

Project 1

The re-locatable palletizing problem (due April 17).

This project is a component of most robotic applications. The whole point of robotics is flexibility. You should be able to call up library functions of pre-defined tasks and execute them anywhere in the robot's workspace with a minimum of reprogramming. But this means you must have programmed the tasks using *relative* locations. Also, the programs should be robust enough that if somebody tries to execute them incorrectly (e.g., outside the workspace of the robot) an error message will be generated.

The Task:

The task is to load/unload a pallet of precisely fixtured parts. In this case we are using a Duplotm pallet and blocks. Using your program, a person like me should be able to (i) teach the robot the new position and orientation of the pallet and, (ii) tell the robot how many rows and columns to transfer. The robot should then execute the placement task. You can either load/unload the pallet using blocks from a fixed input location of your choosing, or you can simply transfer them from one part of the pallet to another (or stack them up or whatever).

Issues:

1. Duplo and Lego have tight tolerances! Unlike Lego, Duplo has also been designed for ease of assembly (by preschoolers) so that only moderate care is required in programming. Still, the tolerances are tight enough that you will have to think about accuracy to succeed with this project. Think about how to space teaching points for best orientation accuracy.
2. Be careful to write your "place" operation so that you don't ever jam misaligned Duplos down and cause a crash. This can damage the Duplos, the grippers and, in the case of Robot World, the Z axis of the robot. See us for how to write "fault tolerant" gentle placements that won't crash when there's an error. **Believe us, errors will happen...and remember, a software crash on a robot is usually a hardware crash too!** So be sure to always crash gently.
3. There is generally an offset between the grasped part or tool and the robot wrist. In some cases this offset may seem like a simple translation along the robot wrist axis, but it usually involves small X and Y adjustments as well. People tend to forget this when designing their programs. So, to enforce the "tool-offset" issue, we have made grippers with exaggerated offsets. This will cause the first project to be a bit more time consuming, but it will pay off later in the re-usability and robustness of your code. You may want to write a small offset-specifying and checking program (the offsets tend to change when somebody has had a crash).

Requirements:

1. It should be possible to locate the pallet anywhere, with any orientation, in the horizontal plane of the workspace.
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 more 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.

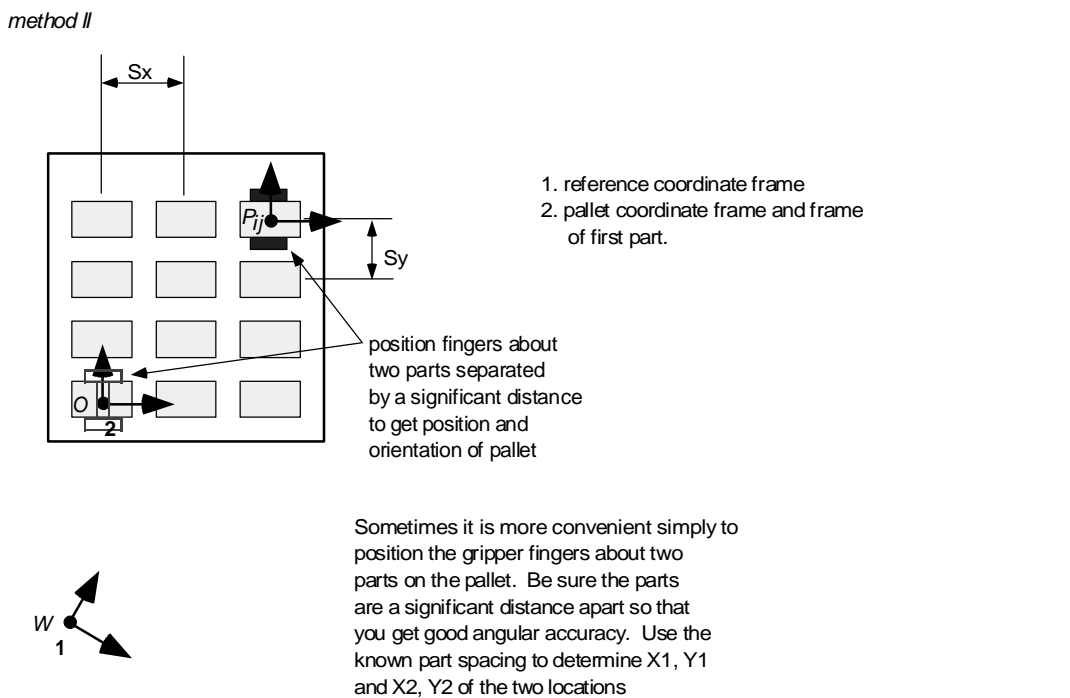
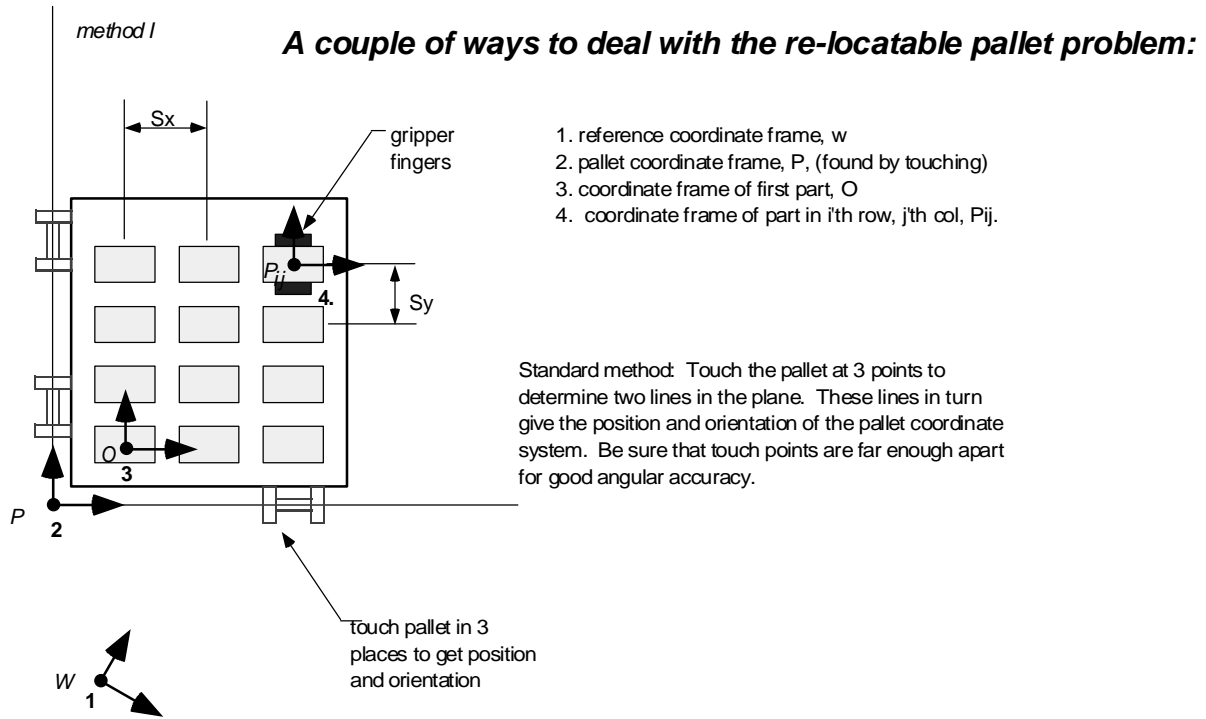


