

**Requirements**

1. It should be possible to locate the pallet anywhere in the valid workspace of the robot.
2. The user should be able to specify the numbers of rows and columns of blocks at run time.
3. All blocks should be successfully grasped and placed.

*If the above 3 requirements are not met, you will be asked to reschedule your demonstration for another try until you succeed.*

Goodies (not mandatory, but worth extra credit):

4. If the pallet is placed wholly or partly outside the valid workspace, the program should catch this and report an error BEFORE starting the task, so that the user can reposition the pallet and try again. This is important for having a robust "utility" function.
5. Any clever measures to ensure high accuracy.
6. General convenience or coolness of user interface and/or swift execution. For example, one trick is to make the first fiducial somehow asymmetrical (or perhaps a collection of two or three dots or an oblong shape) so that the camera gets enough information to know approximately where to go to find the second and third fiducials. This obviously saves teaching time and is nice for the operator. See us for ideas about how to do this if interested.

**Demonstrations:**

I will ask for a show of hands on Thursday May 1. I hope to see demos the next week. At any rate, I will pass out instructions for Project 3 on May 1 so you can start planning for it. As with the first project, please email me a copy of your source code after the demo.

## Project 2

The relocatable pallet problem, with vision. Due week of May 5.

As you've heard, the second project is basically "repeat the first project, but using vision" (and a different robot). Here are some things to keep in mind:

7. The basic idea is to use the camera to identify some "fiducial" marks on the pallet instead of teaching points by hand. On both robots, the easiest thing is to use binary "blob" vision. You can easily get the centroid of a black or white dot, for example. You are discouraged from permanently marking the Duplo pallets, but you are free to modify duplo *blocks* by gluing or taping paper "hats" with dots or other fiducial marks on them. There are many possibilities. It is also possible to use edge-finding software to look directly to the edge of a pallet, although this is often not as easy as blob finding. See us if you want to try this.
8. As with the first project, there will be offsets: (i) robot wrist to gripper (ii) pallet origin to [i,j] block, (iii) camera to world. As before, a divide-and-conquer strategy is needed. Do tests to make sure you have each aspect of the problem exactly right before putting them together. This provides an opportunity to divide tasks among team members -- let one person focus<sup>1</sup> on the vision problem, for example.
9. Lighting can be a problem. Shiny surfaces (e.g., Duplos) can cause glare and be identified as false objects. Carefully narrowing the selection criteria on blob area, roundness, etc. helps to distinguish true fiducials from false ones. Also, there is nothing wrong with making a paper or cardboard "mask" to cover parts of the pallet while teaching. This would be an "operator instruction" in an industrial situation. (In a true industrial situation you would also have more ability to control lighting and surface textures & colors for pallets.)

### Vision transformations

Introducing vision adds a couple of robot-dependent twists to the transformations.

Adept: With the Adept, a complication is that the camera is not at the wrist or hand; it is on the second link. So even though the Adept knows where its wrist is and, by doing the camera calibration, it can tell you the world [X,Y] location of an object in the field of view, that is not the same as knowing what angles Joint1 and Joint2 should have to position the *camera* over a known world [X,Y] location. Depending on how you do the project (see "extra goodies" below) you may need a solution to this problem.

There are basically 2 solutions. The first is to use the Adept's built-in inverse kinematics to figure out how to position the robot wrist over a known [X,Y] location and then work backwards up the arm to the camera (undo the wrist rotation, undo the Zaxis translation and then do a constant offset transformation out to the camera). This approach is covered in the Adept manual pages copied at the end of the ME319 Helpful Hints binder. The second approach is simply to do your own inverse kinematics for a 2-link arm. Use a ruler to approximately measure the length of the first link and the distance from the elbow to the camera. The first link length is probably given in the Adept hardware installation manual too (It's on the Adept bookshelf). With either solution you do not need an exact answer because you only need to do a good enough job of positioning the camera so the object at [X,Y] will be in the approximate middle of the field of view.

Robot World: With Robot World, you may need to consider that there is an offset between the centerline of the quill of the mobile devices and their bases. So the total offsets could be [quill XY offset]:[theta offset]:[gripper XY offset]. The first [quill XY offset] did not come up in the first project because all locations were taught and grasped with the same device.

---

<sup>1</sup>(sorry :-)