Controls for Assistive Robots

Guangzheng Zang

Controls for Assistive Robots

by Guangzheng Zang

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Anca Dragan

Research Advisor

* * * * * * *

Professor Jitendra Malik

Second Reader

05/11/2017

Abstract

The industry of robotics is on the rise; an increasing number of robots are being used in factories and households. Our capstone project – Controls for Assistive Robots – deals with this market demand. We aim to implement and integrate various subsystems of UC Berkeley's InterACT lab to enhance its infrastructure. My technical contribution is a robot perception system that gives the robot the ability to "see" and interact with the environment. Specifically, we implemented a vision system using Microsoft's Kinect v2 sensor. We managed to make the Kinect a quantitative sensor ready to use in research. We also integrated it with other subsystems in the lab and build a system so that the robot could detect the pose of certain objects in real world, and plan to grab them in a 3D virtual environment. With the systems we developed, future researchers in the lab will be able to conduct their own research and keep making a meaningful impact to the robotics industry.

Table of Contents

INTRODUCTION

Our team aims to implement and integrate certain components of UC Berkeley's InterACT Laboratory in the EECS department. In order to achieve this, we built a system that allows a robot arm to perceive the pose of an object and plan to grab it. More specifically, we want to set up and calibrate Microsoft's Kinect v2 sensor as the robot arm's perception system, have it work with AprilTags, a fiducial marker developed by april robotics laboratory (april robotics laboratory, 2016) so that the system could recognize an object's 6D pose (x, y, z, yaw, pitch, roll), and apply trajectory smoothing algorithms in order for a robot arm to grab it. Table 1 shows a work breakdown of our project.

Team member	**Project component**	**Description**
Guangzheng (Jeff)	Perception System	Set up and calibrate Kinect v2 sensor
Tyler	AprilTag Recognition	Recognize the 6D pose of objects using AprilTags
Geronimo	Motion planning	Configure a smooth trajectory to grab the object

Table 1, Work breakdown of project

PROBLEM STATEMENT

This paper focuses on the implementation of an imaging system for the InterACT Laboratory. The aim is to properly set up the imaging sensors, allowing the system to get reliable data from the environment so that the robot could "see" the world. In future steps, the imaging system will be used to take images containing certain objects with AprilTags on it. Therefore, setting up the imaging system is the first step of our project.

