# Camera to Robot Calibration

## Introduction

This report summarizes the work performed to install and test an automated camera-robot calibration system using the ROS-Industrial calibration library. The system was installed at NIST in the cooperative robotics lab using a UR10 robot and 6 different cameras. A calibration script was designed which, upon command, re-calibrates the extrinsic parameters of the cameras. Analysis software was also developed and tested which predicts the accuracy of the network of cameras based on the calibration results. The efficacy of the analysis software has not yet been verified due to some technical difficulties. A short tutorial and demonstration of the system’s function and technical details was also provided.

The calibration system is designed to be part of a ROS-Industrial installation. Once configured, it provides repeatable calibration results for all the cameras with little input from the user. The calibrated extrinsic parameters are updated such that subsequent launches of the ROS system incorporate the latest calibration results. In the future, users may add, remove or re-position some or all of the cameras with very little effort. Users simply modify the camera definition file and modify or add new observation commands to the scene definition file.

## Setup Details

The calibration system requires a basic ROS installation for the UR10 and a camera interface. The ROS installation process is straight forward, but involves installing all required packages used by the installation. The non-standard packages used here are the Ceres-Solver and the Aravis-Camera, ROS-Industrial-Calibration and the Universal-Robot packages. Once these packages were installed, a number of files had to be created to build the robot-camera system. One set of files launches a node for each camera which publishes rectified images from each camera. Another set of files interfaces ROS-MoveIt to the UR10 so that collision free motion may be automated. Another set of files adds the calibration service to the system.

### Camera Setup

The camera nodes are all launched using the “*nist\_cameras.launch*” file. This file individually sets the exposure, frame rate, and other parameters for each camera. It is responsible for connecting to each camera, streaming images raw images and also for processing them to create rectified images using each camera’s unique intrinsic calibration parameters. After adjusting its focus, aperture and exposure, each camera had to intrinsically calibrated to account for lens distortion. The calibration data is used both for generating streams of rectified images, and for computing the extrinsic calibration parameters. It should be noted that camera setup is non-trivial. Without proper focus, white balance, etc, the system will be unable to reliably detect targets for extrinsic calibration. Finally, an initial guess for each camera’s pose has to be entered into the system. The initial pose should be within a few feet and within 10-15 degrees in roll, pitch and yaw of the correct values.

### Robot Setup

Setting up the robot for automatic path planning is significantly simplified using the MoveIt Setup Assistant. The main task is to add the necessary geometry to the URDF. For this installation, we modeled the cameras, the table and other structure that the robot might collide with using simple geometry. We also added the calibration target to the end-effector. The main customizations necessary for calibration is to use the “*calibration\_transform\_macro.xacro*” to attach each camera to the world frame. This macro generates a sequence of 6 links and joints, one each for x, y, z, roll, pitch and yaw. In addition, the launch files which load the robot controller and the robot model need to combine the robot’s joint states with the calibration joint states. Finally, the “mutable\_joint\_state\_publisher” node must be included to provide an interface to the calibration values.

### Calibration Setup

To setup the calibration one needs to define three input files, one to define the target, one to define the cameras and one to define observation scenes. The target definition file is self explanatory. The camera definition file has one critical aspect, the transform interface. In this example, all the cameras use the “*ros\_camera\_scti*” transform interface. This interface provides a programmatic link to the x,y,z,roll,pitch and yaw values defined by the “*calibration\_transform\_macro.xacro*” in the URDF. The file defining the observation scenes is the most complex. To ease the creation of this file, an interactive script was written to automatically generate scenes from robot poses. To use the script, one simply moves the robot around, verify that the desired camera has the target within its field of view and then set the ROS parameter “/caljob\_creator/capture\_scene” to true. For this application, we generated a sequence of scenes for each camera designed to cover as much of the field of view as possible at several different distances.

## Calibration Execution

As is typical of ROS robot installations, the system is started by running one or more launch files. Three separate launch files are currently necessary to start the system, but these could easily be combined into a single launch file. These launch scripts include one to set up the interfaces between MoveIt! and the robot controller, one to run all the cameras, and one to run the calibration service. An optional 4th launch script starts up *rviz* for visual feedback. Once all these nodes are running the calibration routine is executing by calling the “*calibration*” service. Typical installations will incorporate the calling of this service in their operator GUI.

