



Lab 4: Transformation functions and geometric transforms.

Objective:

The objective of this lab is:

- To apply Log transformation on images and visualize results.
- To apply power-law transform to correct gamma in images.
- To apply various geometric transformations (like translation, rotation, scaling, sheering and affine) on images and see their effects.

Theory:

Log Transform:

Applying a simple Logarithmic transformation on image can be used to process darker images to a relatively natural images (see Figure 1) while inverse logarithmic transform can also be used to transform saturated images to a natural ones.

In principal, a simple log transformation can be written as:

$$s = c \cdot \log(1 + r)$$

Where c is a scaling constant which can be calculated based on the maximum allowed output value (255 for an 8-bit image) and the maximum value $\max(I_{\text{input}}(i,j))$ present in the image:

$$c = \frac{255}{\log[1 + \max(I_{\text{input}}(i,j))]}$$

The effect of the logarithmic transform is to increase the dynamic range of the dark regions in an image and decrease the dynamic range in the light regions.

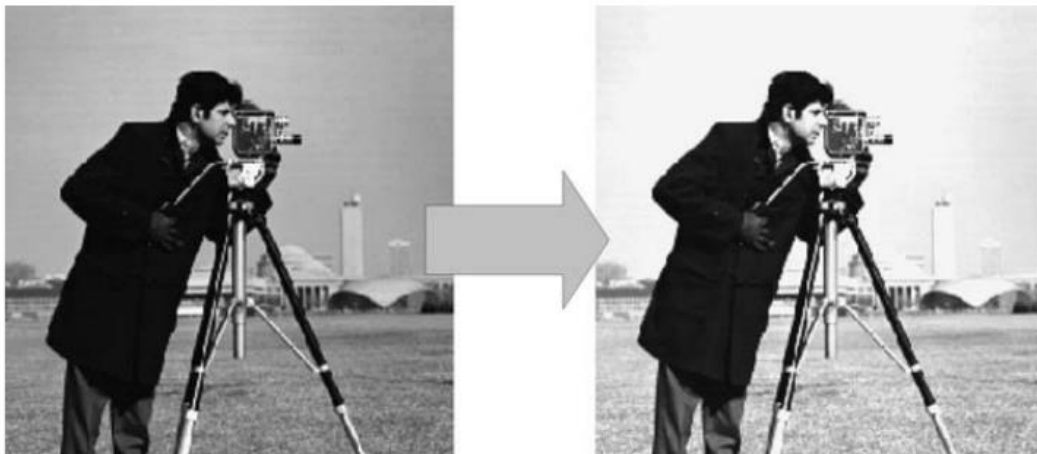


Figure 1: Application of Log transform.



Similarly, an inverse logarithmic transformation function will increase dynamic range in the light regions while decreasing the dynamic range in dark regions thus can be used to process saturated and washed out images.

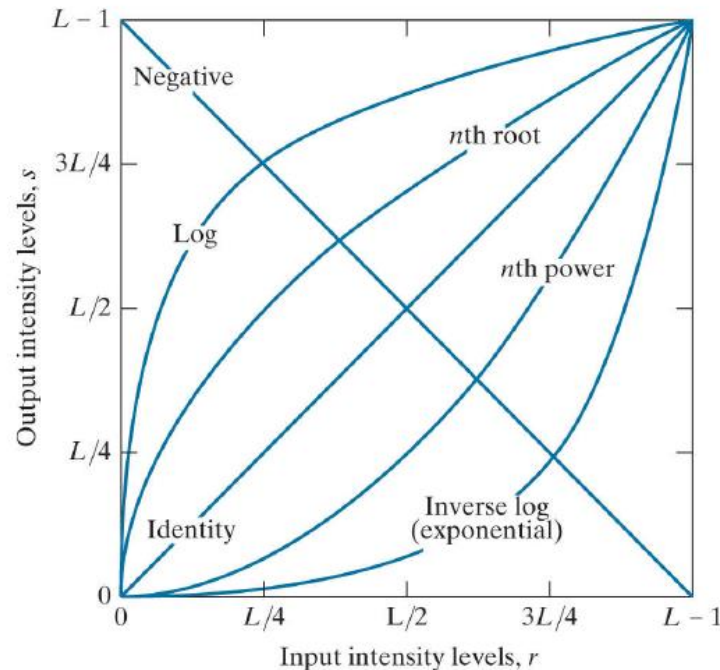


Fig 2: Some basic intensity transformation functions.

Power-Law transformations:

The n th power and n th root curves shown in fig. 2 can be given by the expression:

$$s = cr^\gamma$$

This transformation function is also called as gamma correction. For various values of γ different levels of enhancements can be obtained. This technique is quite commonly called as Gamma Correction. If you notice, different display monitors display images at different intensities and clarity. That means, every monitor has built-in gamma correction in it with certain gamma ranges and so a good monitor automatically corrects all the images displayed on it for the best contrast to give user the best experience.

The difference between the log-transformation function and the power-law functions is that using the power-law function a family of possible transformation curves can be obtained just by varying the λ .

