Kendama-bot:
The Kendama-playing UR5

530.707 Robot System Programming
Prof. Simon Leonard
Nicholas Maritato, Seena Vafaee, and Charlie Watkins

Table of Contents:

Project Overview & How to Run

Overview

The aim of this project is to create the Kendama-bot: a modified UR5 with an attached Kendama that can throw and catch the Kendama ball. The project can be run entirely in simulation or it can be run on real hardware (note that only the ball launch procedure works on real hardware at the moment). The code for this project is completely open-source and can be accessed via the project's GitHub repo: https://github.com/Nick7244/RSP_Final_Project. Download the code and required dependencies into a catkin workspace, and build it using the "*catkin build"* command from within the workspace directory. Package descriptions and API for the codebase can be found in the documentation that follows.

Running in Gazebo Simulation

To run the project in the Gazebo simulation environment, run the launch file for starting the Gazebo world and loading the robot, controllers, etc.:

roslaunch ur5_kendama_gazebo ur5_kendama.launch

Then, in a separate terminal window, launch the deployer script for starting and interfacing the controller:

rosrun rtt_ros deployer -s {path to ur5_kendama_deployer.ops}

You can now use the various deployer operations defined by the controller to interface the UR5 via the deployer terminal. See the below documentation for the *ur5_kendama_control* package for a list of deployer operations and their function (NOTE: be sure to run "*cd robot"* in the deployer terminal before executing any of the operations).

Running on Real Hardware

To run the project in the Gazebo simulation environment, run the launch file for starting the Gazebo world and loading the robot, controllers, etc.:

roslaunch ur5_kendama_hardware_control ur5_kendama.launch

Then, in a separate terminal window, launch the deployer script for starting and interfacing the controller:

rosrun rtt_ros deployer -s {path to ur5_kendama_deployer.ops}

The same deployer operations can be used on the real robot as in the simulation.

ball_plugin

Overview

This package defines the Gazebo model plugin for the ball. The plugin gets the ball's cartesian position from Gazebo and publishes it.

API

Published Topics

/ball_position (ur5_kendama_msgs/ball_position)

Cartesian position of the ball in Gazebo with respect to the world coordinate frame.

ball_position_subscriber

Overview

This package contains the ball_pos_subscriber class. This class creates a subscriber to the ball position topic, and has getter functions for returning the ball's position and velocity. The position is gathered from the ball model plugin topic, and instantaneous velocity calculations are completed to get an estimate of the ball's vertical velocity (z-direction).

API

Subscribed Topics

/ball_position (ur5_kendama_msgs/ball_position)

Cartesian position of the ball in Gazebo with respect to the world coordinate frame.

ball_position_transformer

Overview

This package is intended to receive ball position messages in the camera coordinate frame and transform them to the world coordinate frame using a coordinate transformation from the hand-eye calibration (posted to the tf from the hand-eye package). Incoming ball positions in the camera frame are streamed from the ball_tracker node, and outgoing ball positions in the world frame are published to the ball_position topic.

Package Dependencies

This package depends on some standard ROS packages, including roscpp, geometry_msgs, tf2, and tf2_geometry_msgs. The package also uses the ROS realsense_camera package in order to stream images from the Intel RealSense R200 Camera. The realsense_camera package requires the librealsense package, which has prerequisites for installation which can be found here: http://wiki.ros.org/librealsense. These two packages can be installed with the following commands:

sudo apt-get install ros-kinetic-librealsense
sudo apt-get install ros-kinetic-realsense-camera

The package also depends on an internal package, ur5_kendama_msgs, which is a custom message package for this project. This package is included in the git repository for this project.

API

Subscribed Topics

/ball_pos_camera (ur5_kendama_msgs/ball_position)
Ball position in camera coordinate frame. Contains (x,y,z) in Float64 format.

Published Topics

/ball_position (ur5_kendama_msgs/ball_position)
Ball position in world coordinate frame. Contains (x,y,z) in Float64 format.

Launch Files

ar_tag_track.launch
Launches the realsense r200 nodelet, the aruco AR tag tracker, and the ball_position_transformer node, as well as the rqt_gui to visualize the AR tag tracking.

Running the Node

The package can be used with the following command, which launches the realsense r200 nodelet and rqt_gui to display camera images, as well as the ball_position_transformer node:

roslaunch ball_position_transformer ar_tag_track.launch

ball_tracker

Overview

This package wraps a blob tracker provided by the ViSP library. The tracker receives both color and depth images from the Intel RealSense R200 Camera, and the ViSP blob tracker is used to detect the centroid of a blob and publish the blob's (x,y,z) position relative to the camera.

