Actuators & Motion

Manuela Veloso

15-491, Fall 2008

http://www.andrew.cmu.edu/course/15-491

Computer Science Department

Carnegie Mellon University

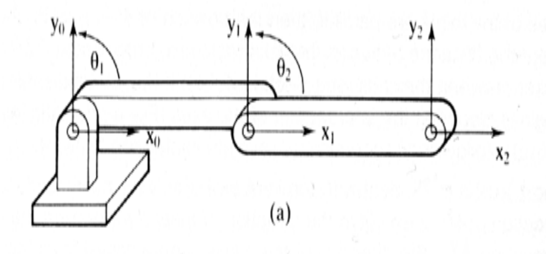
Robot Motion

- Forward and Inverse Kinematics
- PID Control
- Frame-Based Motions
- Modeling Effects of Motions

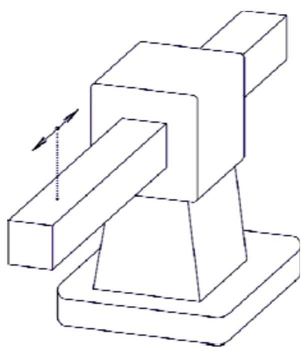
Forward Kinematics, Inverse Kinematics, & PID Control in a Nutshell

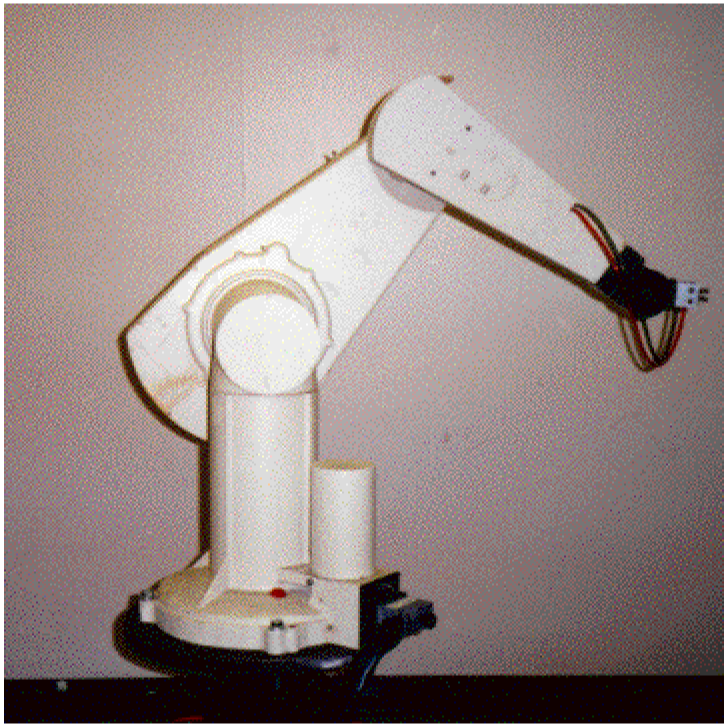
Robotic Arms

Revolute Joint



Prismatic Joint

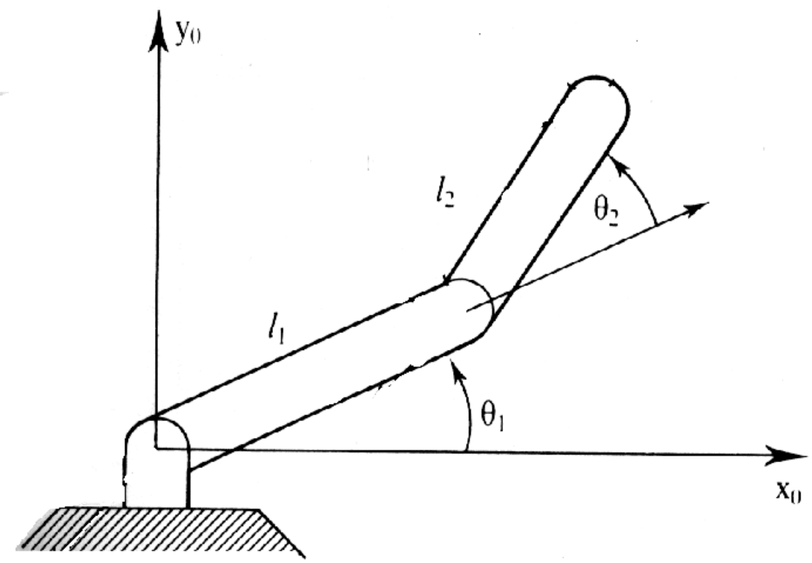




Forward Kinematics

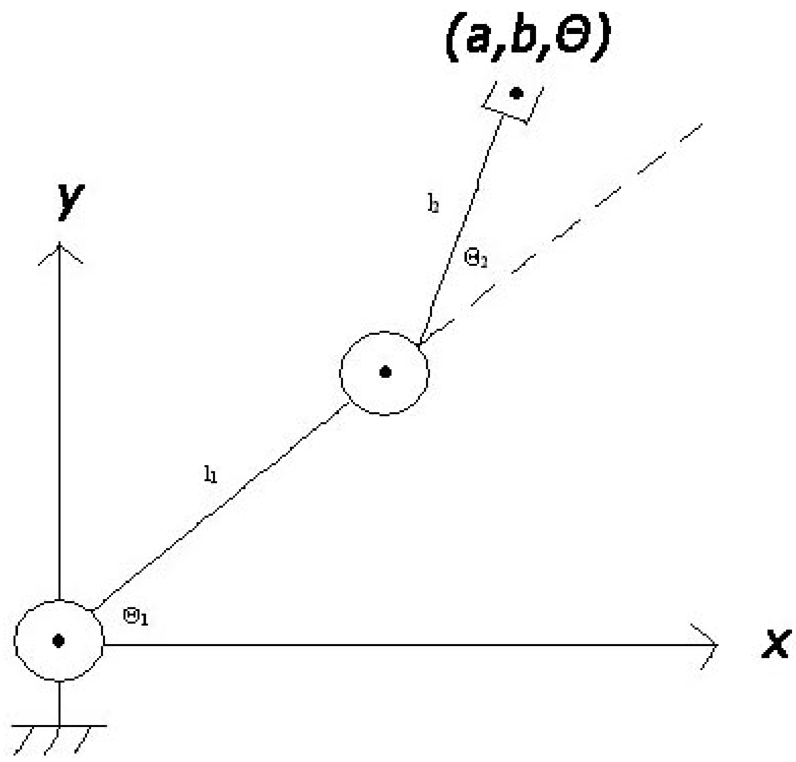
- The problem: determine the position of the end of the robotic arm given θ_1 and θ_2 and l1 and l2

- Geometric Approach
