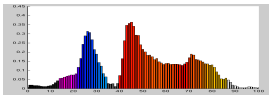
*Dr. Qadri Hamarsheh*



# Digital Image Processing (750474)

# Lecture 10

## Intensity Transformation Functions using Matlab

### Outline of the Lecture

- **Function "imadjust".**
- **Logarithmic and Contrast-Stretching Transformations.**
- **Matlab Example: Utility M-function for Intensity Transformations**

## Function "imadjust"

**Imadjust**- the basic IPT tool for intensity transformation of gray scale 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*** respectively.
- 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 `unit8`, and **65535** for `unit16`.
- 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 value in **f** to create **g**.
  - **gamma <1**: the mapping is weighted toward brighter output values.
  - **gamma = 1** (default): the mapping is linear>
  - **gamma > 1**: the mapping is weighted toward darker output values.

**Examples:**

- ✓ Obtaining the **negative image** using ***imadjust*** function.

```
>> g1 = imagjust (f, [0  1], [1   0]);
```

- ✓ Obtaining the **negative image** using IPT function ***"imcomplement"***

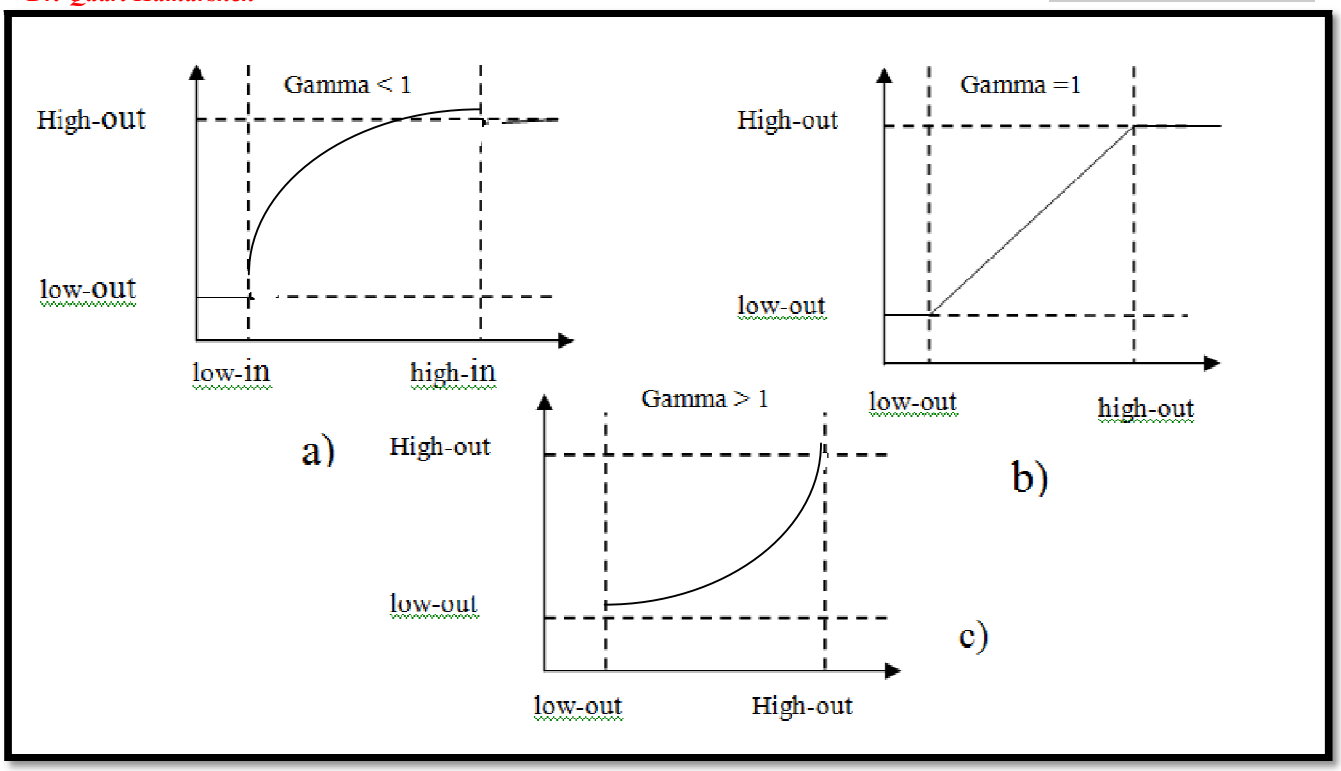
```
>> g= imcomplement (f);
```

- ✓ **Expanding** the gray scale region between 0.5 and 0.75 to the full [0.1] range.

```
>> g2 = imadjust (f, [0.5     0.7,5], [0  1]);
```

- ✓ **Compression** the low end and **expanding** the high end of the gray scale.

```
>> g3 = imadjust (f, [   ], [   ], 2);
```




## Logarithmic and Contrast-Stretching 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]**
  - im2unit8 brings them to the range **[0 255]**

**Contrast- Stretching Transformation Function:**

- The ***contrast- stretching*** transformation function has the form:

$$S = T(r) = \frac{1}{1 + \left(\frac{m}{r}\right)^E}$$

**r-** ***Input image.***

**s-** ***Output image***.

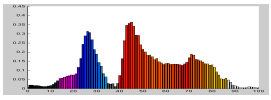
**E**- Controls the ***slope*** of the function.

- This equation is implemented in matlab for an entire image as:

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

**eps**- to prevent ***overflow*** if f has any **0** values

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



# Matlab Example: Utility M-function for Intensity Transformations

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

**Handling a variable number of inputs and/or outputs**

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

**n= nargin**

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

**n= nargout**

**Example:**

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

- 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.
```

- To write functions in which the number of input and/or outputs arguments is ***variable***, we use **varagrin** and **varargout**.

**Example:**

- Accepts a variable number of inputs into function **testhv3**.

```
function [m , n]= testhv3 (varargin)
```

- Returns a variable number of outputs from function **testhv4**.

```
function [varargout] = testhv4 (m, n, p)
```

- Function **testhv5** has one fixed input argument x, followed by a variable number of input arguments, similar comments apply to **varargout**. (it's acceptable to have a function in which both the number of input and output arguments is variable)

```
function [m, n] = testhv5 (x, varargin).
```

- When **varargin** or **varargout** are used: the matlab sets it to a **cell array**.

**Example:**

```
>> [m, n] = testhv5 (f, [0   0.5   1.5], A, 'label');
%f is an image, the second argument is row vector,
%A is a matrix, label is a string.
```

## Matlab code

**Write a function that computes the following transformation functions: negative, log, gamma and contrast stretching, in writing this function we use function "changeclass" which has the syntax:**

**g= changeclass (newclass, f)**

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

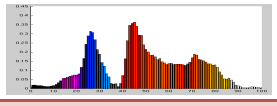
**Valid values for newclass are 'uint8', 'uint16' and 'double'.**

**Matlab code intran.m**

```
function g= intrans (f, varargin)
% INTRANS performs intensity (gray-level) transformation.
% G= INTRANS (f, 'neg') computes the negative of input image f.
% 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 'unit8' or 'uint16'
% if parameter CLASS is omitted, the output is of the
% same class as the input.
```

```
% G= INTRANS (f, 'gamma', GAM) performs a gamma-
% transformation on the input image using parameter
% GAM (a required input).
% G= INTRANS (f, 'strech' M, E) computes a contrast-
% stretching transformation using the expression 1./ (1+(m./ f+ eps)) .^E
% 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 image are converted to double first using im2double.
% for the 'log' transformation, double images are transformed
% 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 specified 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);
```



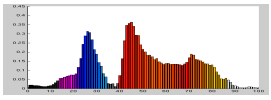
```
else % convert to double, regardless of class (f).
        f= im2double (f);
end;
% Determine the type of transformation specified.
method = varargin {1};
% Perform the intensity transformation specified.
switch method
case 'neg'
              g= imcomplement(f);
case 'log'
        if length (varargin) == 1
            c= 1;
       elseif 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;
   elseif length (varargin) ==3
         m = varargin {2};
          E = varargin {3};
   else
        error ('Incorrect number of inputs for the stretch 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 directly 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 = CHAGE CLASS (CLASS, X, 'indexed');
switch class
case 'uint8'
        image = im2uint8 (varargin {:});
case 'uint16'
        image = im2uint16 (varargin {:});
case 'double'
        image = im2double (varargin {:});
otherwise
        error ('unsupported IPT Data class.');
end
```