1 Introduction

The goal of this tutorial is to increase your comfort level with matlab and to introduce some techniques that will be useful as we move through the course. Good introductions to matlab are available at [http://www.math.utah.edu/lab/ms/matlab/matlab.html](http://www.math.utah.edu/lab/ms/matlab/matlab.html) and in the Day6 directory. The matlab help files are also very helpful.

2 Preliminaries

Matlab is a great interactive tool. Once you get a small chunk working, I recommend that you capture it in a script (a file with the extension of .m). You can execute the script simply by typing the name of the file (minus the extension). You can also use the `save` and `load` commands save and reload you work space. This can be quite handy in picking up where you left off.

3 Entering Matrices

Creating a matrix in matlab is very straightforward. Square brackets ("[" and "]") are used to identify a matrix. Elements of a row are separated by spaces and rows are ended by a semicolon. For example, the matlab command

\[
a = [2 4; 3 1]
\]

will make the matrix assignment

\[
A = \begin{bmatrix} 2 & 4 \\ 3 & 1 \end{bmatrix}.
\]

Matlab has several useful matrix building functions: `eye(n)` creates an \( n \times n \) identity matrix; `zeros(m, n)` creates an \( m \times n \) matrix of zeros; and `ones(m, n)` creates an \( m \times n \) matrix of ones. Matrices can also be constructed from other matrices. For example, if \( x \) and \( y \) are column vectors then the matlab command

\[
a = [x \ y]
\]

constructs a matrix with 2 columns. Note that putting a semicolon at the end of the assignment will prevent matlab from printing the matrix. This can be very helpful when working with images.
4 Submatrices

The matlab command

\[ A(1,1) \]

refers the element of matrix \( A \) that is in the first row and first column. Ranges and sets of indices can also be used to reference elements of a matrix. For example,

\[ A(:,1) \]

extracts the first column of \( A \),

\[ A(:,1:3) \]

extracts columns 1 through 3, and

\[ A(:,[1 3]) \]

extracts columns 1 and 3. This can be used to select a single color from an image for use in tools which require gray scale image (e.g. \texttt{cpselect}). Submatrices can also be target of an assignment.

5 Matrix Operators

Matlab provides a wide variety of matrix operators. Matrices can be added (+), subrtraced (-), multiplied (*), divided (both left \( \backslash \) and right /) and exponentiated (^). These commands can also be performed element-wise by preceding the operator with a “.”. Other matrix operations include:

\[
\begin{align*}
A' & \quad \text{returns the transpose of } A \\
\text{inv}(A) & \quad \text{returns the inverse} \\
\text{size}(A) & \quad \text{returns the size} \\
\text{det}(A) & \quad \text{returns the determinant} \\
\text{rank}(A) & \quad \text{returns the rank} \\
\text{norm}(A) & \quad \text{returns the norm}
\end{align*}
\]

6 Vector Operators

Useful vector operations include:
cross(x, y) returns the cross product of vectors x and y
min(A) returns the minimum element
max(A) returns the maximum element
mean(A) returns the mean value
var(x,1) returns the second moment about the mean

7 Solving Linear Equations

Non-homegeneous equations of the form \( Ax = b \) can be solved using

\[ x = A\backslash b \]

They can also be solved using a variety of factorization methods. The matlab help pages on “solving linear equations” have more info.

Homogeneous equations of the form \( Ax = 0 \) can be solved using

\[ \text{null}(A) \]

if the system is exactly determined. If the system is over determined and noisy singular value decomposition is your best bet. For example,

\[ [u \ d \ v] = \text{svd}(A) \]

decomposes the coefficient matrix into 3 other matrices. The solution to the system of equations is the last column of \( v \). The last column of \( v \) can be reshaped into a matrix using something like this

\[ \text{reshape}(v(:,9),3,3) \]

Note: matlab takes the elements column-wise and we probably want the transpose of what matlab returns.