- Algebraic Approach



A simple example

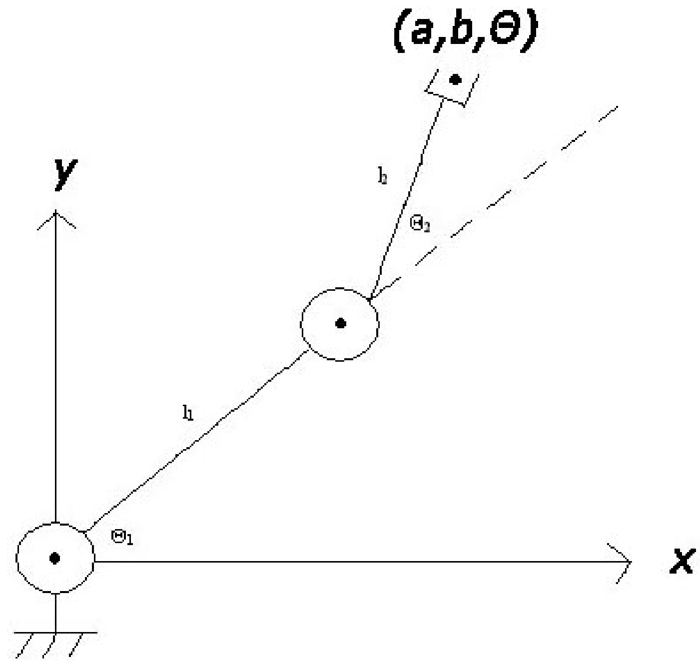
- Two links connected by rotational joints to a stable platform
- Given θ_1 and θ_2, solve for *a*, *b* and θ



Solution

- Can be solved trigonometrically:

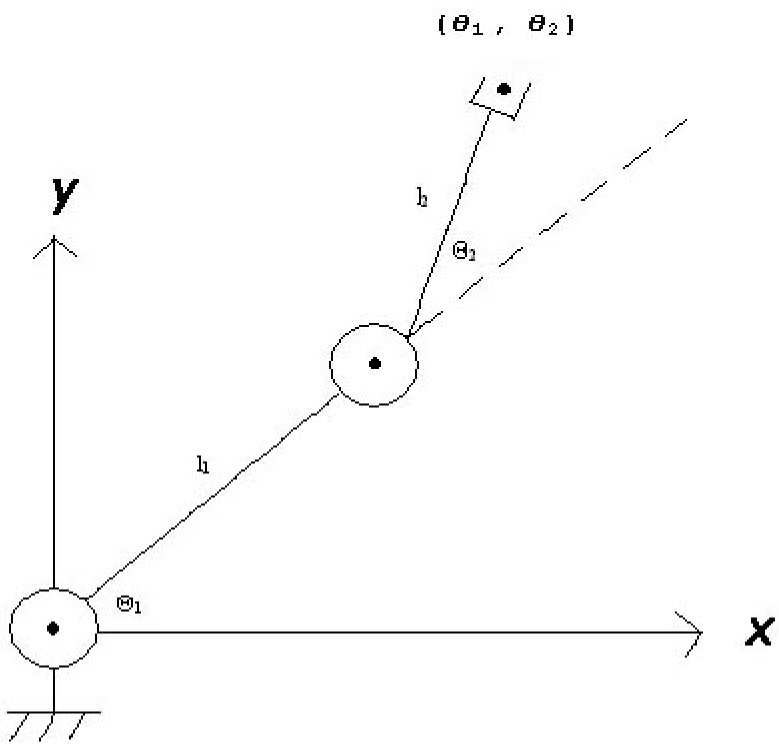
$$a = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$
$$b = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2)$$
$$\Theta = \theta_1 + \theta_2$$



Inverse Kinematics

- Going backwards
- Find joint configuration given position & orientation of tool (end effector)
- More complex (path planning & dynamics)
- Usually solved either algebraically or geometrically
- Possibility of no solution, one solution, or multiple solutions