## Calibration Results

The calibration script for this installation consists of 150 robot poses. Some cameras have a wider variety of observations than others. More diverse data generally results in more consistent results. However, other issues may affect the performance of the system. For example, the target is imprecisely mounted on the robot’s end effector using oversized mounting holes. Therefore, the transform between the end effector and the origin of the circle grid is not well defined. Differences between the modeled and the actual transform to the target can manifest as increased variance in camera poses estimates. When all observations of the target by a camera have similar orientation, the result is a bias in the camera’s pose. When the target is observed with multiple target locations this offset instead increases the variance observed in the camera’s pose estimate. The following table displays the variance in pose for each camera observed from 16 different runs of the calibration process. Note, that the fourth and fifth cameras have much larger variance while the sixth camera has the lowest variance. The fourth and fifth cameras are a stereo pair and have very similar observations. For these two cameras, we used lots of rotation changes between the target and the cameras. On the other hand, for the sixth camera, we did not modify the rotation at all, and collected lots of poses throughout this camera’s field of view.

Table 1 The precision of the cameras' extrinsic calibration

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Precision | X (mm) | Y (mm) | Z (mm) | Pitch(degrees) | Roll(degrees) | Yaw (degrees) |
| Basler 1 | 0.17188 | 0.12766 | 0.12766 | 0.0071 | 0.0044 | 0.0048 |
| Basler 3 | 0.36114 | 0.36901 | 0.02599 | 0.0108 | 0.0142 | 0.0143 |
| Basler 4 | 2.19701 | 3.82132 | 1.07848 | 0.3348 | 0.1552 | 0.1704 |
| Basler 5 | 0.81040 | 0.93087 | 0.66695 | 0.1071 | 0.1026 | 0.0237 |
| Basler 6 | 0.0177 | 0.01254 | 0.01466 | 0.0007 | 0.0006 | 0.0013 |
| Basler 7 | 0.02336 | 0.04156 | 0.03981 | 0.0025 | 0.0013 | 0.0015 |

## Accuracy Analysis

The calibration library was modified to export the observation data for an accuracy analysis. The analysis assumes that fiducial localization noise drives the noise model of the system. Typical residual error in the intrinsic calibration of one tenth of a pixel per observation was with a Monte-Carlo process to estimate the accuracy of the calibrated camera network for localization of targets within the working volume. The analysis has two parts. First the pose accuracy is estimated by re-running the calibration process with observations corrupted by randomly generate zero-mean noise with a standard deviation of 1/10th pixel. One hundred synthetic calibrations were run to generate the expected pose statistics for each camera.

Next, the triangulation accuracy of the camera system was estimated with a second Monte-Carlo process. For this process, we simulated localizing each point on a uniform grid covering the workspace using the camera network for triangulation. Each camera that can observe a point contributes to its location. The location of each point was computed 100 times to estimate statistical variations due to both pose estimation and fiducial localization error. Camera poses were perturbed using samples generated from the statistics estimated in the previous simulation. The variance in position at each grid point is affected by the number of cameras that have it within their field of view, its geometric relationship to those cameras and the precision of their pose estimates. The following figures show the predicted accuracy at 0.0, 0.2, 0.4 and 0.6 meters above the table’s surface respectively.

|  |  |
| --- | --- |
| heatimage0.jpg | heatimage2.jpg |
| 0.0 meters | 0.2 meters |
| heatimage4.jpg | heatimage6.jpg |
| 0.4 meters | 0.6 meters |

Figure 1 Triangulation Accuracy for a 6 camera system calibrated using a UR10 robot

## Conclusions

The Industrial Calibration library was designed to provide robot installations with an automated method for repeatable and accurate determination the pose of cameras and robots relative to one another with little or no user intervention. The system was installed on a UR10 robot in the NIST cooperative robotics laboratory. A calibration process was defined and executed a number of times to assess the repeatability of its results. The camera network’s accuracy was also predicted using a Monte-Carlo simulation. Future testing will attempt to verify the predicted localization accuracy is achieved.