MEAM 520

More Velocity Kinematics

Katherine J. Kuchenbecker, Ph.D.

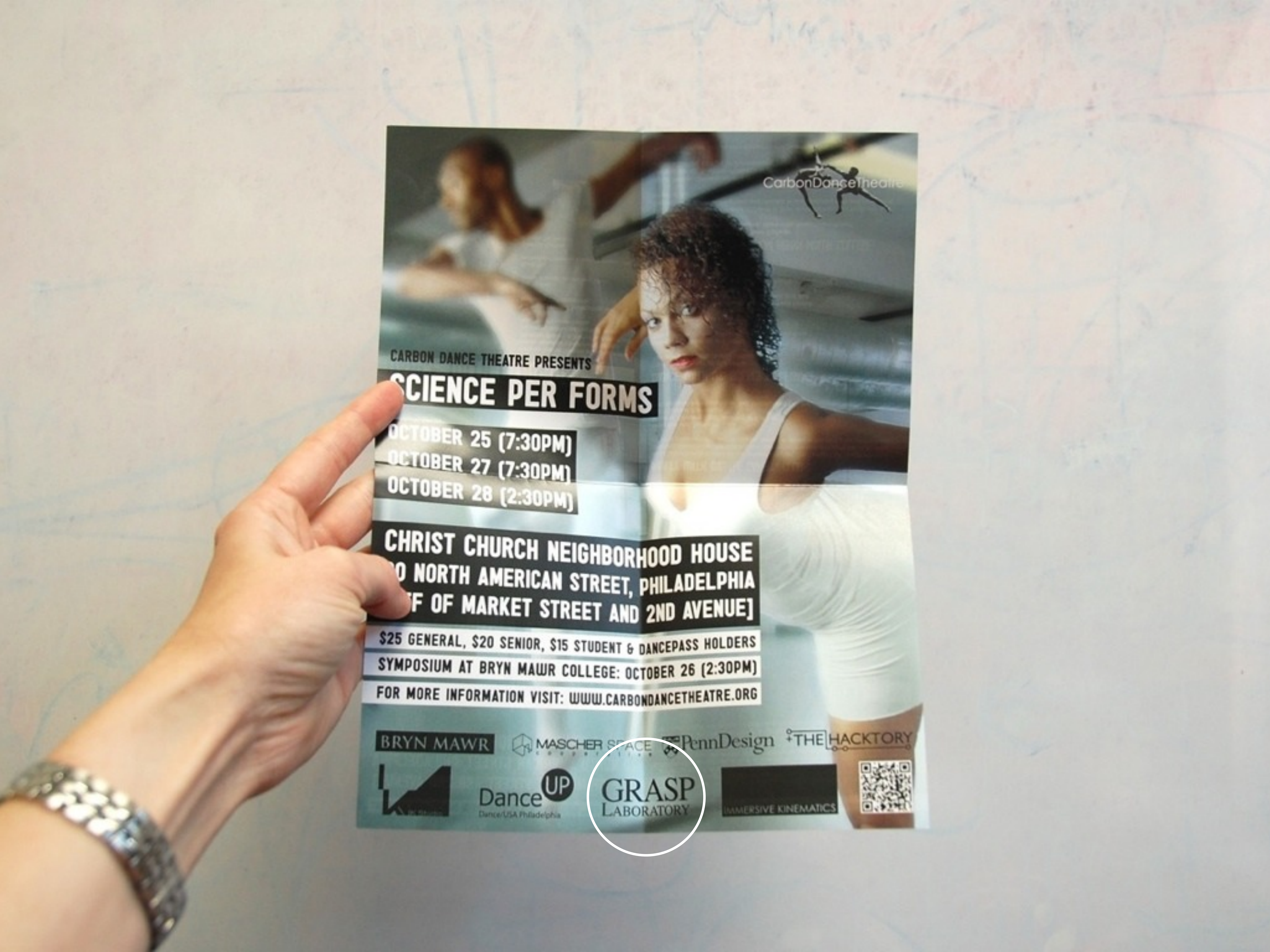
General Robotics, Automation, Sensing, and Perception Lab (GRASP)

MEAM Department, SEAS, University of Pennsylvania

Lecture 12: October 18, 2012


CARBON DANCE THEATRE PRESENTS
SCIENCE PER FORMS
OCTOBER 25 (7:30PM)
OCTOBER 27 (7:30PM)
OCTOBER 28 (2:30PM)
CHRIST CHURCH NEIGHBORHOOD HOUSE
20 NORTH AMERICAN STREET, PHILADELPHIA
[OFF OF MARKET STREET AND 2ND AVENUE]
$20 SENIOR, $15 STUDENT & DANCEPASS HOLDERS


DANCE WITH CYBORGS
Carbon Dance Theatre presents a new dance piece by co-choreographers Meredith Rainey & Marcel Williams Foster exploring digital technology and robotics through dance.
Simon Kim
School of Design (Architecture), UPenn
Mark Yim
Mechanical Engineering & Applied Mechanics, UPenn
Georgia Guthrie
Director, The Hacktory
PARTY WITH US: THE CARBON CLUB
DANCE WITH US: SCIENCE PER FORMS
Thursday October 25 at 7:30PM
Saturday October 27 at 7:30PM
Sunday October 28 at 2:30PM
Performances at Christ Church Neighborhood House
20 North American Street, off of Market and 2nd Ave
THINK WITH US: SYMPOSIUM @BRYN MAWR COLLEGE
October 27th at 2:30PM at Bryn Mawr College
Linda Caruso-Haviland, Director of Dance at Bryn Mawr College, will moderate a discussion on digital technologies and robotics in contemporary time and movement based art forms:


CarbonDanceTheatre
CARBON DANCE THEATRE PRESENTS
SCIENCE PER FORMS
OCTOBER 25 (7:30PM)
OCTOBER 27 (7:30PM)
OCTOBER 28 (2:30PM)
CHRIST CHURCH NEIGHBORHOOD HOUSE
0 NORTH AMERICAN STREET, PHILADELPHIA
F OF MARKET STREET AND 2ND AVENUE]
$25 GENERAL, $20 SENIOR, $15 STUDENT & DANCEPASS HOLDERS
SYMPOSIUM AT BRYN MAWR COLLEGE: OCTOBER 26 (2:30PM)
FOR MORE INFORMATION VISIT: WWW.CARBONDANCETHEATRE.ORG
BRYN MAWR
MASCHER SPACE
PennDesign
THE HACKTORY
DanceUP
Dance/USA Philadelphia
GRASP LABORATORY
IMMERSIVE KINEMATICS

Carbon Dance Theatre

http://www.carbondancetheatre.org/#!projects

CarbonDanceTheatre

ABOUT | ARTISTS | PROJECTS | EVENTS | MEDIA | CALENDAR | SUPPORT | CONTACT

Science per Forms

October 25, 27 & 28, 2012
Neighborhood House Theater, Christ Church
Philadelphia, PA
$25 general admission $20 seniors $15 students and dancepass holders

Showtimes:

