Audio-Visual Speech Processing

COM 4110 / COM 6070

Jon Barker

j.barker@dcs.shef.ac.uk

http://www.dcs.shef.ac.uk/~jon

Department of Computer Science

University of Sheffield

Lecture 3: Image Processing with MATLAB

Objectives

- To understand how images can be represented.
- To learn how to manipulate images using MATLAB.

Topics

- Representing Images Digitally.
- An Introduction to MATLAB.
- Manipulating Images with MATLAB.

Reading

- *Getting Started With MATLAB*,
  http://www.mathworks.com/access/helpdesk/help/techdoc/learn_matlab/learn_matlab.shtm

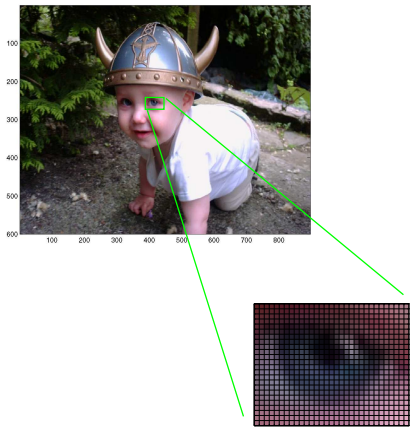
Lecture 3: Image Processing with MATLAB

Overview

- Digital Representation of Images.
- An Introduction to MATLAB.
- Manipulating Images with MATLAB.

Digital Images

A digital image is constructed from a grid of discrete ‘pixels’.



The above image has 600 rows of 840 pixels, i.e. about half a million pixels.

This is typical for a high quality digital video camera frame.

Colour Images - The RGB Colour Space

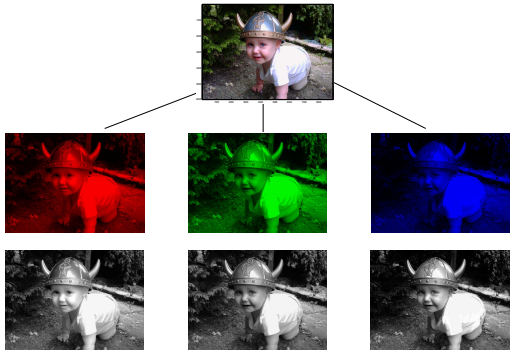
Digital video cameras encode colour by decomposing it into a mixture of red (R), green (G) and blue (B) light, (hence **RGB**).

A typical colour image employs an 8-bit number to store each of the R, G and B components for each pixel. So the previous image would require 600 x 840 x 3 bytes (1.5 Mb) of storage.

- (R,G,B) = (255,0,0) = pure red,
- (R,G,B) = (0,255,0) = pure green,
- (R,G,B) = (255,255,255) = white,
- (R,G,B) = (0,0,0) = black,
- (R,G,B) = (0,100,255) = greeny-blue
- (R,G,B) = (255,255,0) = yellow.

Colour Images - The RGB Colour Space

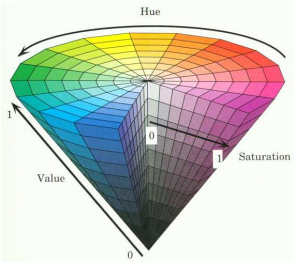
Here we see the image broken down into its red, green and blue components.



Note the varying contributions of each colour component. For the bushes the green component is dominant, for the face the red component dominates, whilst the helmet is mostly blue.

Colour Images - The HSV Colour Space

RGB is just one way of describing a colour. A popular alternative is to use Hue (H), Saturation (S) and Value (V) - hence HSV.



Some simple equations will convert between RGB and HSV.

HSV is useful as the hue value is robust to changes in the lighting conditions.

HSV is sometimes called HSB (Hue, Saturation, Brightness). It is not to be confused with a closely related colour space known as HLS (Hue, Lightness, Saturation).

[see http://www.fho-emden.de/ hoffmann/hlscone03052001.pdf]

Greyscale Images

Grey scale images use one value to represent each pixel and are displayed using shades of grey varying between white and black.

The grey scale is usually employed to depict the brightness (or 'luminance') of the image.

Note - **gray** = US spelling, **grey** = British English spelling.

RGB to Greyscale Conversion

We need to estimate the brightness of the colour image at each point.

Three different techniques:

- Brightness = R + G + B (left)
- Brightness = $0.3 \times R + 0.59 \times G + 0.11 \times B$ (centre)

  The coefficients 0.3, 0.59 and 0.11 model the way that red, green and blue light make a different contribution to the **perception** of brightness.
- RGB -> HSV, and take Brightness = V (right)



Lecture 3: Image Processing with MATLAB

Overview

- Digital Representation of Images.
- **An Introduction to MATLAB.**
- Manipulating Images with MATLAB.
- The Assessed Practical Assignment.

What is MATLAB?

