University of Alberta

CMPUT 501

Visual Servoing Sensitivity Analysis

Author:
Oscar Ramirez

Professor:
Martin Jagersand

December 29, 2013

1 Visual Servoing

In this project I look at the sensitivity of the Jacobian matrix used during visual servoing. As it's main objective visual servoing will minimize an error:

$$\boldsymbol{e}(t) = \boldsymbol{s} - \boldsymbol{s}^* \tag{1}$$

Where $\boldsymbol{s}$ and $\boldsymbol{s^*}$ are the image features for different tracked points. These can then be used to set up several types of constrains like point to point, or point to line constraints. Once the error function has been defined we perform the servoing by performing a simple linear control based on the image Jacobian:

$$J = \begin{bmatrix} \frac{\partial f_1(\boldsymbol{q})}{\partial q_1} & \cdots & \frac{\partial f_1(\boldsymbol{q})}{\partial q_m} \\ \vdots & & \vdots \\ \frac{\partial f_k(\boldsymbol{q})}{\partial q_1} & \cdots & \frac{\partial f_k(\boldsymbol{q})}{\partial q_m} \end{bmatrix} \tag{2}$$

Where $\boldsymbol{q}$ is a vector containing the joint angles, and $\boldsymbol{f}$ a vector with the different image features. One simple way to find an estimate of the Jacobian, $\hat{J}$, is to find a $\Delta\boldsymbol{e}$ by performing small motions at each joint:

$$\hat{J} = \begin{bmatrix} \begin{bmatrix} \vdots \\ \Delta \boldsymbol{e}_{q_1} \\ \vdots \end{bmatrix} & \cdots & \begin{bmatrix} \vdots \\ \Delta \boldsymbol{e}_{q_m} \\ \vdots \end{bmatrix} \end{bmatrix} \tag{3}$$

Having both (1) and (3) we can use them in a simple control law to minimize the error:

$$\dot{\boldsymbol{q}} = -\lambda \hat{J}^{\dagger} \boldsymbol{e} \tag{4}$$

2 Simulation

2.1 Plannar Motion

Using the Robotics Toolbox for Matlab I implemented a simulation environment where I can test different experimental setups for visual servoing. I first began with a simple 2DOF robot that performed planar motions. This allowed me to test my implementation of the visual servoing environment, this setup is shown in Figure 1. At first I used a simple pinhole camera model, and later moved to a projective perspective model using the intrinsic camera parameters of one of the Grasshopper cameras from the lab. I found the intrinsic parametrs using the ROS camera calibration package and a 8x6 calibration grid.

