Extensible Telepresence Mobile Controller with Autonomous Track-and-Follow and Grasp Detection

Zichuan Huang (zh2280), Ying Zhu (yz3039), Xindong Zhang (xz2443), Yu Gu (yg2466)

Overview

This project aims to build a mobile platform to provide an extensible controller interface to control robots with mobile phone. The mobile platform chosen is iOS and the robot model chosen is Fetch. But since the main communication is through UDP sockets and it is interfacing with ROS, it is adaptable to any robot model using ROS and it could be easily changed to adapt to other mobile platforms. Majority of the tests are carried out using Fetch Gazebo simulator but it should work on real robots as well.

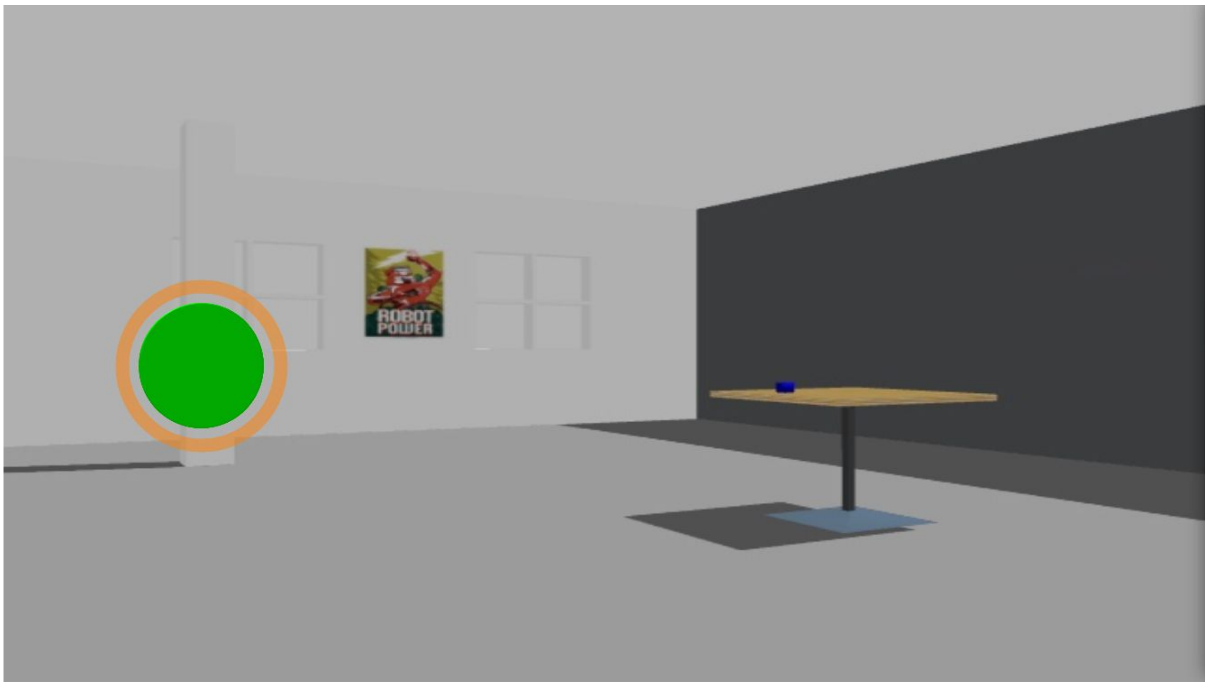
Components

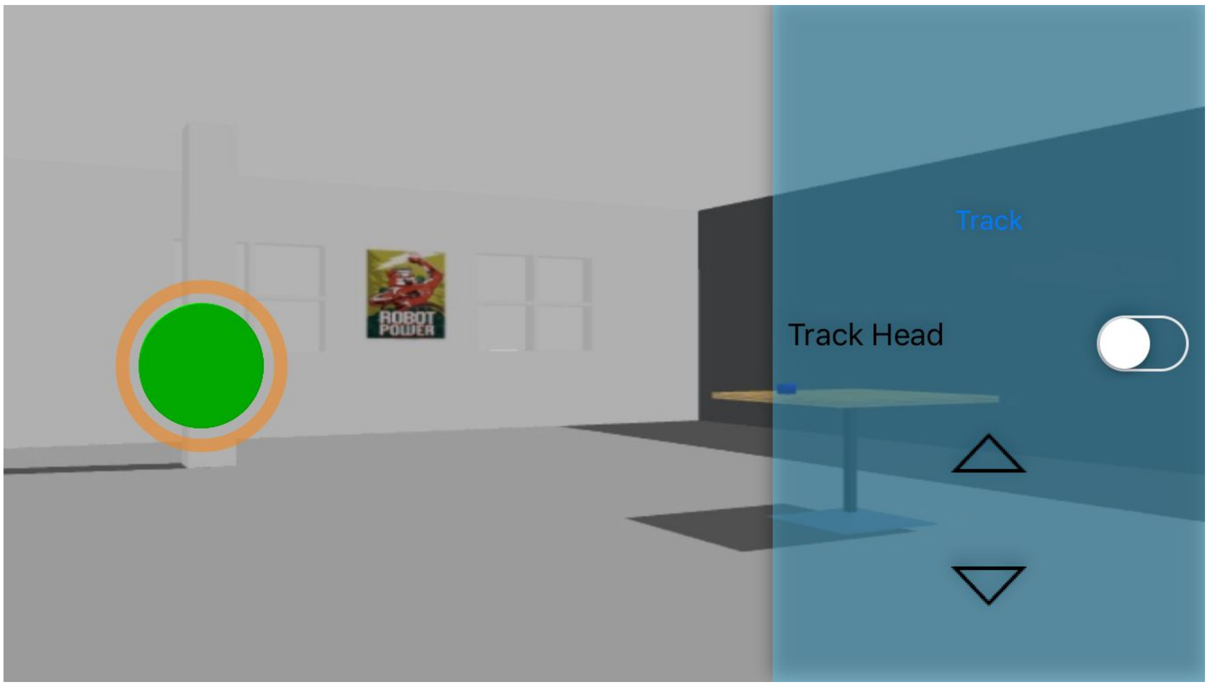
Enabling two-way data transfer

Our main source of data transfer between computer and mobile phone is through UDP sockets. Computer listens for UDP packet on port 10086 and mobile phone listens on port 10010. Packets are JSON that have the following structure:
{‘type’:’AAAA’,’data’:{SOMEDATA}}. UDP server would call corresponding functions according to the ‘type’ field in JSON received.

Building an iOS application to provide controller interface