- MATLAB is a matrix-vector-orientated system that supports a very wide range of things
- It is a powerful tool for work in audio, image or video processing, network analysis, pattern recognition, scientific computing...
- It deals naturally and efficiently with vectors and matrices and the operations you want to do with such objects
- It is an excellent environment for plotting graphs
- Availability: on both the CICS and DCS networks; to have your own copy you have to buy the *Student Edition of MATLAB*

Setting up Vectors

```
>> x = [5.5 6.6 7.7]
x = 5.5000    6.6000    7.7000

>> y = [5.5; 6.6; 7.7]
y = 5.5000
    6.6000
    7.7000

>> y'
ans =  5.5000    6.6000    7.7000

>> z = [1 2 3 4];
>> z
z =  1     2     3     4

>> z(2)=5
z = 1     5     3     4

>> n = length(z)
n = 4
```

Vector Operations

MATLAB can perform standard operations

`v = [1 1 1]; w=[2 3 4];`

Addition `v + w` returns `[3 4 5]`

Subtraction `v - w` returns `[-1 -2 -3]`

Scalar multiplication `9*v` returns `[9 9 9]`

Magnitude `norm(v)` returns `1.7321`

Scalar product `dot(v,w)` returns `9`

Vector product `cross(v,w)` returns `[1 -2 1]`

Matrix Arithmetic (2)

- Adding a scalar

```
>> x + 1
ans =
     4     5     6
     4     0     2
     1     2     1
```

- Element by element multiplication

```
>> x .* y
ans =
     3     4     5
     4    -1     0
     0     1     0
```


Matrix Arithmetic (1)

- To create two 3-by-3 matrices from the keyboard:

```
» x = [1 1 1; 2 -1 0; 0 1 1]; y = [3 4 5; 2 1 2; 1 1 0];
```

- Matrix addition

```
>> x+y
ans =
     4     5     6
     4     0     2
     1     2     1
```

- Matrix multiplication

```
>> x * y
ans =
     6     6     7
     4     7     8
     3     2     2
```

```
>> y * x
ans =
    11     4     8
     4     3     4
     3     0     1
```


More matrices

- **Identity matrix** The 3-by-3 identity matrix is given by `eye(3)`
- **Zero matrix** The 3-by-3 zero matrix is given by `zeros(3,3)`
- **Ones** A 3-by-3 matrix of ones is given by `ones(3,3)`
- **Size of a matrix** [rows, cols] = size(x)
- **Submatrix** The ':' symbol can be used to select parts of a matrix

```
>> x(1:2,2:3)
ans =
     1     1
    -1     0
>> x(1:2,:)
ans =
     1     1     1
     2    -1     0
```

Determinant and Inverse

- **Determinant** Finding the determinant is easy:

```
>> det(x)
ans = -1
```

- **Inverse** Finding the inverse of a square matrix is done with `inv`

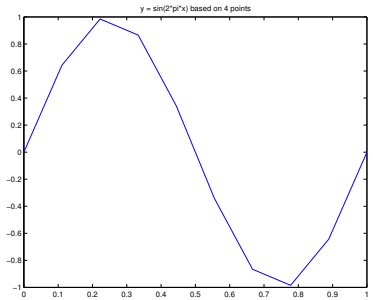
```
>> inv(x)
ans =
    1.0000    0.0000   -1.0000
    2.0000   -1.0000   -2.0000
   -2.0000    1.0000    3.0000
```


Programming in MATLAB

- Variables are not declared by the user but created as needed in the MATLAB workspace
- In MATLAB every simple variable is a matrix of numbers indexed from 1. Scalars are 1-by-1 matrices; Vectors are “skinny” matrices.
- Sequences of commands are called MATLAB **scripts**, which may be stored in files (with the suffix `.m`)
- You can also defined MATLAB **functions**, also stored in m-files. A function file should contain a single definition and the function name and file name should match (see next slide).
- Variable within a function are local; variables within a script are global (part of the general workspace)

A MATLAB Function

```
function y=sinplot(n)
%SINPLOT plots sin(2*pi*x) between 0 and 1
%
% Usage: sinplot(b)
% where n is the number of points the plot is based on
x = linspace(0,1,n+1)
y = sin(2*pi*x);
plot(x,y)
title(sprintf('y = sin(2*pi*x) based on %d points', n))
```



Lecture 3: Image Processing with MATLAB

Overview

- Digital Representation of Images.
- An Introduction to MATLAB.
- **Manipulating Images with MATLAB.**
- The Assessed Practical Assignment.