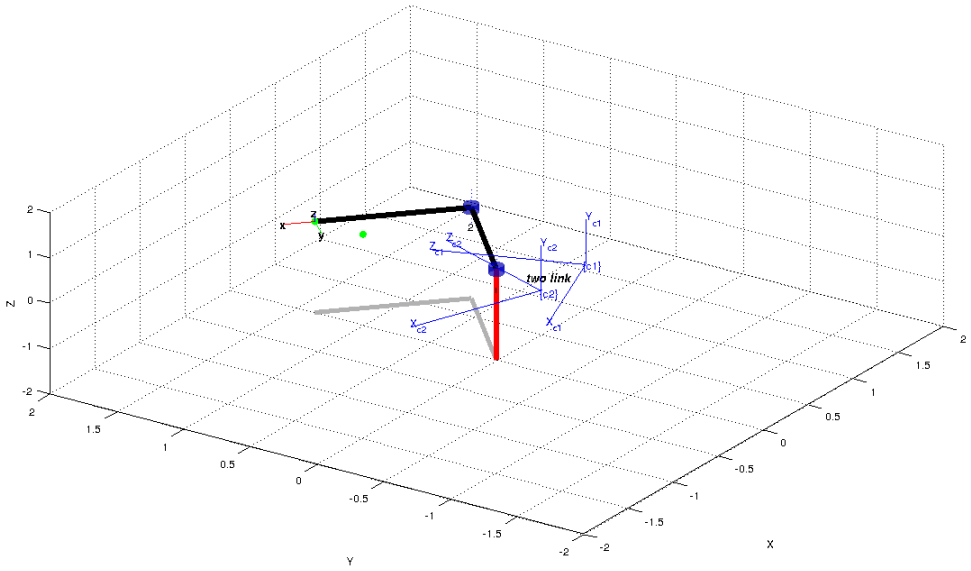
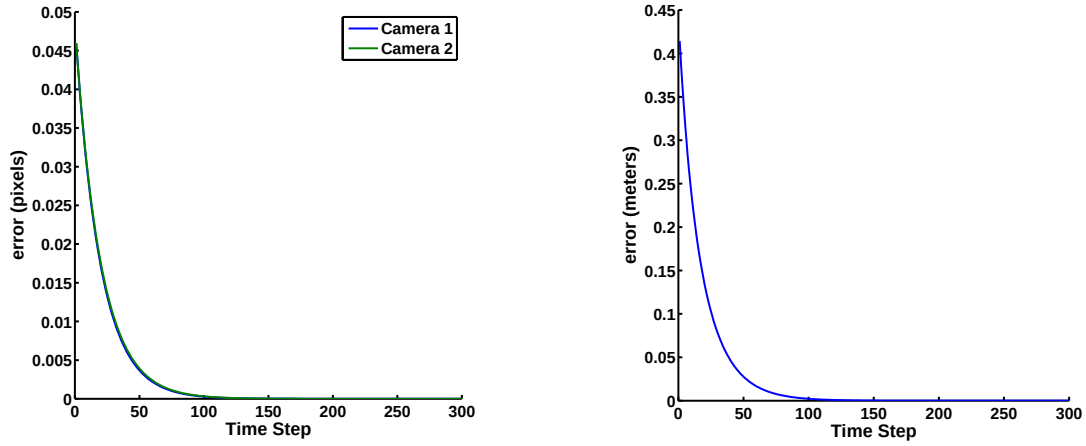


Figure 1: Simple 2DOF robot simulation setup



(a) Image Error

(b) World Error

Figure 2: Error as the visual servoing progresses for the 2DOF setup

Once implemented I was able to perform simple point to point tasks. Where one point was defined to be the target, and anotherone a point that was tracked at the end effector of the robot. The error was defined simply as $\boldsymbol{e} = \boldsymbol{s}_t - \boldsymbol{s}_{eef}$ and the system

quickly converged when updating the pose of the robot by following (4). To visualize how the error was reduced I tracked the sum of the magnitudes of the errors as shown in Figure 2.

2.2 Puma Robot Simulation

Using a model for the Puma I also tested different scenarios. First I looked at a point to point configuration similar to the one performed in the 2 DOF scenario. This setup is not sufficient to drive the robot towards the target since sevearal joints have no effect on the image features. This is also reflected in the condition number of the jacobian which ranged between the 10^4 to 10^6 range depending on where the target was placed.

