CSE4421/COSC5323

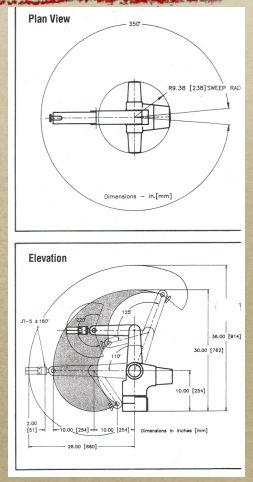
ek 4: Kinematics

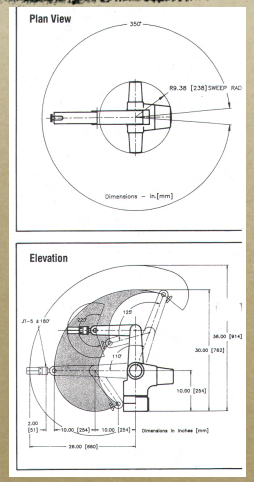
Forward kinematics

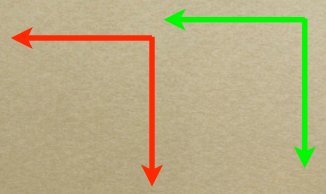
- *Forward kinematics for manipulators*
- *Essentially computing the transformation that describes points in a frame aligned with the end effector to points in the world coordinate frame.*

CRS A150

- *Waist -175..175 (0=straight ahead)*
- *Shoulder 0..110 (0==horizontal)*
- *Elbow -125..0 (0==straight)*
- *Wrist bend -110..110 (0=straight)*
- *Wrist twist -180..180 (0=no*



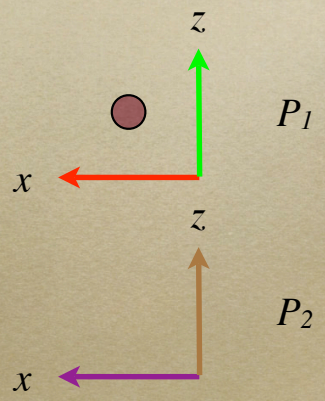




Frame Transformations

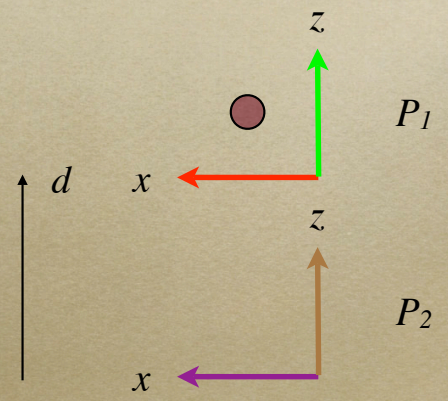
${}^{2}_{1}P$ *Maps points given in Frame 1 to Frame 2*

$P_2 = {}^{2}_{1}PP_1$



So here...

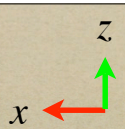
${}^{2}_{1}P = T(0,0,d)$

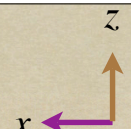


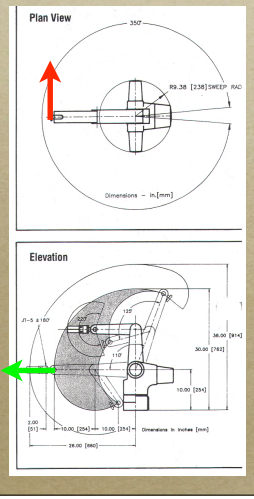
Lets assume the following

- *$T(dx,dy,dz)$ - translate by (dx,dy,dz)*
- *$R_x(theta)$ - rotate about x by theta*
- *$R_y(theta)$ - rotate about y by theta*
- *$R_z(theta)$ - rotate about z by theta*

The CRS A150

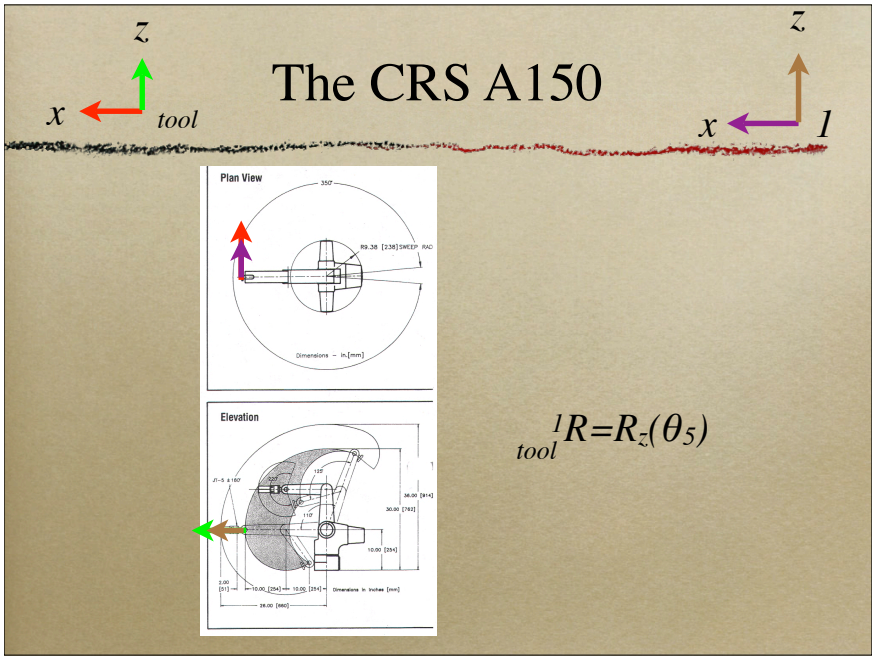






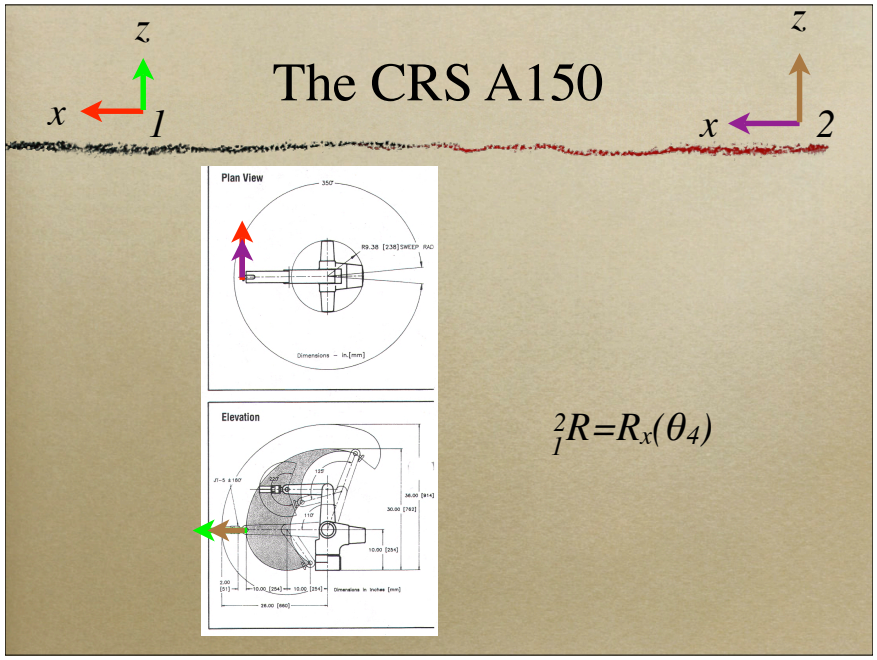
Define a 'tool' frame. say z pointing forward