Another example

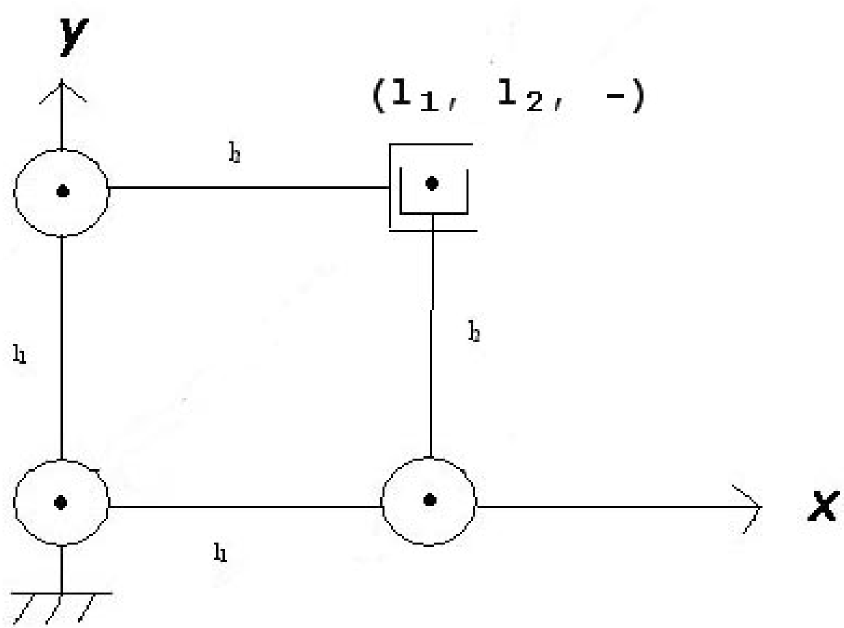


Lets assume l1 = l2

What is the configuration of each joint if the end effector is located at (l_1, l_2,-)?

(ie. Find θ_1 *and* θ_2)

Solution



$\theta_1=0, \theta_2 = 90$

Or

$\theta_1=90, \theta_2 = 0$

(Two Solutions)

The Math

- That was an easy one… what does the math look like?

$$c^2 = a^2 + b^2 - 2ab\cos C$$

$$(x^2 + y^2) = l_1^{\ 2} + l_2^{\ 2} - 2l_1l_2\cos(180 - \theta_2)$$

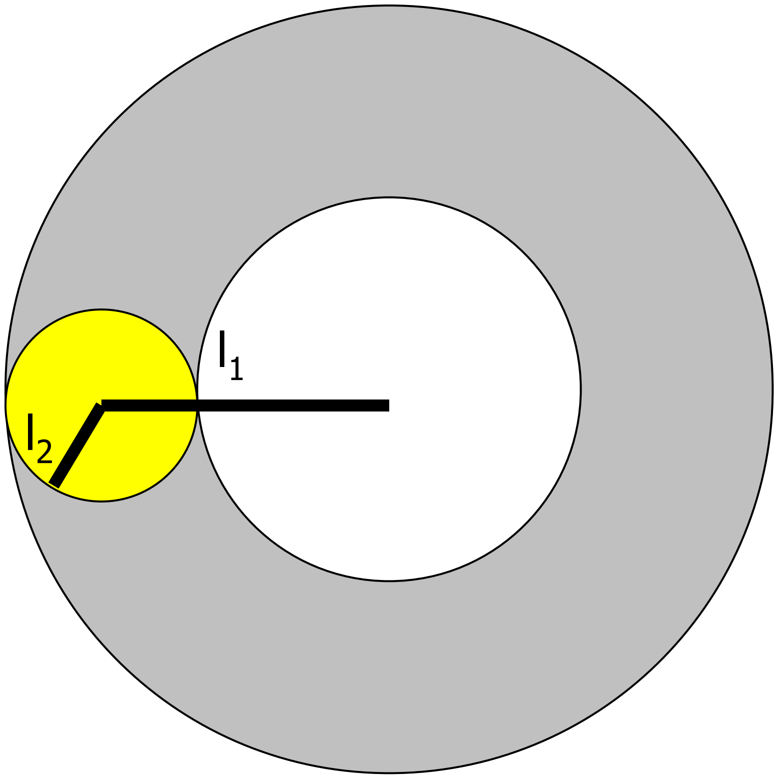
$$\cos(180 - \theta_2) = -\cos(\theta_2)$$

$$\cos(\theta_2) = \frac{x^2 + y^2 - l_1^{\ 2} - l_2^{\ 2}}{2l_1l_2}$$

$$\theta_2 = \arccos\left(\frac{x^2 + y^2 - l_1^{\ 2} - l_2^{\ 2}}{2l_1l_2}\right)$$

$$\theta_1 = \arcsin\left(\frac{l_2\sin(\theta_2)}{\sqrt{\mathrm{x}^2 + \mathrm{y}^2}}\right) + \arctan 2\left(\frac{\mathrm{y}}{\mathrm{x}}\right)$$

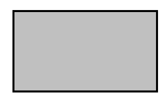
Reachable Workspace



Joint Limits:

