

## MATLAB Lecture 6. Images and Movies in MATLAB

Images are represented as three dimensional matrices: for every point we store with its coordinates  $(x, y)$  the red, green, and blue intensities. The animations in MATLAB are more like real movies, as we can adjust the position and aim of the camera.

### 6.1 Images in MATLAB

From the website of this course, you can download the image “Library.jpg”, using the web address <http://www.math.uic.edu/~jan/mcs320/Library.jpg>

To load and display an image in MATLAB, we type the following sequence of commands:

```
>> x = imread('h:\Library.jpg'); % image saved on drive h:
>> image(x)
>> axis off
```

If we type `size(x)`, we see three numbers, the dimensions of the three dimensional matrix. In particular,  $x$  is a  $m \times n \times 3$  matrix, the elements in  $x(:, :, 1)$  are the red intensities,  $x(:, :, 2)$  are the green intensities, and  $x(:, :, 3)$  are the blue intensities.

For example, to remove the blue intensities, we can do the following :

```
>> noblue = x;
>> noblue(:, :, 3) = zeros(size(x,1),size(x,2));
>> image(noblue);
```

The entries in  $x$  are unsigned 8-bit integers, whereas the operations for MATLAB are mainly done with doubles. Therefore, we need to convert to doubles before we manipulate the images:

```
>> dx = double(x) % convert to double
>> y = uint8(dx) % convert to unsigned 8-bit integer
>> image(y) % same image as before
```

We can make the picture more blue by multiplying the blue intensities with the factor 1.5:

```
>> moreblue(:, :, 1) = x(:, :, 1);
>> moreblue(:, :, 2) = x(:, :, 2);
>> moreblue(:, :, 3) = dx(:, :, 3)*1.5;
>> image(uint8(moreblue));
```

Assigning the red, green, and blue intensities to their average, we obtain a gray scale picture:

```
>> gy = (dx(:, :, 1) + dx(:, :, 2) + dx(:, :, 3))/3;
>> y(:, :, 1) = gy;
>> y(:, :, 2) = gy;
>> y(:, :, 3) = gy;
>> image(uint8(y));
```

By selection of submatrices and the proper choice of axes, we can zoom the picture.

We can encrypt a message by multiplying the red, green, and blue intensities with a random matrix. To decrypt we then multiply with the inverse of that random matrix. Assuming we loaded the image into  $x$ , we generate a blurring matrix as follows:

```
>> blur = randn(size(x,1));    % matrix to blur
>> key = inv(blur);          % matrix to reconstruct image
```

To encrypt or decrypt we can use the m-file `crypto`:

```
function y = crypto ( c, i )
% returns either blurred or reconstructed image
y(:, :, 1) = c*i(:, :, 1);
y(:, :, 2) = c*i(:, :, 2);
y(:, :, 3) = c*i(:, :, 3);
```

Here is how we use `crypto`:

```
>> bx = crypto(blur,double(x)); % bx is matrix of doubles
>> y = crypto(key,bx);          % restore original image x
>> image(uint8(y));            % display to verify
```

## 6.2 Movies in MATLAB

Storing several plots as frames in a movie allows us to make animations. An extra feature of MATLAB is that it allows to adjust the viewpoint.

### 6.2.1 Position and Aim the Camera

The graphics in MATLAB offer some direct tools to create interesting views of a surface. We illustrate these features on the logo of MATLAB. In the sequence of instructions below we first search to locate the position and target of the camera. We do this on the global view. In the other subplot we ask MATLAB to generate the picture from those camera positions.

```
>> subplot(2,1,1)                % global view
>> membrane                      % logo of MATLAB
>> hold on
>> xlabel('x'); ylabel('y'); zlabel('z'); % useful for positioning
>> s = [-0.9 1.0 0.2];           % start position
>> plot3(s(1),s(2),s(3),'r+');   % mark start on plot
>> p = [0.5 0.0 1.1];           % peak of membrane
>> plot3(p(1),p(2),p(3),'bx');   % mark peak on plot
>> subplot(2,1,2)                % create local view
>> membrane
>> hold on
>> set(gca,'CameraPosition',s)    % place the camera
>> set(gca,'CameraTarget',p)     % aim the camera
>> set(gca,'Projection','Perspective') % perspective view
```

The command `gca` returns a handle to the current axis in the current figure. To see all properties of the current graphical object, we invoke `get(gca)`. With `set` we can set the properties of objects. In the three `set` instructions above we set the position of the camera at one corner of the plane field of the membrane, aim the camera towards the peak of the membrane, and create a perspective view.