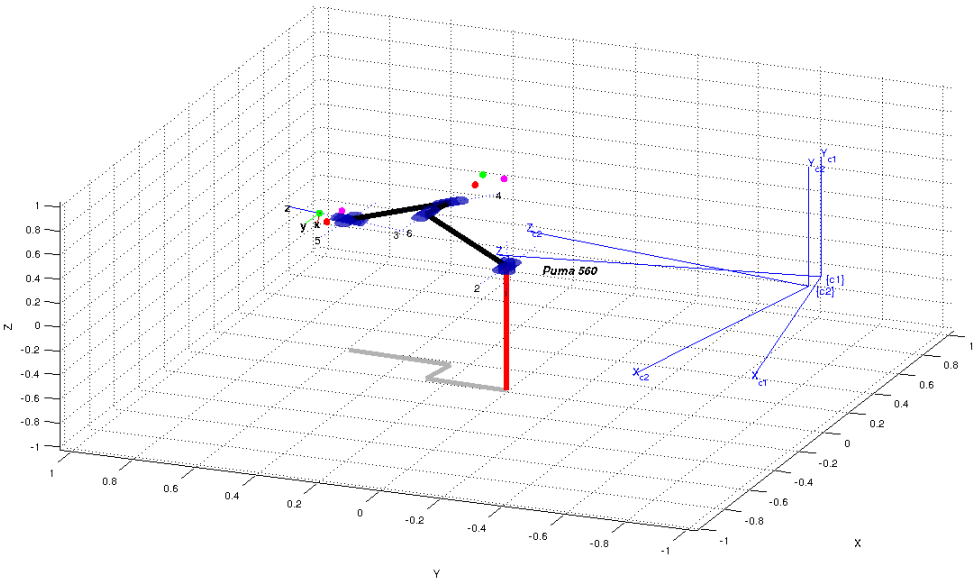
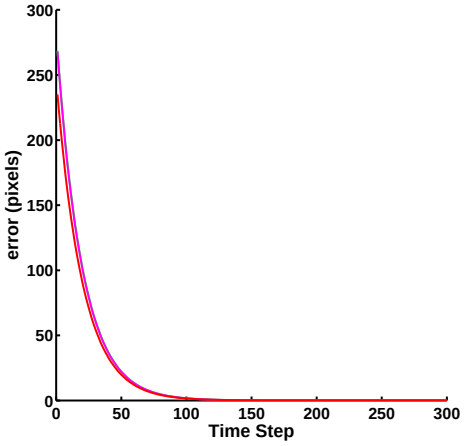


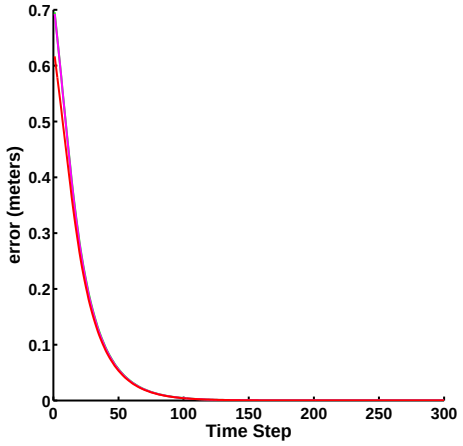
Figure 3: Puma robot simulation using 3 points as a target and as end effector features.

Then I tried two simple configuartions with point to point constrains using 3 and 4 points as the target, and the tracked features of the end effector. The 3-point scenario is shown in Figure 3.

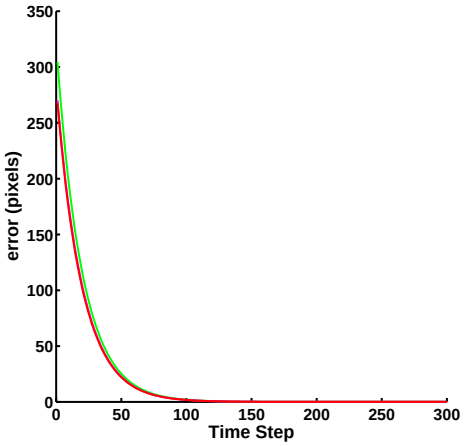
When performing the simulation it was interesting to see that the 3 point configuration converged faster than the 4 point configuration. This was done while having the same start and end poses, the error for this simulation is shown in Figure 4. As part of the simulation I also looked at the condition number of the Jacobian as the simulation went on. To do this I found the Jacobian at each pose in which the robot was at throughout the simulation. Although there is some variation, the system is fairly stable throughout the whole motion. The plot for this is shown in Figure 5.