Package Dependencies

This package depends on some standard ROS packages, including roscpp, std_msgs, and sensor_msgs. The package also depends on the visp and visp_bridge packages, which can be installed with the following commands:

sudo apt-get install ros-kinetic-visp
sudo apt-get install ros-kinetic-visp-bridge

The package also uses the ROS realsense_camera package in order to stream images from the Intel RealSense R200 Camera. The realsense_camera package requires the librealsense package, which has prerequisites for installation which can be found here: http://wiki.ros.org/librealsense. These two packages can be installed with the following commands:

sudo apt-get install ros-kinetic-librealsense
sudo apt-get install ros-kinetic-realsense-camera

The package also depends on an internal package, ur5_kendama_msgs, which is a custom message package for this project. This package is included in the git repository for this project.

API

Subscribed Topics

/camera/color/image_raw (sensor_msgs/Image)
    Color rectified RGB image published by realsense nodelet.
/camera/depth/image_raw (sensor_msgs/Image)
    Raw depth image published by realsense nodelet. Contains uint16 depths in mm.

Published Topics

/ball_pos_camera (ur5_kendama_msgs/ball_position)
    Ball position in camera coordinate frame. Contains (x,y,z) in Float64 format.

Launch Files

ball_tracker.launch
    Launches the realsense r200 nodelet, and the ball_tracker node.

Running the Node

The package can be used with the following command, which launches the realsense r200 nodelet, the aruco AR tag tracker, and the ball_tracker node:

roslaunch ball_tracker ball_tracker.launch

easy_hand_eye

Overview

This package handles the eye-on-base hand-eye calibration and ArUco marker tracking. The git repository (https://github.com/IFL-CAMP/easy_handeye) contains instructions for installing dependencies and further description of the package and its uses.

Four additional launch files have been added to this package. aruco.launch launches a node defined in the aruco_ros package with remapped topics and parameters to match the rest of the program. The marker id and size are set to match the large marker we use for the calibration. aruco_cube.launch is identical, but uses a different marker size to match the size of the cube marker used to demonstrate tracking.

kendama_calibrate.launch and kendama_publish.launch follow the outlines provided in the repository's README file.

API

Launch Files

kendama_calibrate.launch

> Launches the ur_modern driver node (for connecting to the hardware), the realsense r200 nodelet, arucos AR tag tracker, the hand-eye calibration node, and an rqt_gui for visualization of the video feed/AR tag tracking.

kendama_publish.launch

> Launches the calibration publisher to publish the hand-eye transformation to the tf.

Note

The README for the easy_handeye package on GitHub shows that when the publisher launch file is run, it should publish the transformation to a topic called "/calibration_result," which is why the ball_position_transformer subscriber that we created subscribes to that topic to get the transformation. We found that, despite what is noted in the README, this topic is actually never created and there is no indication of its creation in the source code.

industrial_msgs

Overview

This is a necessary dependency for ur_modern driver package. Check out the ROS wiki page for this package here: http://wiki.ros.org/industrial_msgs.

kendama_description

Overview

This package defines the kendama's description. The meshes folder contains .STL files that define the visual and collision meshes for the kendama, ball, kendama mount, and checkerboard mount. In the urdf folder, the kendama.xacro file defines the geometry of the kendama and mount and their relationship to the UR5's flange. The xacro also defines a joint to a link positioned at the center of the surface of the kendama's catch (cup_link).

ur5_kendama_control

Overview

This package defines the control procedures for the UR5 to launch and catch the kendama ball. This includes the trajectory generation and tracking via a ROS control position group controller. The main controller logic is in ur5_kendama_controller.cpp, which is an Orocos Task Context class type. This is so that the controller can be utilized as a Orocos Deployment Component, to be run at a specified frequency and so it can be interfaced with user commands from the Deployer terminal. The ur5_kendama_controller class creates the forward/inverse kinematics solvers for converting positions/velocities between cartesian space and joint space (utilizing the Orocos KDL library). It also creates a ReflexxesAPI object to be utilized for trajectory generation.

The Reflexxes library allows for creating smooth spline trajectories under a set of desired position/velocity/acceleration parameters and with the constraints of a set of maximum

velocity/acceleration/jerk limits. We utilize this library for creating joint space trajectories in all separate stages of the controller. The UpdateHook() of the ur5_kendama_controller class is called once every iteration of the Deployer Component timer, and iterates through the current desired Reflexxes trajectory, posting the commanded joint positions at that stage of the trajectory to the UR5.

