

EN-74 ECE: Introduction to Image Processing
Tufts University
Fall 2007
Problem Set 3
Due Sept. 26, 2007

1. Can computers represent numbers like π or e ? Explain

The problem here is that these are irrational numbers and hence cannot be represented *exactly* in a finite number of digits. Since computers by their very nature have finite sized memories, they can only allow for approximate (though still highly accurate) representations of these numbers.

2. Why does `imshow` work so poorly for binary images that are of type `uint8`?

For type `uint8`, `imshow` is expecting numbers between 0 and 255 with zero being black and 255 being white. When we cast a binary image as a `uint8`, the zeros and ones are interpreted on this 0 to 255 scale. Hence all the zeros are black and the ones are only slightly less black. The overall image looks pretty poor.

3. Using `mod` and `floor`, rewrite `bitplane.m` to get rid of the `for` loops.

```
function B = bitplane1(I,num)

% Function to extract bitplane num from image I
% Inputs:
%   I = grayscale image of type uint8
%   num = number of bitplane we want
% Outputs
%   B = output bitplane image

% Let's do some error checking to help the user of the program
if (~isa(I,'uint8'))
    error('Input image must be a uint8')
end
if ((num > 8) | (num < 1) )
    error('Bitplane must be between 1 and 8')
end

B = logical(mod(floor(I/2^num),2));
```

4. Using **bitget**, rewrite **bitplane.m** to get rid of the **for** loops.

```
function B = bitplane(I,num)

% Function to extract bitplane num from image I
% Inputs:
%   I = grayscale image of type uint8
%   num = number of bitplane we want
% Outputs
%   B = output bitplane image

% Let's do some error checking to help the user of the program
if (~isa(I,'uint8'))
    error('Input image must be a uint8')
end
if ((num > 8) | (num < 1) )
    error('Bitplane must be between 1 and 8')
end

B = logical(bitget(I,num));
```

5. Using **tic** and **toc**, determine the speeds of the three versions of **bitplane.m**

The code I used for this is as follows:

```
% Read in any old color image
co = imread([imbase,'flowers.tif']);      % RGB color

% Bitplane with for loop
tic
bitplane3a = bitplane(co,3);
toc

% Bitplane without for loop, mod-floor version
tic
bitplane3a = bitplane_no_for_1(co,3);
toc

% Bitplane without for loop, bitget version
tic
bitplane3a = bitplane_no_for_2(co,3);
toc
```

The results clearly show the performance hit associated with for loops in Matlab:

```
Elapsed time is 0.522393 seconds. % Using for loops
Elapsed time is 0.030519 seconds. % mod-floor
Elapsed time is 0.058534 seconds. % bitget
```