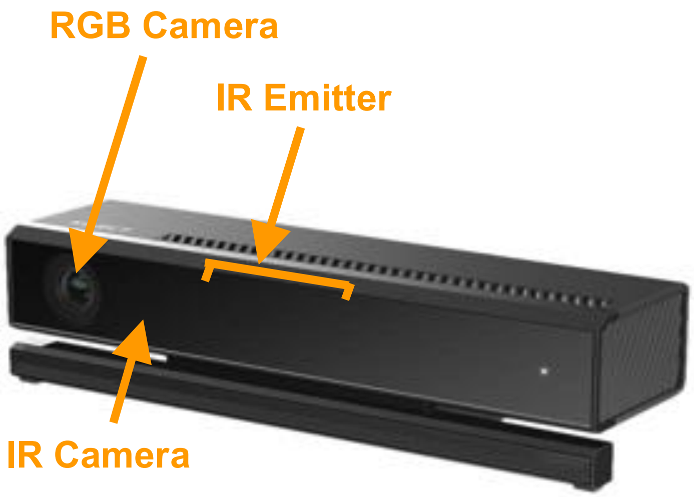

Image 1 (Derivative, n.d.), *Kinect V2 sensor*

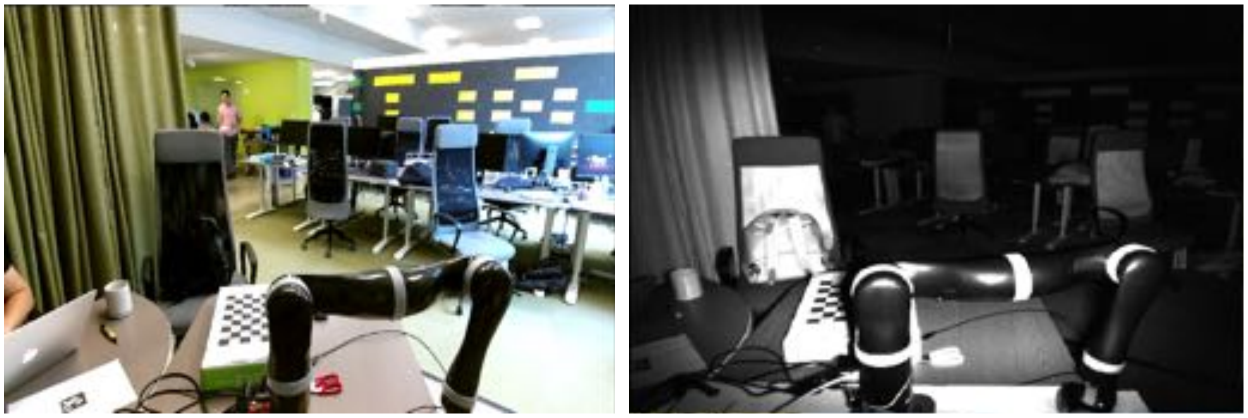

Image 2, Sample RGB image (left) and Depth image (Right)

To get the images, we use Microsoft's Kinect v2 sensor (shown in Image 1), which is able to take both RGB and depth images (shown in Image 2). The RGB images are simply color images, which are no different from the images seen by human's eyes; the depth images record the depth of each pixel from the camera, giving the robot a sense of the 3D shape of objects. This sensor has one color camera, one IR camera, and three IR emitters. The IR emitters project IR light which is reflected once hitting an object; the reflected light is perceived by the IR camera. Based on this set up, the depth of each pixel is calculated based on the Time of Flight (ToF) scheme: the distance from the camera to each pixel is proportional to the time it takes for the IR signal to travel to that pixel and bounce back to the IR camera. Some of the technical features of the sensor is shown in Table 2.

IR camera resolution	512 x 424 pixels
RGB camera resolution	1920 x 1080 pixels
Field of view	70 x 60 degrees
Frame rate	30 frame per second
Operative measuring range	0.5 to 4.5 meters
Object pixel size	1.4mm (@0.5m) to 12mm (@4.5m)

Table 2, Technical Features of Kinect V2 sensor (Lachat, 2015)

However, simply hacking into the sensor and extract the color and depth frames would give imprecise images: there are three pitfalls in this seemingly simple procedure. First, the Kinect sensor uses Digital Single Lens Reflex (DSLR) cameras, and therefore geometric distortion exists. As shown in Image 3(a), this distortion is most observable when there are pairs of parallel lines in the image. Using these distorted images would lower the accuracy of the image and hurt the performance of the project. Second, due to the imperfections of the camera, there is a mismatch between the actual image taken and the image expected by a pinhole camera model. These imperfections could include shift in the principal point (the center point of the image plane), different pixel scaling factor along the two axes of the image, etc. Third, as shown in Image 1, there is a small distance (~2.5 cm) between the centers of IR and RGB camera on the Kinect sensor. This causes a shift between the color and depth images. This shift can be seen in Image 4 (a).

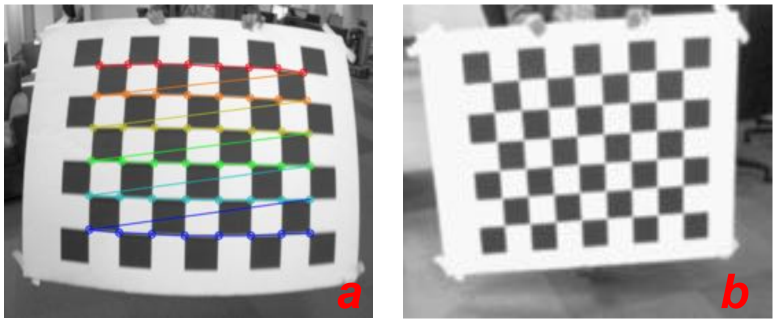

Image 3 (ROS Wiki, 2015)*, Intrinsic/distortion Calibration of Monocular Camera: (a) with geometric distortion, straight lines appear curved; (b) after calibration, straight lines appear straight*