The ball_launch_controller and ball_track_controller classes are created by the ur5_kendama_controller class. Upon creation, they are passed references to the ReflexxesAPI object as well as the forward and inverse kinematics solvers. The ball_launch_controller is responsible for creating the trajectory required to launch the ball up in the air, and the ball_track_controller is responsible for creating the trajectory that tracks the ball while it is in the air, and then cradling the ball into the kendama cup to catch it.

The ur5_kendama_controller class sets up an Orocos RTT output port that is utilized by the Deployment Component to stream the commanded joint positions to the UR5 position group controller topic. This class also sets up all the Deployment Component operations necessary for interfacing the controller from the Deployer terminal.

API

Subscribed Topics

/joint_states
: Current joint position and velocity posted by the joint state controller, which is connected to the UR5.

Published Topics

/joint_group_position_controller/command
: Commanded joint positions from the MsrJntState RTT output port are posted to this topic to command the UR5 to a desired position.

Deployer Scripts

ur5_kendama_deployer.ops
: Creates the ur5_kendama_controller object and sets the frequency of the controller to 100Hz. This allows for interfacing with the controller via the below deployer operations.

Deployer Operations

GetJointPos
: Returns the joint state of current iteration of the Reflexxes trajectory.

SetJointPos
: Takes in a set of desired joint positions and velocities and commands Reflexxes to generate a trajectory to that desired state.

TPose
: Creates the trajectory necessary for the UR5 to achieve T-Pose, which is the initial launch position of the robot.

ZeroPose
: Creates the trajectory necessary for the UR5 to return to the zero pose.

LaunchBall
: Calls upon the ball_launch_controller to generate the trajectory necessary to launch the ball in the air. Once the ball is launched, the controller switches to call on the ball_track_controller to generate the trajectory necessary to track and catch the ball.

TrackBall
: Calls upon the ball_track_controller to track and catch the ball. This is useful if you desire to simply drop the ball from an arbitrary position and want the UR5 to catch it after it is dropped (as opposed to having the UR5 launch the ball up in the air on its own and then catch it).

Running the Node

The package can be used with the following command, which launches the deployer script to start and interface the controller:

rosrun rtt_ros deployer -s {path to ur5_kendama_deployer.ops}

ur5_kendama_description

Overview

This package contains an xacro that joins the kendama model (kendama and mount) to the ur5 and a launch script that uploads the robot description and ball_plugin to the parameter server.

API

Launch Files

ur5_kendama_upload.launch
: Loads the UR5 with attached kendama to the parameter server under the robot_description parameter. If the simulated UR5 is utilized, this also loads the ball_plugin to the parameter server to publish the position of the ball in Gazebo to the /ball_position topic.

ur5_kendama_gazebo

Overview

This package is the highest level package to initiate connection to the simulated UR5. The worlds folder contains a saved gazebo world file with a simulation environment containing the kendama ball. The launch folder contains a launch script that starts the world and spawns the robot and ball_plugin. Parameters in this launch file will be passed on to set parameters of the controller.

API

Launch Files

ur5_kendama.launch

> Starts gazebo with the specified world file, calls upon ur5_kendama_upload.launch to load the UR5 and ball_plugin xacros to the parameter server, spawns the UR5 and ball_plugin into the Gazebo world, and launches the joint_group_position_controller to be able to control the position of the joints of the UR5.

ur5_kendama_hardware_control

Overview

This package is the highest level package to initiate connection to the hardware UR5. It contains the launch files necessary for connecting to the hardware UR5 via ethernet, starting the position controller, and sending the robot_description to the parameter server.

API

Launch Files

ur5_kendama.launch

> Calls upon ur5_ros_control.launch to connect to the robot ethernet with the correct port, sets the “launch_only” parameter to false (this is utilized in the ur5_kendama_controller class to turn the controller into launch only mode, which disables the ball tracking controller since we were unable to get the ball_tracking working for the real system).

ur5_ros_control.launch

Connects to the hardware UR5 via ethernet utilizing the ur_moder driver package, calls upon ur5_kendama_upload.launch to load the UR5 xacro to the parameter server, and launches the joint_group_position_controller to be able to control the position of the joints of the UR5.

ur5_kendama_msgs

Overview

This package contains one message definition: ball_position.msg. The message contains three std_msgs/Float64 (x, y, and z) that define the cartesian position of the ball. This message is used to stream the ball cartesian position on the /ball_position topic.