$-180^\circ \leq \theta_1 \leq 180^\circ$

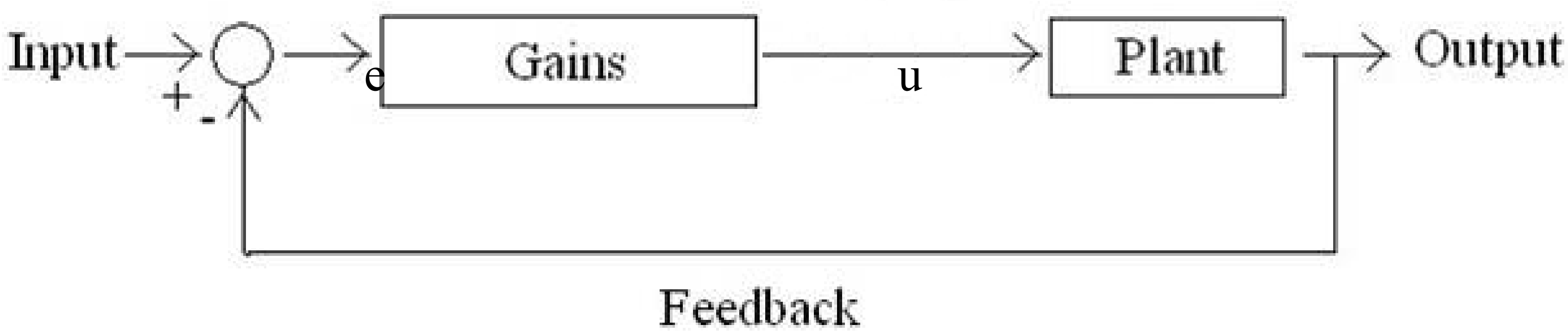
$-180^\circ \leq \theta_2 \leq 180^\circ$



PID Control

- Proportional Integral Derivative Control
- The Basic Problem:
  - We have n joints, each with a desired position which we have specified
  - Each joint has an actuator which is given a command in units of torque
  - Most common method for determining required torques is by feedback from joint sensors

The Control Loop



What is PID Control?

- Proportional, Integral, & Derivative Control
  - Proportional: Multiply current error by constant to try to resolve error
  - Integral: Multiply sum of previous errors by constant to resolve steady state error (error after system has come to rest)
  - Derivative: Multiply time derivative of error change by constant to resolve error as quickly as possible

Summary

- These concepts make up the low level functionality of general robot motion
- Much more information about PID Control and Forward & Inverse Kinematics in other Robotics courses

Example: AIBO Actuators

- 18 degrees of freedom with a continuously controllable range of motion
  - 3 DOF in each leg (12 total)
  - 3 DOF in the head
  - 2 DOF in the tail
  - 1 DOF in the jaw
- Each joint is controlled by specifying a desired *joint angle*
- 2 binary motors for the ears
- A speaker for general sound production

Frame-Based Motion

- Each motion is described by a series of "frames" which specify the position of the robot, and a time to interpolate between frames

- Movement between frames is calculated through linear interpolation of each joint

Examples: Valid Motion Frames

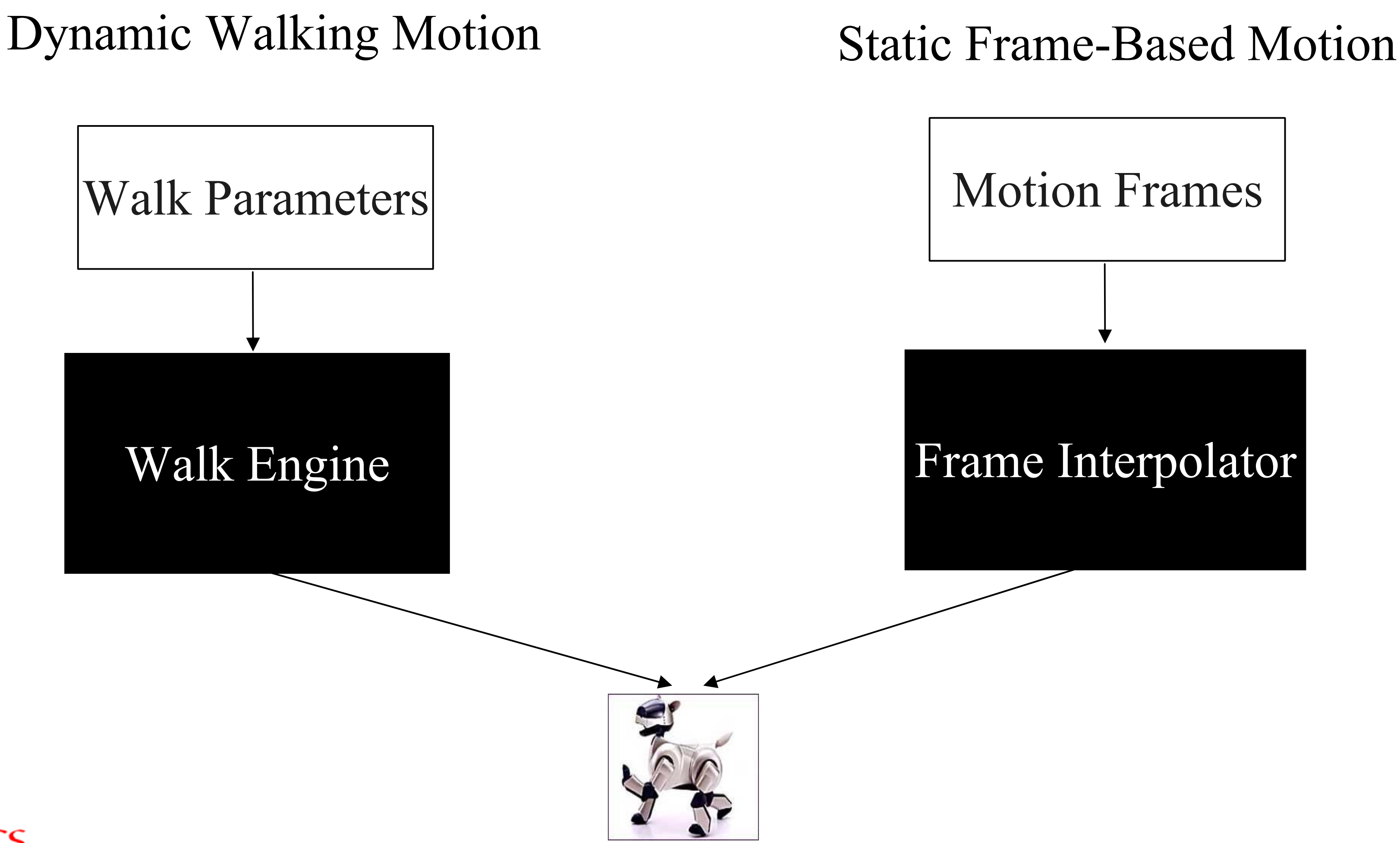
```
BodyPos(b,98,RAD(16));
HeadAng(b, 0.5, 1.5, 0.0);
LegPos(b,0, 123,  85, 0);
LegPos(b,1, 123,-85, 0);
LegAng(b,2, 0.1, 0.0, 0.2);
LegAng(b,3, 0.1, 0.0, 0.2);
m[n].body = b;
m[n].time = 100;
n++;

LegAng(b,0, 0.0, 1.5, 0.0);
LegAng(b,1, 0.0, 1.5, 0.0);
LegAng(b,2, 0.1, 0.0, 0.2);
LegAng(b,3, 0.1, 0.0, 0.2);
m[n].body = b;
m[n].time = 100;
n++;
```

```
BodyPos(b,98,RAD(16));
HeadAng(b, 0.5, 1.5, 0.0);
MouthAng(b,-.7);
LegPos(b,0, 123, 85,0);
LegPos(b,1, 123,-85,0);
LegPos(b,2, -80  , 75,0);
LegPos(b,3, -80  ,-75,0);
m[n].body = b;
m[n].time = 100;
n++;

m[n].body = b;
m[n].time = 100;
n++;
```