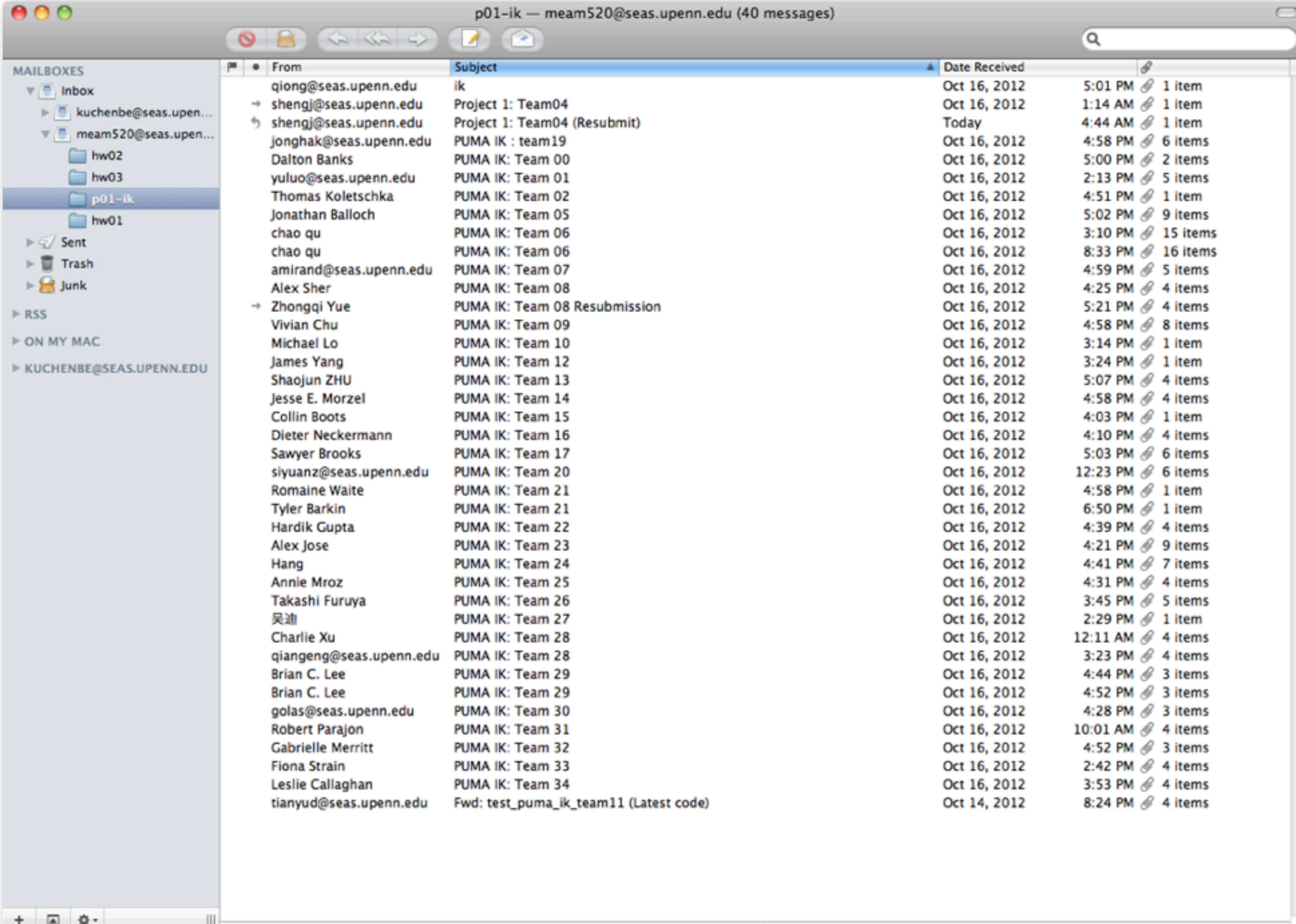
Thursday 10/25 @ 7:30PM
Saturday 10/27 @ 7:30PM
Sunday 10/28 @ 2:30PM

for tickets click here

More information on Carbon Dance Theatre's

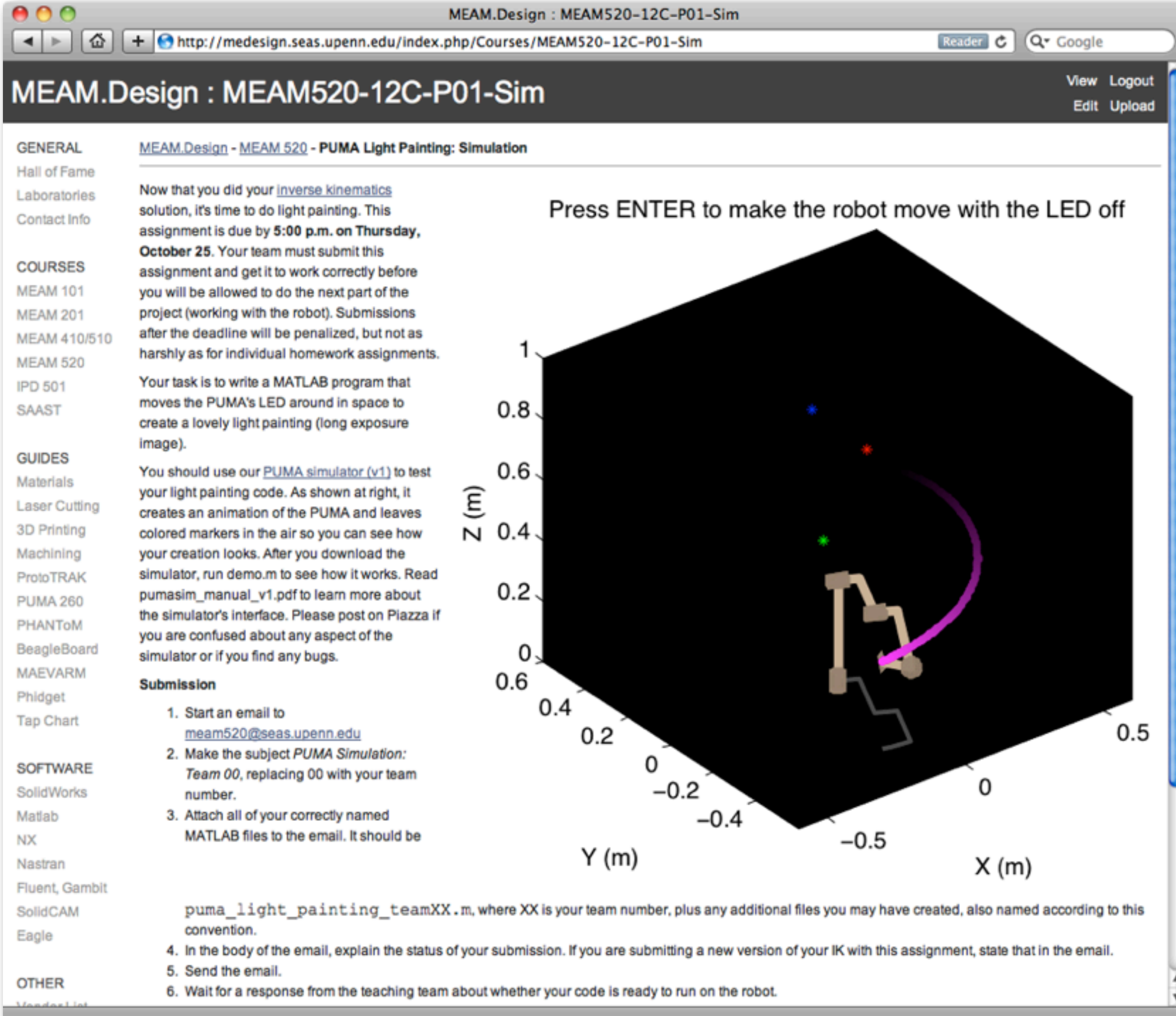
CARBON DANCE THEATRE creates projects that empower performers and entertain audiences by creating work that is rooted in classical ballet and infused with the collaborative process of theatre.

meredith@carbondancetheatre.com | 2920 Cambridge Street Philadelphia, PA 19130 | graphic design: tori lawrence | www.torilawrence.com



