**Algorithm Interface Overview**

**Hand-Eye Calibration**

*Outputs*

- Homogenous Cartesian transformation between the robot's end-effector and the center of the camera lens, commonly known as the "hand-eye" transformation.
- The pose of the calibration checkerboard in the robot base frame.

*Inputs*

- Checkerboard dimensions (number of corners in each dimension and square size).
- Set pictures of the checkerboard taken from a wide range of angles. At least 30 images is recommended for best results.
- Corresponding set of robot end-effector poses as homogenous transforms for each picture taken above (see Data Acquisition below).

*Assumptions/Setup*

- Hardware Dependencies:
    - KUKA iiwa robotic manipulator
    - Intel RealSense RGB-D F200 camera mounted on KUKA end effector plate
    - Calibration checkerboard
- Calibration checkerboard is placed ~ 600mm in front of the robot at the same vertical level as the robot base.
- The checkerboard does not move during image collection.
- The camera is rigidly mounted to the robot end effector, and thus the camera-to-end-effector transformation is constant.
- A 1 meter radius around the KUKA iiwa system is free of obstructions.

*Data Acquisition*

The data necessary to run this calibration is acquired by sending the robot to a number of poses which give the Intel RealSense camera a complete view of the calibration checkerboard. At each of these poses, the RGB camera image is saved, along with the pose of the robot end effector. Grayscale images would also be acceptable for this purpose. The camera's depth image is not needed for this calibration. Each image/robot-pose pair will be referred to as a "frame" from now on.The KUKA iiwa reports its end-effector pose as an x,y,z cartesian position measured in mm and a set of euler angles a,b,c reported in degrees. For our purposes, this is processed into a homogeneous transformation matrix, still using mm for the translational units.

*Workflow*

1. Camera Calibration: Camera calibration can be performed using the same set of checkerboard images which are used for the camera calibration. The output of this

process is the camera intrinsics matrix (focal length, skew, etc) as well as distortion correction parameters. There are many tools to perform this process automatically, such as the Matlab calibration toolbox.

2. Collect Camera Extrinsics for each Image: The camera extrinsics represent the pose of the camera with the respect to a "world" frame defined by the calibration checkerboard. By convention, the camera frame is defined such that the center of the lense is the origin, and the positive z-axis is the direction the camera faces. The camera extrinsics are computed as part of the camera calibration above, so we need only collect this set of homogenous transformations for use in the next stage of the calibration.
3. Compute Camera to End-Effector Transformation and Checkerboard Position: At this stage we have two sets of transformations: the transformation between camera and the checkerboard "world" frame (${}^{w}H_{c}$) and the transformation between the robot base and the robot end-effector aka the "gripper" (${}^{g}H_{b}$). The goal of the hand eye calibration is to acquire the camera to end-effector transformation (${}^{c}H_{g}$). We assume that the camera to end-effector transformation is constant, so for two different frames of data i,j we know:

$$ {}^{c}H_{g,i} = {}^{c}H_{g,j} = {}^{c}H_{g} $$

Also, since we assumed the checkerboard does not move during image acquisition, we know that for two different frames:

$$ {}^{w}H_{c,i}\ {}^{c}H_{g}\ {}^{g}H_{b,i} = {}^{w}H_{b,i} = {}^{w}H_{b,j} = {}^{w}H_{c,j}\ {}^{c}H_{g}\ {}^{g}H_{b,j} $$

By rearranging the above equation, we construct an AX = XB problem, where X is the camera to end-effector transformation:

$$ {}^{w}H_{c,j}^{-1}\ {}^{w}H_{c,i}\ {}^{c}H_{g} = {}^{c}H_{g}\ {}^{g}H_{b,j}\ {}^{g}H_{b,i}^{-1} $$

Theoretically, we only need two images to solve for ${}^{c}H_{g}$, the camera to end-effector transformation. However, by taking many images at many different angles, we can reduce the error in the hand-eye calibration. Using a least square error solver, we find the optimal camera-to-end-effector transformation for all the frames data taken.

4. Compute Checkerboard pose in Robot Base Frame: Using the computed hand-eye transformation, we can find the checkerboard pose with respect to the robot base:

$$ {}^{w}H_{c,i}\ {}^{c}H_{g}\ {}^{g}H_{b,i} = {}^{w}H_{b,i} $$

We find the board pose for each data frame and take the average as the final checkerboard pose, ${}^{w}H_{b}$.

**Robot-Robot Calibration - Generalized**

The outputs, inputs, and assumptions of this section apply to **ALL** of the following robot-robot calibration methods, in addition to each method's specific assumptions.