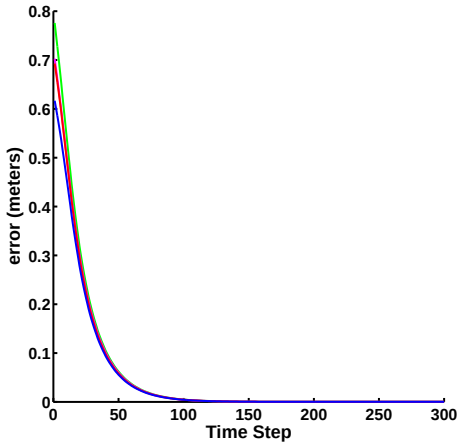
(a) 3 Points Image Error Camera 1



(b) 3 Points World Error



(c) 4 Points Image Error Camera 1



(d) 4 Points World Error

Figure 4: Error as the visual servoing is performed for 3 and 4 point scenarios on the Puma robot.

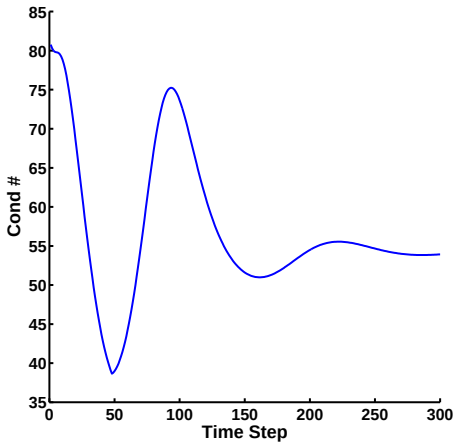


Figure 5: Condition number of J as the robot moves throughout the visual servoing simulation. Only the initial Jacbian was used during the movement.

2.3 Complex Scenarios

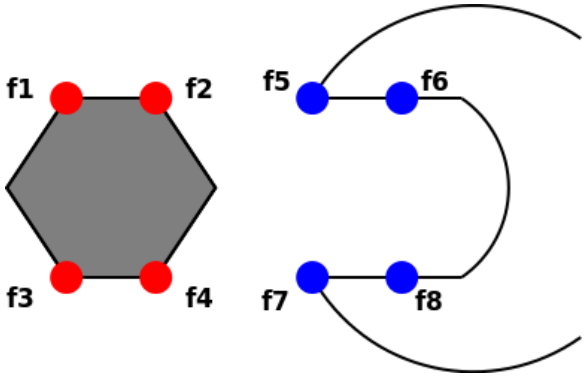


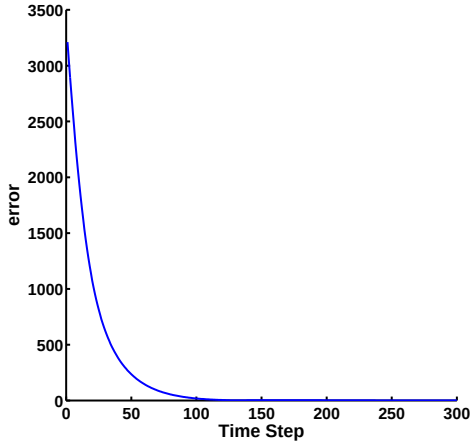
Figure 6: Feature points tracked in wrench task.

After performing the point to point tasks I reproduced two more complex scenarios. First a point to line task, and then a more interesting task for fitting a wrench to a nut. The image features defined for the wrench setup are shown in Figure 7. Once tracked, the error is setup as a point to line constraint between the line formed with the top wrench points $f_1 f_2$ and the top tip of the wrench f_5. A similar constraint is formed with the bottom points. Point to point constraints are also set with f_2 and f_6, and f_4 with f_8. The final form of the error function for the wrench example will then stack

