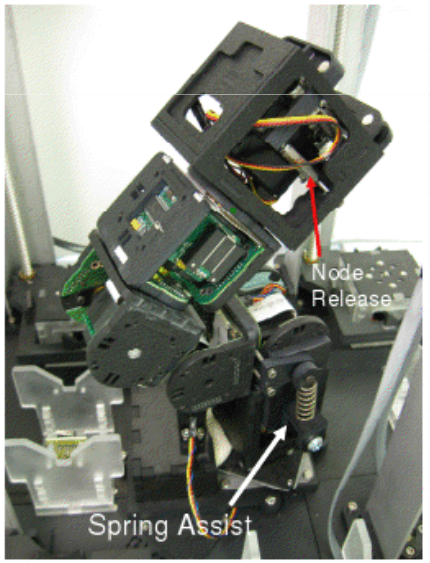
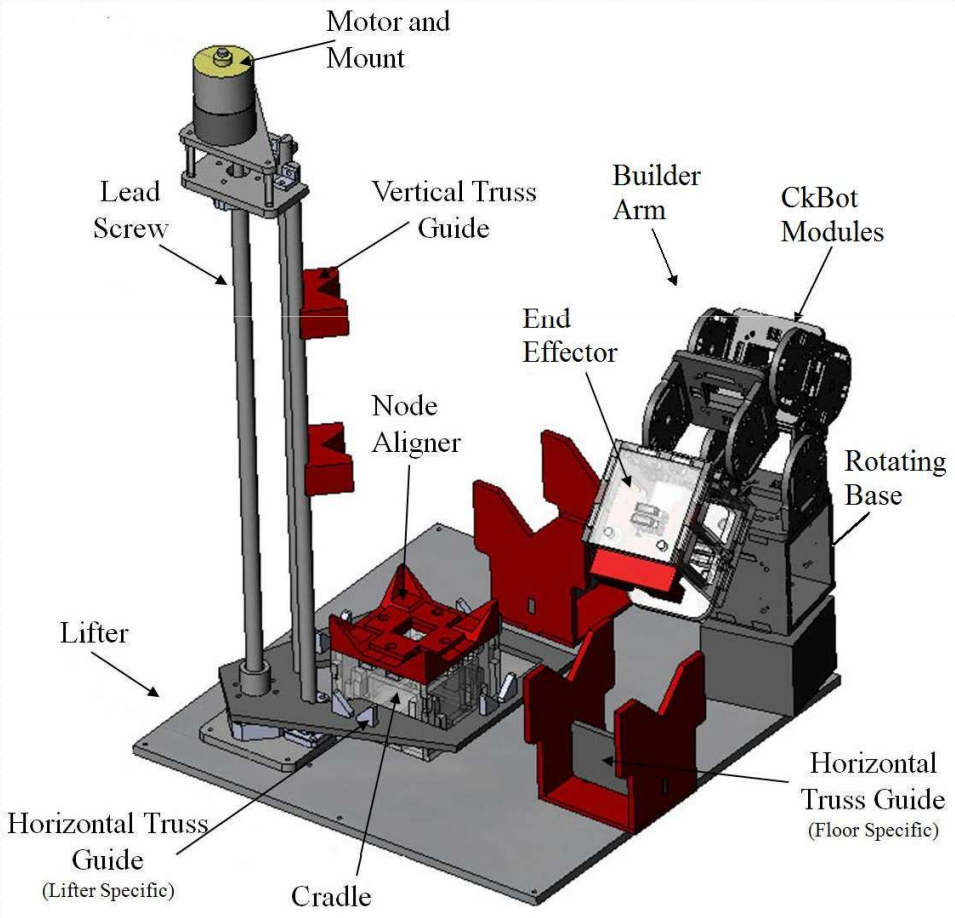
Factory Floor Testbed MS2

Stefan Kristjansson
Andrew Lawrence
Richard Wood

Table of Contents



System


Motor and Mount
Lead Screw
Vertical Truss Guide
Builder Arm
CkBot Modules
End Effector
Node Aligner
Rotating Base
Lifter
Horizontal Truss Guide (Floor Specific)
Horizontal Truss Guide (Lifter Specific)
Cradle

Project Update – Initial Goal

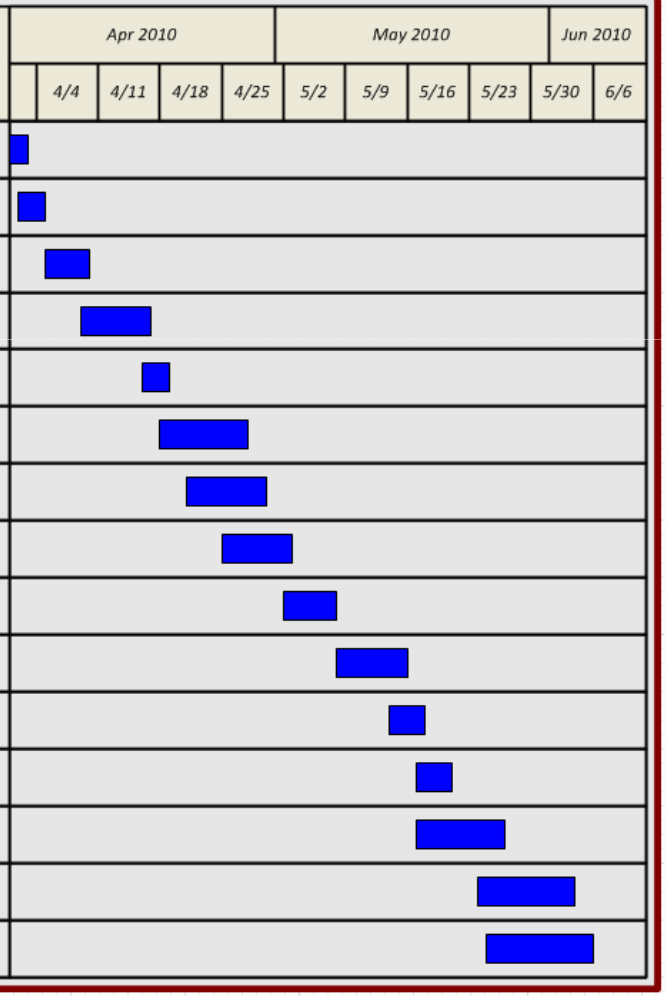
- To design high level control system to:
  - Build rectangle -> cube -> structure
  - Manage resources for single tile construction
  - Manage resources for multiple tile construction
  - Detect errors, faults and make repair as necessary