From	Subject	Date Received		
qiong@seas.upenn.edu	ik	Oct 16, 2012	5:01 PM	1 item
shengj@seas.upenn.edu	Project 1: Team04	Oct 16, 2012	1:14 AM	1 item
shengj@seas.upenn.edu	Project 1: Team04 (Resubmit)	Today	4:44 AM	1 item
jonghak@seas.upenn.edu	PUMA IK : team19	Oct 16, 2012	4:58 PM	6 items
Dalton Banks	PUMA IK: Team 00	Oct 16, 2012	5:00 PM	2 items
yuluo@seas.upenn.edu	PUMA IK: Team 01	Oct 16, 2012	2:13 PM	5 items
Thomas Koletschka	PUMA IK: Team 02	Oct 16, 2012	4:51 PM	1 item
Jonathan Balloch	PUMA IK: Team 05	Oct 16, 2012	5:02 PM	9 items
chao qu	PUMA IK: Team 06	Oct 16, 2012	3:10 PM	15 items
chao qu	PUMA IK: Team 06	Oct 16, 2012	8:33 PM	16 items
amirand@seas.upenn.edu	PUMA IK: Team 07	Oct 16, 2012	4:59 PM	5 items
Alex Sher	PUMA IK: Team 08	Oct 16, 2012	4:25 PM	4 items
Zhongqi Yue	PUMA IK: Team 08 Resubmission	Oct 16, 2012	5:21 PM	4 items
Vivian Chu	PUMA IK: Team 09	Oct 16, 2012	4:58 PM	8 items
Michael Lo	PUMA IK: Team 10	Oct 16, 2012	3:14 PM	1 item
James Yang	PUMA IK: Team 12	Oct 16, 2012	3:24 PM	1 item
Shaojun ZHU	PUMA IK: Team 13	Oct 16, 2012	5:07 PM	4 items
Jesse E. Morzel	PUMA IK: Team 14	Oct 16, 2012	4:58 PM	4 items
Collin Boots	PUMA IK: Team 15	Oct 16, 2012	4:03 PM	1 item
Dieter Neckermann	PUMA IK: Team 16	Oct 16, 2012	4:10 PM	4 items
Sawyer Brooks	PUMA IK: Team 17	Oct 16, 2012	5:03 PM	6 items
siyuanz@seas.upenn.edu	PUMA IK: Team 20	Oct 16, 2012	12:23 PM	6 items
Romaine Waite	PUMA IK: Team 21	Oct 16, 2012	4:58 PM	1 item
Tyler Barkin	PUMA IK: Team 21	Oct 16, 2012	6:50 PM	1 item
Hardik Gupta	PUMA IK: Team 22	Oct 16, 2012	4:39 PM	4 items
Alex Jose	PUMA IK: Team 23	Oct 16, 2012	4:21 PM	9 items
Hang	PUMA IK: Team 24	Oct 16, 2012	4:41 PM	7 items
Annie Mroz	PUMA IK: Team 25	Oct 16, 2012	4:31 PM	4 items
Takashi Furuya	PUMA IK: Team 26	Oct 16, 2012	3:45 PM	5 items
吴迪	PUMA IK: Team 27	Oct 16, 2012	2:29 PM	1 item
Charlie Xu	PUMA IK: Team 28	Oct 16, 2012	12:11 AM	4 items
qiangeng@seas.upenn.edu	PUMA IK: Team 28	Oct 16, 2012	3:23 PM	4 items
Brian C. Lee	PUMA IK: Team 29	Oct 16, 2012	4:44 PM	3 items
Brian C. Lee	PUMA IK: Team 29	Oct 16, 2012	4:52 PM	3 items
golas@seas.upenn.edu	PUMA IK: Team 30	Oct 16, 2012	4:28 PM	3 items
Robert Parajon	PUMA IK: Team 31	Oct 16, 2012	10:01 AM	4 items
Gabrielle Merritt	PUMA IK: Team 32	Oct 16, 2012	4:52 PM	3 items
Fiona Strain	PUMA IK: Team 33	Oct 16, 2012	2:42 PM	4 items
Leslie Callaghan	PUMA IK: Team 34	Oct 16, 2012	3:53 PM	4 items
tianyud@seas.upenn.edu	Fwd: test_puma_ik_team11 (Latest code)	Oct 14, 2012	8:24 PM	4 items

Project 1 : PUMA Light Painting



MEAM.Design : MEAM520-12C-P01-Sim

http://medesign.seas.upenn.edu/index.php/Courses/MEAM520-12C-P01-Sim

MEAM.Design : MEAM520-12C-P01-Sim

View Logout
Edit Upload

GENERAL
Hall of Fame
Laboratories
Contact Info

COURSES
MEAM 101
MEAM 201
MEAM 410/510
MEAM 520
IPD 501
SAAST

GUIDES
Materials
Laser Cutting
3D Printing
Machining
ProtoTRAK
PUMA 260
PHANToM
BeagleBoard
MAEVARM
Phidget
Tap Chart

SOFTWARE
SolidWorks
Matlab
NX
Nastran
Fluent, Gambit
SolidCAM
Eagle

OTHER

MEAM.Design - MEAM 520 - **PUMA Light Painting: Simulation**

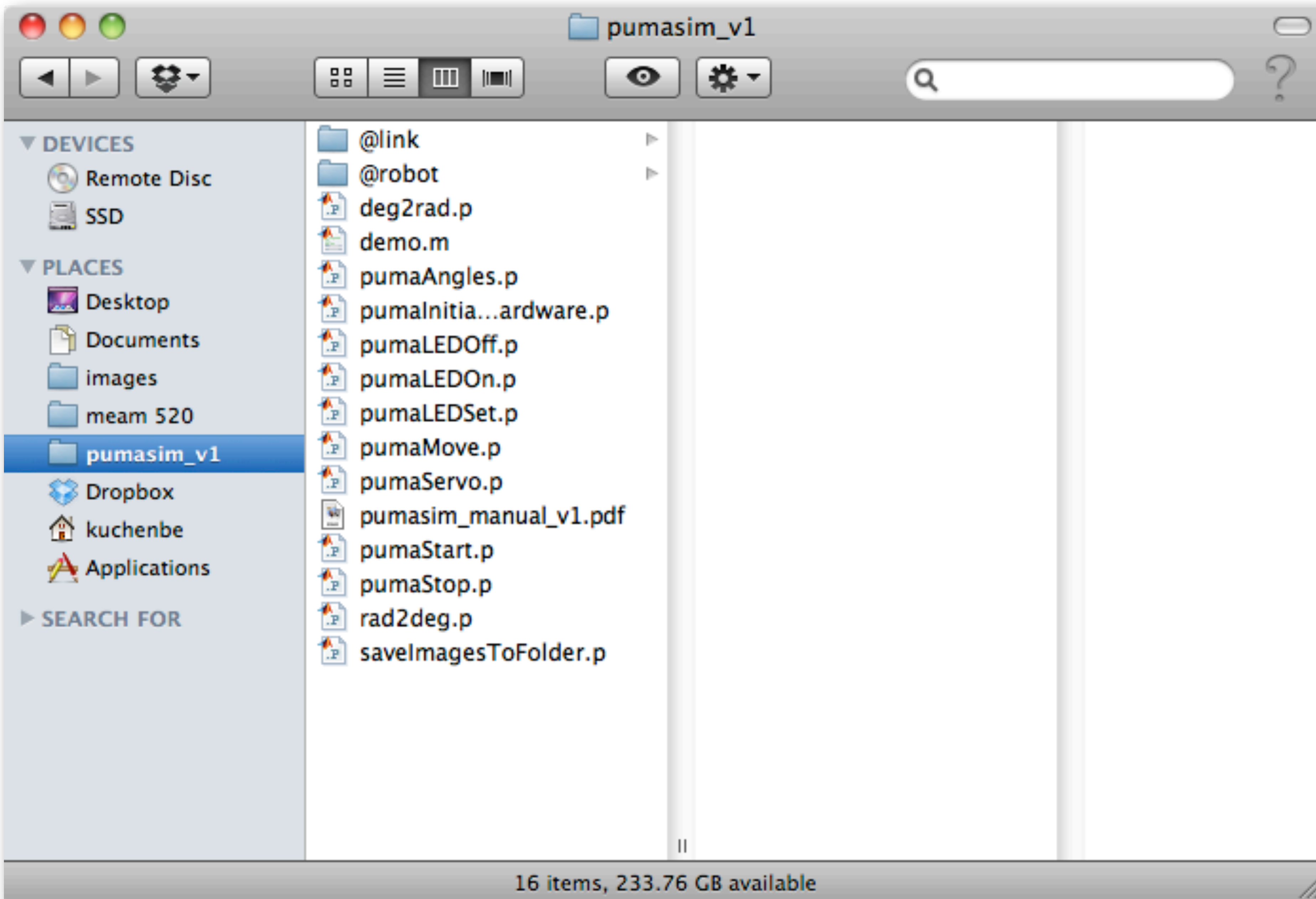
Now that you did your inverse kinematics solution, it's time to do light painting. This assignment is due by **5:00 p.m. on Thursday, October 25**. Your team must submit this assignment and get it to work correctly before you will be allowed to do the next part of the project (working with the robot). Submissions after the deadline will be penalized, but not as harshly as for individual homework assignments.

