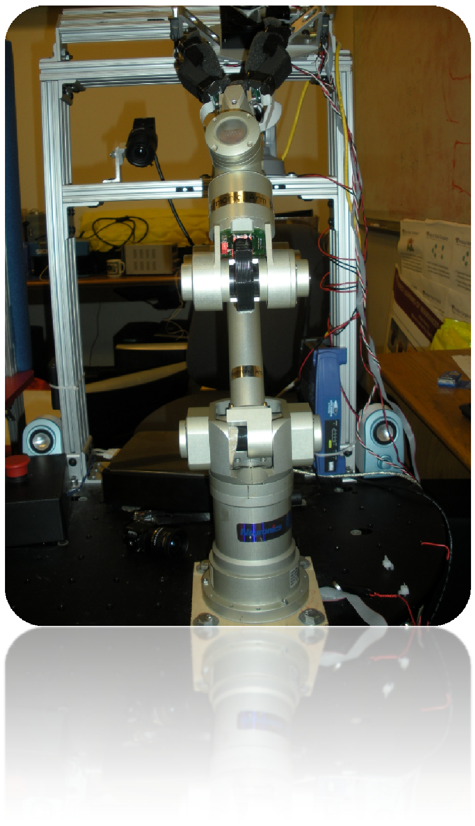
Automatic Robotic arm calibration using parameter optimization techniques

Project by: Chetan Kalyan

Introduction:

Robotic manipulation tasks need accurate calibration between sensors (eyes) and arms. Standard manual methods of calibration require recording 3D coordinates of the end effector and in the arm and eye frames.



We propose to perform this calibration using optimization techniques on the parameters of the robotic arm:

There are a few parameters in the arm which are assured to be accurate, such as the join angles of each link in the arm and the length of some of the links in the arm. However, the length of the last link can vary due to wear and tear or due to the user adding add-ons at the tips of the arm. The state-of-the-art is to manually measure the length of these links and feed them to the arm. This is unreliable as manual measurements are prone to errors.

The scope of this project is thus to reduce the calibration error of the arm to a negligible level. This involves optimization over the lengths of the links of the arm, using as few parameters as possible. This will allow us to use the arm without having to perform time-intensive manual measurements.

Normally, such an optimization over a large number of variables is a fairly complicated process, since it involves finding gradients and hessians of all the parameters involved in order to optimize. For our purposes, since the optimization is primarily over numeric data, we opted to choose the Levenberg-Marquardt algorithm, which performs numeric minimization over repeated iterations of the algorithm. This suited our purposes, since the functions governing the kinematics of the arm are sufficiently complicated to prevent easy calculation of gradients.

Optimizing the lengths of all the links as well as the rotation and translation matrices returned the best results. This shows that the transformation of co-ordinates from the arm frame to the sensor frame is also prone to errors and can be optimized.

In subsequent sections, we describe the setting up of the optimization problem, the experimental set-up and the results that we obtained.

Fundamentals:

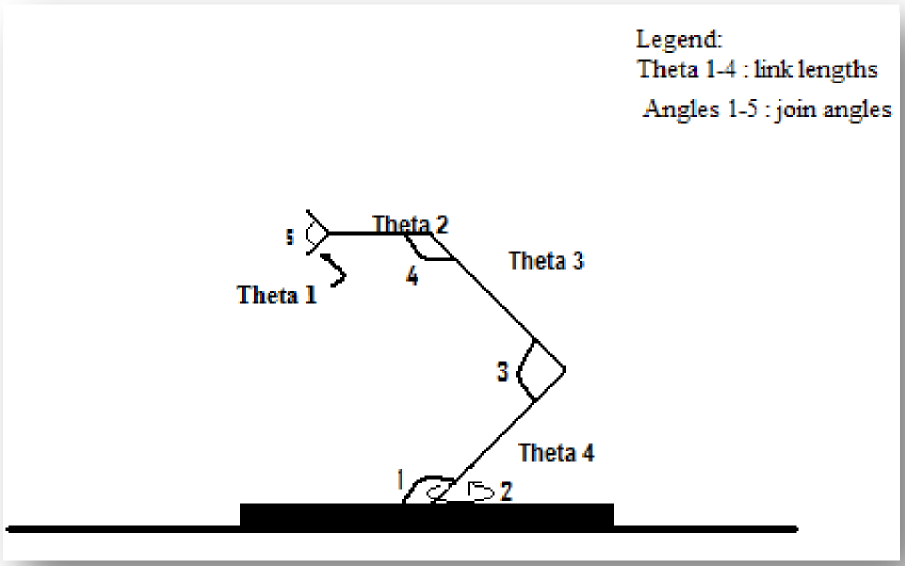


Figure 1 Schematic of arm parameters

We make the assumption that, even though the lengths of the links are not entirely accurate, the errors in θ_1 to θ_n are small and can be used to train our model effectively.

Let $J \in R^5$ be the vector of the 5 join angles of the arm. The measurements of J are assumed to be accurate.

In order to predict the length of θ_0, we need to first find the position of the arm in a 3D space. This means that we need to be able to accurately say what the (x,y,z) co-ordinates of the end-effector are.

We have 2 frames of reference in this problem- the frame of the robotic arm *R* and the frame of the on-board sensor *S:*

Let X_S^i be the value of (x,y,z) returned by the sensor for the ith training sample;

X_R^i be the co-ordinates of the arm in the robot space.

We need to find X_R^i in terms of X_S^i, and match the two, in order to find the correct (x,y,z) for computation of θ_0.