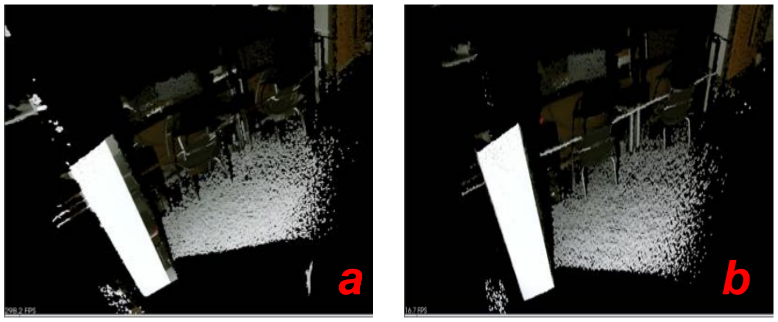

Image 4 (Wiedemeyer, Kinect2 Calibration , 2016)*, Calibrate IR to RGB camera: (a) with the shift, the flat surface (white) does not perfectly cover its color image; (b) after calibration, the flat surface (white) perfectly covers its color image.*

In order to eliminate these distortions and to achieve precise image data, we need to perform calibration on the cameras. Specifically, distortion calibration would counter the effect of lens (radial and tangential) distortion, intrinsic calibration would counter the imperfections of the camera, and a dual camera calibration on the IR and RGB camera would solve the mismatch problem of the two images. Details on this will be discussed in "Theoretical Background" section.

Unfortunately, being able to get precise images does not equal to a successfully set up sensing system; in most cases, including this capstone project, the 6D pose of the camera itself relative to a shared coordinate system is needed in order to make sense of the images. In the next steps of our project, we will use AprilTags to detect the 6D pose of an object relative to the camera. If the robot arm cannot access the position and orientation of the camera in some shared coordinate system, the detected pose of the target object is meaningless because there is no way for the robot arm to convert that detected pose (relative to camera) to the pose relative to itself, thus forbidding the arm to really perceive the object.

In summary, it comes to the following challenges in order to build the imaging system:

1. Figure out the intrinsic parameters of the camera (Intrinsic calibration)
2. Fit the image to a non-linear transform to counter lens distortion (Distortion calibration)
3. Calibrate the IR camera to RGB camera (Dual lens calibration)
4. Obtain the 6D pose of the camera itself in a shared coordinate system (Extrinsic calibration)

Theoretical Background

As discussed in "Problem Statement" section, in order to make Kinect v2 a quantitative sensor, we need to apply certain camera calibration procedures. Therefore, this section will emphasize on the 3D to 2D projection process in a camera as well as the role that camera calibration is playing.

A camera can be generally modeled as a pinhole camera, assuming that the image plane is in front of the camera center – shown in Image 5. Here, a point X in 3D with coordinate (X, Y, Z) is projected onto the image plane with respect to the camera center C. The resulting pixel is denoted as x. The Z-coordinate of x (or any other point on the image plane) is defined as the focal length (f) of the camera. To calculate the Y-coordinate of x, we analyze the similar triangle shape shown on Y-Z plane located at right side of Image 5. The law of similar triangle gives us:

$$y = fY/Z$$

Similarly, from the similar triangle on the X-Z plane, we have:

$$x = fX/Z$$

Therefore, the pixel x has coordinate $(x, y) = (\frac{fX}{Z}, \frac{fY}{Z})$.