Your task is to write a MATLAB program that moves the PUMA's LED around in space to create a lovely light painting (long exposure image).

You should use our PUMA simulator (v1) to test your light painting code. As shown at right, it creates an animation of the PUMA and leaves colored markers in the air so you can see how your creation looks. After you download the simulator, run demo.m to see how it works. Read pumasim_manual_v1.pdf to learn more about the simulator's interface. Please post on Piazza if you are confused about any aspect of the simulator or if you find any bugs.

Submission

1. Start an email to meam520@seas.upenn.edu
2. Make the subject *PUMA Simulation: Team 00*, replacing 00 with your team number.
3. Attach all of your correctly named MATLAB files to the email. It should be `puma_light_painting_teamXX.m`, where XX is your team number, plus any additional files you may have created, also named according to this convention.
4. In the body of the email, explain the status of your submission. If you are submitting a new version of your IK with this assignment, state that in the email.
5. Send the email.
6. Wait for a response from the teaching team about whether your code is ready to run on the robot.


pumasim_v1
DEVICES
Remote Disc
SSD
PLACES
Desktop
Documents
images
meam 520
pumasim_v1
Dropbox
kuchenbe
Applications
SEARCH FOR
@link
@robot
deg2rad.p
demo.m
pumaAngles.p
pumaInitia...ardware.p
pumaLEDOff.p
pumaLEDOn.p
pumaLEDSet.p
pumaMove.p
pumaServo.p
pumasim_manual_v1.pdf
pumaStart.p
pumaStop.p
rad2deg.p
saveImagesToFolder.p
16 items, 233.76 GB available

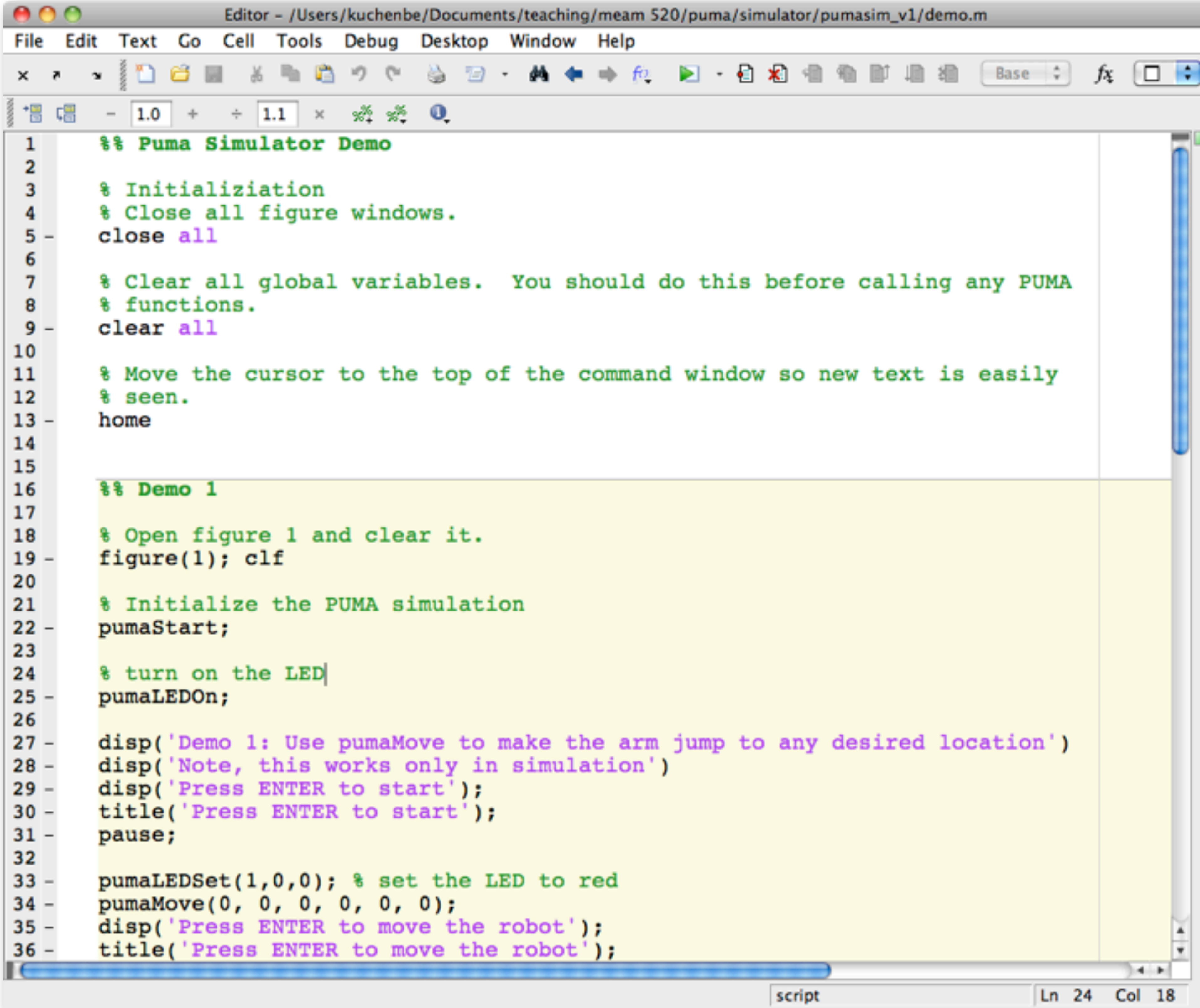
RUNNING THE PUMA

The basic work flow when using the PUMA 260 arm is

1. Make sure the Emergency Stop (E-stop) button is engaged (pressed down).
2. Call `pumaStart('Hardware', 'on', 'Delay', 10)`, where the number following delay is the minimum allowable time, in milliseconds, between calls to `pumaServo`. This may be set to any value above 0.5ms. This will display a warning that the PUMA will return to the home position. Ensure that the workspace is clear and manually move the PUMA closer to the home position if you think it may hit the table or an object.
3. Type 'y' or 'yes' then hit Enter to continue.
4. Release the E-stop by pulling up on the button, at which time the PUMA will return to the home position.
5. In a separate MATLAB process, call `startFrameBuffer` to begin capturing video from the webcam.
6. Make a light painting, using `pumaServo` to command the robot to move. Remember to call `pumaLEDOn` to enable the LED and use `pumaLEDSet` to select the color.
7. Call `stopFrameBuffer` to finish capturing video.
8. Return the PUMA to the home position, if possible.
9. Call `pumaStop` to disable the controller.
10. Engage the E-stop by pressing the button down.
11. Use `makeVideoAndImage` to create long-exposure picture and video. Optionally, you may wish to save the image files to a different location using `saveImagesToFolder` before starting a new video.

There are several other things to keep in mind:

- Do ***NOT*** use the clear all command once the PUMA has been initialized before calling pumaStop, otherwise MATLAB will crash.
- Test the video capture before running your whole light painting.