The Motion Interface



Kicking

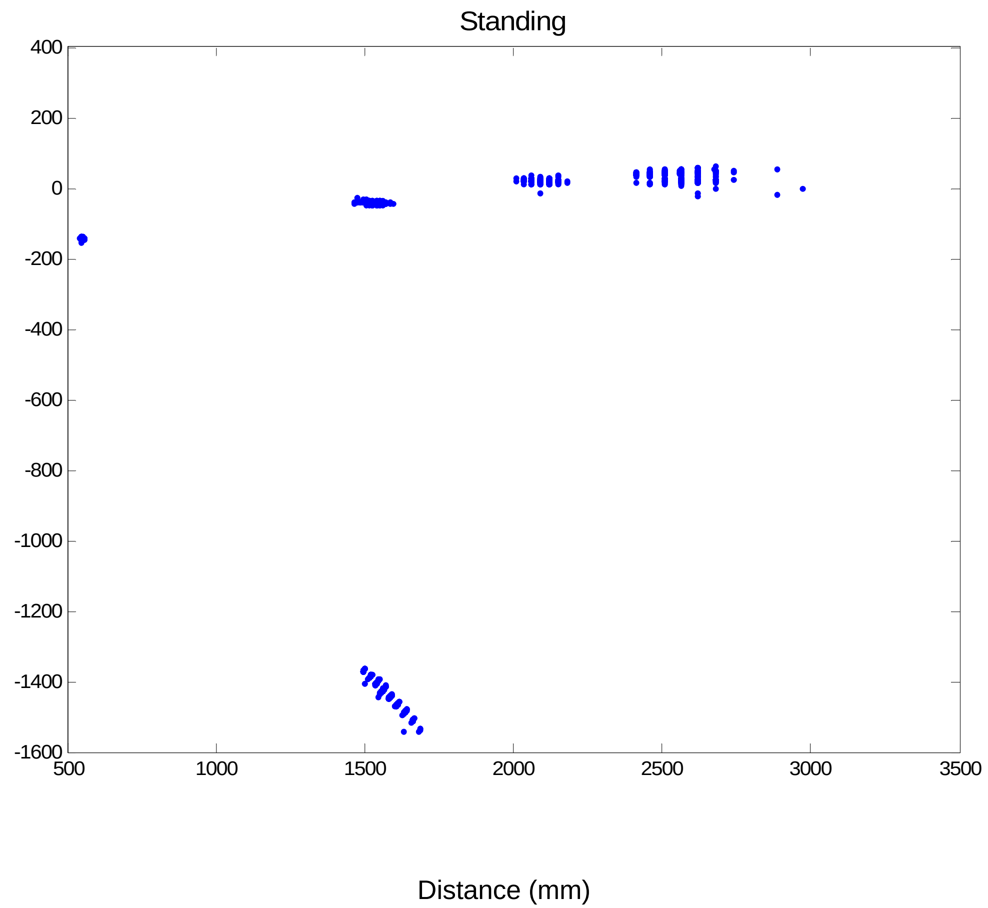
- A series of set positions for the robot
- Linear interpolation between the frames
  - Kinematics and interpolation provided by CMWalkEngine
- Set robot in desired positions and query the values of the joints