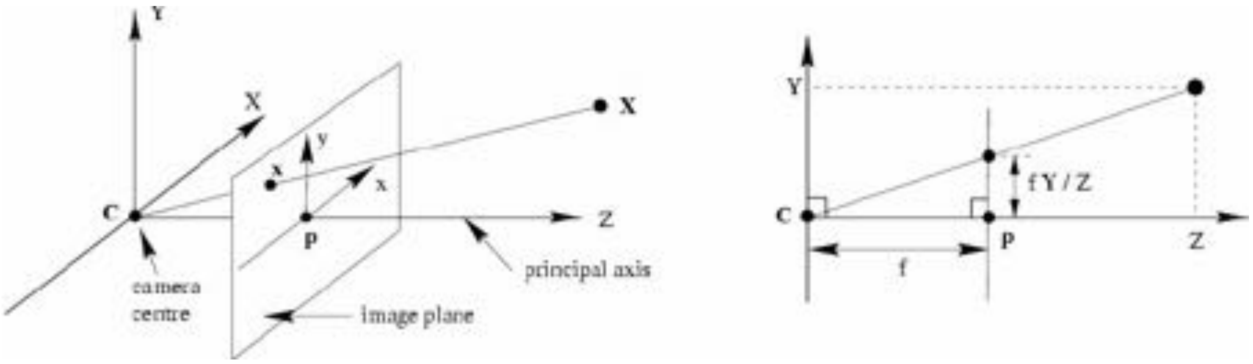

Image 5 (Forsman, 2011)*, Simple Pinhole camera model. Notation: X is point in 3D world, x is point on 2D image, C is Camera center, P is principal point, f is focal length*

In homogeneous coordinates (see Appendix A), the transformation from (X, Y, Z) to $(\frac{fX}{Z}, \frac{fY}{Z})$ can be written as:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{fX}{Z} \\ \frac{fY}{Z} \\ 1 \end{pmatrix} = \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

This model is ideal and does not take into account three practical non-idealities. First, the (x, y) coordinate of the pixel in the ideal model might not be reflected on the real image. Three intrinsic imperfections of the camera might contribute to this effect:

1. The center of the image, i.e. point $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ in homogeneous coordinate, might be shifted in the real image. The shift is denoted as $\begin{pmatrix} u_c \\ v_c \end{pmatrix}$.

2. The camera might have different scaling factors (α_x, α_y) along x and y directions when taking the image.
3. There might also be a skew coefficient (γ) between the x and y axis.

These factors along with the focal length f are commonly referred to as the intrinsic parameters of a camera. Taking all these factors into account, the real image coordinate $\begin{pmatrix} u \\ v \end{pmatrix}$ becomes:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \alpha_x & \gamma & u_c \\ 0 & \alpha_y & v_c \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_x & \gamma & u_c \\ 0 & \alpha_y & v_c \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} f\alpha_x & f\gamma & u_c \\ 0 & f\alpha_y & v_c \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = K \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

where $K = \begin{pmatrix} f\alpha_x & f\gamma & u_c \\ 0 & f\alpha_y & v_c \\ 0 & 0 & 1 \end{pmatrix}$, defined as the camera intrinsic matrix. The idea of intrinsic calibration is to estimate these constants; knowing this matrix, we could apply its inverse and achieve images predicted by the ideal pinhole camera model.

The second non-ideality is that point X's coordinate is often hard to measure correctly in camera's frame. In order words, $\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$ is often calculated from point X's coordinate in the world frame $\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix}$. This process is also known as rigid body transform (shown in Image 6). Two transforms constitute a rigid body transform: rotation and translation. Rotation is a Euclidean Transform and can be represented as a 3-by-3 matrix R; translation can be represented as a 3D vector $t = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$. In this sense, the target point $P_{target} = R \cdot P_0 + t = \begin{pmatrix} R_1 \cdot P_0 + t_x \\ R_2 \cdot P_0 + t_y \\ R_3 \cdot P_0 + t_z \end{pmatrix}$, where R_i is the i-th row of R. In homogeneous coordinates, this becomes:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

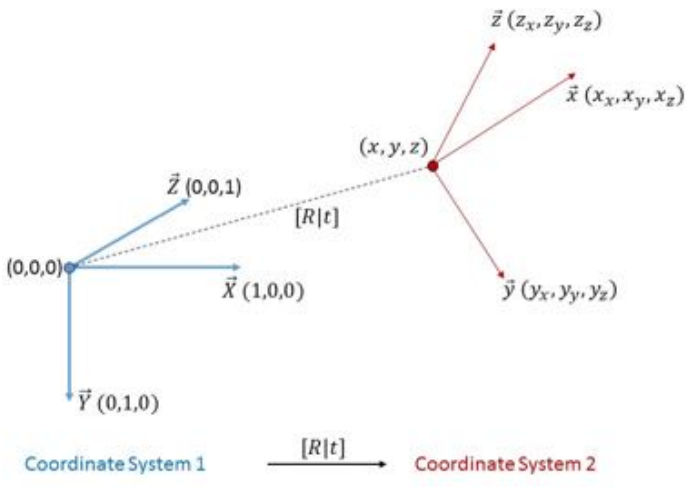

Image 6, Rigid body transform. Transform from coordinate system 1 (blue) to coordinate system 2 (red).

