

**CZ4003 Computer Vision**

**Laboratory 1**

**Point Processing + Spatial Filtering + Frequency Filtering + Imaging Geometry**

Mittal Madhav (U1822408H)

**Introduction**Computer vision is a field of artificial intelligence that trains computers to interpret and understand the visual world. Computer vision tasks include methods for acquiring, processing, analyzing and understanding (classifying) digital images.  
  
Lab 1 introduces us to processing of these digital images in order to remove unwanted noise through various processing and filtering techniques, and lab 2 introduces us to the classification of objects in images through different edge detection techniques.  
  
  
  
**Lab 1 Objectives :-**

* 1. Become familiar with the MATLAB and Image Processing Toolbox software package.
  2. Experiment with the point processing operations of contrast stretching and histogram equalization.
  3. Evaluate how different Gaussian and Median filters are suitable for noise removal.
  4. Become familiar with the frequency domain operations
  5. Understand imaging geometry.

**Experiments**

1. **Contrast Stretching**
   1. Inputting image, finding type of image, and converting to grayscale  
        
      ****
   2. Viewing the image   
        
      ****
   3. Checking the minimum and maximum intensities present in the image  
        
      ****Minimum intensity was found out to be 13  
      Maximum intensity was found out to be 204
   4. The values of a and b show that the image has poor contrast. Thus, we do contrast stretching on the given image.   
        
      Using the formula :   
        
        
        
      **  
      A double photo of a bus

      Description automatically generated A double photo of a bus

      Description automatically generated  
        
       Figure 1 : MRT Before and After Contrast Stretching**The minimum and maximum intensities for the MRT Photo obtained after contrast stretching were found to be 0 and 255 respectively.
   5. Displaying the contrasted image   
        
      ****
2. **Histogram Equalization**
   1. Displaying the image intensity histogram of P using 10 and 256 bins respectively.  
        
      **  
      Chart, histogram

      Description automatically generated Chart, histogram

      Description automatically generated  
        
       Figure 2 : Image Intensity Histograms with 10 and 256 bins respectively**  
        
      Bins represents the number of gray levels to be shown in a histogram. If there are N bins, all the pixels in the picture will be spread across these N bins.  
      As the number of bins increase, we get a more detailed and accurate histogram because of accurate representation of pixel grayscale intensities by the individual bins.
   2. Histogram equalization using 10 and 256 bins  
        
      **  
      Chart, histogram

      Description automatically generated Chart, histogram

      Description automatically generated  
        
       Fig 3 : Histogram Equalization with 10 and 256 bins respectively**Similarities : The same number of pixels (320 \* 443) are spread out in both the histograms. Both histograms are more equalized than their Image Intensity Histogram counterparts.  
      Differences : Since the histogram with 10 bins has only 10 bins to fill, thus it is more equalized.
   3. Rerunning histogram equalization to observe changes in output histograms.  
        
      The histogram remains the same, i.e, if you run a histogram equalization algorithm for the second time on the same image, it does not change the output. This is because in the first iteration, the pixels are mapped to their corresponding bins based on an algorithm, which when run again, will map them to the same bins.  
      This proves the idempotency of histogram equalization.
3. **Linear Spatial Filtering**
   1. Generating the Gaussian averaging filters and displaying them as 3-D graphs using mesh functions   
        
      **  
      Chart

      Description automatically generated Chart, radar chart

      Description automatically generated  
         
       Fig 4 : Gaussian Averaging Filters with X,Y = 5 and σ = 1 and 2 respectively**
   2. Downloading the NTU image with additive gaussian noise and viewing it.  
        
       A large brick building

      Description automatically generated
   3. Filtering the image using the gaussian filters created above and displaying the results.  
        
        
      A large building

      Description automatically generated A large building in the background

      Description automatically generated  
        
      **Fig 5 : NTU additive Gaussian noise image filtered using Gaussian Filters with σ= 1,2**If we increase the variance, the distribution will be more spread out, and the peak will be less spiky. That de-emphasizes the sharp gradient changes in the image. Thus if we increase the variance, the image will be blurrier.   
        
      It was thus concluded that as we increase the variance, the noise is more effectively removed, but the image gets more and more blurred. This is the trade-off. Therefore, filter with variance = 1 outputted a higher quality image, but with more noise, as compared to the filter with variance = 2.   
        
      Note : While using conv2, it was found out that the output image always had a black outline. After google-searching and looking at the image matrix, it was concluded that conv2 had a default setting of “full”, which results in the formation of a new increased size image, with very less border values, with eventually results in the formation of the black lines. Thus, we need to change the setting to “same”, so that only the central part of the convolution is returned, and not the whole image.
   4. Downloading the NTU image with additive speckle noise and viewing it.  
        
       A large brick building

      Description automatically generated
   5. Using the two gaussian filters from part (c) to remove the speckle noise, and the comparing the results  
        
        
        
        
      A large building in the background

      Description automatically generated A large building in the background

      Description automatically generated   
        
       **Fig 6 : NTU additive speckle noise image filtered using Gaussian Filters with σ = 1,2**  
      It was found out that the speckles were not completely removed, and thus the gaussian filter is not appropriate for removing additive speckled noise.
4. **Median Filtering**
   1. Filtering the image with Gaussian noise using the median and displaying the results.  
        
