Alvar Tag Tracking using Intel Realsense D435 Camera Repository Structure

Note: *This document provides a summary of the various files stored in the GitHub repository and provides instructions of moving files to particular locations on the user's machine. This document should only be utilised and the respective files moved AFTER the installation process outlined in the README file has been successfully completed.*

The GitHub repository developed for the Alvar Tag tracking functionality for the Algae Lab KUKA robot consists of two primary directories:

- realsense_ar_tracking_2 – primary ROS package containing all required launch files, source C++, header and XML files needed to complete the required functionality of the project.
- RealsenseFiles – modified URDF and launch files which need to be moved into the correct location after installation of all the dependent packages.

'RealsenseFiles' Directory

This directory consists of three files:

- rs_d435_camera.launch – This launch file was produced during the development of the code and launches the Intel Realsense node (setting up the connected camera) and begins publishing all the internal camera frame transforms to the TF node. This file can be modified to **change the resolution** of the various cameras in the Intel Realsense D435 camera, by editing lines 9, 10, 13, 14, 17 and 18. Upon successfully installing all the dependent packages outlined in the README file, this launch file should be copied to the following location:

  ~/catkin_ws/src/realsense-ros/realsense2_camera/launch/

- _d435.urdf.xacro – This URDF file is used to define the various internal transforms of the Intel Realsense D435 camera. Only a minor modification was made to this file as the latest iteration of the realsense ROS node modified this code but had not been tested and validated. The only modification made was the commenting of lines 144 – 150 to allow the node to successfully launch and operate correctly. Upon successfully installing all the dependent packages outlined in the README file, this URDF file should be copied to the following location:

  ~/catkin_ws/src/realsense-ros/realsense2_description/urdf/

- test_d435_camera.urdf.xacro – This URDF file is instantiated by the launch file described above and simply defines the reference frame against which the remainder of the above URDF file is built upon. Again, only a minor modification to the link name was made so that conflicts with the existing system would not occur. Upon successfully installing all the dependent packages outlined in the README file, this URDF file should be copied to the following location:

  ~/catkin_ws/src/realsense-ros/realsense2_description/urdf/

‘realsense_ar_tracking_2’ Directory

Launch Files

Numerous launch files were generated throughout the development of the ROS package. These launch files are described in more detail below:

- camera_calibration.launch – This launch file sets up all the required ROS nodes and runs the required C++ files to conduct the Robot Base to Camera calibration process. This launch file starts the Alvar tag tracking package as well as the camera_calibrator node.

- sealer_fridge_calibration.launch – This launch file sets up all the required ROS nodes and runs the required C++ files to conduct the calibration of the Sealer/Fridge tags to the respective points of interaction. This launch file again starts the Alvar tag tracking package as well as a broadcaster node which publishes the calibrated Base to Camera transform and the sealer/fridge calibration node.

- realsense_track.launch – This launch file is the fundamental launch file which runs the required tag tracking functionality for the overall intended performance of the Algae Lab robot. This is the primary launch file which runs the required ROS packages and nodes to produce the intended output of the relative positions of the sealer/fridge with respect to the robot. This launch file instantiates the Alvar package, a broadcaster node to publish the calibrated Base to Camera transform and the tag tracking node.

- test_alvar.launch – Finally, this last launch file was developed to provide ease of conducting the testing of the precision of the Alvar package. It instantiates the Alvar tag tracking package as well as a simple alvar testing node used to simply print the transforms to the terminal for the user to utilise throughout the testing procedure. This launch file and the associated nodes are not integral to the implementation of the overall system and the intended output.

Source Files

The functionality of the package was written using the C++ programming language. This section provides a detailed description of the numerous C++ and Header files that were written during the development of the package.

- alvar_tester.cpp & alvar_tester.h – These C++ and Header files were used to develop a node which tested the precision and accuracy of the Alvar tag tracking package. This node was fairly simple in nature in that it would only obtain the relative transforms from the camera to the calibration tag and display the current transform and the rolling average to the terminal. The user would then utilise this data to perform an analysis of the overall precision and accuracy of the Alvar ROS package.