Therefore, the real image coordinate becomes:

$$\begin{pmatrix} u \\ v \end{pmatrix} = K \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = K \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

$$= K \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} = K[R \quad t] \begin{pmatrix} P_w \\ 1 \end{pmatrix}$$

where R is 3-by-3 rotation matrix, t is 3D translation matrix, and P_w is the measured coordinate in world frame. The process of extrinsic calibration deals with the measurement of R and t, which is essentially the 6D pose of the camera in world frame.

The third factor that makes the simple model of pinhole camera non-ideal is the existence of geometric distortion, which includes radial and tangential distortion. Radial distortion exists because the shape of the lens causes variations in light refractions. Radial distortion cannot be estimated by a linear transform or matrix; practically, it is often modeled by a polynomial function with sufficiently high accuracy. An effective formula to correct radial distortion is:

$$x_{corrected} = x(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6)$$

$$y_{corrected} = y(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6)$$

where r is the distance from the pixel to the center: $r^2 = x^2 + y^2$. Tangential distortion comes from the fact that the lens might not be perfectly parallel to the image plane. Similar to radial distortion, this effect can be corrected using the following non-linear equations:

$$x_{corrected} = x + [2p_1xy + p_2(r^2 + 2x^2)]$$

$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2xy]$$

k_1, k_2, k_3, p_1, p_2 are coefficients that need to be estimated by distortion calibration.

Implementation and Validation

This section will discuss the steps we took to perform the necessary calibration procedures discussed above; it will also show the results of these procedures.

First of all, to hack the Kinect v2 sensor, we are using an open source package called iai_kinect2. This driver has three ROS packages:

1. kinect2_bridge: create the bridge between libfreenect2, an open source driver for Kinect v2 sensor, and ROS nodes that broadcast images detected and sensor information
2. kinect2_viewer: create visualization of detected images
3. kinect2_calibration: a package dedicated for intrinsic and distortion calibration, as well as dual camera calibration of IR and RGB cameras

Therefore, we first installed all the above software by following iai_kinect2's online tutorial (Wiedemeyer, IAI Kinect2, 2017). We made sure that image can be streamed to the viewer before continuing to the next step.

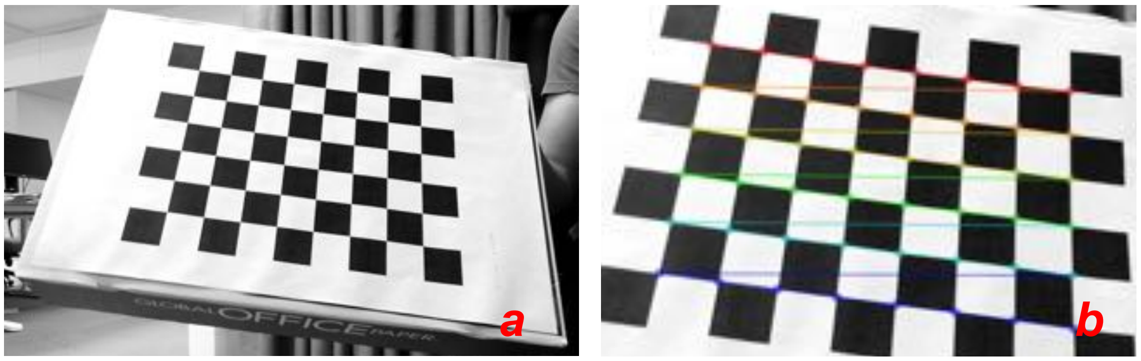

Image 7, Checkerboard pattern for intrinsic/distortion/dual camera calibration. (a) sample input image used; (b) inner corners detected (marked by color dots)