How do we do this? We know the value of J^i, the join angles of the ith training sample of the data set. We also know θ, the vector of link lengths. We can hence find X_R^i as a function of J and θ as:

$$x_r = \cos(J(1)) * [\ \vartheta(1)*\sin(J(2)) + \vartheta(2)*\sin(J(3)) + (\vartheta(3)+\vartheta(4))*\sin(J(4))\ ];$$

$$y_r = \sin(J(1)) * [\ \vartheta(1)*\sin(J(2)) + \vartheta(2)*\sin(J(3)) + (\vartheta(3)+\vartheta(4))*\sin(J(4))\ ];$$

$$z_r = \vartheta(1)*\cos(J(2)) + \vartheta(2)*\cos(J(3)) + (\vartheta(3)+\vartheta(4)) * \cos(J(4));$$

We can use the rotation matrix $R' \in R^{3x3}$ and the translation vector $T' \in R^3$ to transform the robot space to the sensor space, and thus match our robot and sensor co-ordinates:

$$€^R_s = R' * X_r + T'$$

Optimizing this reduces to a least squares problem, and once the optimal solution has been found, we can proceed to use *R',T'* and J to find the optimized value of θ,as :

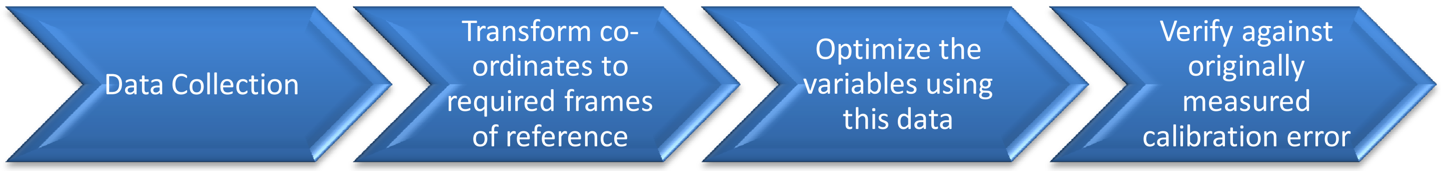
$$\text{Min}_{R',T',\vartheta}\ [\ k^*(\text{actual dist. between 2 pts on the board}) + (\ €^{R}_{S}\,(X_r) - X_s\,)^2\ ]$$

Here, k is a scaling factor which imposes a constraint on the optimization and ensures that the matrices are not changed too much. We measure the distance between any 2 points on the board and use this distance to constrain the optimizer to find the length of the links satisfying the distance equation.

In this optimization, we cannot find the gradient or the hessian efficiently, since the function governing X_R^i is complex. So, we use the Levenberg-Marquardt algorithm to find local optima without finding gradient or hessians. We iterate over this step till we reach the global optimum.

Experimental Set-up:

Procedure for execution of project:


Data Collection
Transform co-ordinates to required frames of reference
Optimize the variables using this data
Verify against originally measured calibration error

- **Data collection**. Collect join angles of the arm at various positions, in order to find J^i.

  - On a board, draw some crosses, representing the X_S^i's. These will be used as the points for data collection.
  - Scan the board with the laser and record the positions of each data point. This gives us the co-ordinates in the sensor frame.

- Move the arm to each of the crosses and record join angles and pose at that position. This gives us the parameters of the robot frame.

- **Transform co-ordinates from robot space to sensor space**.

  The transformation to sensor frame of reference allows us to compare the values from the arm and the sensor values.

- **Optimize** the transformation, using Levenberg-Marquardt algorithm, in Matlab.

  We ran the Levenberg-Marquardt algorithm on the collected data for 1000 iterations, initializing each of the 3 optimization parameters (*R′,T′, theta*) to some default values, not assured to be accurate. We found that the algorithm stabilized somewhere around 800 iterations.

- **Test** the optimized parameters in calibration and find change in calibration error.

  The output of the optimization was the 3 parameters- (*R′,T′, theta*). We used this output to re-calibrate the robot and check the calibration error against the original error.

Results:

We collected 87 distinct data points. We ran the Levenberg-Marquardt algorithm on this dataset with default initializations and performed 1000 iterations on the optimizer.

We then gave faulty initializations to the optimizer and checked for convergence to the correct values. We found that the length of the last link was optimized to an error of less than 1mm and the rotation and translation matrices were also optimized to get better results.

Calibration error = mean error between the predicted co-ordinates and actual co-ordinates of each sample in the sensor frame

Kinematics error= mean error between the predicted co-ordinates and actual co-ordinates of each sample in the arm frame

	Initial	Optimized	Initial	Optimized
Length of last link	134.3	135.3	167	135.2011
Rotation matrix change		0.4410		0.4357
Translation vector change		0.0120		0.0173
Calibration error	7.552966 mm	0.079319 mm	7.552966 mm	0.752 mm

Kinematics error	0	0.816763 mm	1.0693 mm	0.87 mm

Conclusion and further work:

Thus, Parameter optimization of robotic arm parameters is much more reliable and efficient than manual configurations. In the future, we plan to expand the optimization technique to help in calibrating most, if not all, parameters of the arm, thus reducing time and effort in calibration. We also plan to devise more sophisticated ways of measuring calibration errors in a more varied environment.

Acknowledgements:

I would like to thank Mr. Quoc Viet Le for his patience and his counsel throughout the project. I would also like to thank everyone at the Robot learning lab for their support, encouragement and help during my work.

References:

1. The Levenberg-Marquardt algorithm: Implementation and theory, Jorge J.More, Springer Berlin / Heidelberg, 1978.
2. Ng, Andrew, “Least squares method”, “Parameter optimization”, CS 229: Machine Learning Course Notes (2008 Aut), Available at http://www.stanford.edu/class/cs229/materials.html