Editor - /Users/kuchenbe/Documents/teaching/meam 520/puma/simulator/pumasim_v1/demo.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Base
1.0
1.1
%% Puma Simulator Demo
% Initializiation
% Close all figure windows.
close all
% Clear all global variables. You should do this before calling any PUMA
% functions.
clear all
% Move the cursor to the top of the command window so new text is easily
% seen.
home
%% Demo 1
% Open figure 1 and clear it.
figure(1); clf
% Initialize the PUMA simulation
pumaStart;
% turn on the LED
pumaLEDOn;
disp('Demo 1: Use pumaMove to make the arm jump to any desired location')
disp('Note, this works only in simulation')
disp('Press ENTER to start');
title('Press ENTER to start');
pause;
pumaLEDSet(1,0,0); % set the LED to red
pumaMove(0, 0, 0, 0, 0, 0);
disp('Press ENTER to move the robot');
title('Press ENTER to move the robot');
script
Ln 24 Col 18

Confirmed Midterm Date

Thursday, November 8, in class

~ email KJK if you have a severe conflict ~

Velocity Kinematics

Slides created by
Jonathan Fiene

How do the velocities of the joints affect
the linear and angular velocity of the end-effector?

These quantities are related by the **Jacobian**,
a matrix that generalizes the notion of an ordinary
derivative of a scalar function.

Jacobians are useful for planning and executing smooth trajectories, determining singular configurations, executing coordinated anthropomorphic motion, deriving dynamic equations of motion, and transforming forces and torques from the end-effector to the manipulator joints.

explore how **changes** in joint values affect the end-effector movement

could have **N joints**, but only **six** end-effector velocity terms (xyzpts)

The **Jacobian** matrix lets us calculate how joint velocities translate into end-effector velocities (depends on configuration)

look at it in two parts - position and orientation

$$\boxed{v_n^0 = J_v \dot{q}} \qquad \omega_n^0 = J_\omega \dot{q}$$

How do we calculate the position Jacobian?

$$\dot{\mathbf{p}} = \mathbf{J}_p(q)\ \dot{\mathbf{q}}$$

endpoint velocity

Jacobian matrix

joint velocity

$$\mathbf{J}_p = \begin{bmatrix} \frac{\delta x}{\delta q_1} & \frac{\delta x}{\delta q_2} & \cdots & \frac{\delta x}{\delta q_n} \\ \frac{\delta y}{\delta q_1} & \frac{\delta y}{\delta q_2} & \cdots & \frac{\delta y}{\delta q_n} \\ \frac{\delta z}{\delta q_1} & \frac{\delta z}{\delta q_2} & \cdots & \frac{\delta z}{\delta q_n} \end{bmatrix}$$

Prismatic

$$J_{v_i} = z_{i-1}$$

Revolute

$$J_{v_i} = z_{i-1} \times (o_n - o_{i-1})$$

$$v_n^0 = J_v \dot{q}$$

What joint velocities should I choose to cause a desired end-effector velocity?
(inverse velocity kinematics)

$$\dot{q} = J_v^{-1} v_n^0$$

This works only when the Jacobian is square and invertible (non-singular).

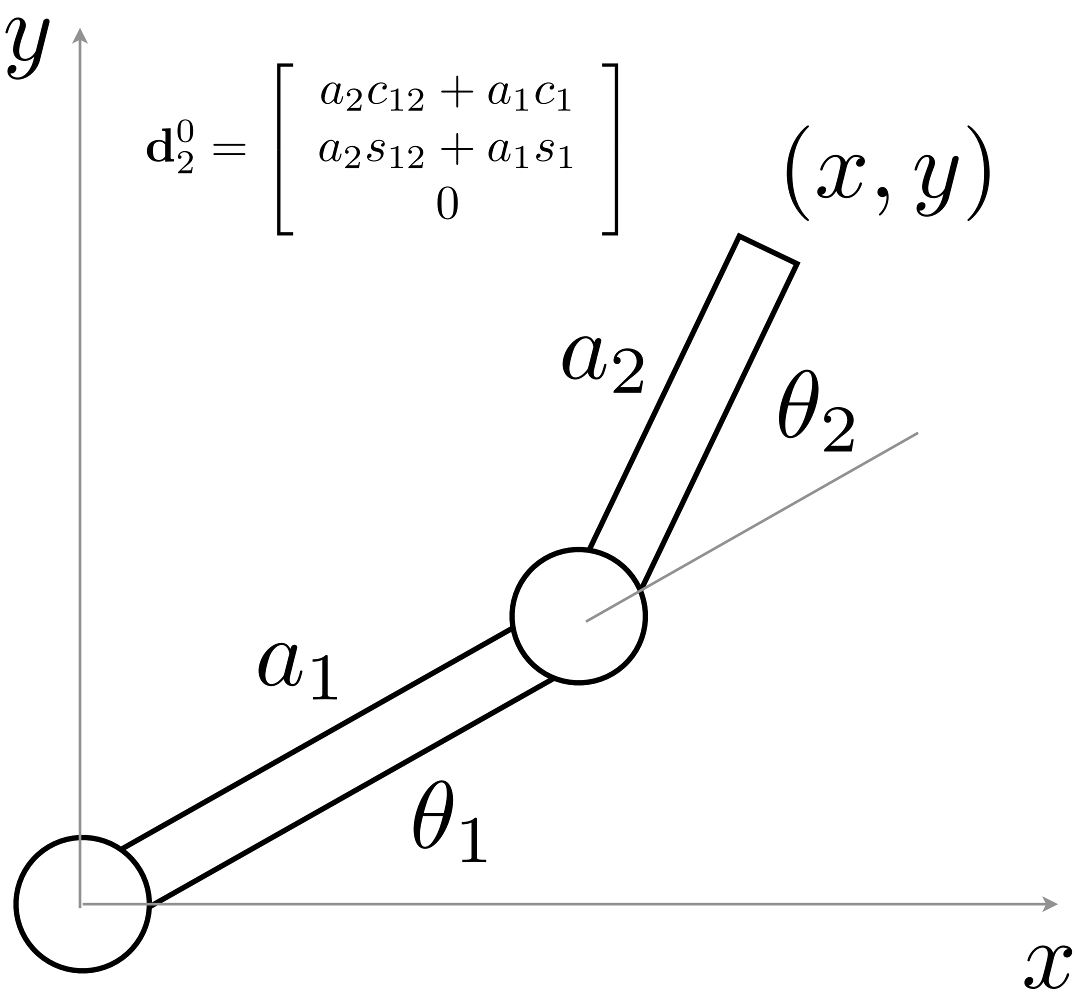
SHV 4.11 explains what to do when the Jacobian is not square:
rank test (v is in range of J)
use J^+ (right pseudoinverse of J)
when the robot has extra joints, there are many solutions

Singularities are points in the configuration space where infinitesimal motion in a certain direction is not possible and the manipulator loses one or more degrees of freedom

Mathematically, singularities exist at any point in the workspace where the Jacobian matrix loses rank.

a matrix is singular if and only if its determinant is zero:

$$\det(\mathbf{J}) = 0$$



$$\mathbf{J} = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} \\ a_1c_1 + a_2c_{12} & a_2c_{12} \end{bmatrix}$$

$$\det(\mathbf{J}) = ?$$

$$\det(\mathbf{J}) = a_1\, a_2\, (c_1s_{12} - s_1c_{12})$$

When does $\det(\mathbf{J}) = 0$?

$\det(\mathbf{J}) = 0$ when $\theta_2 = 0$

Is that the only time?