Second, we followed the procedure in kinect2_calibration package to get precise images. The first step in this procedure is intrinsic and distortion calibration for both RGB and IR cameras. We showed camera a checkerboard pattern (shown in Image 7 (a)) with known dimensions. Corner detection algorithms are applied to detected the inner corners of the checkerboard pattern, as shown in Image 7 (b). Because the shape and scale of the checkerboard pattern is known to the software, it is able to use these inner corner points to fit for a precise model of intrinsic matrix $K = \begin{pmatrix} f\alpha_x & f\gamma & u_c \\ 0 & f\alpha_y & v_c \\ 0 & 0 & 1 \end{pmatrix}$, as well as the distortion parameters k_1, k_2, k_3, p_1, p_2. For each camera, around 100 images are used to make sure the data points are sufficient enough to predict these parameters accurately. The second step is dual camera calibration for the two cameras. Similar to the previous step, we showed the checkerboard pattern to both cameras at the same time to estimate the essential matrix. The calibration results are:

$$K_{RGB} = \begin{pmatrix} 1067.6486954727648 & 0 & 964.13734842242093 \\ 0 & 1067.9665181210512 & 558.48297731126331 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} k_{RGB,1} \\ k_{RGB,2} \\ p_{RGB,1} \\ p_{RGB,2} \\ k_{RGB,3} \end{pmatrix} = \begin{pmatrix} 0.097691856090063728 \\ -0.17002329009962283 \\ 0.00079887031789669760 \\ 0.0030040866129423297 \\ 0.081674308190985132 \end{pmatrix}$$

$$K_{IR} = \begin{pmatrix} 360.74855402784794 & 0 & 251.45892721502329 \\ 0 & 360.61065953563514 & 211.30081753336793 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} k_{IR,1} \\ k_{IR,2} \\ p_{IR,1} \\ p_{IR,2} \\ k_{IR,3} \end{pmatrix} = \begin{pmatrix} 0.093080426557230964 \\ -0.28251132382012545 \\ -0.0010445405024170492 \\ 0.00049370947897657178 \\ 0.10772278110128708 \end{pmatrix}$$

$$d_{shift} = -23.622009807933534$$

The calibrated images are shown in Image 8.

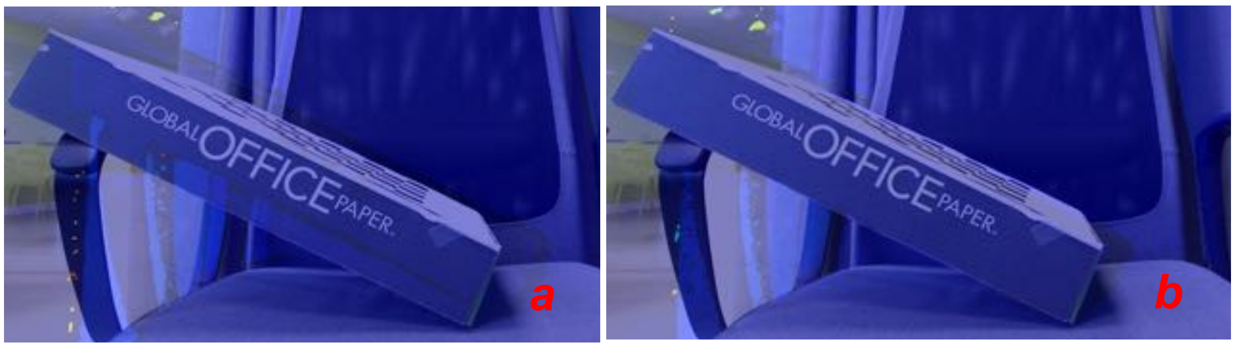

Image 8, overlay of IR and RGB images before and after calibration. (a) before calibration, there is a shallow IR "shadow" of the box to the right of its color image. (b) after calibration, the "shadow" overlaps completely with its color image.

