CS 117: Project in Computer Vision

Spring 2011

Lecture 1: Introduction
Professor Deva Ramanan

Department of Computer Science
dramanan@ics.uci.edu
Outline

• Overview of this course:
  ○ Goals of this class
  ○ Assignments, Labs, and MATLAB
  ○ Grading policies

• Today’s lecture:
  ○ Overview of course
  ○ Visual interfaces
  ○ Introduction to MATLAB

• Assignment 1
Introductions

• Professor Deva Ramanan
  http://www.ics.uci.edu/~dramanan/

Bio: I graduated with a PhD from UC Berkeley, where I was advised by David Forsyth. Previously, I was a research professor at TTI-Chicago. I was fortunate enough to be a visiting researcher at the Visual Geometry Group at Oxford, the Robotics Institute at CMU, and the Interactive Visual Media Group at Microsoft Research. I work primarily in computer vision, but am also interested in machine learning and computer graphics.
Overview of Course

• See class Web page

• Overview
  ○ Goals of this class
  ○ Schedule
  ○ Assignments, Labs, and MATLAB
  ○ Grading policies
Enrollment for the class

• Yes, we will have this class (I hope!)

• I’m allowing students to replace CS116 with CS112, CS171, or CS 178

• I’ll hand out pre-req waivers that you’ll have to fill out; bring them in on Thursday and I will sign
Organizational Issues

• Questions
  ○ Email class distribution list
    ▪ cs117-S11@classes.uci.edu
  ○ Everyone benefits from questions/answers!
    ▪ I will e-mail responses within a day
  ○ Office hours: 12:30 to 2 on Wed

• If you can’t make office hours, request an appointment by email
Projects in Computer Vision: Visual Interfaces

human-computer interaction

gesture recognition

Xbox Kinect

Nintendo Wii

We’ll study the technical algorithms behind the above
The future....

http://www.bruceongames.com/2007/10/02/the-future-it-is-all-a-gesture/
Kinect holds the Guinness World Record of being the "fastest selling consumer electronics device". It sold an average of 133,333 units per day with a total of 8 million units in its first 60 days.[18][19][20] 10 million units of the Kinect sensor have been shipped as of March 9, 2011.[1]

Wikipedia Article
Real-Time Hand-Tracking with a Color Glove

Robert Y. Wang¹ Jovan Popovic¹,²,³

¹Computer Science and Artificial Intelligence Lab
Massachusetts Institute of Technology
²Advanced Technology Labs, Adobe Systems Incorporated
³University of Washington
Some demos...

http://www.youtube.com/watch?v=Jd3-eiid-Uw&feature=player_embedded
We will build up pieces for...
This course

- We will learn the “science” behind tracking and gesture classification, and focus less on the (important!) human interface issues
- Fun application of probability and linear algebra
- MATLAB will become your friend
  - pre-loaded image libraries
  - powerful tool for prototyping
- You will learn/implement
  - Color detection, Bayesian Probability Models, Multivariate Gaussians, 2D Tracking, Dynamic Programming, Nearest Neighbor Classification, Optical-flow
More motivation

• Our vision lab has lots of projects involving tracking... looking for (additional) undergraduate involvement

Biomechanical analysis for sports
More motivation (open projects)

Tracking people in Bren Hall
More motivation (ongoing projects)

Activity analysis
Schedule


Project 1: Already up

Mainly an introduction to MATLAB

A large part of your success in this course depends on your ability to pick up MATLAB in the first assignment!
Introduction to MATLAB
MATLAB

• Introduction to MATLAB
  ○ Expressions
  ○ Data types
  ○ Matrices/arrays
  ○ Functions
  ○ Arithmetic operations
  ○ MATLAB environment
  ○ MATLAB graphics
  ○ MATLAB images

• Note: In class we will cover only a subset of the capabilities of MATLAB. I expect you to learn more about MATLAB outside class, e.g., via tutorials in Assignment 1, etc
  ○ Success in your project assignment will depend on how well you learn MATLAB in the first few weeks of class
Introduction to MATLAB

• MATLAB
  ○ technical computing environment for numeric computation and visualization

• Why it is useful for prototyping vision projects:
  ○ large toolbox of numeric/image library functions
  ○ very useful for displaying, visualizing data
  ○ high-level: focus on algorithm structure, not on low-level details
  ○ allows quick prototype development of algorithms

