Orienting and sorting parts for robot pickup

In a robot assembly facility, always keep track of the part’s pose.

Robot loading and unloading machining centers
https://www.youtube.com/watch?v=4TOotC_Q3sU

When parts arrive ‘randomly’ then must ‘reacquire’ pose

Getting parts to align, consider the vibratory feeder bowl:
https://www.youtube.com/watch?v=QsJzSFVAnhk
https://www.youtube.com/watch?v=9pWtjwkrPCs

And/or various other attachments to conveyor belts …
https://www.youtube.com/watch?v=FvLBEMLEUmA

Vibratory loading table with flip station:
https://www.youtube.com/watch?v=wgK4jxrWH3A
Most gripper designs are ‘single-purpose’

Fingers are typically designed to maintain pose for a specific part.

To grab part from jumbled mix, with a multi-purpose gripper, new startup has appeared…

https://www.righthandrobotics.com

See article at
http://gizmodo.com/these-self-learning-robot-arms-teach-each-other-how-to-1794029246
Task descriptions

Task: Insert pin in block
In order to manipulate objects in space, must consider:

**OBJ**: Object coordinate frame w.r.t. **O**

**F**: Coordinate frame attached to feature defined w.r.t. **OBJ**

**G**: Pose of gripper tip (located by **E**) defined w.r.t. feature **F**

The vectors shown point to the origin of each coordinate system but do not show the orientation captured by the H.T.

\[ Z \cdot T_6 \cdot E = (\text{OBJ}) \cdot F \cdot G \]
Task Sequence:

Move P1  Move to position over pin
Move P2  Approach pin from overhead
Grasp
Move P3  Lift pin vertically
Move P4  Move to position above hole with pin oriented along hole axis
Move P5  Insert pin
Release:  Let go of pin
Move P6  Move gripper back away from pin
Note: P1 and P3 are typically the same

Define following transformations capturing the position and orientation, or pose:

- $P$: Pose of pin in base coord
- $B$: Pose of block in base coord
- $^B{H_i}$: Pose of $i$th hole w.r.t. block
- $^P{PG}$: Pose of gripper* when holding pin w.r.t. pin (* as defined by E)
- $^P{PA}$: Pose of approach above pin
- $^P{PD}$: Pose of departure above pin
- $^{H}{PHA}$: Pose of pin above hole for approach
- $^{H}{PIN}$: Pose of inserted pin
To define $\mathbf{P}$:

Place gripper on pin at pick up pos’n

$$ZT_6E = \mathbf{P} (\mathbf{PG})$$

$$\mathbf{P} = ZT_6E(\mathbf{PG})^{-1}$$

$\mathbf{PG}$ typically just involves a known rotation

Simpler solution: Let pin axes align with gripper axes, or … incorporate rotation into $\mathbf{E}$
Redefine P (Pose of the pin in base coordinates, i.e. the pin’s homogeneous transformation matrix)

PG depends on P and E. Once P and E are set, PG is set
Assuming $^{p}PA$ is desired at dist $Z$ above $P$

With modified $P$ coordinate frame …

Then

$$^{p}PA= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

If alternate departure point desired, lift pin in gripper to position $PD$

$$ZT_6E = P \; PD \; PG$$

Define \[ PD = P^{-1}ZT_6E(PG)^{-1} \]

To define $B$:

Move manipulator around to a position so that the pin held by the gripper is inserted to the appropriate depth

$$ZT_6E = B \; H_i \; PIN \; PG$$

$$B = ZT_6E(H_i \; PIN \; PG)^{-1}$$

$H_i$ & $PIN$ are predefined by specs

Note: Pin/gripper orientation must not change in the above development
Note: In above diagram, pin has been elongated for clarity

Note:

P is defined from the workspace origin to locate the pin coordinate system’s position and orientation.

All transformations defined w.r.t. \( P \) apply just as well w.r.t. any H.T. defining the pin coordinate system’s position.