The CRS A150



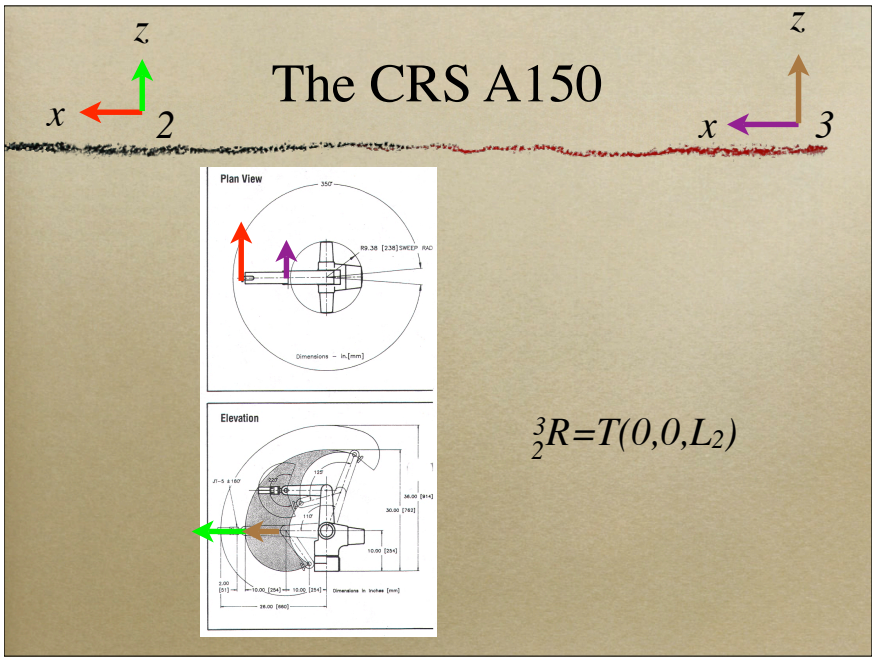
$${}^{1}_{tool}R = R_z(\theta_5)$$

The CRS A150



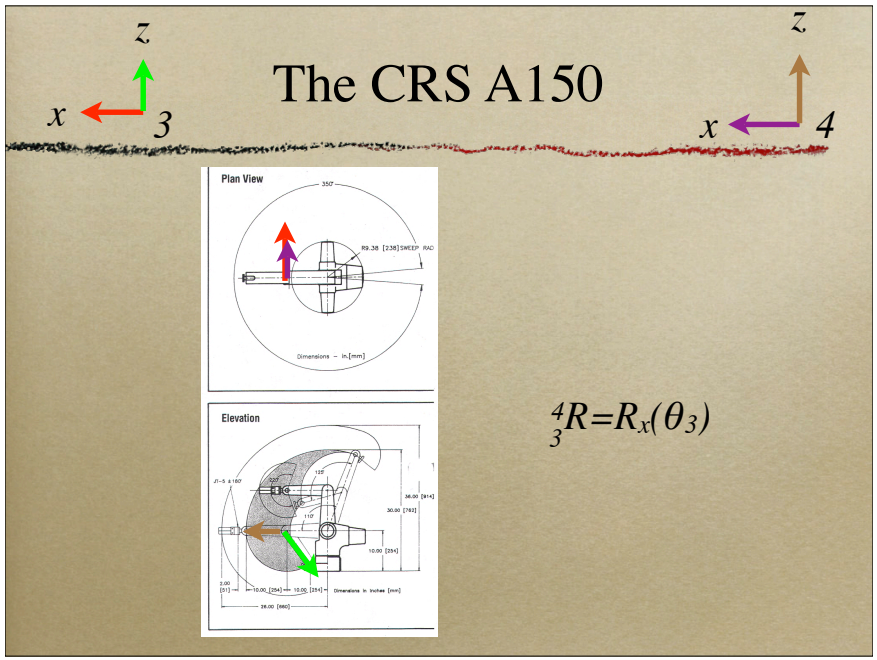
$${}^{2}_{1}R = R_x(\theta_4)$$

The CRS A150



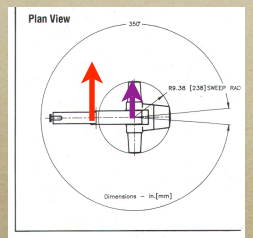
$${}^{3}_{2}R = T(0,0,L_2)$$

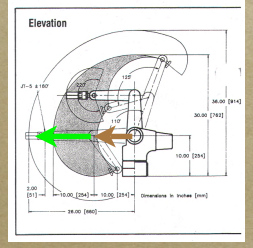
The CRS A150



$${}^{4}_{3}R = R_x(\theta_3)$$

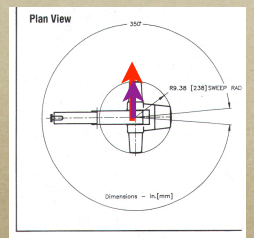
The CRS A150

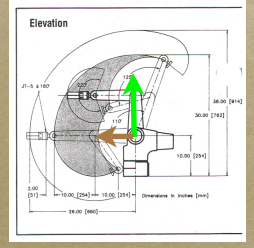




$${}^{5}_{4}R=T(0,0,L_1)$$

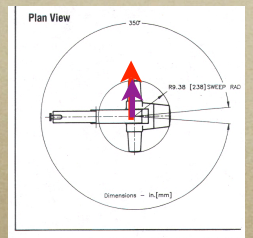
The CRS A150

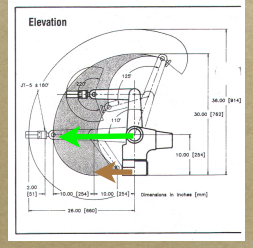




$${}^{6}_{5}R=R_x(\theta_2)$$

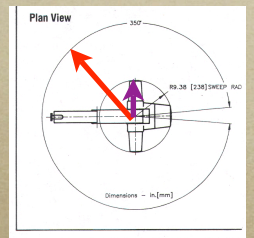
The CRS A150

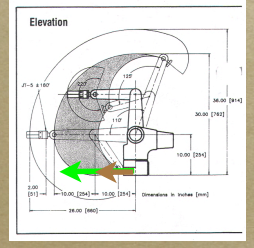




$${}^{7}_{6}R=T(0,-L_0,0)$$

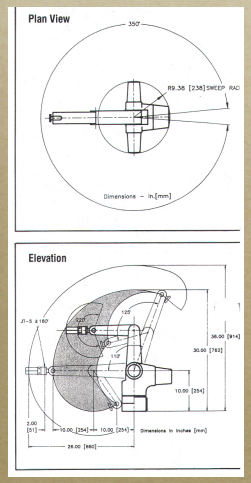
The CRS A150





$${}^{world}_{7}R=R_y(\theta_1)$$

The CRS A150



$$^{1}_{tool}R=R_z(\theta_5) \qquad ^{world}_{7}R=R_y(\theta_1)$$

$$^{2}_{1}R=R_x(\theta_4)$$

$$^{3}_{2}R=T(0,0,L_2)$$

$$^{4}_{3}R=R_x(\theta_3)$$

$$^{5}_{4}R=T(0,0,L_1)$$

$$^{6}_{5}R=R_x(\theta_2)$$

$$^{7}_{6}R=T(0,-L_0,0)$$

The CRS A150