High Sensitivity to Parameters Good Settings for Effective Kick

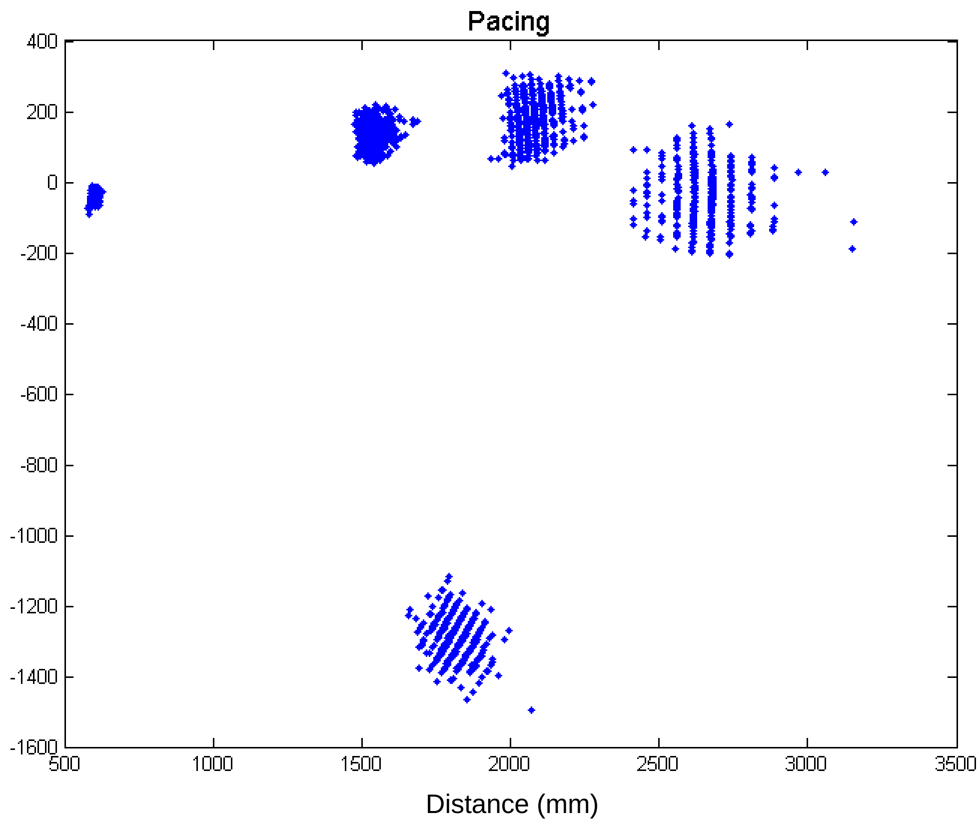
Use of Kicks in Behaviors

- Modeling *effects* of kicking motions
  - Ball vision analysis
  - Ball trajectory angle analysis
  - Kick strength analysis
- Kick selection for behaviors
  - Selection algorithm
  - Performance comparison

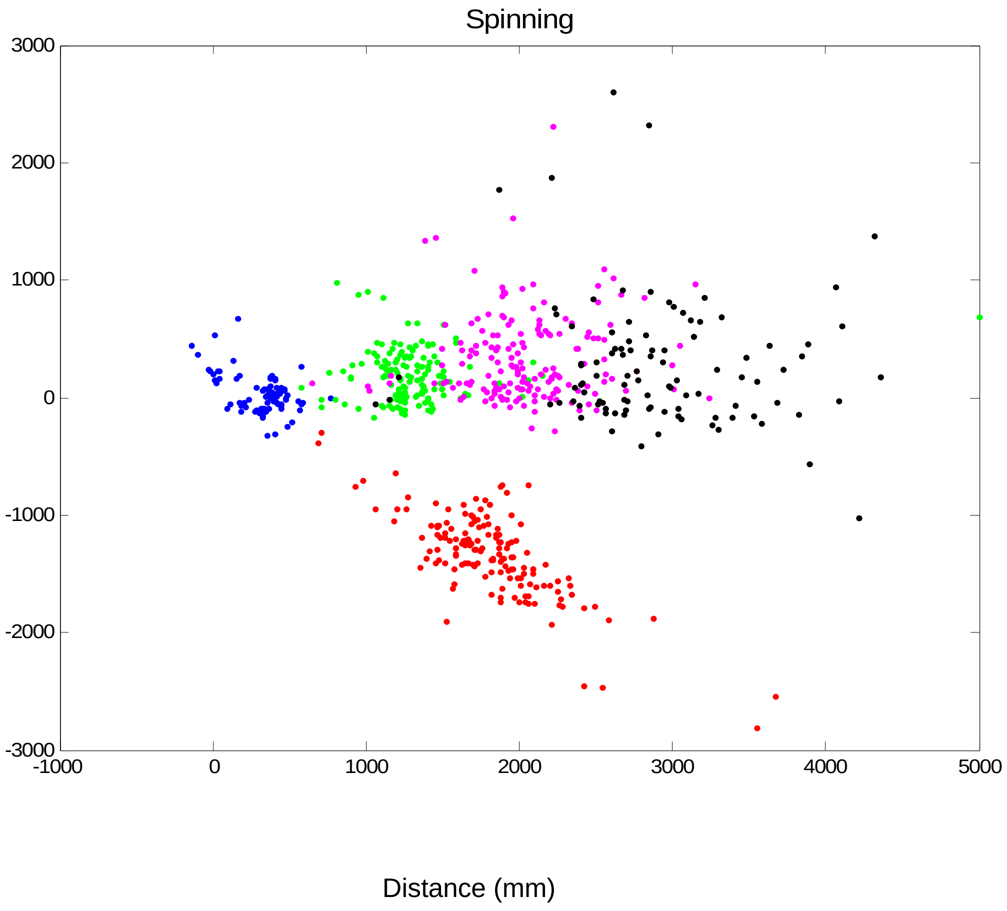
Accuracy of Object Detection Varies -- Robot Standing --