8 Image Functions

Matlab provides a number of functions that are helpful for manipulating images. Browsing the help pages for the “image processing toolbox” is a good place to start. Some of the key functions are:
i = imread('filename'); \hspace{1em} \text{read image filename into matrix i}

imshow(i) \hspace{1em} \text{display image i}

figure \hspace{1em} \text{used to control the window in which data is display}

hold \hspace{1em} \text{allows overlays on a figure}

p = ginput(4) \hspace{1em} \text{Puts cross-hairs on the current figure and allows the selection}
\hspace{1em} \text{of “4” points from the image.}

\textbf{Note:} be careful moving between images, which we typically access as \((x, y)\), and the matrix
\hspace{1em} \text{representation, which matlab accesses as \texttt{(row, col)}.}

\begin{itemize}
  \item \texttt{maketform} \hspace{1em} \text{creates a transformation structure that can be applied to an}
  \hspace{1em} \text{image}
  \item \texttt{imtransform} \hspace{1em} \text{applies a transform structure to an image}
\end{itemize}

\textbf{Note:} Internally matlab uses pre-multiplication. The book, most vision and graphics folks and I
\hspace{1em} \text{use post-multiplication. Therefore \texttt{maketform} needs the transpose of our transformation}
\hspace{1em} \text{matrices.}

\begin{itemize}
  \item \texttt{cpselect(i1, i2)} \hspace{1em} \text{allows interactive selection of corresponding points (control}
    \hspace{1em} \text{points) in a pair of images}
  \item \texttt{cpcorr} \hspace{1em} \text{refines the locations of corresponding points}
  \item \texttt{cp2tform} \hspace{1em} \text{creates a transformation which will map one image to the other}
\end{itemize}

\section{Functions}

It may be useful to create matlab functions for small pieces of reusable code. The matlab help
\hspace{1em} \text{pages on “m-files” have lots of good info on how to do this.}

\section{Exercises}

1. Find the plane defined by the points \(\mathbf{p1} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T\), \(\mathbf{p2} = \begin{bmatrix} 2 & 1 & 1 & 1 \end{bmatrix}^T\) and
\hspace{1em} \(\mathbf{p3} = \begin{bmatrix} 1 & 2 & 1 & 1 \end{bmatrix}^T\), using
\hspace{1em} \[\begin{bmatrix} \mathbf{p1}^T \\
\mathbf{p2}^T \\
\mathbf{p3}^T \end{bmatrix} \pi = 0\]
\hspace{1em} and the matlab command \texttt{null}.

2. Find the plane which best fits the points \(\mathbf{p1} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T\), \(\mathbf{p2} = \begin{bmatrix} 2 & 1 & 1 & 1 \end{bmatrix}^T\),
\hspace{1em} \(\mathbf{p3} = \begin{bmatrix} 1 & 2 & 1 & 1 \end{bmatrix}^T\) and \(\mathbf{p4} = \begin{bmatrix} 2 & 2 & 0.9 & 1 \end{bmatrix}^T\), using
\hspace{1em} \[\begin{bmatrix} \mathbf{p1}^T \\
\mathbf{p2}^T \\
\mathbf{p3}^T \\
\mathbf{p4}^T \end{bmatrix} \pi = 0\]
and the matlab command \texttt{null}.

3. Find the plane which best fits the points $\mathbf{p}_1 = \begin{bmatrix} 1 & 1 & 0.9 & 1 \end{bmatrix}^\top$, $\mathbf{p}_2 = \begin{bmatrix} 2 & 1 & 1 & 1 \end{bmatrix}^\top$, $\mathbf{p}_3 = \begin{bmatrix} 1 & 2 & 1 & 1 \end{bmatrix}^\top$ and $\mathbf{p}_4 = \begin{bmatrix} 2 & 2 & 0.9 & 1 \end{bmatrix}^\top$, using

$$
\begin{bmatrix}
\mathbf{p}_1^\top \\
\mathbf{p}_2^\top \\
\mathbf{p}_3^\top \\
\mathbf{p}_4^\top
\end{bmatrix} \pi = 0
$$

and the matlab command \texttt{svd}.