Fig 1: A couple of ways to define a pallet.

Tips on the first project

Here are some tips that may help you if you start to feel frustrated by lack of progress. You have already discovered why the simple transform loop on the “warm up” assignment from the first day of class is missing some information to make it work. Note that there are 3 types of spatial or offset information that you are trying to determine. It helps to separate these three problems and test your solutions to each them individually (divide and conquer).

1. The offset from the robot wrist or quill to the grasped part.

The offset will have two main components: an angular component and an XY component. The angular component means that when the robot wrist is ostensibly aligned with something like the world X or Y axis, an object grasped in the hand is actually rotated a bit. One way to measure the angular offset is to grasp a long bar and then touch each end of the bar to a fixed location. The angle will be given by $\delta x/\delta y$ (ask us for an explanation of this method.) Once you have found the angular offset you can create a "correction" transform that corresponds to rotating that amount about the Z axis.

Now you are ready to find the X and Y offsets in the corrected wrist frame. The XY offsets can be inserted into a second transformation that would be post-multiplied after the rotate offset. Rotations of 90deg, 180deg, -90deg, -180deg are particularly easy for debugging X and Y offsets. For example, on RobotWorld you could have the gripper grasp one of the aluminum cylinders and bring it to a height just above a post. Errors of 1mm or less will be clearly visible.

Ultimately, if you can successfully command a grasped block to pirouette about its own centerline then you can be sure you have your tooling offsets computed accurately.

2. The position and orientation of the pallet in the world frame

Meanwhile, if you touch several Duplo blocks on the pallet, always touching the same (e.g., lower right) corner of each block with the same corner of the gripper, while keeping the world orientation of the gripper constant, then you can check if you have found the pallet position and orientation correctly -- *independent* of whether you have the correct wrist/gripper offset or not!

Try moving the robot along the X and Y axes of the pallet. Does it track straight and true? If there is an error in the pallet position or orientation you will quickly see it. Note that the effect of an angular error in pallet orientation gets worse as you go further from the pallet origin; but XY errors in pallet location remain constant in magnitude.

With both Adept and Robot World you can get the pallet frame either using trigonometry or using FRAME (Adept) or BUILDFRAME (Robot world) commands. Note that the frame commands have a couple of idiosyncracies in both cases. Read the manual descriptions carefully. Remember to think about singularities (is it possible you could get a divide by zero for any orientations?) and the difference between two-argument $\text{atan2}(y, x)$ and single argument $\text{tan}(y/x)$ functions.

3. The location of the [i,j]'th part on the pallet with respect to the pallet frame.

This can often be obtained most accurately AFTER you have the pallet position and orientation accurately (as above). You'll get better accuracy if you count out several spaces, measure the total distance and then divide by the number of spaces. Try moving from the [1,1] location out to a position near the edge of the pallet and lowering a grasped block to just above the point where it would start to assemble. You'll soon see if you are off by a millimeter. Use that information to adjust your block spacing parameter.

Placement of parts:

For both robots, a good idea when programming the placement is to first move to a location just above the place location and then proceed slowly for the last 1/8 inch or so. This prevents hard crashes that damage Duplos. Remember on Robot World that the vertical speed must be set separately using the speedz function (not speed) -- and don't set it below 10, to avoid hanging up the servo.

Printing and saving your programs:

On the Adept you can format a 3.5inch floppy to take to any PC clone. I think the format is /H (?) You can read about the FORMAT instructions in the manual. We also have a version of "V+ for windows" which allows you to do some editing and syntax checking on a PC. Unfortunately, it requires a hardware key, so it's limited to PCs in the lab.

On Robot World you use standard Macintosh floppy disks.

In both cases, you have to use the older 800K formatting (not 1.3 Mbytes). I will try to set up another Mac or PC with a printer near the robots soon. Also, we hope soon to have the lab on the SU-Net, which will make it possible to upload and download files from the robots.