Accuracy of Object Detection Varies -- Robot Pacing --



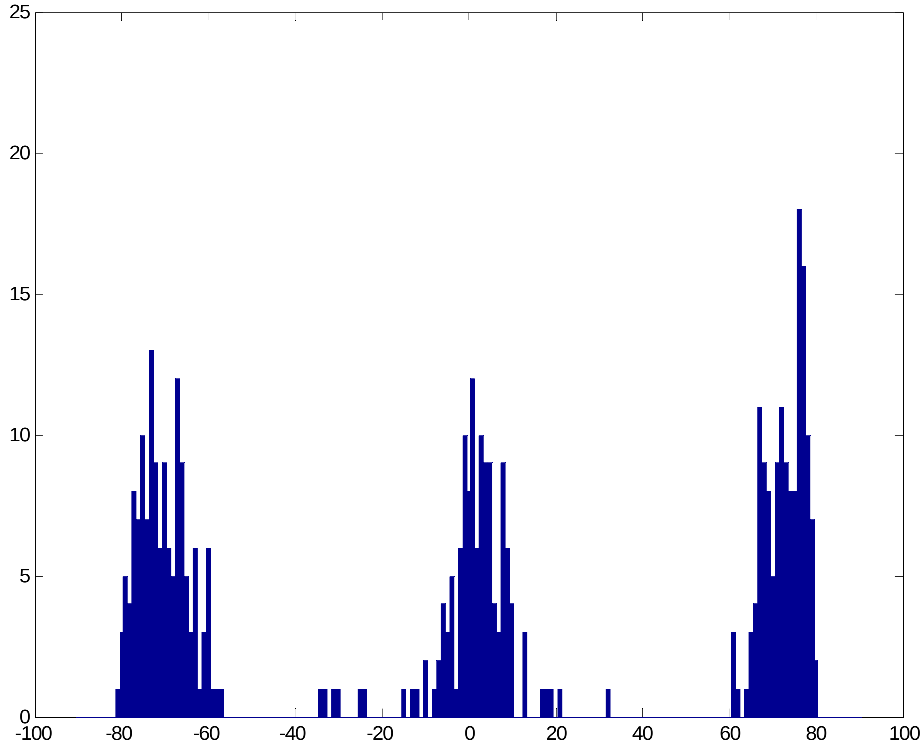
Accuracy of Object Detection Varies – Robot Spinning --



Ball Trajectory Angle

- Estimate the angle of the ball's trajectory relative to the robot
  - Track ball's trajectory after the kick
  - Retain information about ball position in each vision frame
  - Calculate angle of trajectory using linear regression