Project Update – New Goal

- Characterize the system
- Derive dynamic and kinematic equations to define joint/module forces and interaction
- Develop Path Planning based on Torque Minimization
- Once controller design developed, continue with original project plan

New Schedule

ID	Task Name	Start	Finish	Duration
1	Assemble robot and modules; wire system	4/1/2010	4/2/2010	2d
2	Compile existing CCL code	4/2/2010	4/4/2010	3d
3	Establish Communication, PC > Hardware through Ipython in Linux	4/5/2010	4/9/2010	5d
4	Develop initial path planning	4/9/2010	4/16/2010	8d
5	Establish communication through windows and GUI	4/16/2010	4/18/2010	3d
6	Calibrate and Characterize the System	4/18/2010	4/27/2010	10d
7	Develop dynamic and kinematic equations	4/21/2010	4/29/2010	9d
8	Design Integral controller from system equations	4/25/2010	5/2/2010	8d
9	ICRA Competition	5/2/2010	5/7/2010	6d
10	Develop final path planning	5/8/2010	5/15/2010	8d
11	Develop high level module to build rectangles	5/14/2010	5/17/2010	4d
12	Nominal demonstration: Build a cube	5/17/2010	5/20/2010	4d
13	Get CCL to work	5/17/2010	5/26/2010	10d
14	Large simulation execution	5/24/2010	6/3/2010	11d
15	Disturbance detection and correction	5/25/2010	6/5/2010	12d