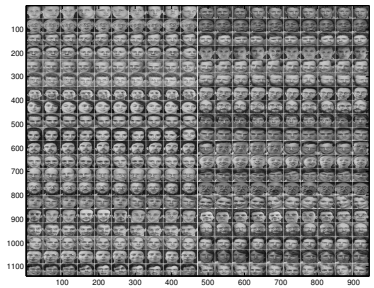
Working with Images

- **Loading images** - this couldn't be easier:

```
>> y = imread('faces.png');
>> size(y)
ans =   1140          942
```

- **Displaying images** - use `image` or `imagesc`

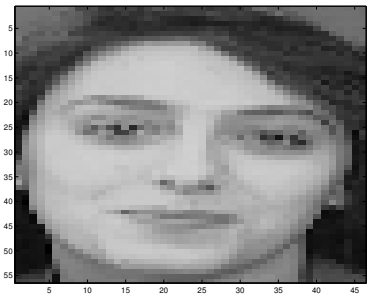
```
>> image(y);
```



Creating a Sub-Image

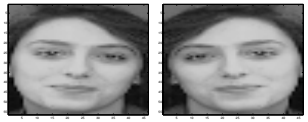
Greyscale images are simply matrices. So we can cut out a subimage by extracting the corresponding submatrix with the ':' symbol. e.g. to cut a single face from the previous image:

```
>> face=y(400:455,330:375);
>> image(face);
```

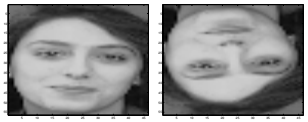


More Matrix Operations Applied to Images

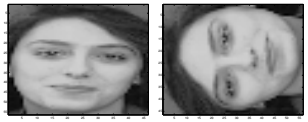
- **Flipping left-to-right** - `newface=fliplr(face);`



- **Flipping up-to-down** - `newface=flipud(face);`



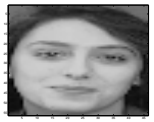
- **Matrix transpose** - `newface = face';`



More Image Manipulation

- **Increasing the brightness** - i.e. increase the mean value

```
newface=double(face)+40.0;
```

+0  +40  +80  +120 

- **Increasing the contrast** - i.e. increase the variance

```
>> dface=double(face);
>> meanface = mean(mean(dface));
>> newface = meanface + (dface-meanface) * 1.5;
```

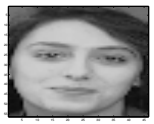
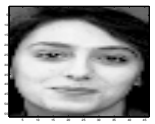
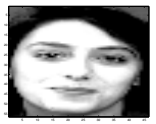
x1.0  x1.5  x2.0  x3.0 

Image Rotation - The Hard Way

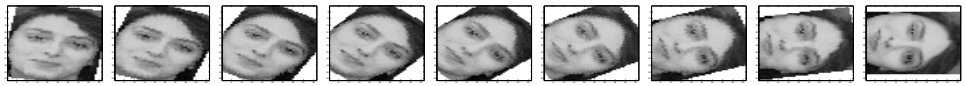
```
function rotface=rotate_face(face, theta)
theta=theta*pi/180;
[xmax,ymax]=size(face);
xc=xmax/2; yc=ymax/2;

fpad=255*ones(xmax+2,ymax+2);
fpad(2:xmax+1,2:ymax+1)=face;
fout=zeros(xmax+2,ymax+2);

for x=1-xc:xmax+2-xc
  for y=1-yc:ymax+2-yc
    phi=atan(y/(x+0.00001)); r=sqrt(x*x+y*y); xsign=1-2*(x<0);
    fout(x+xc,y+yc)=fpad(min(xmax+2,max(1,uint8(xc+xsign*r*cos(phi+theta)))),
                         min(ymax+2,max(1,uint8(yc+xsign*r*sin(phi+theta)))));
  end
end
rotface=fout(2:xmax+1,2:ymax+1);
imagesc(rotface);
```

Image Rotation - The Easy Way

```
rotface=imrotate(face, theta, 'bilinear', 'crop')
```

Rotates face by theta degrees, employing bilinear interpolation and cropping the rotated image to fit in the same area as the original.

Bilinear interpolation removes the grainy effect produced by the non-interpolated function on the previous slide (below).



`imrotate` is part of the **MATLAB Image Processing Toolbox**. Toolboxes are extensions to MATLAB which provides many ready made functions. Unfortunately, the Image Processing Toolbox is not presently available on the DCS teaching network.

Summary

- MATLAB is a power system for doing matrix based computation
- It provides a simple interpreted programming language
- MATLAB code may not be as efficient as compiled languages, but it allows rapid experimentation.
- MATLAB is a very convenient tool for image processing.
- There are 3rd party MATLAB 'toolboxes' (like libraries) which provide support for a wide range of applications, e.g. statistics, signal processing, machine learning, image processing, etc.

Lecture 4&5 Preview: Face Detection

The lectures to follow will examine the processing stages necessary in the design of an **automatic audio-visual speech recognition system**.

The first step is to **detect the faces** of the speakers. This crucial step will be the focus of lectures **4** and **5** - *and of the assessment*.

The face detection lectures will cover the following techniques:

- Colour-based segmentation,
- Fisher faces (i.e. Linear Discriminant Analysis),
- Eigen-faces (i.e. Principal Component Analysis),
- Neural network-based face detection.