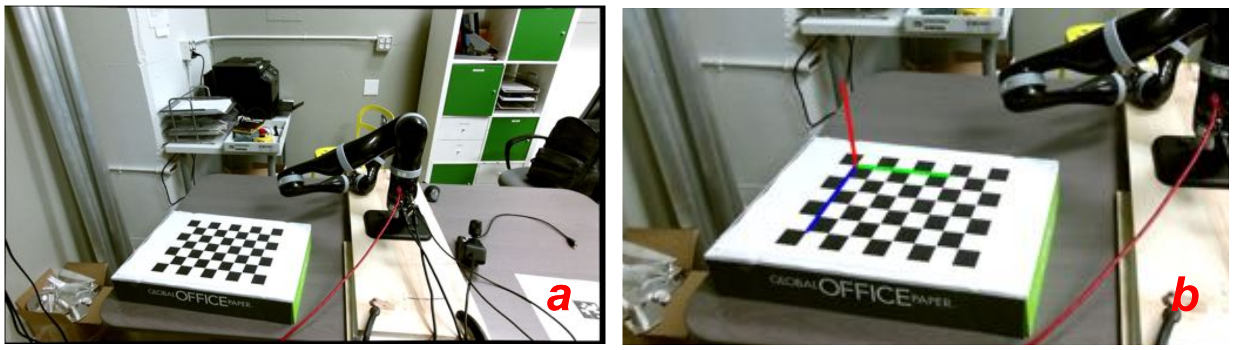

Image 9, Image of checkerboard pattern on lab table. (a) an example of input image to the extrinsic calibration node. (b) after extrinsic calibration, the node is able to detect the x, y, z axis and draw them on the image.

Third, we need to estimate the pose of the camera itself in a shared coordinate system. This shared coordinate system is defined by a checkerboard pattern that is fixed to a lab table (an example is shown in Image 9). An extrinsic calibration ROS node is used to read in an image containing this checkboard pattern and broadcast the 6D pose of the camera relative to this shared coordinate system. As an example, given an image shown in image 9 (a), the extrinsic calibration ROS node is able to calculate the pose of the camera in this example as:

$$R = \begin{pmatrix} -0.15032725 & 0.46206395 & -0.87401294 \\ 0.98855974 & 0.08127456 & -0.12706148 \\ 0.01232449 & -0.88311458 & -0.46899547 \end{pmatrix}$$

$$t = \begin{pmatrix} 0.75806523 \\ 0.43898541 \\ 0.59396386 \end{pmatrix}$$

Also, the x, y, z axis of the shared frame are detected as shown in Image 9 (b). Using this setup, a researcher should perform two steps before taking images:

1. fix the pose of the sensor and have it take an image containing the checkerboard pattern;
2. give this image to the extrinsic calibration ROS node to broadcast its 6D pose.

CONCLUSION

In conclusion, we have successfully made the Kinect v2 sensor a quantitative sensor. After performing calibration procedures, we are able to take accurate color and depth images containing objects to be perceived by the robot. Moreover, we are able to obtain and broadcast the 6D pose of the