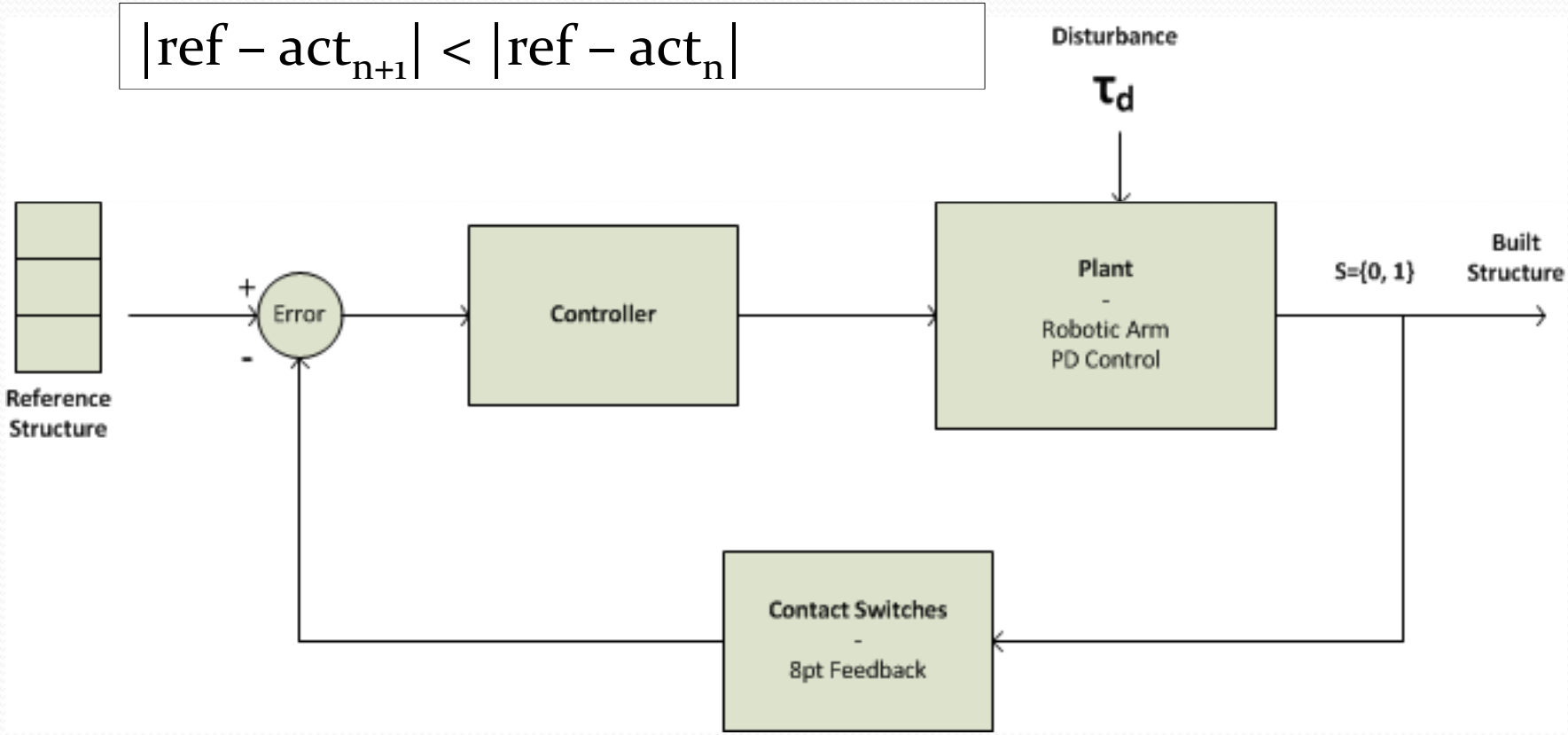


High Level System Inputs/Outputs

- Inputs:
  - Reference structure
    - Send program
    - Robot constructs nodes and trusses
    - If fails to accurately place a node or truss, tries again
    - Continues building until all feedback matches reference
- Outputs
  - Contact Switches
  - Compare to reference structure

High Level Block Diagram

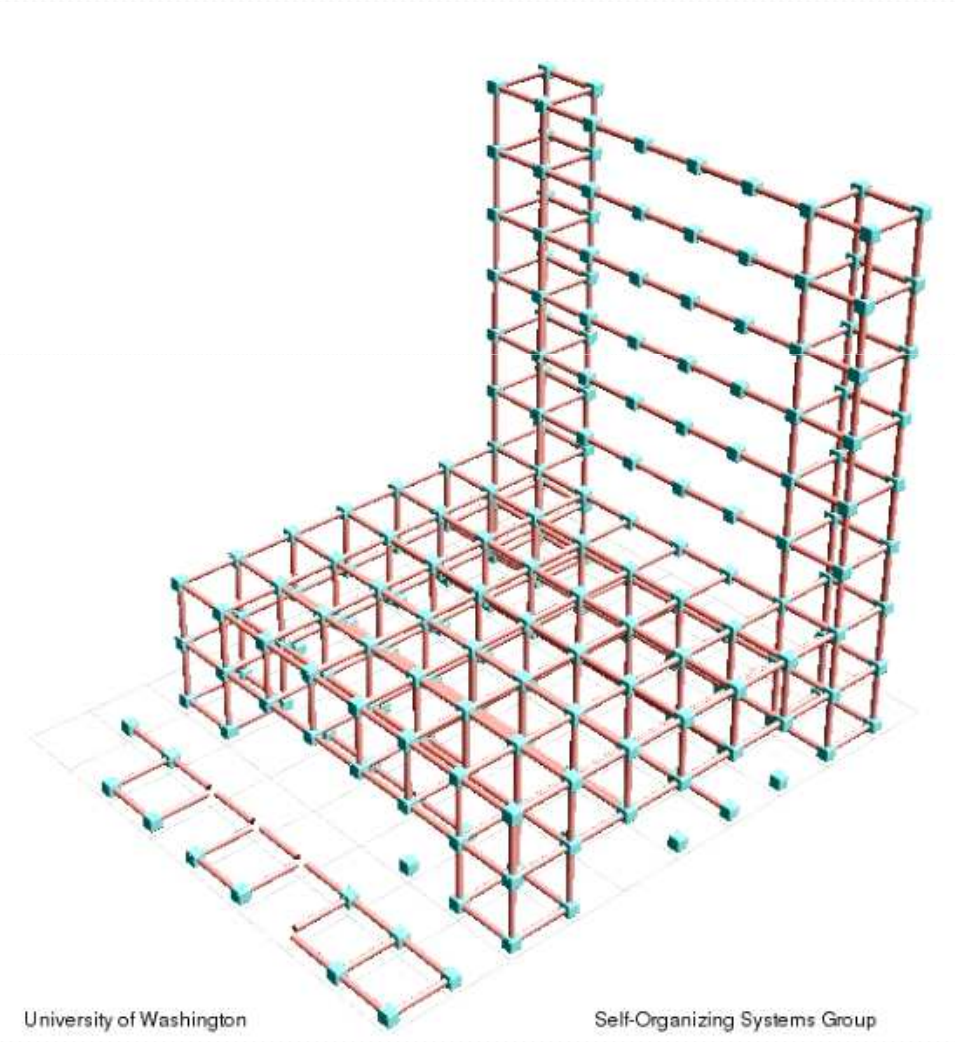
$$|ref - act_{n+1}| < |ref - act_n|$$



High Level System State

Input		N(1,1)	N(1,2)	N(1,3)	N(1,4)	Th(1,1)	Th(1,2)	Th(1,3)	Th(1,4)	Tv(1,1)	Tv(1,2)	Tv(1,3)	Tv(1,4)	
		N(2,1)	N(2,2)		…	…			Th(2,4)	…			Tv(2,4)	
		…							…				…	
		N(n,1)			…	…			Th(n,4)	…			Tv(n,4)	
Output		N(1,1)	N(1,2)	N(1,3)	N(1,4)	Th(1,1)	Th(1,2)	Th(1,3)	Th(1,4)	Tv(1,1)	Tv(1,2)	Tv(1,3)	Tv(1,4)	
		N(2,1)	N(2,2)		…	…			Th(2,4)	…			Tv(2,4)	
		…							…				…	
		N(n,1)			…	…			Th(n,4)	…			Tv(n,4)	