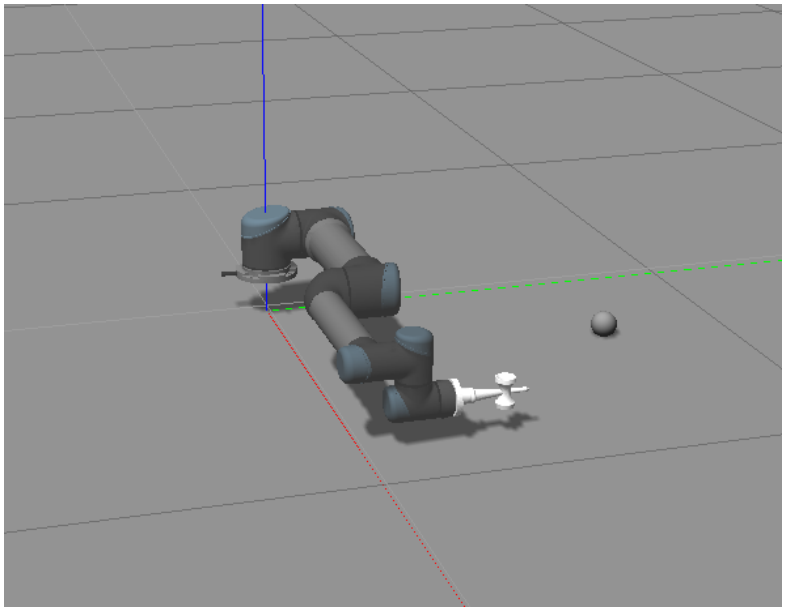
ur_modern_driver

Overview

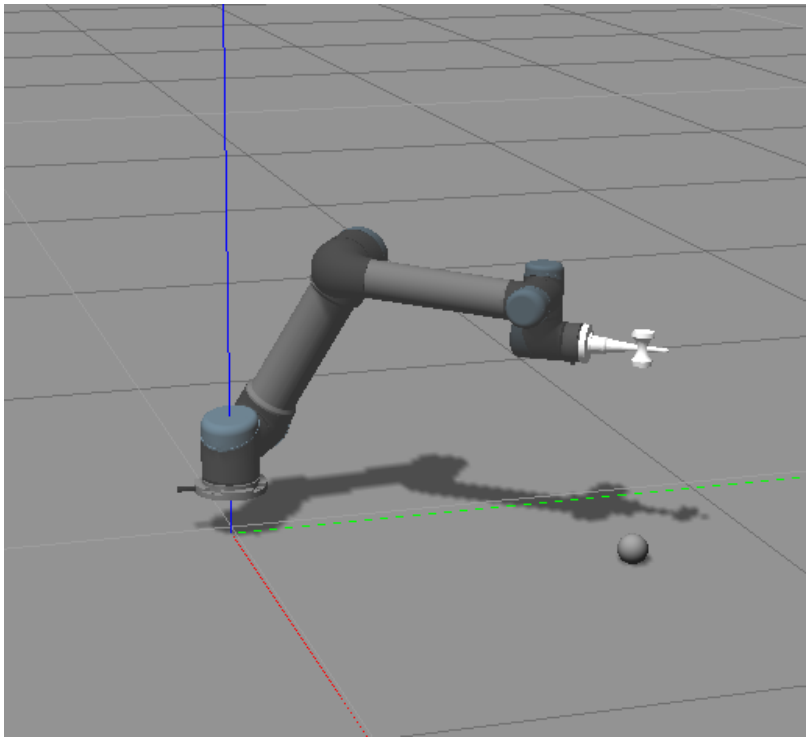
This is a necessary package to connect to the hardware UR5 over ethernet. The github repo for this package is found here: https://github.com/ros-industrial/ur_modern_driver/tree/kinetic-devel.

Media

Zero Pose:



T-Pose:



Video Demonstrations:

Launch & catch in simulation:
https://drive.google.com/file/d/1e1bIxPwHXHm0sWLDqQTGbGQsdl5RmOAp/view?usp=sharing

Track & catch ball from drop in simulation:
https://drive.google.com/file/d/12dDdwwc-xz-ZK-HuJ4svwzMW4inStn50/view?usp=sharing

Launch ball on hardware:
https://drive.google.com/file/d/1tAfh9NtdSdnuiOwJ8pfCJzHRjNh9VhXG/view?usp=sharing