by the two 2 throwing arms. The two F&P Robotics P-Rob 3 arms will have 8 different balls to throw for the fetching robot. However, the Tiago Steel robot will only find the correct yellow ball, travel to it, and attempt to pick it up. Every other object will be ignored by Tiago.

Equipment:

- **F&P Robotics P-Rob 3**
    - The original purpose of using P-Rob 3 was to have it pick up one ball and have our Sony Aibo fetch the ball and bring it back. Due to some changes in our project we decided to develop code for two separate P-Rob3 arms which would pick up and throw a total of 8 balls. Then the Tiago Steel would go and pick up the one yellow ball. The red/yellow p-rob3 just picks up 4 black balls at specified positions and throws those balls at predetermined angles. The white/blue p-rob3 picks up 3 black balls and the 1 yellow ball at specified locations, then throws each of those balls at a randomized motor angle which is meant to make the task of retrieving the ball more difficult for Tiago. Not only can the white/blue p-rob3 throw the balls at randomized motor angles but it also has the functionality for the user to manually use the arrow keys to choose which motor angle they want to throw the yellow ball at. In order to put the white/blue p-rob3 into manual mode, you must uncomment and comment our certain code within the RoboticArmController.c. It tells you how to do it within the controller.
- **Sony Aibo ERS7**
    - For our week-long time spent working with the Sony Aibo ERS7, we spent a majority of our time getting the robot to walk since it uses four individual limbs rather than two wheels and object detection along with a motion that allowed the robot to pick up the object it recognized. While working on object recognition we discovered that the camera built into the robot had no recognition node, so we had to add in another camera to the robot. We discovered that the robot had very little documentation about it, with the only examples available showing very little

about the robot. While attempting to pick up objects, we discovered that the jaw of the robot had no physics and objects would clip through, we made the decision to move to a different robot that could pick up a ball.

- **Tiago Steel**
    - Since we had worked with the Tiago Steel robot before, we were not diving into the unknown. But there were still some features that were new to us that we had to figure out. The first was implementing Camera recognition. When adding a camera to the robot, we discovered that the camera slot on the robot was on its shoulder at a set height, meaning if the ball was too close to the robot then when it would start spinning to search for the ball, the ball would not be visible to the camera and would go undetected. When detecting objects, we used the objects id that could be grabbed from the camera recognition to make sure we found the correct object. Once we found the correct object, we were able to get the object's coordinates and use inverse kinematics to move the robot closer to the object to be picked up. Once the robot was at the ball, it would use inverse kinematics with the arm to pick up the ball. After many test cases and test files of inverse kinematics for the robot, we struggled to get a fully working robot that could pick up a ball from the ground. Though it could locate it, it could not fully pick it up.

Individual Contributions:

- **Megan McGinnis**
    - Initially handled working with the Sony Aibo ERS7, using camera recognition for object detection, and working on getting the robot to walk around. After the switch to Tiago Steel, continued working on camera recognition and object detection. Attached a camera to Tiago Steel's camera slot, and using the camera's recognition node, had the robot identify the location of the ball. For an object to be detectable using the camera's recognition node, it must be a base node and have the recognitionColors set to a color. Once that has been done, if the object is visible to the camera it can be detected. Using inverse kinematics, created a path from Tiago Steel's starting location to the ball's ending location in the world which varied due to the P-Rob 3 being coded to throw the balls to random locations. Worked with Lukas to finalize Tiago Steel's controller, making some small adjustments to object recognition and the stopping distance between Tiago Steel and the ball.
- **Trevor Green**
    - Responsibilities:
        - Create the webots world
        - Develop controller for P-Rob3 (white/blue) that picks up 3 black balls and 1 yellow ball(target ball for Tiago Steel), then throws the balls at

randomized motor angles. To make the task of finding and retrieving the yellow ball more difficult for Tiago.
        - Added a manual control functionality to allow the user to manually change the angle at which the yellow ball is thrown.
    - Assist the rest of the team with complex coding issues i.e the inverse kinematics with Lukas
    - Organize the structure of directories
- Overview of project:
    - First week I focused on developing the webots world with all of the robots and objects that were going to be used within our project. During the second week I collaborated with Jonathan to develop the controller for P-Rob3 which allowed the arm to pick up and throw our balls. After some review from our professors and TAs during the third week we decided to split up mine and Jonathan's work for grading purposes. So I took over all of the work on the (white/blue) P-Rob3 arm which was responsible for picking up and throwing 3 black balls and 1 white ball, at a randomized motor angle. During the last week of our project I decided that I wanted to implement some extra functionality to my arm. So I developed a control_angle_to_throw_yellow_ball() function in c code that allows the user to manually control the angle at which the arm throws the yellow ball using the left, right, and down arrow keys. Aside from working on the P-Rob3 (white/blue) controller I also assisted Lukas with developing the inverse kinematics for the Tiago Steel arm. Lastly, I compiled all of the controllers and robots that everyone worked on into an intuitive directory structure.

- **Lukas Bush**
    - Handled working with originally the Sony Aibo ERS7 and object detection along with pickup objects. After complications, and switching our robot to the Tiago Steel, focused heavily on incorporating inverse kinematics with the Tiago Steel's arm. Created 3 different variants of the inverse kinematics which all worked to different degrees. The first followed the sample world inverse_kinematic.wbt which worked but the end effector was the wrist joint rather than the actual end. The second variant was with help from Shivendra and was the most accurate. The third came from the piazza post that gave an example of how to do so from a github. Tested but did not work properly for us. Worked with Megan to bring the Tiago Steel's controller together and make small adjustments to camera recognition.
- **Jonathan Conser-**

- P-Rob 3(red/yellow) arm code creation and maintenance was handled. Problems were included within trying to implement a throwing motion with limited velocity of motors,and not throwing balls that interrupt the other arms throwing motion. Jonathan implemented and coded an autonomous throwing motion that was slightly different behavior than the original throwing robot, does not have to throw the ball that is detected by the Tiago Steel, so the range of throwing does not have to be limited to a certain area that makes it detectable by the Tiago's side cameras.

Future Goals:

- For improving along with what we have, we would continue to work with picking up the correct ball once located. We also talked about making a true fetching simulator, meaning that the retrieving robot returns the ball to the vicinity of the throwing robot and the simulation continues into a loop where the throwing robot can detect that the ball has been dropped into its vicinity and will continue to pick it up and throw it while the retrieving robot will continue to retrieve and return it.

Downloading and Running Instructions:

Directory Structure:

- Final_Project_Submission
  - IKPY
    - Controllers
      - Ball_Tracker
        - Ball_Tracker.py
        - TiagoSteel.urdf
      - Ball_Tracker2
        - Ball_Tracker2.py
        - TiagoSteel.urdf
      - Ball_Tracker3
        - Ball_Tracker3.py
        - TiagoSteel.urdf
    - Worlds
      - .IKPY.wbproj
      - IKPY.wbt
  - Robotics_Final_Project
    - controllers
      - P-Rob3_RY_Jonathan_controller
        - P-Rob3_RY_Jonathan_controller.c
      - P-Rob3_WB_Trevor_controller
        - P-Rob3_WB_Trevor_controller.c
      - TiagoSteelController
        - TiagoSteelController.py
        - TiagoSteel.urdf

        - Worlds
            - Final_project_world.wbt
            - .final_project_world.wbproj

Explanation of main folders:

1. IKPY: This directory has our 3 inverse kinematics attempts, along with a small world in which to test. The first controller being the original inverse kinematics. The second being the most accurate one. The third is an attempt to try a different way then what was found in documentations.
2. Robotics_Final_Project: This directory contains our overall project submission where we combined all of our robots and controllers into one world to show off their capabilities.

How to download and run the worlds:

1. Go to https://github.com/trgr5899/Robotics_Final_Project and download the Final_Project.zip
2. Unzip the folder
3. Open Webots
4. Go to file=>open world
5. Then choose IKPY.wbt to look at our work with inverse kinematics or Final_project_world.wbt to look at our main project submission
6. In order to run any of the controllers within the directories you must have IKPY installed, which can be done using “pip install ikpy”.
7. Lastly before running the simulation make sure to build the P-Rob3_WB_Trevor_controller.c and the P-Rob3_RY_Jonathan_controller.c. This can be done by clicking on the little gear at the top right of the webots text editor while having one of the C files open in the text editor window. Make sure to do it for both files
8. Some other basic packages might need to be installed before running the simulation, but many of these can be installed by referencing google.
9. Now you are able to run our amazing simulations!