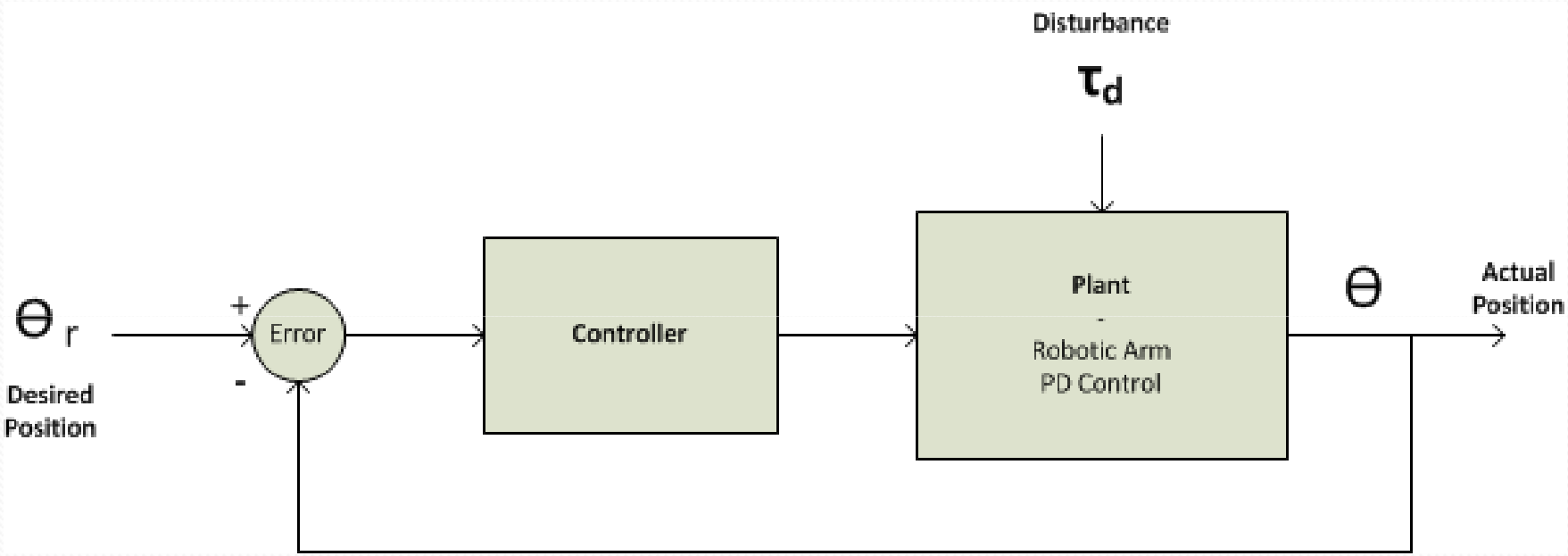
Plant Simulation



High Level Characterization

- Node Placement
  - Total: 10 Success: 6 Failure: 4
  - Success Rate: 60%
- Truss Placement (Horizontal)
  - Total: 10 Success: 8 Failure: 2
  - Success Rate: 80%
- Truss Placement (Vertical)
  - Total: 10 Success: 7 Failure: 3
  - Success Rate: 70%

Low Level Block Diagram



$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + N(\theta,\dot{\theta}) = \tau$$

$$\dot{\theta} = k(\theta_{r,i} - \theta_i) + \tau_i$$

$$u = \theta_r$$
$$y = \theta$$

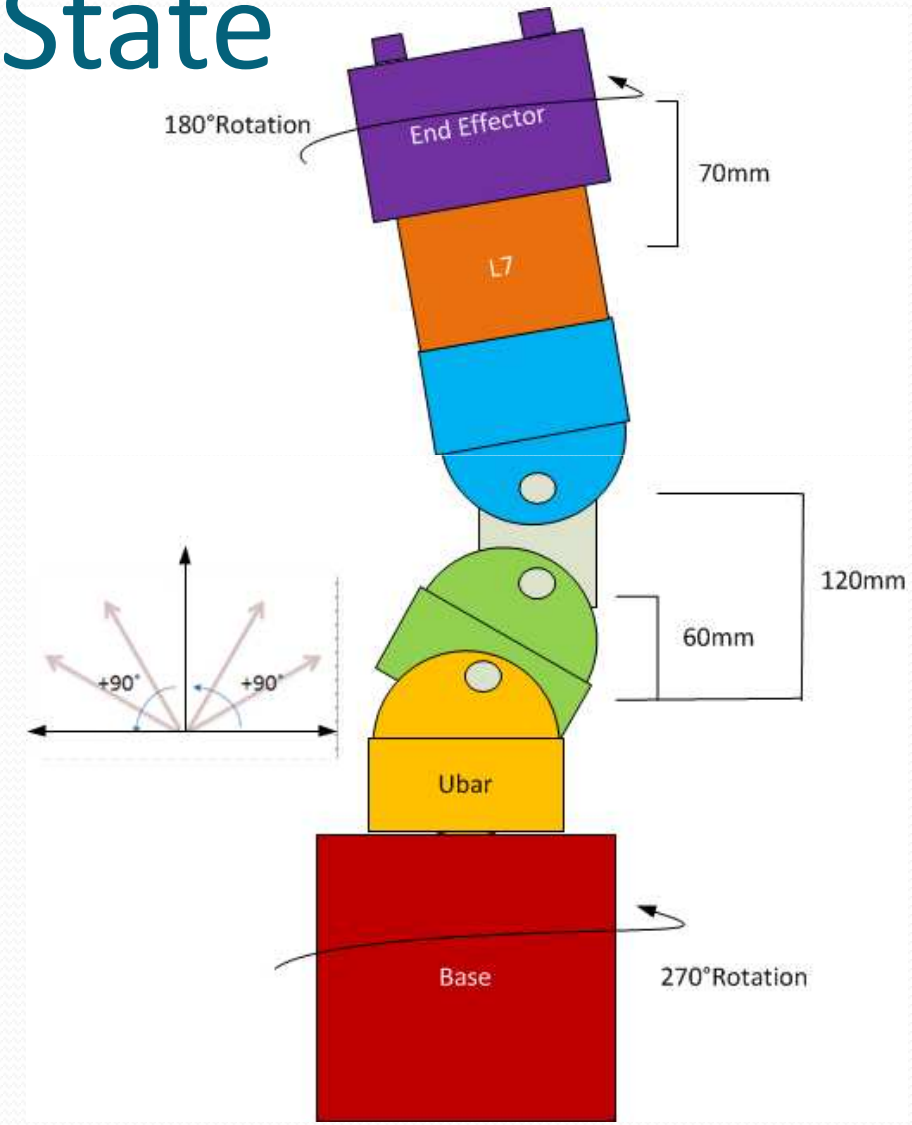
$$\dot{\theta} = -k_p\theta + k_p u$$

Low Level System Inputs/Outputs

- Inputs:
  - θ_r – Desired motor position
    - 5 modules -> 5 controllers -> 5 desired motor positions
- Outputs:
  - θ – Actual motor position
    - 5 modules -> 5 controllers -> 5 actual motor positions

Internal System State

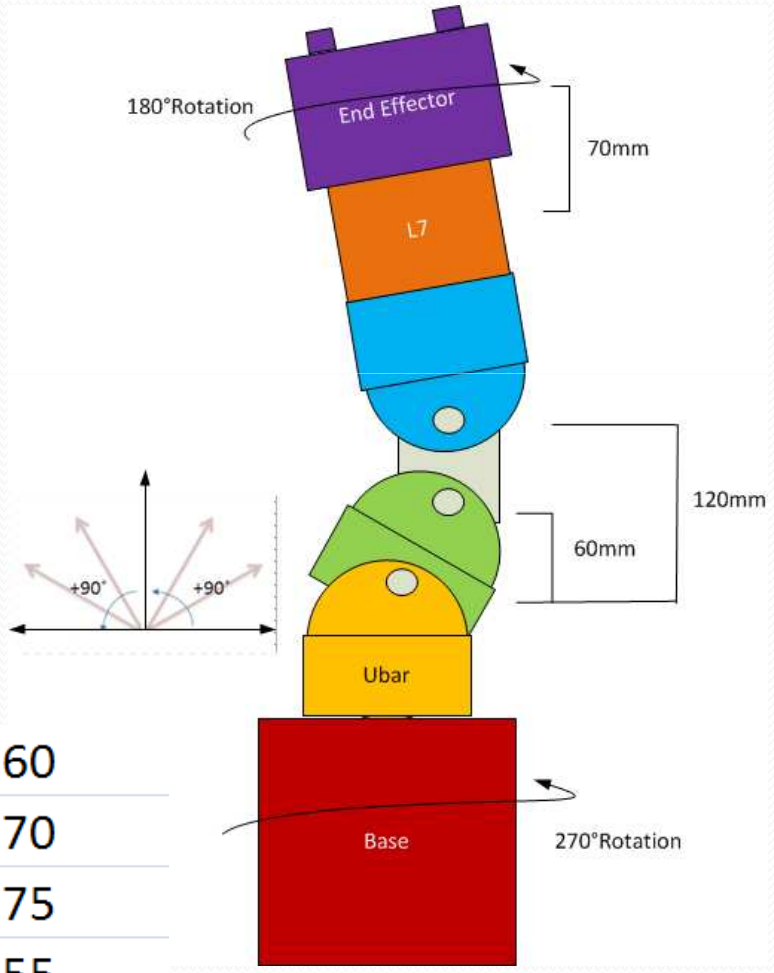
- 1 Base
  - Rotates arm 270°
- 3 Ubar modules
  - Rotates 180° in the xy plane
- 1 L7 module
  - Rotates end effector 180°
- 1 End Effector
  - Module and Truss manipulation



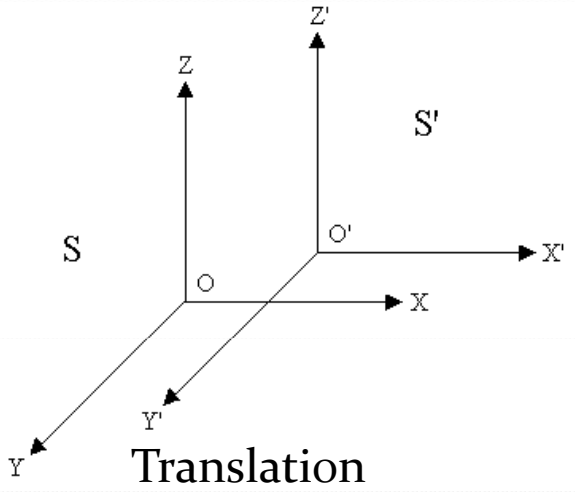
Model Parameters and Value Estimations

Module	Mass (g)	Dimensions (mm)
Ubar	138	W60xL60xH60
L7	138	W60xL60xH60
Base	200	W60xL60xH85
End-Effector	130	W60xL67xH80
Node	129	W58xL58xH58
Truss	137	W35xL235xH35

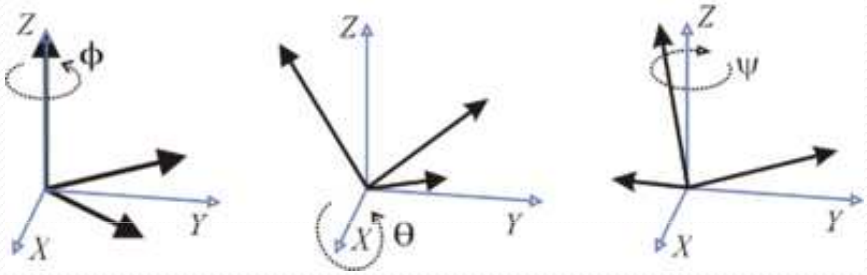
Distance between ubar pivot points (mm)	60
Distance between C7 and EEf (mm)	70
Vertical Distance from EEf to Node (mm)	75
Lateral Distance from Eef to Truss (mm)	55



Kinematics



Translation

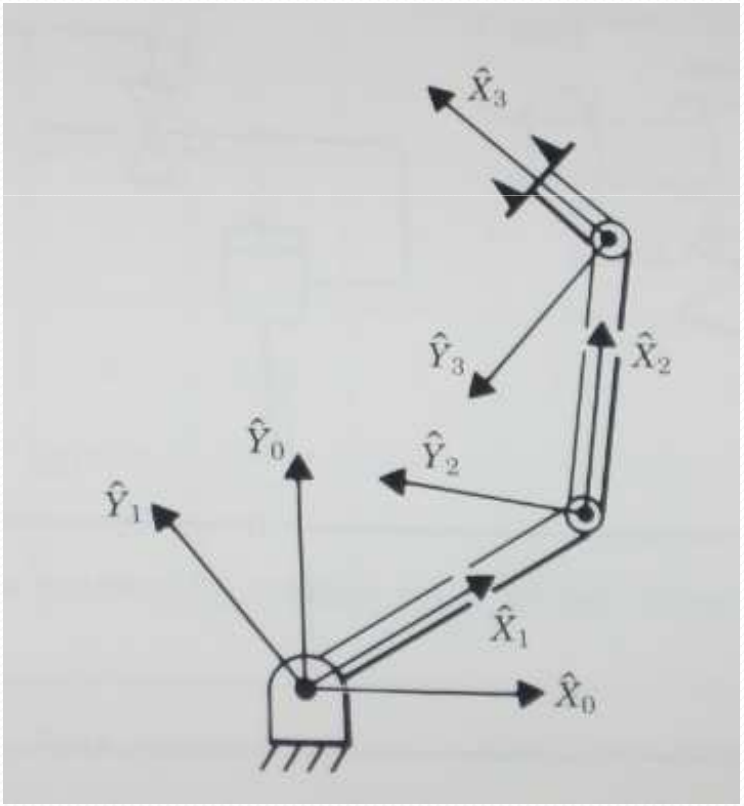


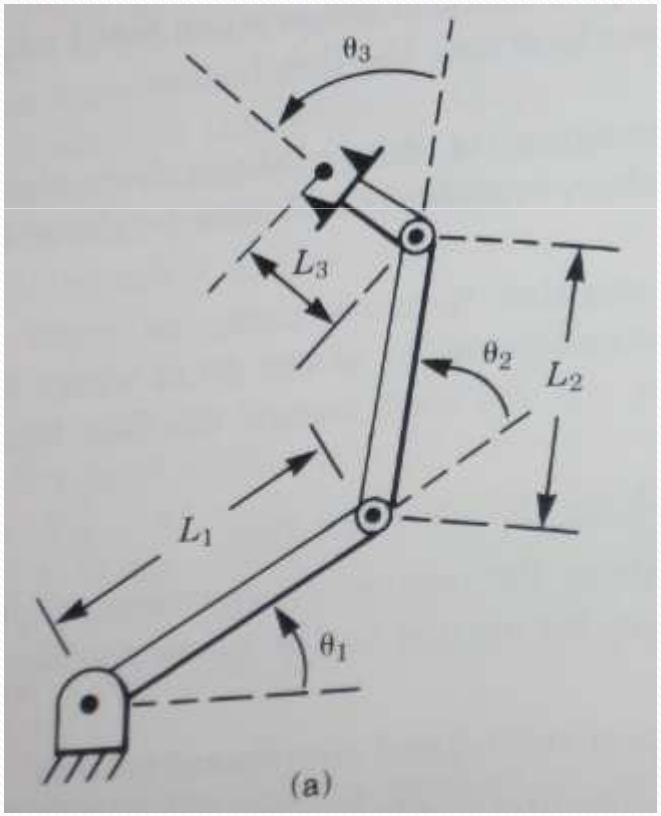
Rotation

Relation of system i to i-1

$$ {}^{i-1}_{i}\boldsymbol{T} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \cos(\theta i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