Thus \( ^{p}A \), \( ^{p}D \), \( ^{p}G \) may or may not represent an H.T. defined w.r.t. the “P” pose in space, but they do represent an H.T. defined w.r.t. the reference coordinate system attached to the pin.
$^{h}\text{PHA}$ is defined to be located at dist $h$ above block

Thus $^{h}\text{PHA}= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -h \\ 0 & 0 & 0 & 1 \end{bmatrix}$

To carry out task:

1. Move P1 $T_6=Z^{-1}P \ PA \ PG \ E^{-1}$
2. Move P2 $T_6=Z^{-1}P \ PG \ E^{-1}$
3. Grasp
4. Move P3 $T_6=Z^{-1}P \ PD \ PG \ E^{-1}$
5. Move P4 $T_6= Z^{-1}B \ H_i \ PHA \ PG \ E^{-1}$
6. Move P5 $T_6= Z^{-1}B \ H_i \ PIN \ PG \ E^{-1}$
7. Release
8. Move P6 $T_6= Z^{-1}B \ H_i \ PHA \ PG \ E^{-1}$
9. Go to 1 Back to instruction 1.
Note form of expressions

\[ T_6 = (\text{coord}) (\text{pos}) (\text{tool})^{-1} \]

Where

**coord**: Transformation for working coordinate frame

**pos**: Transform for desired pose of tool tip or object

**tool**: Transform for describing the tool tip or object to be moved

The above may also be re-written in another form:

\[ T_6(\text{tool}) = (\text{coord}) (\text{pos}) \]

By specifying **tool** and **coord**, all motion instructions can be written in terms of the desired positions and orientations
Task Program:

1. Tool = E PG^{-1} → Attach tool
2. Coord = Z^{-1}P → Set up coord w.r.t pin
3. Move PA → \( T_6 = \text{coord} \times \text{PA} \times (\text{tool})^{-1} \)
4. Move Origin* → \( T_6 = \text{coord} \times (\text{tool})^{-1} \)
5. Grasp

6. Move PD → \( T_6 = \text{coord} \times \text{PD} \times (\text{tool})^{-1} \)
7. Coord = Z^{-1}BH_i → Set up coord w.r.t hole
8. Move PHA → \( T_6 = \text{coord} \times \text{PHA} \times (\text{tool})^{-1} \)
9. Move PIN → \( T_6 = \text{coord} \times \text{PIN} \times (\text{tool})^{-1} \)
10. Release

11. Move PHA → Move straight out w.r.t hole axis
    \( T_6 = \text{coord} \times \text{PHA} \times (\text{tool})^{-1} \)

12. Go to 1 → Back to instruction #1

* Origin = \[
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
\]
Programming languages to control manipulators

Example: VAL (Variable Assembly Language) has 111 Commands.
See http://en.wikipedia.org/wiki/Variable_Assembly_Language

◆ Monitor commands (50)
- Defining locations 7
- Program editing 11
- Program and location- data listing 2
- Program and location- data storage 11
- Program control 7
- System status and control 6
- System switches 3
- System diagnostics and modification 2

◆ Program Instructions (56)
- Robot configuration control 6
- Motion 13
- Hand control 6
- Integer arithmetic (5 operations) 2
- Location assignment and modification 7
- Program control 13
- Trajectory control 7
- Continuous path motion 2

For open source Robot Operating System, see
and http://www.ros.org
Programming Robots is Problematic:

Interactions with real world is difficult:

- Unpredictability
- Disastrous physical effects of errors
- Irreversible
- Cannot reinitialize the environment

Expressing physical actions is difficult:

- Speed
- Acceleration
- Via points
- Types of motion
- Force control
- Sensors
- Parallel operations

Open source ROS development:

https://www.willowgarage.com

Folding 5 towels (March 17, 2010):

https://www.youtube.com/watch?v=gy5g33S0Gzo
On-line programming:
- Interactive environment for debugging and testing requires use of robot

Off-line Programming:
- Doesn’t require robot
- Does not necessarily capture the use of sensors. Can one simulate sensors?
- More complex software (simulators, graphics system, etc.)

List of robot simulation packages can be found at:


Simulation packages in the early days of the industry: