Lect "7"

Intensity Transformation functions Using Matlab.

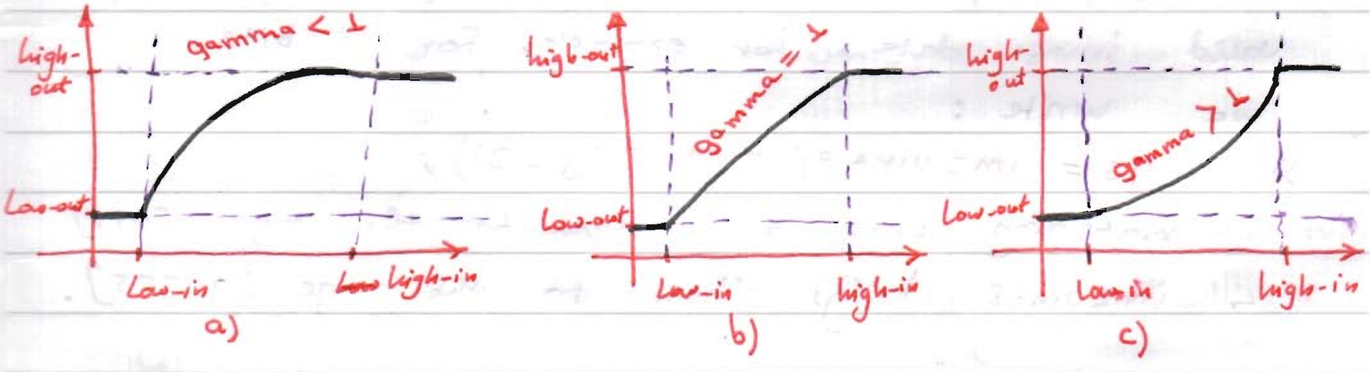
⊛ Function "imadjust"-

imadjust - the basic IPT tool for intensity transformation of grayscale image, the syntax is

g = imadjust(f, [Low-in high-in], [Low-out high-out], gamma)

this function maps the intensity values in image f to new values in g, such that the values between Low-in and high-in map to values between Low-out and high-out.

- values below Low-in and above high-in are clipped to Low-out and high-out respectivelly.
- input image can be of class uint8, uint16 or double.
- Low-in, high-in, Low-out and high-out must be between (0 - 1), the imadjust function multiplies these values : by 255 for uint 8, and 65535 for uint16.
- Using empty matrix [ ] for [Low-in high-in] or for [Low-out high-out] results in the default values [0 1]
- if high-out is less than Low-out the output intensity is reversed.
- Parameter gamma : specifies the shape of the curve that maps the intensity values in f to create g



a)- gamma < 1 : the mapping is weighted toward brighter output values b- gamma = 1 .(default) : the mapping is linear.

c- gamma > 1 : the mapping is weighted toward darker output values.

examples:

⊛ 1-
```
» g1 = imadjust (f, [0 1], [1 0]);
```
obtaining the negative image using imadjust function.

⊛ 2)
```
» g = imcomplement (f).
```
obtaining the negative image using IPT function "imcomplement"

3)
```
» g2 = imadjust ( f, [0.5 0.75], [0 1]);
```
this command expands the grayscale region between 0.5 and 0.75 to the full [0, 1] range.

4)
```
» g3 = imadjust (f, [ ], [ ], 2);
```
this command compresses the low end and expands the high end of the gray scale.

* Logarithmic and Contrast- Streching Transformations.

⊛ Logarithmic transformations are implemented in Matlab using the expression:

```
g = c * Log ( 1 + double ( f))
```

c- is a constant.

* the shape of the gamma curve is variable, whereas the shape of the Log function is fixed.

- when performing a Logarithmic transformation, it is often desirable to bring the result back to valid image data, for example, For 8 bits, we write :

```
» gs = im2uint8( mat2gray (g));
```

- mat2gray brings the values to the range [0, 1]
- im2uint8 brings them to the range [0 255].

⊛ Contrast_ streching transformation function :

The contrast- streching transformation function has the form :

$$s = T(r) = \frac{1}{1+\left(\frac{m}{r}\right)^{E}}$$; r_. input image. s- output image.

E- Controls the slope of the function.

this equation is implemented in Matlab for an entire image as :

```
g = 1 ./ (1 + (m./(double(f) + eps)).^E)
```

eps_ to prevent over flow if f has any 0 values

```
>> g = im2uint8( mat2gray( Log(1 + double( f)))); //- to
>> imshow (g) ;   % get valid image data.
```

* MatLab Example : Utility M- functions for intensity Transformatio

- The code contains error checking.
- - " - Can handle a variable number of input and/or outputs.

* Handling a Variable number of inputs and/or outputs

1) To check the number of arguments input into an M- function we use function nargin

n = nargin.

2) To check the number of arguments output into an M- function, we use function nargout

n = nargout .

example :

>> T= testhv (4, 5) ; Use of nargin within the body of this function would return a 2, while use of nargout would return a 1.

3) To check if the correct number of arguments were passed, we use nargchk, The syntax is

msg = nargchk (low, high, number),

This function returns the message:

- Not enough input parameters : if number is less than low.
- Too many input parameters : if number is greater than high.
- empty matrix : if number is between low and high (inclusive).

A frequent use of function nargchk is to stop execution via the error function if the incorrect number of arguments is input.

example :

```
function G = testhv2(x, y, z)
    :
  error( nargchk( 2, 3, nargin));
    :
```

Typing >> testhv2(6) ;% will produce the error

Not enough input arguments % and execution would terminate.

4) To write functions in which the number of input and/or outputs arguments is variable, We use Varagrin and Var argout.

example : 1) function [m, n] = testhv3(~~Varg~~ Varargin) -

accepts a variable number of inputs into function testhv3.

2) function [Varargout] = testhv4(m, n, p)

returns a variable number of outputs from function testhv4.

3) function [m, n] = testhv5(x, Varargin) .

function testhv5 has one fixed input argument x, followed by a variable number of input arguments, Similar comments ~~supplied~~ apply to Varargout. (it's acceptable to have a function in which both the number of input & output arguments is variable).

* When Varagin or varargout are used : the MatLab sets it to a ~~cell~~ cell array.

For example :

>> [m, n] = testhv5 (f, [0 0.5 1.5], A , 'Label'); % f is an image

the second argument is row vector, A - matrix, Label - string,

~~the elements of Array cell~~

"Matlab code"

Write a function that computes the following transformation functions: negative, log, gamma and contrast streching, in writing this function we use function "Changeclass" which has the synatx:

```
g = Change class( newclass, f ).
```

This function Converts image f to the class specified in parameter newclass and output it as g.

Valid Values for newclass are 'uint8', 'uint16' and 'double'.

```
function g = intrans ( f , varargin)
% INTRANS Performs intensity (gray-level) transformation.
% ① G= INTRANS (F, 'neg') Computes the negative of input image f.
% ② G= INTRANS( F, 'Log', c, CLASS) Computes C*Log(1+F)
% and multiplies the result by (positive) constant C. if the
% last two parameters are omitted, c defaults to 1.
% Because the log is used frequently to display Fourier
% spectra, Parameter CLASS offers the option to
% specify the class of the output as 'uint8' or 'uint16'
% if parameter CLASS is omitted, the output is of the
% same class as the input.
% ③ G = INTRANS ( F, 'gamma', GAM) performs a gamma-
% transformation on the input image using parameter
% GAM ( a required input).
% ④ G= INTRANS ( F, 'strech', M, E) Computes a contrast-
% streching transformation using the expression 1./(1+(M./F+eps));
% parameter M must be in the range [0,1], the default
% Value for M is mean2 ( im2double( F)), and the
% default value for E is 4.
% For the 'neg', 'gamma' and 'strech' transformations,
% double input images whose maximum Value is greater
% than 1 are scaled, first using MAT2GRAY.
% other images are converted to double first using im2double.
% For the 'Log' transformation, double images are transformated
% without being scaled; other images are converted to double
% first using IM2DOUBLE.
%   The output is of the same class as the input,
% except if a different class is specifed for the
% 'Log' option.

% Verify the correct number of inputs.
        error(nargchk(2, 4, nargin))
% Store the class of the input for use later.
        Classin = class(f);
% if the input is of class double, and it is outside
% the range [0,1], and the specified transformation is
%   not 'Log', convert the input to the range [0,1].
      if strcmp(class(f),'double') & max(f(:)) > 1 & ...
         ~ strcmp(varargin{1},'Log')
              f = mat2gray(f);
      else  % Convert to double, regardless of class(f).
              f = im2double(f);
      end
% Determine the type of transformation specified.
        method = Varargin{1};
% Perform the intensity transformation specified.
    switch method
    Case 'neg'
          g = imcomplement(f);
    case 'Log'
        if   length(varargin) == 1
                c = 1;
        else if  length(varargin) == 2
                c = varargin{2};
    elseif   length( Varargin) == 3
          c = Varargin{2};
          Classin  =  varargin{3};
    else
          error('Incorrect number of inputs for the log option.')
    end
        g =  c * (Log(1+ double(f)));
Case 'gamma'
    if length(varargin) < 2
        error('Not enough inputs for the gamma option.')
    end
     gam = varargin{2};
    g = imadjust(f, [ ], [ ], gam);
Case 'strech'
    if  length(varargin)== 1
        % use  defaults.
          m= mean2(f);
          E = 4.0;
    else if    length(varargin) == 3
          m =  Varargin{2};
          E =  Varargin{3};
    else
          error('Incorrect number of inputs for the strech option.')
    end
        g = 1./(1 +(m./(f +eps)).^E);
otherwise
    error('Unknown enhancement method.')
end
% Convert to the class of the input image.
    g = changeclass( classin, g);
```

* As an illustration of function intrans :

```
>> g = intrans (f, 'strech', mean2 (im2double(f)), 0.9);
>> figure, imshow (g) .
```

* m = mean2(A) - Computes the mean (average) value of the elements of matrix A.

* mean2 (im2double (f)) ⇒ was used directtly inside the function call, the result value was used for m. image f was Converted to double with range [0,1], So the mean would also be in this range, as required for input m,
The value E was determined interactively.

"function Changeclass"

```
function image = changeclass (class, varargin)
 % CHANGECLASS changes the storage class of an image.
 %   I2 = CHAGECLASS(CLASS, I);
 %   RGB2 = CHAGECLASS( CLASS, RGB);
 %   BW2 = CHAGECLASS (ClASS, BW);
 %   X2 = CHAGECLASS (CLASS, X, 'indexed');
   switch Class
      case 'uint8'
            image = im2uint8 ( varargin {:});
      Case 'uint16'
            image = im2uint16 ( varargin {:});
      case 'double'
            image = im2 double ( varargin {:});
   otherwise
            error (' Unsupported IPT Data Class.');
   end
```