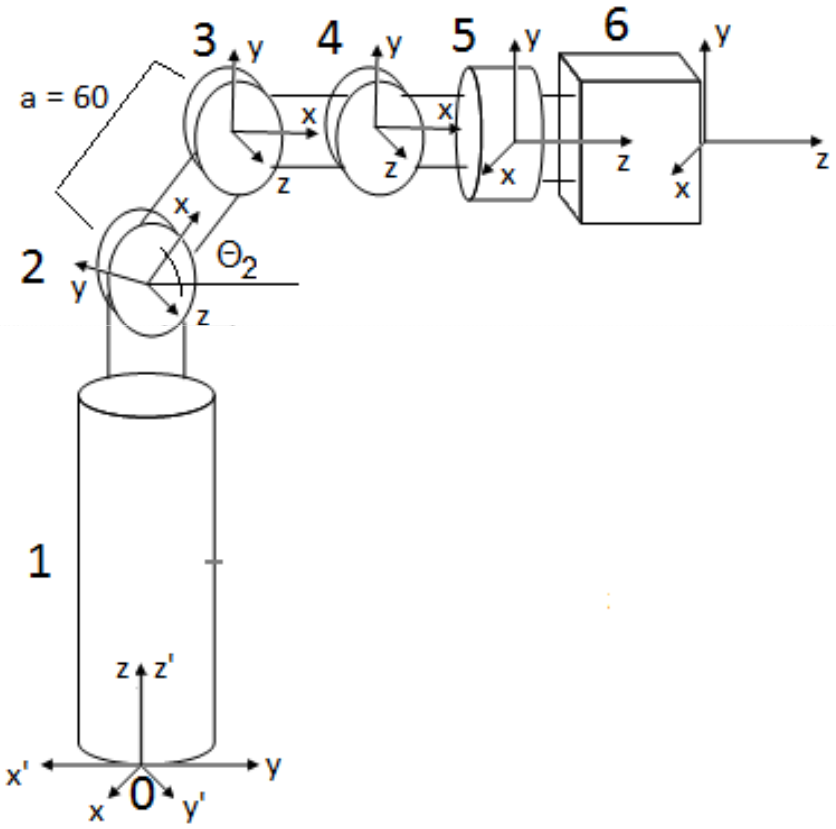
Planar Robotic Arm





Kinematic Link Parameters

i	α_{i-1}	a_{i-1}	d_i	Θ_i
1	α_1	0	0	0
2	0	155	0	Θ_2
3	0	60	0	Θ_3
4	0	60	0	Θ_4
5	α_5	60	0	0
6	0	100	0	0



'a' distances are measured in mm

$\alpha_1 \rightarrow$ Angle of Base Rotation

$\Theta_2 \rightarrow$ Angle of Bottom UBAR

$\Theta_2 \rightarrow$ Angle of Mid UBAR

$\Theta_2 \rightarrow$ Angle of Top UBAR

$\alpha_1 \rightarrow$ Angle of L-7

Reverse Kinematics

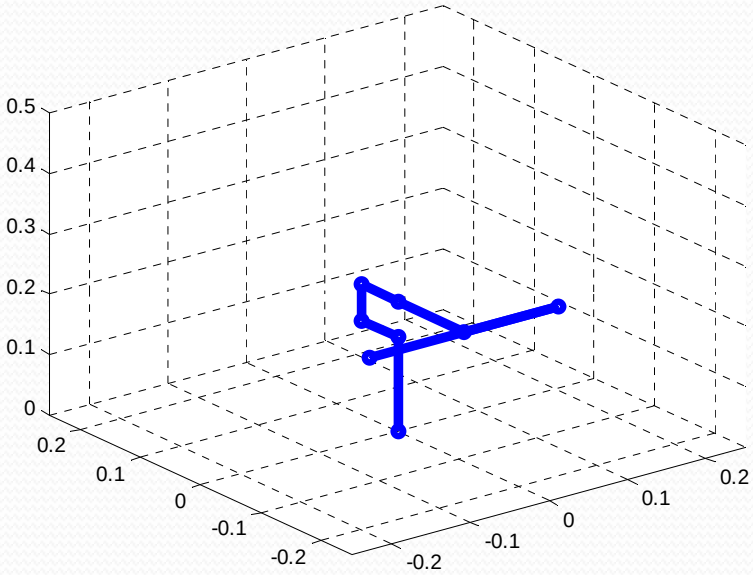
- Positions of frames related back to origin through