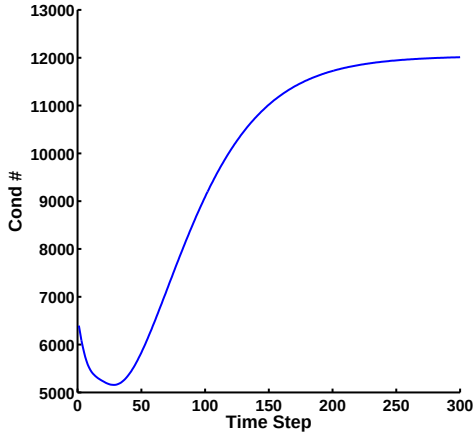
the features from each camera:

$$\boldsymbol{e} = \begin{bmatrix} \boldsymbol{f}_{l2} - \boldsymbol{f}_{l6} \\ \boldsymbol{f}_{l4} - \boldsymbol{f}_{l8} \\ (\boldsymbol{f}_{l1} \times \boldsymbol{f}_{l2}) \cdot \boldsymbol{f}_{l5} \\ (\boldsymbol{f}_{l3} \times \boldsymbol{f}_{l4}) \cdot \boldsymbol{f}_{l7} \\ \boldsymbol{f}_{r2} - \boldsymbol{f}_{r6} \\ \boldsymbol{f}_{r4} - \boldsymbol{f}_{r8} \\ (\boldsymbol{f}_{r1} \times \boldsymbol{f}_{r2}) \cdot \boldsymbol{f}_{r5} \\ (\boldsymbol{f}_{r3} \times \boldsymbol{f}_{r4}) \cdot \boldsymbol{f}_{r7} \end{bmatrix} \tag{5}$$

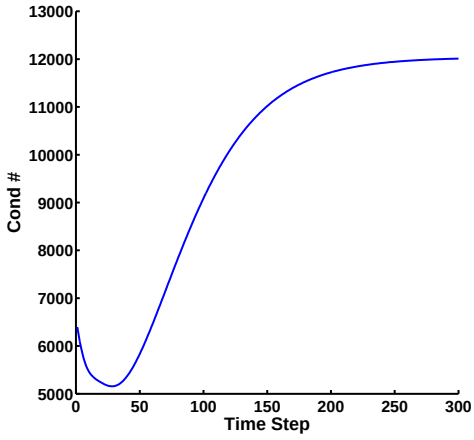
For this setup, when placing the starting position of the robot to be fairly close to the target, the system converges and the task specification is sufficient to place the wrench at the desired location. The condition number of the Jacobian however is significantly larger than the previous scenarios.



(a)



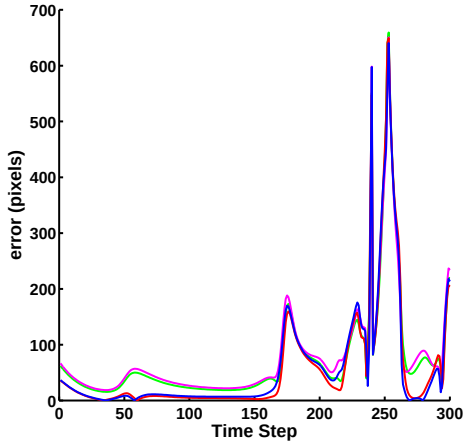
(b)



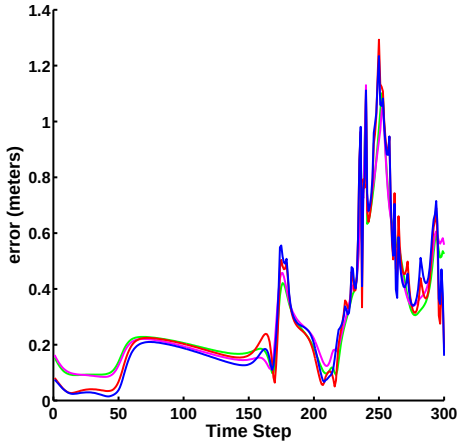
(c)

Figure 7: Errors and Condition of the Jacobian as the visual servoing is performed for the wrench scenario on the Puma robot.