*Output*

- Homogenous Cartesian transformation between the two robot base frames.

*Inputs*

- Intrinsic parameters of the Intel RealSense camera (see step 1 of Hand-Eye Calibration).
- Camera to End-Effector transformations for both robots (see Hand-Eye Calibration).
- Calibration image data (see specific methods).

*Assumptions/Setup*

- Hardware Dependencies:
    - KUKA iiwa robotic manipulator (x2)
    - Intel RealSense RGB-D F200 camera mounted on each KUKA's end effector.
    - Calibration object(s) (see specific methods)
- The KUKA's are rigidly attached to each other, so the robot-robot transformation is constant throughout the calibration. The robot bases should be at approximately equal vertical level.
- The KUKA's both face a central common workspace.
- Calibration objects are placed in the common workspace ~ 600mm away from both robots, at approximately the same vertical level as both robot bases. The objects are visible to both robots.
- Calibration objects do not move during image collection.
- The camera is rigidly mounted to the robot end effector, and thus the camera-to-end-effector transformation is constant.
- A 1 meter radius around the both KUKA iiwa system is free of obstructions, if the other KUKA is in "candle pose." When one KUKA is making movements, the other is in candle pose (arm pointed straight up) to reduces likelihood of robot-robot collision.

**Robot-Robot Calibration - Checkerboard**

*Additional Input*

- Checkerboard dimensions (number of corners and square size)
- Optional: checkerboard poses from hand-eye calibrations if this calibration is done directly after hand-eye calibration.

*Assumptions/Setup*

- Calibration Object: Checkerboard

*Data Acquisition*

The data collected for this calibration is exactly the same as the hand-eye calibration (see above). The only difference is that we are now collecting images from both robot cameras. The only additional stipulation is the the board must be in the same position when each robot collects its images, as this method depends on the board pose being constant. If collected this way, the same set of checkerboard images/robot poses can be used to produce the camera calibration and hand-eye calibration for each robot individually as well as the robot-to-robot transformation. Now that the hand eye calibration is known, we really only need one image/gripper-pose from each robot to identify the pose of the checkerboard. However, we recommend using multiple images to minimize error.

*Workflow*

1. Identify the Pose of the Checkerboard in both Robot Base Frames: The checkerboard pose is computed in the same manner as step 4 of the hand-eye calibration (see above). If this calibration is run right after the hand-eye calibration of each individual robot, and if the board was in the same place for both robot's images, then we can simply use the board pose computed in the hand-eye calibration. This is an "optional" input.
2. Compute Base-to-Base Transformation: Once the pose of the checkerboard is known with respect to both robot base frames, it is trivial to compute the base-to-base transformation through the common "world" frame defined by the calibration object:

   ${}^{w}H_{b1}$ - homogenous transformation from robot 1 base to world frame
   ${}^{w}H_{b2}$ - homogenous transformation from robot 2 base to world frame
   ${}^{b1}H_{b2}$ - homogenous transformation from robot 2 base to robot 1 base

$$ {}^{b1}H_{b2} = {}^{w}H_{b1}^{-1}\ {}^{w}H_{b2} $$

**Robot-Robot Calibration - ARToolKit**

*Assumptions/Setup*

- Calibration Object: Printed ARToolKit Marker

*Data Acquisition*

The use of the ARToolKit marker is analogous to the checkerboard, and thus our method of image collection is the same. We send each robot to poses where it can see the marker, and we then record both the RGB image as well as the depth image and the end effector pose for that image. We only need one image from each robot to perform this calibration, but we plan on testing this method with multiple poses to see if it makes any significant difference on the error.

*Workflow*

1. Identify the Pose of the Marker in both Robot Base Frames: Using ARToolKit along with the depth information to recover the marker scale, we can compute the pose of the marker in each image, which we then transform into the robot base frame. If multiple images are being used with each robot, we will use the average marker pose.
2. Compute Base-to-Base Transformation: Once the pose of the marker is known with respect to both robot base frames, it is trivial to compute the base-to-base transformation through the common "world" frame defined by the marker:

   ${}^{w}H_{b1}$ - homogenous transformation from robot 1 base to world frame
   ${}^{w}H_{b2}$ - homogenous transformation from robot 2 base to world frame
   ${}^{b1}H_{b2}$ - homogenous transformation from robot 2 base to robot 1 base

$$ {}^{b1}H_{b2} = {}^{w}H_{b1}^{-1}\ {}^{w}H_{b2} $$

**Robot-Robot Calibration - RGB-D Features and Depth**

*Assumptions/Setup*

- Calibration Object: Visually Distinctive Fiducial Object(s)
    - Fiducial object(s) should have distinct geometry as well as recognizable visual regions (i.e. text, patterns, etc.). The aim is to have many distinctive features which are observable to both cameras.