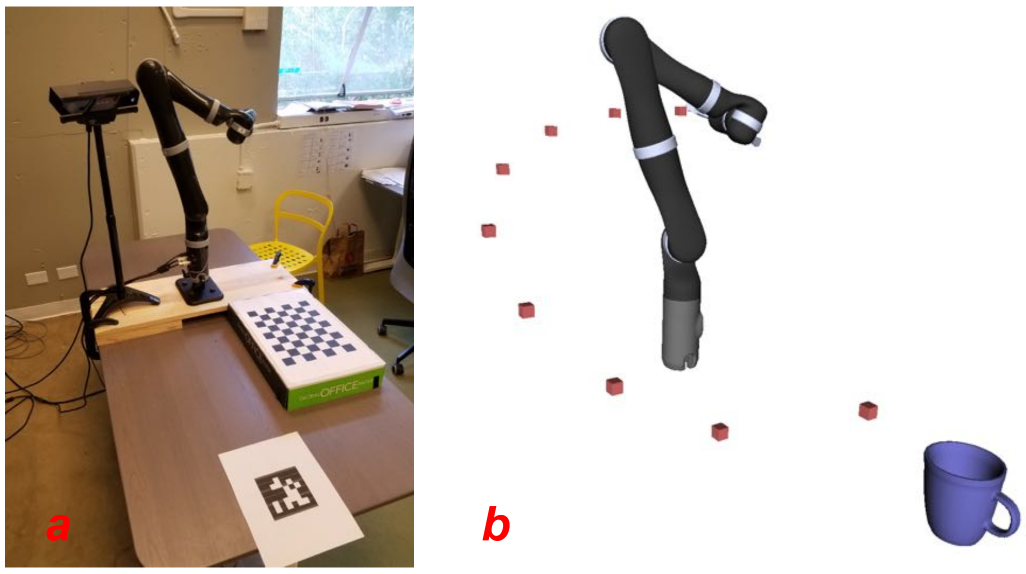

Image 10, (a) Typical setup of the system and (b) corresponding openrave simulations.

camera itself in a shared coordinate system. Please refer to Lab Wiki for how to install all necessary software, perform calibration procedures, and get both color and depth images from the sensor.

Combining the work done by Tyler and Geronimo, we set up our system shown in Image 10 (a). The Kinect sensor is able to accurately detect the pose of the AprilTag (meant to represent a mug) in the coordinate system defined by the checkerboard pattern. Then, it sends its coordinates to an openrave ROS node shown in 10 (b). From here, Geronimo's algorithms allow the robot to smoothly move to the mug.

Appendix

A, Homogeneous coordinates

Homogeneous coordinate is a system that uses a vector of dimension $D+1$ to represent a point in dimension D or any point infinitely far away. For example, a point in 2D $\begin{pmatrix} x \\ y \end{pmatrix}$ can be represented as $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ in homogeneous coordinates. One property of homogeneous coordinates is that $\begin{pmatrix} x \\ y \\ z \end{pmatrix} \equiv \lambda \begin{pmatrix} x \\ y \\ z \end{pmatrix}$, where λ is any real number; in other words, a point in dimension D corresponding to a ray in homogeneous coordinates of dimension $D+1$. Note that the excess degree of freedom derived from the increased dimension is eliminated by this scaling factor. If z is not zero, we normally write $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ as $\begin{pmatrix} x/z \\ y/z \\ 1 \end{pmatrix}$ by fixing z to 1; if z is zero, $\begin{pmatrix} x \\ y \\ 0 \end{pmatrix}$ represents a point infinitely far away.

In the context of this paper, the use of homogeneous coordinates makes it much easier to represent rigid body transform, which is essentially a rotation combined with a translation. Specifically, if we want to apply a rotation matrix R and a translation t to a point $P = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$ in 3D without homogeneous coordinates, we need to write $P' = RP + t$, which is a matrix multiplication and a vector addition. On the other hand, with the help of homogeneous coordinates, we could write $P = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$; this way, the rigid body transform R, t can be written as one matrix $M = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix}$, and $P' = MP$. Only one matrix multiplication is enough to represent rigid body transform.

BIBLIOGRAPHY

april robotics laboratory. (2016, 12). *AprilTag*. Retrieved from https://april.eecs.umich.edu/software/apriltag.html

Derivative. (n.d.). *Kinect*. Retrieved from Derivative Wiki: http://www.derivative.ca/wiki088/index.php?title=Kinect

Forsman, M. (2011). *Point cloud densification.* UMEA˚ UNIVERSITY DEPARTMENT OF PHYSICS.

Lachat, E. (2015, 10). Assessment and Calibration of a RGB-D Camera (Kinect v2 Sensor) Towards a Potential Use for Close-Range 3D Modeling. *MDPI*.

ROS Wiki. (2015). *ROS.org*. Retrieved from How to Calibrate a Monocular Camera: http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration

Wiedemeyer, T. (2016). *Kinect2 Calibration* . Retrieved from github.com: https://github.com/code-iai/iai_kinect2/tree/master/kinect2_calibration

Wiedemeyer, T. (2017). *IAI Kinect2*. Retrieved from Github: https://github.com/code-iai/iai_kinect2