No . . . $\det(\mathbf{J}) = 0$ when $\theta_2 = \ldots, -2\pi, -\pi, 0, \pi, 2\pi, \ldots$

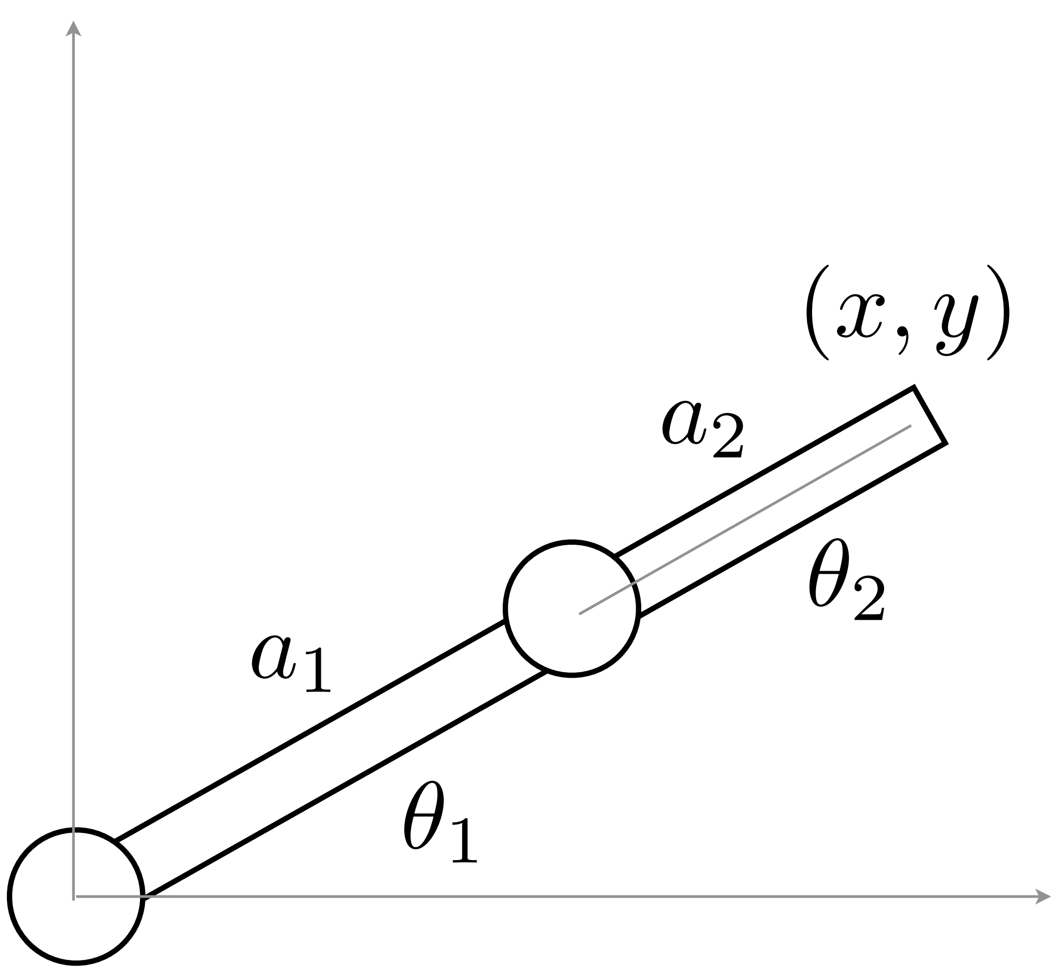
Any other times? $\det(\mathbf{J}) = 0$ when $a_1 = 0$ or $a_2 = 0$

For $\theta_2 = 0$

The Jacobian collapses to have linearly dependent rows

$$\mathbf{J}_{\theta_2=0} = \begin{bmatrix} -a_1 s_1 - a_2 s_1 & -a_2 s_1 \\ a_1 c_1 + a_2 c_1 & a_2 c_1 \end{bmatrix}$$

This means that actuating either joint causes motion in the same direction



Questions ?

explore how **changes** in joint values affect the end-effector movement

could have **N joints**, but only **six** end-effector velocity terms (xyzpts)

The **Jacobian** matrix lets us calculate how joint velocities translate into end-effector velocities (depends on configuration)

look at it in two parts - position and orientation

$$v_n^0 = J_v \dot{q}$$

$$\omega_n^0 = J_\omega \dot{q}$$

How do we calculate the orientation Jacobian?

And now, angular velocity

$$\omega = \mathbf{J}_\omega(q)\ \dot{\mathbf{q}}$$

final frame angular velocity

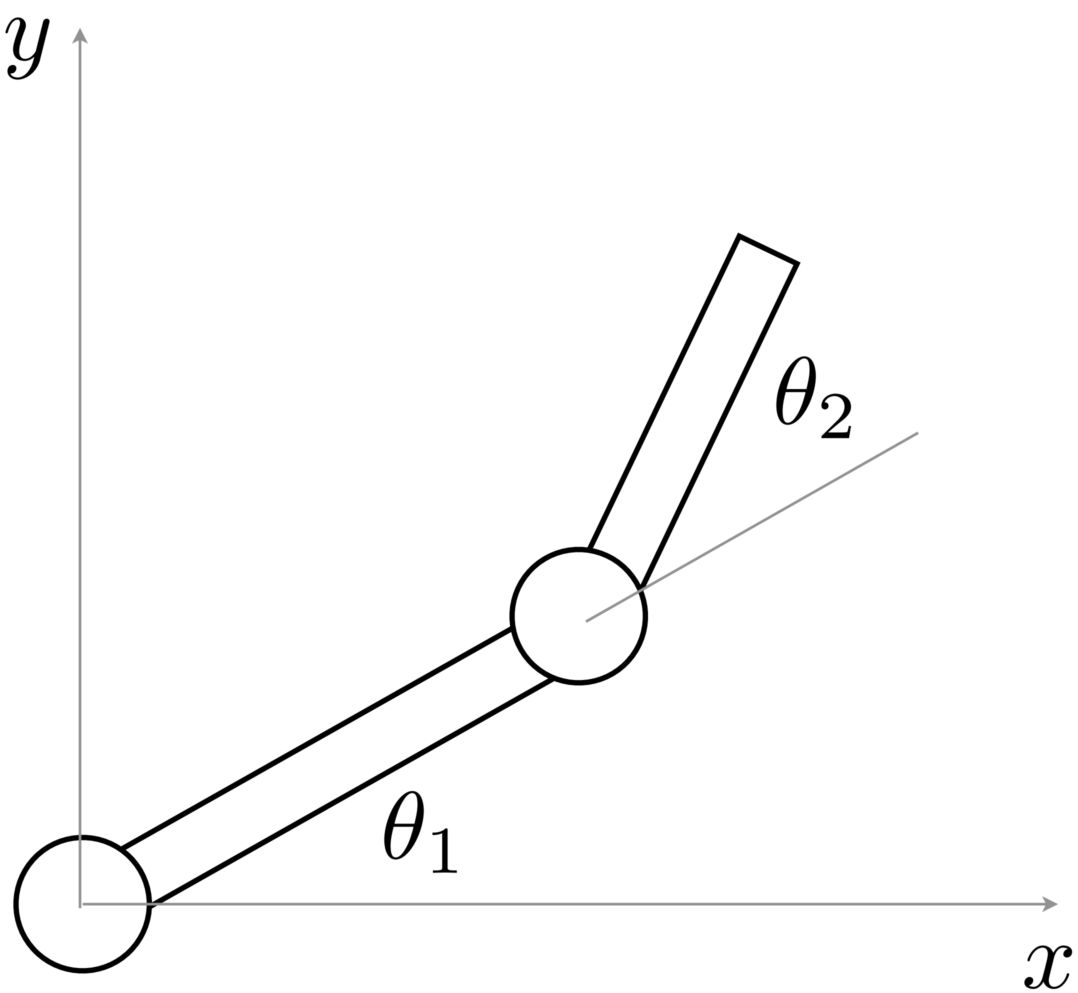
joint velocities

$$\omega_{i,j}^{k}$$

this is the angular velocity of frame **j**
with respect to frame **i**,
expressed in frame **k**

SHV 4.1 gives a good explanation of angular velocity for fixed-axis rotation. SHV 4.2-4.5 go into greater detail.