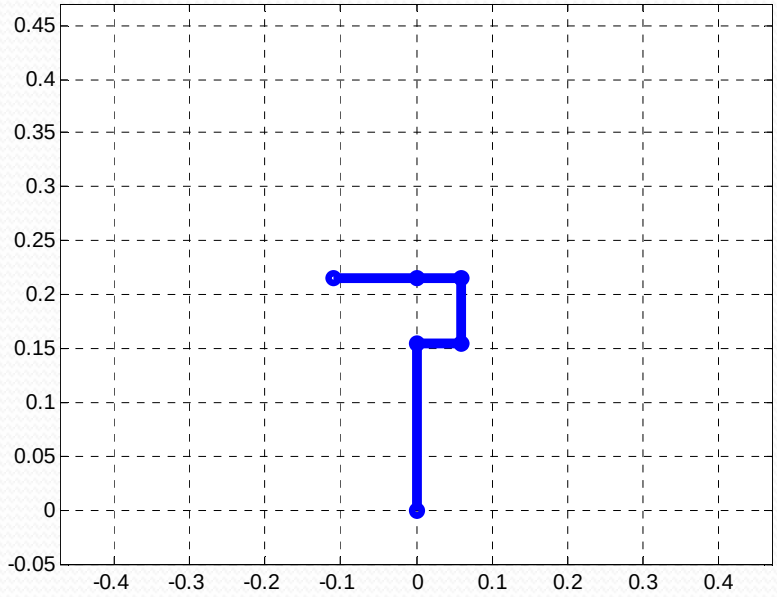
$$ {}^{i-1}_{i}\boldsymbol{T} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \cos(\theta i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha i\text{-}1)d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

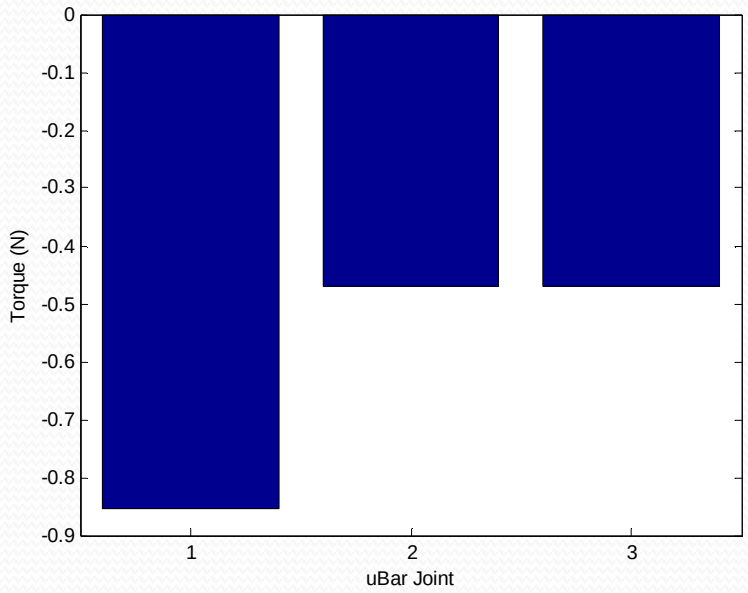
- Torques are calculated about each U-BAR joint.
  - Reverse Kinematics will help plan arm paths by minimizing torques.
  - Each servo has 417 oz-inch of torque

Path Planning
- Torque Minimization

Path Planning
– Torque Minimization







Preliminary Torque Estimation

	ANGLE (degrees)			TORQUE (N)	
Angle Ubar 1	Angle Ubar 2	Angle Ubar 3	Torque Ubar 1	Torque Ubar 2	Torque Ubar 3
-90	0	0	1.1183	0.5592	0.4709
-90	0	45	0.8425	0.4213	0.333
-90	0	90	0.5625	0.342	0
-90	45	0	0.9425	0.5518	0.3879
-90	45	45	0.6424	0.3984	0
-90	45	90	0.2014	0.083	-0.279
-90	90	0	0.326	0	0
-90	90	45	0.109	-0.083	-0.279
-90	90	90	0	0	0

Controllability of System

- In general:
  - Moves where we want
  - Moves when we want

- Known controllability problems include:
  - Inconsistent operation
  - Steady-state error
  - Oscillation
  - Velocity of movement

- Control issues can be addressed through application of:
  - System calibration
  - Integral controller
  - Software function calls or algorithms

Bibliography

CCL: The Computation and Control Language. Retrieved April 05, 2010, from University of Washington, Self Organizing Systems Lab website, http://soslab.ee.washington.edu/mw/index.php/Code
Primary reference for documentation and source code for CCL.

Phidgets. Retrieved April 13, 2010, from Phidgets website, http://www.phidgets.com/
Used as the source for phidget I/O documentation and source code.

Mason, Matthew. (2001). *Mechanics of Robotic Manipulation.* Massachusetts: The MIT Press.
Utilized as a supplemental reference for forward kinematic equations of the robotic arm.

Modlab CKBot Graphic User Interface Manual. Retrieved April 10, 2010, from UPenn, Modular Robotics Laboratory website, http://modlabupenn.org/efri/
Used as the primary reference guide for interfacing with the CKBot modules in the Windows environment.

M. Yim, P. J. White, M. Park, & J. Sastra, "Modular Self-Reconfigurable Robots", 2009, pp. 5618-5631.
A pivotal paper on self-reconfigurable robots; one of the primary CKBot modular robotic design sources.

Nurrat, Richard, & Li, Zexiang, & Sastry, S. (1994). *A mathematical introduction to robotic manipulation.* Florida: CRC Press.
Utilized for instruction on the derivation of torque equations for robotic arm systems.

Craig, John J. *Introduction to Robotics: Mechanics and Control.*(1989) Reading, Mass.: Addison-Wesley.
Used as the primary source for kinematic equation derivation and vector equation manipulation.

IPython Documentation. Retrieved April 12, 2010, from IPython website, http://ipython.scipy.org/moin/
Used for reference documentation on IPython documentation and for the source code for compilation. Ipython is used to interface with the CKBots.