*Data Acquisition*

This method makes use of the RealSense cameras' depth feature to compute the 3D position of features found in the 2D RGB image. We use both robots to collect RGB-D images of the common workspace. We also record the pose of each robot's end-effector when the images are taken. Theoretically, only one image from each robot is needed, however, we plan on taking multiple images of the workspace from different angles to test if this improves calibration accuracy.

*Workflow*

1. Generate Feature Point Clouds: For both robot's images, we use SURF (Speeded Up Robust Features) to generate features. As each feature relates to an x-y region of the image, the basic RGB image only gives us a ray in the camera reference frame which contains the center of the feature. Using the depth information from the RealSense camera, we can recover the 3D positions of features. At the end of this step, we have a 3D feature point cloud in each robot's camera reference frame.
2. Procrustes' Closed Form Solution: Using SURF, we can create a feature correspondence between images taken by the two robots. This gives us a point to point correspondence in the two camera's 3D feature point clouds. From here, we can use Arun's Algorithm or another closed form solution to determine the transformation from one point cloud to the other. It is then possible to compute the robot base-to-base transformation. If we use multiple images, we can compare multiple image pairs and find an optimal base-to-base transformation.

*Base-to-Base Formulation*

- ${}^{g2}H_{b2}$ - transformation from robot 2 base to end-effector 2 (*data acquisition*)
- ${}^{g1}H_{b1}$ - transformation from robot 1 base to end-effector 1 (*data acquisition*)
- ${}^{c2}H_{g2}$ - transformation from camera 2 to end-effector 2 (*hand-eye calibration*)
- ${}^{c1}H_{g1}$ - transformation from camera 1 to end-effector 1 (*hand-eye calibration*)
- ${}^{pc2}H_{c2}$ - transformation from camera 2 to the respective fiducial point cloud (*step 1*)
- ${}^{pc1}H_{c1}$ - transformation from camera 1 to the respective fiducial point cloud (*step 1*)
- ${}^{pc1}H_{pc2}$ - transformation relating the two robots' feature point clouds (*step 2*)
- ${}^{b2}H_{b1}$ - transformation from robot base 1 to robot base 2

$$ {}^{b2}H_{b1} = {}^{g2}H_{b2}^{-1}\ {}^{c2}H_{g2}^{-1}\ {}^{pc2}H_{c2}^{-1}\ {}^{pc1}H_{pc2}^{-1}\ {}^{pc1}H_{c1}\ {}^{c1}H_{g1}\ {}^{g1}H_{b1} $$

**Robot-Robot Calibration - RGB-D Depth Only**

*Assumptions/Setup*

- Calibration Object: Geometrically Distinctive Fiducial Object(s)
    - Fiducial object(s) should have distinct geometry (i.e. contours)

*Data Acquisition*

This method is similar to the RGB-D Features and Depth calibration, however, it only makes use of depth information. We command each robot to several positions where it can measure depth information for the central workspace. We save the RealSense camera's depth image as well as the pose of each robot's end-effector when the images are taken. Theoretically, only one image from each robot is needed, however it is a good idea to collect depth images from multiple angles as it will give a more comprehensive point cloud for the central work space.

*Workflow*

1. Generate Depth Point Clouds: In each frame for each robot, using the depth information of the RealSense camera we can define a point cloud for the scene. Using the recorded end-effector pose and the known hand eye calibration, we can put this point cloud into the robot's base frame. If we record multiple depth images from different angles, we can compound the images' point clouds in the robot base reference frame, as the calibration objects are assumed to be fixed with respect to the robot bases. Compounding multiple depth images will provide a more comprehensive point cloud for the workspace.
2. Generate FPFH Descriptors: Unlike the RGB-D Features and Depth, we cannot use the RGB information to create feature correspondences. Instead, using Point Cloud Library's Fast Point Feature Histogram (FPFH) algorithm, we can compute descriptors for the local geometry around each point in the point cloud. These local descriptors can be used as weighting factors in an ICP algorithm. Essentially, 3D points with like geometric features are more likely to be the same point in different reference frames.
3. Use ICP Variant to find Base-to-Base transformation: By matching the robots' point clouds to one another using a feature-weighted ICP algorithm, we can generate the robot base-to-base transformation. The hallmark of the ICP variant used in this case is its ability to perform robust matching with only subsets of each point cloud. In order to accomplish this, it must be less sensitive to outliers. This ability is necessary because the two robot's point clouds are only partially overlapping.