The Angular Velocity of Connected Rigid Bodies



$$\omega_{0,1}^{0} = 0\,\hat{x}_0 + 0\,\hat{y}_0 + \dot{\theta}_1\,\hat{z}_0$$

$$\omega_{1,2}^{1} = 0\,\hat{x}_1 + 0\,\hat{y}_1 + \dot{\theta}_2\,\hat{z}_1$$

$$\omega_{1,2}^{0} = \mathbf{R}_1^0\,\omega_{1,2}^{1}$$

$$\begin{aligned}\omega_{0,2}^{0} &= \omega_{0,1}^{0} + \mathbf{R}_1^0\,\omega_{1,2}^{1} \\ &= 0\,\hat{x}_0 + 0\,\hat{y}_0 + (\dot{\theta}_1 + \dot{\theta}_2)\,\hat{z}_0\end{aligned}$$

$$\omega_{0,n}^{0} = \sum_{i=1}^{n} \mathbf{R}_{i-1}^{0}\,\omega_{i-1,i}^{i-1}$$

$$\omega_{0,n}^{0} = \sum_{i=1}^{n} (\mathbf{R}_{i-1}^{0}\hat{\mathbf{z}})\,\dot{\theta}_i$$

note: this holds for revolute joints only (by definition, a prismatic joint cannot create angular velocity)

And now, for that Jacobian!

$$\omega_{0,n}^{0} = \sum_{i=1}^{n} \rho_i (\mathbf{R}_{i-1}^{0} \hat{z})\, \dot{\theta}_i \qquad \rho_i = \begin{array}{l} \text{0 for prismatic} \\ \text{1 for revolute} \end{array}$$

$$\omega_{0,n}^{0} = \begin{bmatrix} \rho_1 \hat{\mathbf{z}} & \rho_2 \mathbf{R}_1^0 \hat{\mathbf{z}} & \rho_2 \mathbf{R}_2^0 \hat{\mathbf{z}} & \dots & \rho_n \mathbf{R}_{n-1}^0 \hat{\mathbf{z}} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_n \end{bmatrix}$$

$$\omega = \mathbf{J}_\omega(q)\, \dot{\mathbf{q}}$$

$$\mathbf{J} = \left[\begin{array}{c} \mathbf{J}_p \\ \mathbf{J}_\omega \end{array} \right]$$

(3 x n) cartesian velocity Jacobian

(3 x n) angular velocity Jacobian

(6 x n) Jacobian
a.k.a. manipulator Jacobian
a.k.a. geometric Jacobian

The Jacobian is easily constructed from the manipulator's forward kinematics.

What do you need from the forward kinematics?

4.6.3 Combining the Linear and Angular Velocity Jacobians

As we have seen in the preceding section, the upper half of the Jacobian J_v is given as

$$J_v = [J_{v_1} \cdots J_{v_n}] \tag{4.56}$$

in which the i^{th} column J_{v_i} is

$$J_{v_i} = \begin{cases} z_{i-1} \times (o_n - o_{i-1}) & \text{for revolute joint } i \\ z_{i-1} & \text{for prismatic joint } i \end{cases} \tag{4.57}$$

The lower half of the Jacobian is given as

$$J_\omega = [J_{\omega_1} \cdots J_{\omega_n}] \tag{4.58}$$

in which the i^{th} column J_{ω_i} is

$$J_{\omega_i} = \begin{cases} z_{i-1} & \text{for revolute joint } i \\ 0 & \text{for prismatic joint } i \end{cases} \tag{4.59}$$

Singularities are points in the configuration space where infinitesimal motion in a certain direction is not possible and the manipulator loses one or more degrees of freedom

Mathematically, singularities exist at any point in the workspace where the Jacobian matrix loses rank.

a matrix is singular if and only if its determinant is zero:

$$\det(\mathbf{J}) = 0$$

$$\xi = J(q)\dot{q}$$

(6 x 1) body velocity

(6 x n) Jacobian

(n x 1) joint velocities

$$\begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \dot{q}$$

For a 6-DOF manipulator with a spherical wrist, we can decouple the determination of singular configurations into two simpler problems.

$$J = [J_{\text{arm}} \mid J_{\text{wrist}}]$$

(the book calls this $J = [J_P \mid J_O]$)

$$J = [J_{\text{arm}} \mid J_{\text{wrist}}] = \left[\frac{J_{11}}{J_{21}} \middle| \frac{J_{12}}{J_{22}}\right]$$

$$J_{\text{wrist}} = \begin{bmatrix} z_3 \times (o_6 - o_3) & z_4 \times (o_6 - o_4) & z_5 \times (o_6 - o_5) \\ z_3 & z_4 & z_5 \end{bmatrix}$$

if we choose $o_4 = o_5 = o_6$

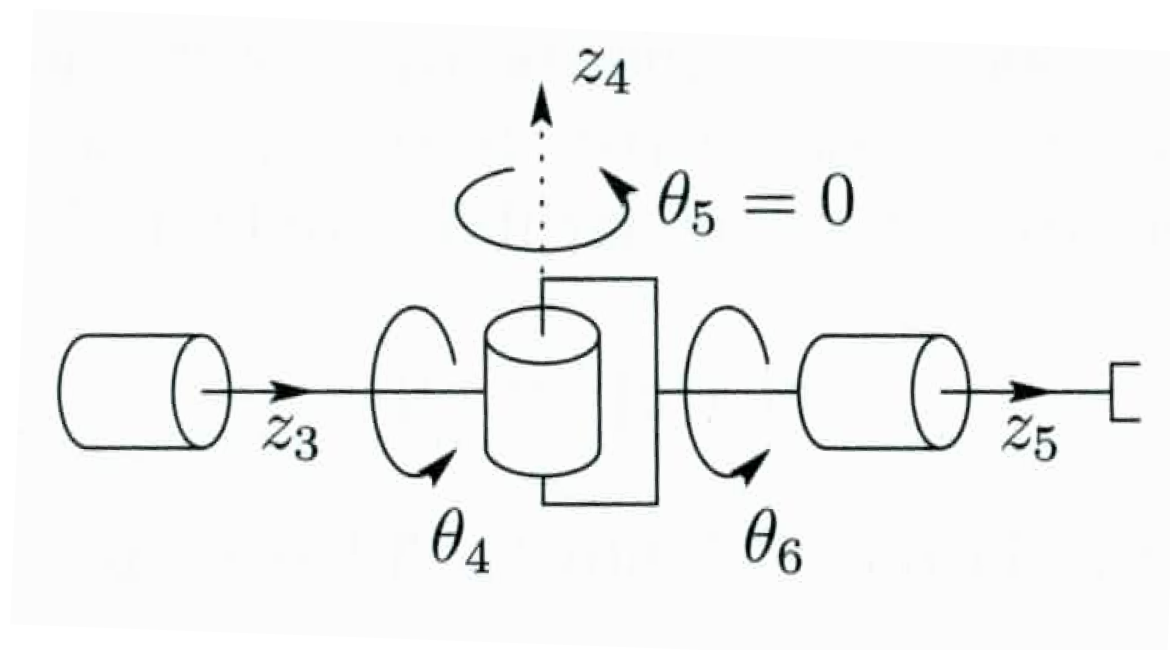
$$J_{\text{wrist}} = \begin{bmatrix} 0 & 0 & 0 \\ z_3 & z_4 & z_5 \end{bmatrix}$$

$$J = \left[\frac{J_{11}}{J_{21}} \middle| \frac{0}{J_{22}}\right] \qquad \det(J) = \det(J_{11}) \det(J_{22})$$

$$\det(J) = \det(J_{11})\det(J_{22})$$

$$J_{22} = \left[\begin{array}{ccc} z_3 & z_4 & z_5 \end{array}\right]$$

Singular when any two wrist axes align



$z_3 \perp z_4$ $\qquad$ $z_4 \perp z_5$

$$z_3 \text{ can become } || \; z_5$$

$$\theta_5 = 0, \pi \text{ are singular configurations}$$

For a specific configuration, the Jacobian scales the input (joint velocities) to the output (body velocity)

$$\xi = J(q)\dot{q}$$

If you put in a joint velocity vector with unit norm, you can calculate in which direction and how fast the robot will translate and rotate.