$$^{world}_{tool}R=\,^{world}_{7}R\,^{7}_{6}R\,^{6}_{5}R\,^{5}_{4}R\,^{4}_{3}R\,^{3}_{2}R\,^{2}_{1}R\,^{1}_{tool}R$$

So given a point $_{tool}P$ you can determine this point in the world coordinate frame

$$_{tool}P=[0\ 0\ 0\ 1]^T$$

Inverse Kinematics

$$^{world}_{tool}R[0\ 0\ 0\ 1]^T=[x\ y\ z\ 1]^T$$

Cannot solve this (in general) for the controllable parameters

Assignment #1

1. Develop the inverse kinematics for the A150. Do the inverse kinematics for the arm. Given a desired position of the end effector plate (x,y,z) compute the joint **angles** that satisfy this or determine that no such joint angles exist. Note that you do not have to work out wrist twist, but wrist bend does contribute to this.

   Use the arm simulator to verify that your derivation is correct.

   NB: There will exist multiple solutions for certain (x,y,z) values. You can use either here, but consider the following question.

2. Write code that drives the real arm from one (x,y,z) setting to another. **In a straight line.** (The centre of the gripper plate must move in a straight line.) Determine if this is possible before moving the arm. Be careful with multiple solutions to the inverse kinematics problem.