Screen background is a webview for displaying what the robot sees. There is a joystick for controlling the movement of the robot. Besides, we have up and down button to control the torso. Moving the phone around can simultaneously control the robot head (Horizontal rotation, vertical rotation). The application can enable the ‘Tracking Head’ mode to allow robot head moving and it also has a track button to ask robot to start tracking an object that the user selects. This application serves as a UDP client to send instructions that user made to the server and also serves as a UDP server to receive object tracking window locations.





Move Controller video:
https://youtu.be/d2Nj9-KOiNQ

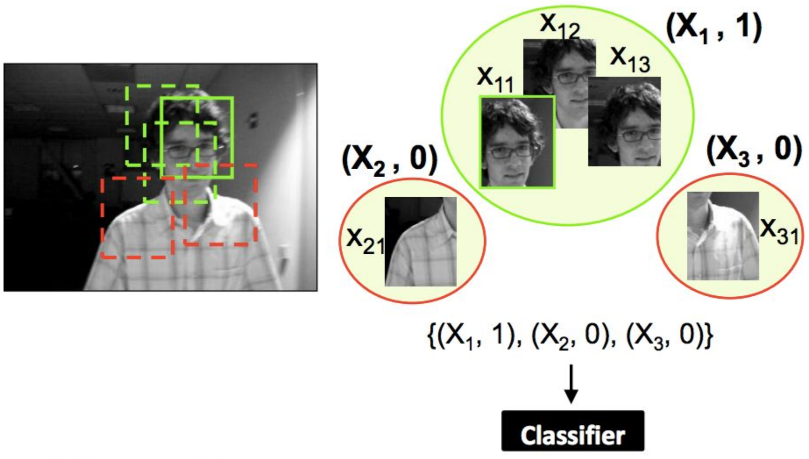
Establish video streaming over HTTP
Background overlay is accomplished using MJPEG streamer and async_web_sockets. We found an existing library web_video_server which combines these two. We changed parts of the code so that the resulting webpage shows nothing but the video. It is an ROS subscriber of topic head_camera/rgb/image_raw. It obtains the raw jpg image using cv_bridge and then

feed it into MJPEG streamer. The mobile phone or any other device in the network could see the stream by visiting the website with port 8080.