Since the positioning may involve some trial and error, we better put our instructions in a script.

### 6.2.2 Make a Movie

With the settings of *start* and *peak* as above, we make a movie to simulate what you would see if you were walking on the plane field towards the peak of the membrane.

First we must decide on the number of frames and give this number as argument to the the command **moviein**. The result of this command is assigned to the name of the movie. Then we make a plot for each frame and invoke the command **getframe** to store the plot into the movie. For a generic plot with 10 frames, we create a movie like

```
>> M = moviein(10);           % movie in 10 frames
>> for i = 1:10               % create the frames
    << here we type all plot commands for current frame >>
    M(:,i) = getframe         % store the frame
end;
>> movie(M)                   % play the movie
```

With the command **movie** we can play the movie as many times as we like at various speeds (as fast as your machine allows).

For the example with the MATLAB logo, we execute the following sequence of commands (using the *s* and *p* we defined earlier):

```
>> figure                     % open new window
>> topeak = moviein(100);     % 100 frames in movie
>> membrane                   % make first plot
>> set(gca,'CameraPosition',s);
>> set(gca,'CameraTarget',p);
>> set(gca,'Projection','Perspective');
>> topeak(:,1) = getframe;     % first frame of movie
>> dt = 0.01;                 % increment for positions
>> for i = 2:100               % create other frames
    s(1) = s(1) + dt;         % move x coordinate of start
    s(2) = s(2) + 0.5*dt;     % move y coordinate of start
    p(3) = p(3) + dt;         % aim camera higher
    set(gca,'CameraPosition',s);
    set(gca,'CameraTarget',p);
    set(gca,'Projection','Perspective')
    topeak(:,i) = getframe;   % store the frame
end;
```

As the loop to create the frames in the movie is executed, we see the movie playing. After creation, we can play the movie by the command

```
>> movie(topeak);           % play the movie
```

### 6.3 Assignments

1. The image “easter-egg.jpg” (available at <http://www.math.uic.edu/~jan/mcs320/easter-egg.jpg>) shows an egg in a nest. Load the image into MATLAB and manipulate the image so that it looks as if the egg has been removed from the nest. To create this impression, replace the color encoding at the coordinates of the egg by the color encoding of pixels of the surrounding nest.
2. The contrast in an image can be enhanced by giving low intensity pixels an even lower intensity and increasing the intensity of the high intensity pixels. One way to achieve this, is the *gamma correction*, defined by  $y_{ij} = [W \cdot (x_{ij})^\gamma / W]$ , where  $W$  equals the maximal value corresponding to the resolution, and for some constant  $\gamma$ . The  $x_{ij}$  represents either the grayscale encoding of the original picture at pixel  $(i, j)$ , or is one of the RGB intensities at  $(i, j)$ .

Take your favorite picture (or use the “easter-egg.jpg”) and try to find a good value of  $\gamma$  to enhance the contrast.

3. Make a sequel to the movie `topeak` above that leaves the plane field of the membrane and climbs up to the peak.
4. Type in the following commands

```
>> [x,y,z] = peaks;  
>> plot3(x,y,z);
```

- (a) Set up the camera position at the far most left corner and aim at one of the peaks. Make a perspective view with those camera positions.
  - (b) Make a movie of what you would see if you were walking on the surface from one corner to one of the peaks (or valleys).
5. Consider the surface  $z = x^3 - 3xy^2$ .
    - (a) Make a plot over the grid  $[-1, 1] \times [-1, 1]$ .
    - (b) Make a perspective view from close above the origin, looking at one of the higher sides.
    - (c) With the camera positioned slightly above the origin, move the target position around to make a movie of the surface, as if you were looking in all directions.
    - (d) Make a movie of what you might see if you were an ant crawling over the surface. Start close to the origin and crawl up to one of the higher places on the surface.