   NB: The arm should move smoothly from one position to another.

3. Write code to drive the mobile base in a square, one meter on a side.

Marking You must hand in a pdf document with written solutions to questions (1) and (2). Questions (2) and (3) also require a software implementation that will be evaluated in two ways. (a) you must hand in (again as a pdf) your code to (1), (2) and (3), and in addition your code will be demonstrated in the lab after February 19th.

Mobile Base

- *Nomad Superscout*
  - *from rob04*
  - *ssh -p 23 cse4421@192.168.2.3*
  - *(password is xxxxxx)*
  - *reactived.scout (port 20000)*
- *on rob04 'ReactiveRobot.java*

ReactiveRobot.java

- *ReactiveRobot(String host, int port)*
- *boolean isAlive()*
- *boolean startup()*
- *void shutdown()*
- *setPose(RobotPose r)*
- *setGoal(RobotPose r)*
- *RobotPose queryPose()*

ReactiveRobot.java

- *int queryState()*
  - *1 if at goal*
  - *0 if moving*
  - *2 if blocked*
- *RobotPose snapPicture()*
- *Image getImage()*

ReactiveRobot.java

- *void playSound(String fname)*
- *float[] getSonar()*

Robot rules

- *Not moving **plug it in***
- *About to hit something **pick it up***
- *Don't abuse the account*
- *If it breaks, let me know ASAP*

Project

- *Proposal is due February 5th (aka next Monday)*
- *Proposal should be 1-2 pages in length*
  - *pdf please*
- *May wish to use a Gantt Chart to structure your time.*

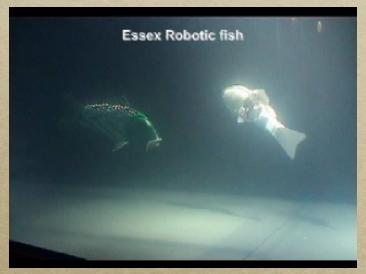
Robot Motion Summary

- *Many (many x many) different ways of getting a robot to move from place to place.*
- *Some astounding different mechanism for robot drive (legs, wheels, tracks, flying, swimming, other)*

Kinematics vs. Dynamics

- *Kinematics - study of motion without considering the forces required.*
  - *Good for simple, static structures.*
- *Dynamics - study of motion while considering the forces*
  - *Good for legged, flying, swimming, (anything for which kinematics are insufficient)*

Essex Fish

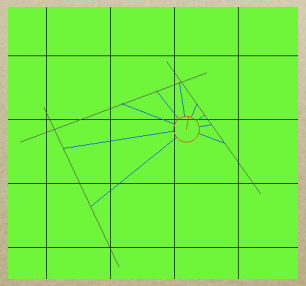


Honda Walking Robot

Others...

Robot Simulation

- *Kinematic Simulation*
  - *Easy to use*
  - *Easy to program*
  - *Simulates the kinematic (geometric) properties of the device.*

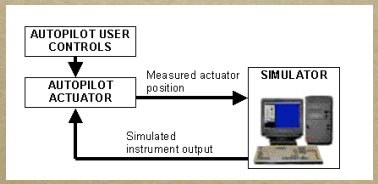


jRobot

Robot Simulation

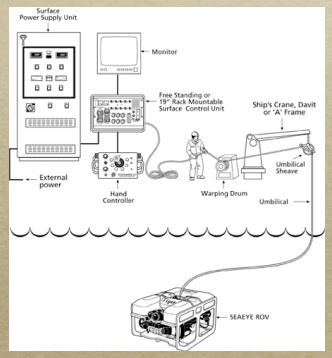
- *Dynamic simulation*
  - *Much more difficult to program*
  - *More accurate*
  - *Simulate the forces that can*

Shipsim



Tethers

- *Reality of many robots (especially ROV, UUV).*
- *Provides power, data, processing, remote operation.*
- *But there is a cost.*



Tethers

- *Expense*
- *Management Issues*
- *Mobility Issues*

Next steps...

- *Given that we can move the vehicle, we need to sense the world in order to interact with it (even if it is only to not hit it).*
- *Chapters 3 & 4 deal with different classes of sensors.*