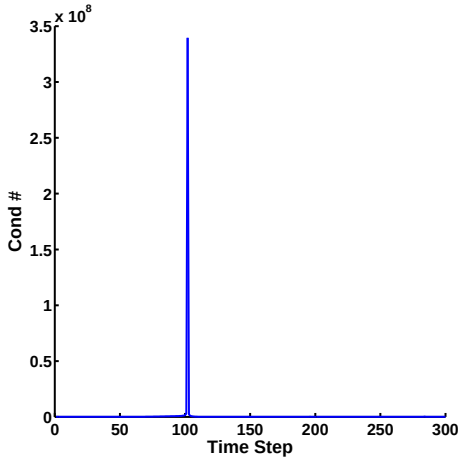
Moving the initial position slightly further away causes the system to diverge if no updates to the Jacobian are made.



(a)



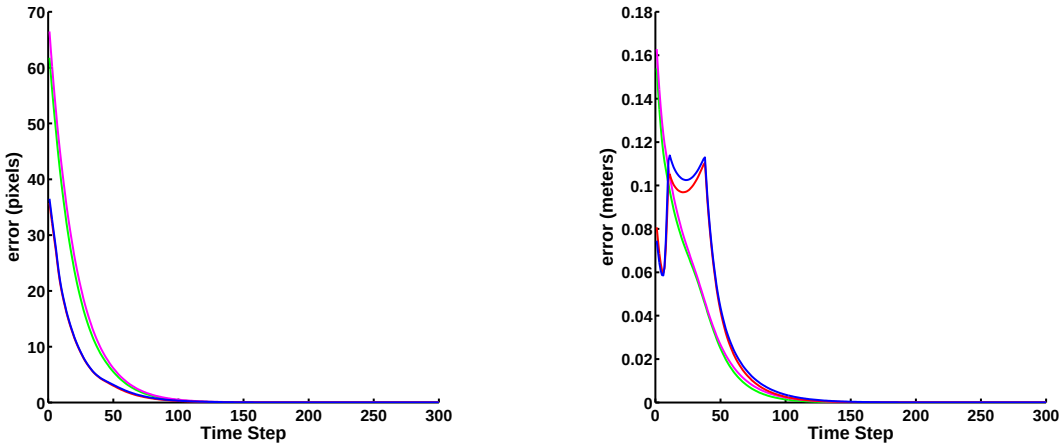
(b)



(c)

Figure 8: Error as the visual servoing is performed for the wrench scenario on the Puma robot.

Interestingly when recalculating the Jacobian at every time step, the task converges in an unexpected way. The error converges towards 0, but as the servoing moves on, the wrench actually flips around thus compleating the visual servoing task, but not in the intended way.



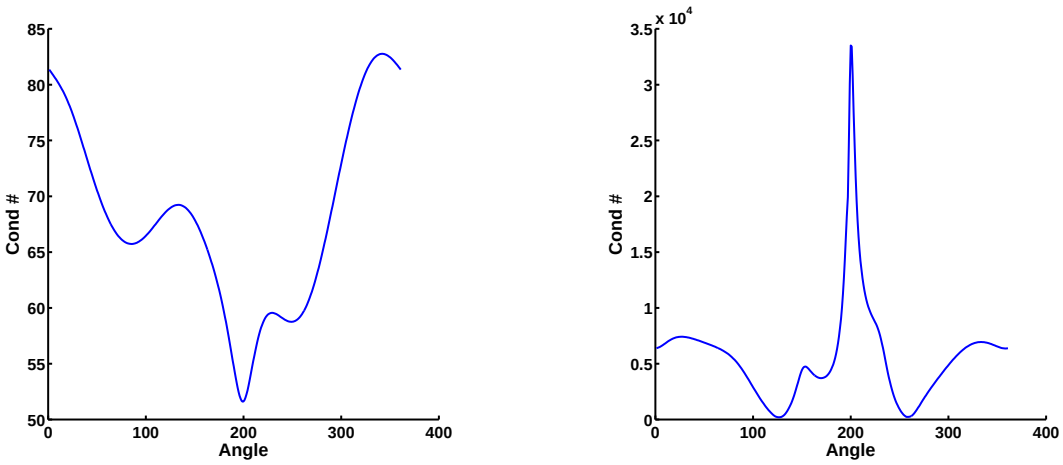
(a) Image Error Camera 1

(b) World Error

Figure 9: Error as the wrench constrain causes the Puma robot to flip the end effector