These are the three basic image enhancement functions for grey scale images that can be applied easily for any type of image for better contrast and highlighting. Using the image negation formula given above, it is not necessary for the results to be mapped into the grey scale range $[0, L-1]$. Output of $L-1-r$ automatically falls in the range of $[0, L-1]$. But for the Log and Power-Law transformations resulting values are often quite distinctive, depending upon control parameters like λ and logarithmic scales. So the results of these values should be mapped back to the grey scale range to get a meaningful output image. For example, Log function $s = c \log(1 + r)$ results in 0 and 2.41 for r varying between 0 and 255, keeping $c=1$. So, the range $[0, 2.41]$ should be mapped to $[0, L-1]$ for getting a meaningful image.



Geometric Transformation:

Unlike intensity transformations where gray-levels of image are transformed, a geometric transformation such as translation can be applied on image facilitate both synthetic and realistic image morphing. In a geometric transform, image pixel location x and y are transformed to x' and y' .

Translation of an image can be expressed as adding an arbitrary value tx in x and ty in y , which can be conveniently written as:

$$\begin{aligned}x' &= x + tx \\ y' &= y + ty\end{aligned}$$

Applying the translation operation can result in moving an image from one location to another. For further simplicity we can write aforementioned equations in a matrix form as:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

In the case of translation, the values of the affine matrix can be found in Figure 3. Similarly other operations such as scaling, sheering and rotation can be performed on image for various digital image processing tasks.

NOTE: Opencv provide a convenient function (**cv2.warpAffine()**) which takes input image **I**, affine_matrix **A** and size of the output image (**width, height**) as arguments and after applying the transformation the output image is returned.

```
import cv2
import numpy as np

img = cv2.imread('messi5.jpg',0)
rows,cols = img.shape

M = np.float32([[1,0,100],[0,1,50]])
dst = cv2.warpAffine(img,M,(cols,rows))

cv2.imshow('img',dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```





National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science



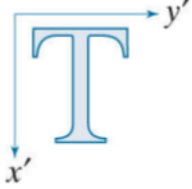
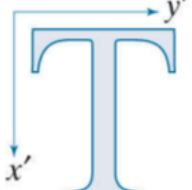
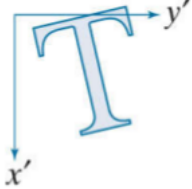
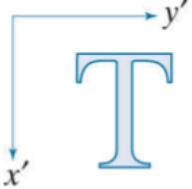
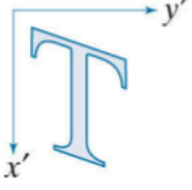
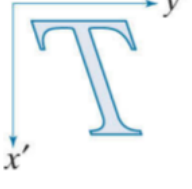
Transformation Name	Affine Matrix, A	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x' &= x \\ y' &= y \end{aligned}$	
Scaling/Reflection (For reflection, set one scaling factor to -1 and the other to 0)	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x' &= c_x x \\ y' &= c_y y \end{aligned}$	
Rotation (about the origin)	$\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x' &= x \cos \theta - y \sin \theta \\ y' &= x \sin \theta + y \cos \theta \end{aligned}$	
Translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \end{aligned}$	
Shear (vertical)	$\begin{bmatrix} 1 & s_v & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x' &= x + s_v y \\ y' &= y \end{aligned}$	
Shear (horizontal)	$\begin{bmatrix} 1 & 0 & 0 \\ s_h & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x' &= x \\ y' &= s_h x + y \end{aligned}$	

Fig 3: Geometric transformations in the form of affine matrix



Lab Tasks:

Task 1:

1. Use python notebook and opencv's imread function and load an image (e.g. dark.tif)
2. Visualize histogram by plotting the histogram.
3. Apply log transform on image and visualize the output histogram.
4. Try to experiment by changing the scaling constant c
5. Summarize your findings by providing a figure containing 4 plots (input image, histogram of input image, output image and histogram of output image)

Task 2:

- Read "aerial.tif" image in python notebook and store that in "img" variable.
- Read the documentation of **Interact** from [ipywidgets](https://ipywidgets.readthedocs.io/en/latest/interact.html) and create a slider named gamma.
- By using the value of gamma, apply a power-law transform on image and figure out which value of **gamma** can result in a contrast enhanced output.
- HINT: since you have to apply this transformation to each pixel, nested for loops including the power-law operation should do the trick.
- Summarize your findings by providing a figure containing 4 plots (input image, histogram of input image, output image and histogram of output image)

Task 3:

- Read "messi5.jpg" image and apply following transformations using cv2.warpAffine() function:
 - Translation
 - Rotation
 - Sheering in x
 - Sheering in y
 - Random affine transform
- Summarize your findings by providing a figure which contains images before and after geometric transformation.

Task 4:

- You may have noticed that in task 2 when using interact with some real-time processing, the system takes some time to process.
- This is usually caused by applying exponential mathematical operation withing nested for loops.
- You can use the concept of mapping to apply the transformation function in an efficient manner!
- Try out both techniques (i.e. power-law in nested for loops and mapping) by increasing the size of input image and calculate the time take by each.
- Conclude your findings in the form of graph in which x-axis should indicate the increasing size of image and y-axis should indicate time taken for processing.