1 Short comment on the Matlab *image* command

The usage of the Matlab *image* command is sometimes a point of confusion. It can be used to display either monochrome or color image data, but the behavior for each of these modes is slightly different, and also sometimes depends on the data type (e.g. integer or floating point). The following sections will describe how to use *image* through examples.

2 Using *image* to display monochrome images

Here's a Matlab example of reading in and displaying a monochrome image.

```
A=imread('mono_img.tif');
image(A);
colormap(gray(256));
axis('image');
```

In this example, *imread* will define the image matrix A as type *uint8* (each value is represented by an 8-bit unsigned integer). If any processing is to be performed on the image, it would need to be converted to type *double* (floating point, double precision). Luckily, this application of the *image* command will work for both *uint8* and *double* data types.

The colormap command is necessary to map the values in A to the correct shades of gray in the displayed image. In a monochrome image, each pixel is represented by a single number. However, in some applications, it may be desired to display a range of colors other than gray values between black and white. For example, in weather imagery, intensity is sometimes represented by the colors green through red. A *colormap* is a function that maps the scalar pixel value to an ordered triple, which specifies the red, green, and blue values of the displayed pixel. To display gray levels, like in black and white photographs, the color components contain equal levels of red, green, and blue, and the particular level determines the "shade" of gray.

In Matlab, the expression `gray(256)` returns a 256×3 matrix, where each row contains contains the red, green, and blue levels for a gray colormap:

```
     0         0         0
0.0039    0.0039    0.0039
0.0078    0.0078    0.0078
0.0118    0.0118    0.0118
   .         .         .
   .         .         .
0.9961    0.9961    0.9961
1.0000    1.0000    1.0000
```

When using the *image* command, each pixel value in the image matrix ("A" in the

Questions or comments concerning this laboratory should be directed to Prof. Charles A. Bouman, School of Electrical and Computer Engineering, Purdue University, West Lafayette IN 47907; (765) 494-0340; bouman@ecn.purdue.edu

example above) is interpreted as an index of a row in the current colormap array. So for this example, a pixel value of 1 would map to the color (0,0,0) (black), and a value of 256 would map to (1,1,1) (white).

Note this works the same for floating point images. In this case the decimal component is just ignored (not rounded). If a pixel value exceeds the index range of the colormap (less than 1 or greater than 256 in our example), the *image* command just maps to the nearest valid index value (i.e. clips to black or white). Alternatively, the command *imagesc* linearly scales the image data values so that the minimum and maximum pixel values map to the first and last rows of the colormap.

3 Using *image* to display color images

Here's a Matlab example of reading in and displaying a color image.

```
A=imread('color_img.tif');
image(A);
axis('image');
```

If *color_img.tif* is indeed a color image, *imread* will read the image into a 3-dimensional array. This array will contain 3 image planes, e.g. $A(:,:,1:3)$, corresponding to the red, green, and blue components of the image. In this example, the matrix "A" will be read in as type *uint8*, so all pixel components will be integer values between 0 and 255.

For color images, the behavior of the *image* command depends on the data type. If the image matrix is of type *uint8*, as in this example, *image* will interpret 0 as the smallest color component value, and 255 as the largest value. However, if the image matrix is of type *double*, the range of possible color component values changes to [0,1], and the function will return an error if any color values are outside this range.

Therefore, if you convert a type *uint8* image to type *double* to apply a filter, you need to either convert back to *uint8* before calling the *image* command, or transform the image values so that all components are in the range 0 to 1. Note the *imagesc* command does *not* work for this purpose (as of release R2008a), since it is not designed for color images.

4 Matlab Help on image

```
IMAGE  Display image.
   IMAGE(C) displays matrix C as an image.  Each element of C
   specifies the color of a rectilinear patch in the image.  C can be
   a matrix of dimension MxN or MxNx3, and can contain double, uint8,
   or uint16 data.

   When C is a 2-dimensional MxN matrix, the elements of C are used
   as indices into the current COLORMAP to determine the color.  The
   value of the image object's CDataMapping property determines the
   method used to select a colormap entry.  For 'direct' CDataMapping
```

```
(the default), values in C are treated as colormap indices
(1-based if double, 0-based if uint8 or uint16).  For 'scaled'
CDataMapping, values in C are first scaled according to the axes
CLim and then the result is treated as a colormap index.  When C is
a 3-dimensional MxNx3 matrix, the elements in C(:,:,1) are
interpreted as red intensities, in C(:,:,2) as green intensities,
and in C(:,:,3) as blue intensities, and the CDataMapping property
of image is ignored.  For matrices containing doubles, color
intensities are on the range [0.0, 1.0].  For uint8 and uint16
matrices, color intensities are on the range [0, 255].

IMAGE(C) places the center of element C(1,1) at (1,1) in the axes,
and the center of element (M,N) at (M,N) in the axes, and draws
each rectilinear patch as 1 unit in width and height.  As a
result, the outer extent of the image occupies [0.5 N+0.5 0.5 M+0.5]
of the axes, and each pixel center of the image lies at integer
coordinates ranging between 1 and M or N.

IMAGE(X,Y,C), where X and Y are vectors, specifies the locations
of the pixel centers of C(1,1) and C(M,N).  Element C(1,1) is
centered over (X(1), Y(1)), element C(M,N) is centered over
(X(end), Y(end)), and the pixel centers of the remaining elements
of C are spread out evenly between those two points, so that the
rectilinear patches are all of equal width and height.

IMAGE returns a handle to an IMAGE object.

C or the X,Y,C triple can be followed by property/value
pairs to specify additional properties of the image.
C or the X,Y,C triple can be omitted entirely, and all
properties specified using property/value pairs.

IMAGE(...,'Parent',AX) specifies AX as the parent axes for the
image object during creation.

When called with C or X,Y,C, IMAGE sets the axes limits to tightly
enclose the image, sets the axes YDir property to 'reverse', and
sets the axes View property to [0 90].

The image object will not render at axes View angles other than
[0 90].  To get a similar effect to rotating an image, use SURF
with texture mapping or PCOLOR.

Execute GET(H), where H is an image handle, to see a list of image
object properties and their current values. Execute SET(H) to see a
```

```
   list of image object properties and legal property values.

   See also imagesc, colormap, pcolor, surf, imread, imwrite.
```

5 Matlab Help on imagesc

```
IMAGESC Scale data and display as image.
   IMAGESC(...) is the same as IMAGE(...) except the data is scaled
   to use the full colormap.

   IMAGESC(...,CLIM) where CLIM = [CLOW CHIGH] can specify the
   scaling.

   See also image, colorbar, imread, imwrite.
```