2.4 Camera Pose

After looking at the different task specifications I experimented with different camera poses. To do this I rotated the camera position around the target point. Looking at the condition number of the Jacobian specific points showed a very high increase. This can be seen in Figure 10



(a) Plane target

(b) Wrench Target

Figure 10: Condition number as the camera is rotated around the target point

Although the plane task specification was for the most part indiferent to the camera pose, the wrench task was very sensitive. Looking at a pose where the camera is at around a 180 degre rotation, see Figure 11, we can see the robot pose, and the target to be almost fully aligned. This seems to be causing the image Jacobian to be much more sensitive to error.

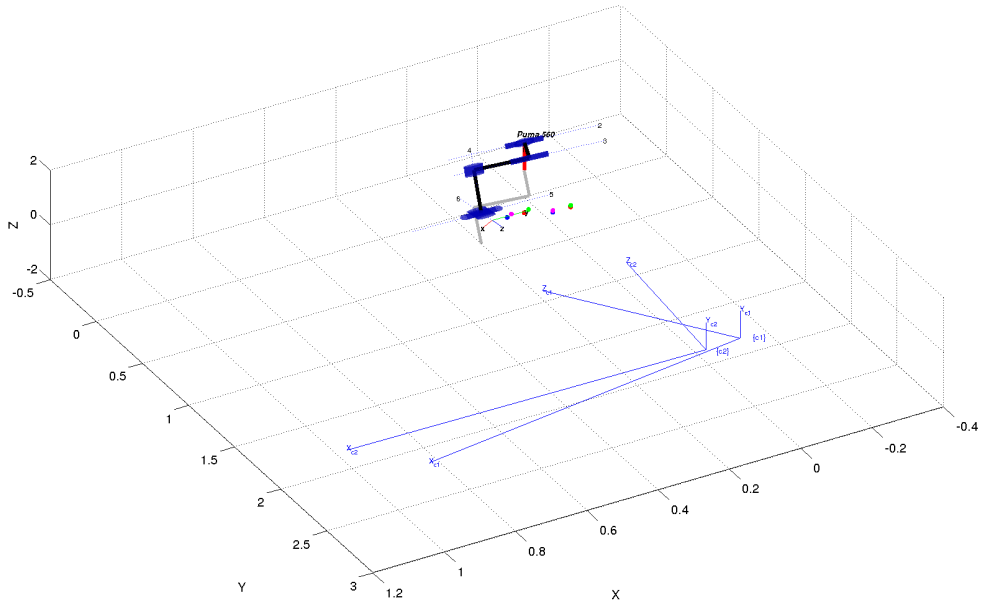


Figure 11: Rotated camera placement for wrench task

References

[1] François Chaumette and Seth Hutchinson. Visual servo control. i. basic approaches. *Robotics & Automation Magazine, IEEE*, 13(4):82–90, 2006.

[2] François Chaumette, Seth Hutchinson, et al. Visual servo control, part ii: Advanced approaches. *IEEE Robotics and Automation Magazine*, 14(1):109–118, 2007.

[3] Peter I. Corke. *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer, 2011.

[4] Seth Hutchinson, Gregory D Hager, and Peter I Corke. A tutorial on visual servo control. *Robotics and Automation, IEEE Transactions on*, 12(5):651–670, 1996.

[5] Nicolas Mansard, Manuel Lopes, José Santos-Victor, and François Chaumette. Jacobian learning methods for tasks sequencing in visual servoing. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 4284–4290. IEEE, 2006.

[6] Azad Shademan, A-M Farahmand, and M Jagersand. Robust jacobian estimation for uncalibrated visual servoing. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5564–5569. IEEE, 2010.