Angle Analysis



Right Head Kick

Forward Kick

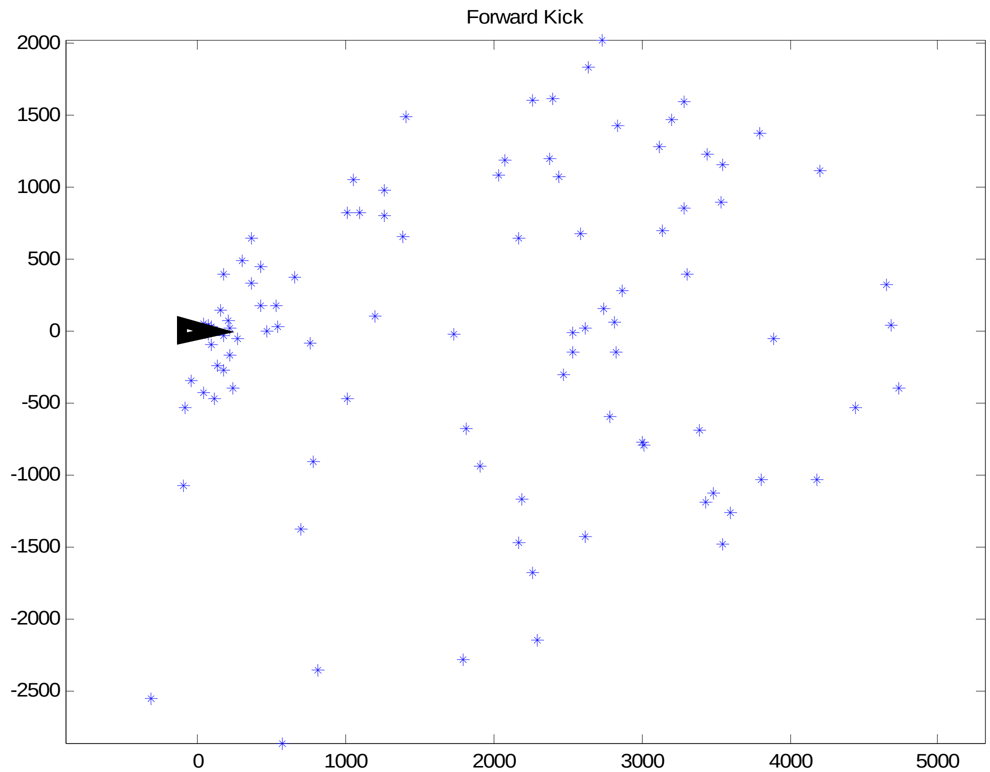
Left Head Kick

Kick Strength

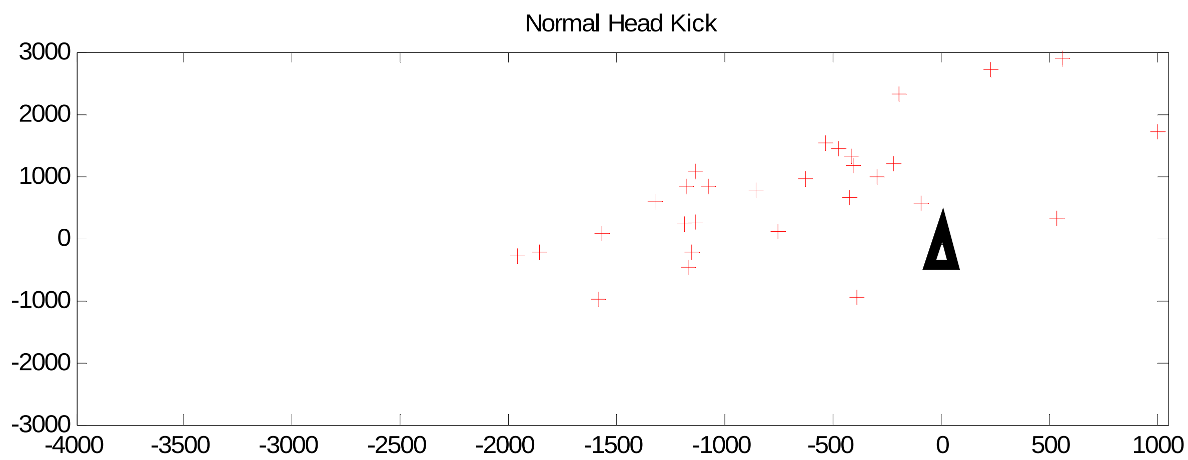
- Estimate the distance the ball will travel after a kick.

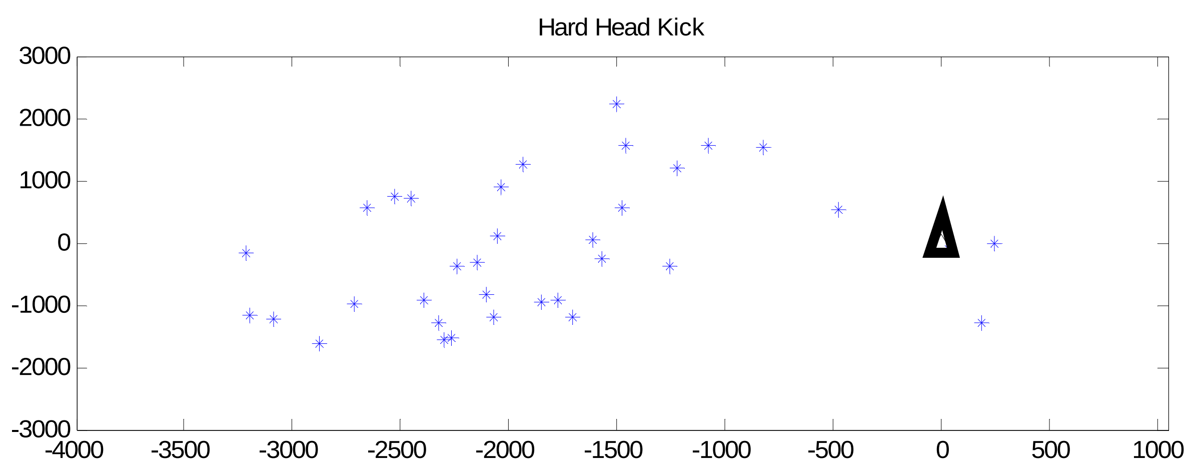
  - Impossible to track entire path of the ball
  - Calculate only the final location of the ball relative to the kick position
  - Estimate failure rate of the kick using distance threshold

Forward Kick Distance Analysis



Head Kick Distance Analysis

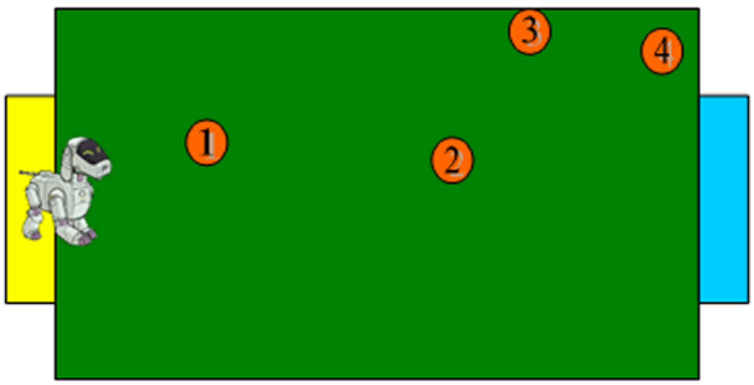




Kick Selection

- Incorporate the kick models into the selection algorithm
  - The robot knows its position on the field relative to the goal and the desired ball trajectory
  - The robot selects appropriate kick by referencing the kick model
  - If no kick fits desired criteria, robot selects closest matching kick and turns/dribbles ball to appropriate position

Kick Selection Performance



Experiment Results

Point	CMPack'02 (sec)	Modeling & Prediction (sec)
P1	**56.7**	**39.8**
P2	**42.5**	**27.2**
P3	**76.5**	**60.0**
P4	**55.0**	**52.0**
Total	**57.8**	**44.8**

Summary

- There is a variety of actuators.
- Effectiveness of motion is highly sensitive to motion parameters.
- We would like to set parameters automatically.