Telepresence by transferring movement of phone to robot's head
We obtain gyroscope reading from iPhone and send UDP packet with type 'track' to the server. The server has a simple action client for 'head_pan_joint' and 'head_tilt_joint' which corresponds to yaw and pitch reading from iPhone. Fetch has a pitch range of -45 to 90 degrees and a yaw range of -90 to 90 degrees, so the values received is clamped to this range and passed to simple action client. The update is sent once every 0.2 second so that the movement is swift.

Telepresence video:
https://youtu.be/3OEAcE1GVLI

Track-and-Follow extension
In order to demonstrate the extensibility of our mobile platform, we decided to incorporate a "Track-and-Follow" functionality. To use this function, the user will first drag a rectangle on the screen for the robot to track. The coordinates of the rectangle will then be transferred to a computer. The track controller is a subscriber of head_camera/rgb/image_raw. Whenever the topic updates, we use OpenCV to calculate the new rectangle in the new image. The program will control Fetch to move using the move controller we wrote according to the size of rectangle. Currently, the tracker algorithm that we use is MIL (Multiple Instance Learning). Given the tracker location, this algorithm will extract positive and negative examples from the current frame and train a classifier in an online manner to separate the object from the background (See the image below). After testing, we found out that this algorithm performs pretty well, however it will fail if there are full occlusion of the initial tracking object. However, in our code, it is easy to change the tracking algorithms to cater different needs.



Track and follow video:
https://youtu.be/TVgvI4vnZxQ

Grasp detection extension
We modified and re-implemented a grasp detection extension which allows the robot to detect graspable objects on a platform and determines the best grasp for each object using 1-dof hands. It is accomplished by first taking a background image before objects are placed on the platform. Then, new image is taken and subtracts the background image to obtain the objects of interest. Then the point cloud 2 is obtained from topic head_camera/depth_registered/points and converted to pcd format and fed into a Matlab program to find the best grasps. Every possible grasp is indicated by a rectangle. For each possible rectangle, the pre-trained deep learning network evaluates how good the grasp is. Then, it picks the one with highest score. It can be done for multiple objects too.
We aim to display the resulting grasps on mobile phone and let user choose which object to pick up and let fetch execute the grasp but this has not been accomplished yet. The system not supports taking saving pcdfile and png image for grasp detection. Though application interface hasn't been implemented.
The model (namely weights and data are trained using cornell database on local machine with GPU enabled). Specs:

Hardware	Alienware
GPU	NVIDIA 1070 16GB

Majority of the Code base was provided by the Cornell University Robot Learning Lab.(Since this is a huge project and it's impossible to implement everything from scratch in such short timeframe) This includes Data Processing, Assistive Functions, Training etc. To avoid copyright issue - all author information of the original code was maintained.

<u>Manual</u>
Zipped files contain folders:
Src: Contains files to be put in a ros-workspace src folder
Humanoid-ios: Contains files to build for ios app
Grasp_Code: Strengthened grasp detection - refer to README contained.

How to use:
Catkin_make in the main ros workspace folder.
After ros core is started (ie using fetch_gazebo or on real machine), run following commands in separate terminal windows:
Rosrun fetch_mobile mobile_controller.py (for udp server to communicate with mobile)
Rosrun web_video_server web_video_server (for camera streaming)
Rosrun fetch_mobile grasping.py (saves pointcloud2 in pcd format, background and detection image)

When app is opened, fill in the IP address of the computer hosting above servers and the update rate (rate at which movement updates, usually 0.2). Swipe left to see tracking, track head and torso controls.

Division of Labor

Zichuan Huang: UDP server, movement controller, head joint controller, video streaming
Ying Zhu: Track-and-Follow extension
Xindong Zhang: UDP client, IOS Development, Mobile Application
Yu Gu: PCD file processing, Grasp detection extension, Platform and robot urdf configuration

References

Lightweight tablet devices for command and control of ROS-enabled robots
https://www.researchgate.net/profile/Michael_Jenkin/publication/271549070_Lightweight_tablet_devices_for_command_and_control_of_ROS-enabled_robots/links/55a49cdf08ae81aec912f9c3/Lightweight-tablet-devices-for-command-and-control-of-ROS-enabled-robots.pdf

Deep Learning for Detecting Robotic Grasps
http://pr.cs.cornell.edu/papers/lenz_ijrr2014_deepgrasping.pdf

Web video server
http://wiki.ros.org/web_video_server

Async web server cpp
http://wiki.ros.org/async_web_server_cpp

Visual Tracking with Online Multiple Instance Learning
http://vision.ucsd.edu/~bbabenko/data/miltrack_cvpr09.pdf