• Some other aspects of MATLAB
  ○ MATLAB is an interpreter -> not as fast as compiled code
    ▪ Typically quite fast for an interpreted language
    ▪ Often used early in development -> can then convert to C (e.g.,) for speed
  ○ Can be linked to C/C++, JAVA, SQL, etc
  ○ Commercial product, but widely used in industry and academia
    ▪ Many algorithms and toolboxes freely available
Matlab links to Kinect

http://acberg.com/kinect/
MATLAB: Basic Concepts

• Variables
  ○ no explicit type declarations or dimension statements
e.g., a = 7.2 or b = [ 1 2 3 4];
  ○ case-sensitive, underscores allowed, etc
  ○ variable created and RHS assigned to memory location

• Numbers
  ○ Default is real-valued, double precision
    ▪ 73.3782, 2.563e-15, 3.00, etc
    ▪ type double (real-valued), 8 bytes
    ▪ type float, 4 bytes
  ○ Also type uint = unsigned integer
    ▪ Uint8: 8 bits, 0 to 255
    ▪ Uint16: 16 bits, 0 to 65536

Common pitfall when dealing with images!
MATLAB: Basic Concepts

- Operators
  - usual set of arithmetic operators
  - + - * / ^
  - a few special ones, e.g., .* ./ \n  - usual set of logical operators: & | ~

- Functions
  - large library of built-in functions (e.g. sqrt, abs, sin)
    - e.g., >> x = sqrt(a);
  - can define your own functions using .m files

- Help reference
  - >> help function gives great reference on every function
  - e.g., >> help sqrt

“SQRT(X) is the square root of the elements of X. Complex results are produced if X is not positive.”
I prefer the non-fancy command-line interface
Matrices and Arrays

- Most common data type is a matrix (2d array):
  - a matrix is of dimension $n$ rows by $d$ columns
  - a vector has dimension $n \times 1$, or $1 \times n$

```
» a = [1 2 3 4]
 a =
  1 2 3 4

» size(a)
ans =
  1 4

» a'
ans =
  1
  2
  3
  4

» size(a')
ans =
  4 1
```

- Define a vector “a”
- Square brackets denote the vector or matrix
- Matlab tells us the size of “a”
- This is the transpose of “a”, it has the “reverse” dimensions of “a”
Images

>> im = imread('arnold.jpg');

>> size(a)
ans =
450 313 3

>> imshow(im); [Requires JVM]

>> image(im); [command line]

Both imshow(im) and image(im) works for color images (3D arrays) and grayscale images (2D matrices)
Matrix subscripts

c = 
[ 1 2 3 4 ;
 5 6 7 8 ];

» c(1,1) 
ans = 
1

» c(2,3) 
ans = 
7

» c(1,:) 
ans = 
1 2 3 4

» c(:,2) 
ans = 
2 6

» c(:,2:4) 
ans = 
2 3 4
 6 7 8

We can access any particular component of the matrix “c”

We can use the “:” (colon) to indicate all rows or all columns, e.g.,
c(1,:) means the first row, all columns
 c(:,2) means all rows, the 2nd column
 c(:,2:4) means all rows, 2nd to 4th columns

How does one
1) Access/display the “red” values of a color image?
2) Access/display the topleft 10x10 “sub-image”?

Thursday, March 31, 2011
Logical Conditions on a Matrix

```matlab
» d = c(1,:)
d = 1 2 5 10

» i = d>2
i = 0 0 1 1

We can define a vector “i” which is a logical function of d

» d(i)
ans = 5 10

We can now use “i” to find the elements of d>2

index = find(d>2)
ans = 3 4

The function `find` can be used to find the indices of non-zero values in an array
```
Functions of Matrices

The `sum` function calculates the sum of a matrix, summing down each column (default) or row columnwise.

How does one compute the average color in an image?
Arithmetic Operations on Matrices

Define two “1 x 3” vectors, “a” and “b”

We can perform the usual arithmetic operations on any matrices “a” and “b” provided they are the same size.
Arithmetic Operations on Matrices

```matlab
» a = [1 2 3]; b = [4 5 6]
» a.*b
ans = 4 10 18
```

There are 2 different ways to multiply matrices:

1. “element by element” multiplication (for vectors this is also called “inner product”)

```matlab
» a*b'
ans = 32
```

2. “matrix multiplication”: multiply a n x m matrix by a m x p matrix to give a n x p matrix: note here how we multiply a 1 x 3 matrix by a 3 x 1 matrix to get a 1 x 1 (scalar) solution

How does one brighten an image (e.g., scale all pixels by 2)?