- calibrator.cpp & calibrator.h – The calibrator C++ and Header files provide all the functionality required for both the Base to Camera and Sealer/Fridge Tag to point of interaction calibration processes. The calibrator.cpp file contains independent functions for each of these calibration processes, however, performs all the required data collection from the Alvar package, transform multiplication and terminal printing required for the user. Upon running the calibration process(es), the user is prompted to move the camera with respect to the Alvar tags (calibration tag for base to camera calibration or the sealer/fridge tag for respective calibration) for a specific number of iterations (10 for base to camera and 5 for equipment calibration). Upon completing the number of iterations, the average relative transform is displayed on the terminal and the user will be required to edit the tf_broadcaster.cpp file with the new calibrated transforms.

- camera_calib.cpp – This is a simple C++ Main function used to establish the ROS node and to instantiate two important threads to run during the Base to Camera calibration process. The first thread runs the calibrateCamera function from calibrator.cpp whilst the second runs a broadcasting function to continuously broadcast the calculated base to camera transform to the TF tree for ease of visualisation throughout the process.

- main.cpp – Another simple C++ Main function used to establish the ROS node for the overall AR tag tracking functionality. This function initialises all the required ROS nodes and communications and instantiates a thread to continuously track the AR tags in the Algae Lab environment. The thread produced in this main function continuously runs the ‘trackTags’ function from the tracker.cpp file

- sealer_fridge_calib.cpp – Similarly to camera_calib.cpp this is a C++ Main function used to establish the ROS node and instantiate two important threads to be run during the Sealer/Fridge tag to target location calibration process. The first thread runs the calibration function which continuously utilises the Alvar ROS package data to calculate the relative transforms between the tags and the intended target locations. Whilst the second thread is used to continuously broadcast the calculated transforms to the TF tree for ease of visualisation throughout the process.

- test_alvar.cpp – The last C++ Main function establishes all the ROS communications and instantiates a thread to run the Alvar package testing functionality. Alongside alvar_tester.cpp, this function creates the required node to test the precision and accuracy of the Alvar tag tracking software.

- tf_broadcaster.cpp – This C++ file is again, another very simple program used to continuously broadcast the calibrated transforms to the TF tree for use in the overall system. This function provides the relative base to camera transforms as well as the sealer/fridge tag to target location transforms to the TF tree. This function is important as it provides the required transformations (calibrated during the set up of the system) to complete the kinematic loop in the Algae Lab. Using this information the relative transforms between the camera and the equipment can be easily obtained and fed into future functionalities of the robot. The relative transforms between the Sealer/Fridge tags and target locations still need to be calibrated and broadcasted in this C++ file.

- tracker.cpp & tracker.h – Finally, these C++ and Header files provide the overall intended functionality of the Algae Lab's Alvar tag tracking processes. This file provides quite simple, yet valuable functionality to the overall system. The main functionality in this file continuously checks the data obtained from the Alvar ROS package and checks if it can see either the Fridge or the Sealer Unit in the camera's FOV. Upon determining which object it can see, the function simply broadcasts the relative transform between the camera and the respective object's tag bundle to the TF tree. This relative transform would then be utilised in the future to determine the relative position between the robot and the point of interaction with the equipment and the robotic arm will be instructed to drive to the required position. This functionality endlessly loops until the ROS node is terminated.

AR Tag Bundles

The 'bundles' directory contains three XML files used to define the relative positioning of the AR tags in the tag bundles with respect to the primary, Master tag. The three following XML files are integral to the operation of the overall system and are specific to each of the pieces of equipment inside the Algae Lab:

- sealer.xml – This XML file defines the relative positioning of tags 1, 2 and 3 with respect to the Master tag 0 for the Sealer Unit. This XML file should only be modified if a clear and obvious error has been made with defining the relative positioning of the tags in the bundle. Changing these parameters will significantly impact on the performance of the Alvar tag tracking package.

- fridge.xml – This XML file defines the relative positioning of tags 5, 6 and 7 with respect to the Master tag 4 for the Fridge. This XML file should only be modified if a clear and obvious error has been made with defining the relative positioning of the tags in the bundle. Changing these parameters will significantly impact on the performance of the Alvar tag tracking package.

- calibration.xml – This last XML file defines the Master ID tag 8 used for the calibration of the Base to Camera transform as well as in testing the performance of the Alvar package. Whilst not technically a 'bundle' (as only a single tag is defined), implementing this XML file significantly reduced the level of effort required in establishing the ROS nodes and allowed for a single Alvar node to be utilised.