If the Jacobian is full rank, you can calculate the manipulability ellipsoid.

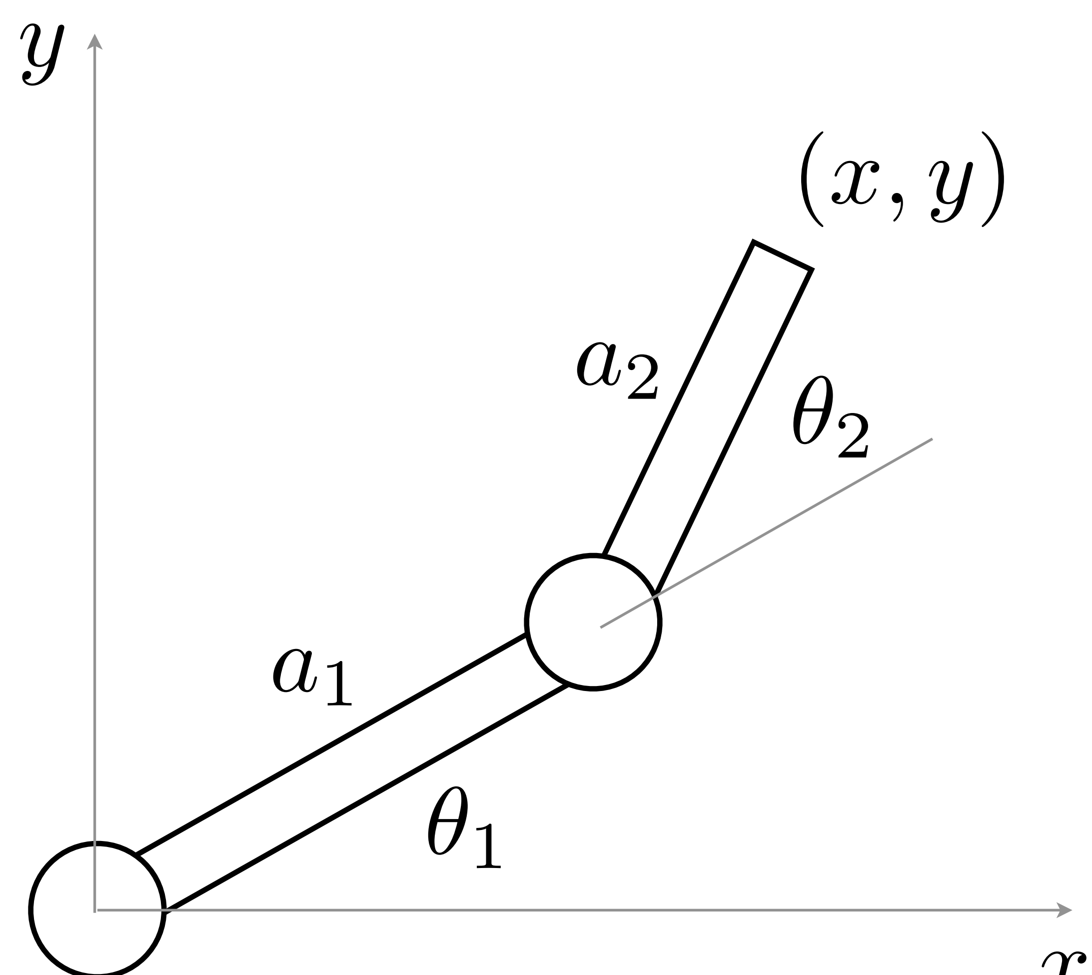
$$\text{choose } \dot{q} = J^{+}\xi$$

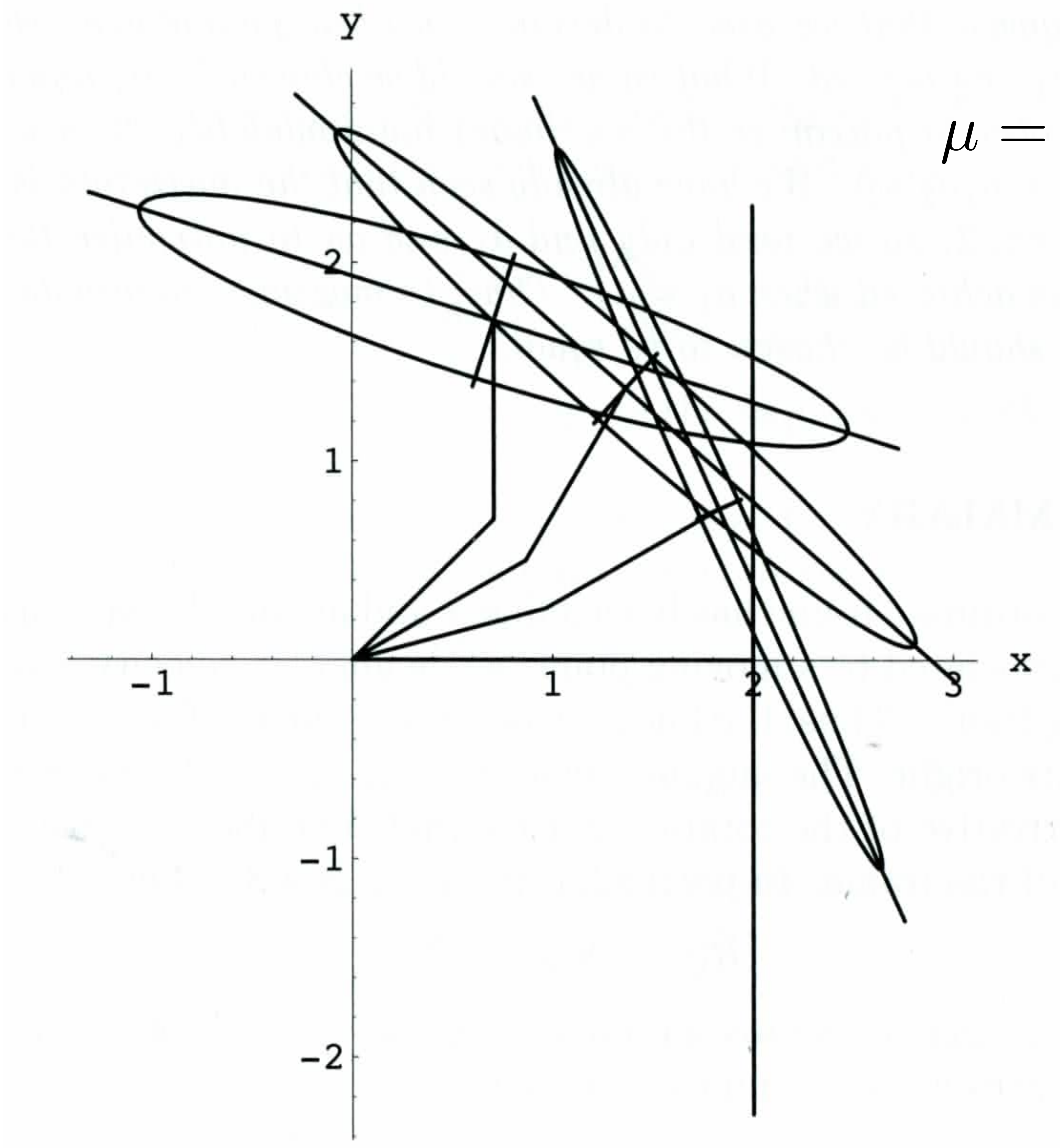
$$||\dot{q}||^2 = \xi^T (JJ^T)^{-1}\xi$$

If not redundant, manipulability

$$\mu = |\det(J)|$$

What does the manipulability ellipsoid look like for the planar RR robot?





$$\mu = |\det(J)| = a_1 a_2 |\sin(\theta_2)|$$

Can be used to tell you where to perform certain tasks.

Also useful for deciding how to design a manipulator.

Soon I will release Homework 4,
an individual assignment on Jacobians

Not sure when it will be due...

Homework 2 and 3 graded