How should one compute the average of a set of images?
Concatenation of Matrices

```matlab
» b = [5 6 7 8];
» c = [a ; b]
c =
1 2 3 4
5 6 7 8
» size(c)
ans =
2 4
» c'
ans =
1 5
2 6
3 7
4 8
» d = [a b]
d =
1 2 3 4 5 6 7 8
```

Define a new vector “b” . Note that using “;” at the end of the line suppresses printing the result.

Define matrix “c” as by concatenating the vectors “a” and “b”. Note the use of “;” within […..] to denote the end of a row.

Here we define matrix “d” as by putting the “a” and “b” side by side: note the different result we get now (different from “c”).
Special Matrices

```matlab
» ones(2,3)
ans =
  1 1 1
  1 1 1

» zeros(1,4)
ans =
  0 0 0 0

» rand(3,3)
ans =
  0.2176 0.4909 0.8985
  0.4054 0.1294 0.5943
  0.5699 0.5909 0.3020
```
Interactive MATLAB Environment

**MATLAB workspace**: area of memory where all variables defined during a working session are stored

```matlab
whos
Name Size Bytes Class

  a 1x4 32 double array
ans 3x3 72 double array
b 1x3 24 double array
c 2x4 64 double array
d 2x2 32 double array
e 1x10 80 double array
i 1x4 32 double array (logical)
```

Grand total is 42 elements using 336 bytes
Interactive MATLAB Environment (ctd)

>> ls
>> whos
>> save work
>> clear
>> whos
>> load work
>> whos

Note: You can save and load variables in the workspace

Note: You can save the interactive commands in a file (e.g., script.m) and run the commands by typing “>> script”
MATLAB graphics

The plot function is very powerful for plotting all sorts of variables. See “help plot” for more details and see the examples in the MATLAB online tutorial.

```matlab
» a = 1:100;
» b = 100:0.01:101;
» c = 101:-1:1;
» d = [a b c];
» e = [d d d d d];
» plot(e)
```

What will this look like?
MATLAB graphics

The plot function is very powerful for plotting all sorts of variables. See “help plot” for more details and see the examples in the MATLAB online tutorial.

```matlab
» a = 1:100;
» b = 100:0.01:101;
» c = 101:-1:1;
» d = [a b c];
» e = [d d d d d];
» plot(e)
```
MATLAB graphics

» a = 0:0.1:10;
» subplot(3,1,1); plot(sin(a))

» r1 = rand(1,length(a))*0.2;
» subplot(3,1,2); plot(sin(a)+r1)

» r2 = rand(1,length(a))*0.8;
» subplot(3,1,3); plot(sin(a)+r2)
MATLAB images

» im = imread('face.jpg');
» whos im
Name Size Bytes Class
im 120x128 122880 double array

Grand total is 15360 elements using 122880 bytes

» hold on;

» plot(10,120,'*','r');
MATLAB images

```matlab
» im = imread('face.jpg');
» whos im
Name      Size         Bytes  Class
im 120x128  122880 double array

Grand total is 15360 elements using 122880 bytes

» hold on;

» plot(10,120,'*','r');
```
MATLAB graphics (ctd)

```matlab
x = rand(1,100);
y = rand(1,100);
plot(x,y,'*')
```
Project 1

• Due next Thursday as 12:30 pm
  ○ Use EEE dropbox, HW1 directory
  ○ No late assignments accepted – submit what you have by 12:30

• 2 parts:
  1. Online MATLAB tutorials (important!)
  2. Write “background subtraction” code

  ○ Compute average image in a video sequence
  ○ Compute a “background subtracted image” using:
  ○ Squared euclidean distance between 2 RGB vectors \( \mathbf{x} \) and \( \mathbf{y} \)
    ▪ \( \text{distance}(\mathbf{x},\mathbf{y}) = \left( \sum_i (x_i - y_i)^2 \right) \)
    ▪ sum is over the components of the vectors,
      ▪ e.g., \( \mathbf{x} = [1 \ 2 \ 3] \), \( \mathbf{y} = [4 \ 5 \ 6] \)
      ▪ \( \text{distance}(\mathbf{x},\mathbf{y}) = (1-4)^2 + (2-5)^2 + (3-6)^2 \) = ...

Thursday, March 31, 2011
Summary

• Today’s lecture
  ○ Class overview
  ○ A broad overview of MATLAB
  ○ Start reading tutorial material in Assignment 1 to get more details

• Next Lecture
  ○ Fancy background subtraction