      A large brick building

      Description automatically generatedA large brick building

      Description automatically generated **Fig 7 : NTU gaussian noise image filtered using Median Filters with size = 3,5**It was observed that as **we increase the size of the median filter, the corners tend to get filtered out, whereas the edges are preserved**. Since the median value must actually be the value of one of the pixels in the neighborhood, the median filter does not create new unrealistic pixel values when the filter straddles an edge. For this reason the **median filter good at preserving sharp edges**.  
        
      In general, the median filter allows a great deal of high spatial frequency detail to pass while remaining very effective at removing noise on images where less than half of the pixels in a smoothing neighborhood have been effected. As a consequence of this, **median filtering is less effective at removing noise from images corrupted with gaussian noise, than gaussian filters**
   2. Filtering the image with Speckle noise using the median and displaying the results.  
        
      **  
      A large building

      Description automatically generated A large building

      Description automatically generated  
        
       Fig 8 : NTU speckle noise image filtered using Median Filters with size = 3,5**It can clearly be seen that the **median filter is better than the gaussian filter to remove noise from speckle images**. When the gaussian filter is applied, the edges are less apparent, as compared to when we use the median filter (as it preserves the edges).
5. **Suppressing Noise Interference Patterns**
   1. Downloading the image with periodic noise.

**A close up of a logo

Description automatically generated**

* 1. Obtaining the Fourier transform of the image, computing and displaying the power spectrum  
       
       
       
      **A screen shot of a computer

     Description automatically generated  
       
      Fig 9 : Periodic noise image power spectrum colormap**
  2. Redisplaying the power spectrum without fftshift and measuring the actual locations of peaks   
       
       
     **A screen shot of a computer

     Description automatically generated**  
     The peaks were found to be at (18,249) and (241, 10)
  3. Removing noise in the Fourier transform and computing and displaying the power spectrum  
       
       
      Chart

     Description automatically generated  
      **Fig 10 : Power Spectrum Colormap after removing the noise**
  4. Computing inverse Fourier transform after removing noise in frequency domain  
       
       
      **A close up of a logo

     Description automatically generated** A close up of a mans face

     Description automatically generated  
       
      **Fig 11 : Periodic noise image before and after filtering**If we want to improve the result, we can set the neighboring 10\*10 pixels to zero, which seems to remove some more noise, but not a lot.  
       
      A close up of a mans face

     Description automatically generated A picture containing photo, sitting, old, box

     Description automatically generated  
       
     **Fig 12 : Periodic noise image by removing 5 and 10 neighboring pixels respectively**  
       
     We can also iterate through the entire picture, pixel by pixel, and check for brightness in each iteration. If the brightness is more than a set threshold value, then we can reduce the corresponding Fourier value to zero.
  5. Downloading the caged-primate photo and attempting to remove the bars  
     1. Downloading the image  
          
         A close up of a rug

        Description automatically generated  
           
         **Figure 13 : Caged Primate**
     2. Displaying the corresponding power spectrum  
          
         **A picture containing chart

        Description automatically generated  
          
         Fig 14 : Power Spectrum Colormap for the primate**
     3. Method to remove the bars :-
        1. Get the 8 peak values using ginput(8)
        2. Store individual x and y coordinates into separate arrays
        3. Round these values
        4. Removing noise in power spectrum using the x, y arrays

After computing this, the contents of x\_arr and y\_arr were found to be  
x\_arr = [11, 6, 17, 22, 247, 253, 243, 238]

y\_arr = [235, 247, 248, 239, 22, 11, 10, 20]

This means that the peaks were found to be (11, 235) (6, 247) (17, 248) (22, 239) (247, 22) (253, 11) (243, 10) (238, 20)  
  
Note that these values are not perfectly accurate, since the peaks weren’t very clear and are subject to user’s vision.

* + 1. Displaying the power spectrum after removing noise  
         
        A screen shot of a computer

       Description automatically generated  
         
        **Fig 15 : Power Spectrum Colormap after removing the noise**
    2. Computing IFT after removing noise in frequency domain and displaying the result  
        A close up of a rug

       Description automatically generated A picture containing bird

       Description automatically generated  
         
        **Fig 16 : Caged Primate Image before and after filtering**

1. **Undoing Perspective Distortion of Planar Surface**
   1. Downloading the photo of the book  
        
       A close up of a card

      Description automatically generated
   2. Using ginput(4) to obtain image corner coordinates   
        
      

The corners were measured in the following order :   
[top left corner, top right corner, bottom left corner, bottom right corner]

* 1. Setting up the matrices and doing the required computation to achieve projective transformation   
       
       
     Yes, the transformation gives back the values of 4 corners.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 210 | 0 | 210 |
| -5.96968964055254e15 | -1.07332358601168e14 | 295.000000000000 | 295 |
| 1 | 1 | 1 | 1 |

**Table 1 : Values inside variable “w”**

* 1. Warping the image  
       
     
  2. Displaying the image  
       
      A close up of a card

     Description automatically generated Graphical user interface, application

     Description automatically generated  
       
      **Fig 17 : Book Image before and after projective transformation**  
     The results were better than expected. However, the top part of the image is blurred. This might be because in the original image, the top part if further away, and hence the text and top part of the image has lower resolution, as compared to the lower part.   
       
     The results might improve if we could either increase the resolution of the original image, or maybe take more than 4 points to increase the accuracy of the projective transformation system.