

**CZ4003 Computer Vision**

**Laboratory 2**

**Edge Detection + Hough Transform + 3D Stereo Vision + Spatial Pyramid Matching**

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 introduced us to processing of these digital images in order to remove unwanted noise through various processing and filtering techniques.

Lab 2 introduces us to various edge detection techniques including the Sobel filters and canny edge detector, followed by an edge linking algorithm called the Hough transform. It is also required for us to implement stereo vision, which aims at computing the relative depth of a 3D point from the stereo cameras.   
  
  
  
**Lab 2 Objectives :-**

1. Explore different edge detection methods.
2. Employ the Hough Transform to recover strong lines in the image.
3. Experiment with pixel sum-of-squares difference (SSD) to find a template match within a larger image. Estimate disparity maps via SSD computation.
4. Optionally, compare the bag-of-words method with spatial pyramid matching (SPM) on the benchmark Caltech-101 dataset. Understand imaging geometry.

**Experiments**

1. **Edge Detection**
   1. **Download `macritchie.jpg’ from edveNTUre and convert the image to grayscale. Display the image.**   
        
       A vintage photo of a group of people in a field

      Description automatically generated  
        
       **Fig 1 : Original macritchie picture**
   2. **Create 3x3 horizontal and vertical Sobel masks and filter the image using conv2. Display the edge-filtered images. What happens to edges which are not strictly vertical nor horizontal, i.e. diagonal?  
        
        
      A picture containing person

      Description automatically generated A picture containing tree, grass

      Description automatically generated  
        
       Fig 2 : Macritchie picture with horizontal (left) and vertical (right )sobel filters**It can be seen from Fig 2 that horizontal and vertical sobel filters are good at detecting horizontal and vertical edges respecitvely. However, they cannot detect the diagonal edges. Only some of the diagonal edges are detected, and this may be explained by the fact that these edges may be constituted of smaller horizontal or vertical edges.
   3. **Generate a combined edge image by squaring (i.e. .^2) the horizontal and vertical edge images and adding the squared images. Suggest a reason why a squaring operation is carried out.  
        
      A close up of a logo

      Description automatically generated A picture containing photo, standing

      Description automatically generated**

**Fig 2 : Macritchie picture – squared and added (left), square rooted (right)**

Since sobel masks can output a negative value, the squaring operation returns the magnitude of the gradient. The adding and squaring operation allows us to detect diagonal edges as well. However, due to this squaring and adding of pixels, the pixel values gets very high, resulting in a very bright image. Thus, a rooting operating is done to get a clearer image.

* 1. **Threshold the edge image E at value t by >> Et = E>t; This creates a binary image. Try different threshold values and display the binary edge images. What are the advantages and disadvantages of using different thresholds?  
       
       
     Qr code

     Description automatically generated  
       
      Fig 3 : Macritchie picture with increasing threshold values**It can be seen from the above figure that we increase the threshold, more noise gets filtered, but also some real edges are not detected due to the high threshold value. Similarly, if the threshold value is very low, then all the noise is not filtered, but the important edges are also not missed out.   
       
     Thus, the correct threshold value must be found out keeping this trade-off in mind.
  2. **Recompute the edge image using the more advanced Canny edge detection algorithm with tl=0.04, th=0.1, sigma=1.0  
       
       
      A close up of a logo

     Description automatically generated  
       
      Fig 4 : Macritchie picture with canny edge detection  
       
       
     Try different values of sigma ranging from 1.0 to 5.0 and determine the effect on the edge images. What do you see and can you give an explanation for why this occurs? Discuss how different sigma are suitable for (a) noisy edgel removal, and (b) location accuracy of edgels.  
       
       
      Qr code

     Description automatically generated  
       
      Fig 5 : Macritchie picture with sigma values of 2, 3, 4, 5 respectively**As we can see from the above figure, as the value of sigma increases, more and more edgels get filtered out. For low sigma values, the output image contains most of the important edgels, however the filter is also unable to filter out the noise. For higher sigma values, the filter filters out most of the noise, but some important edgels are also removed in the process.   
       
     The sigma denotes the standard deviation of the Gaussian kernel which is convolved with the image. A higher sigma means that more pixels are contributing to the edge at the current location, which leads to a lower precision and land thus lower location accuracy. For a smaller sigma, fewer pixels contribute to the edge detection at a location, which means that edges are detected a lot more, sometimes even due to noise.  
       
       
     **Try raising and lowering the value of tl. What does this do? How does this relate to your knowledge of the Canny algorithm?**Qr code

     Description automatically generated **Fig 6 : Macritchie picture with tl values of 0.0001, 0.05, 0.075, 0.0999 respectively**  
     tl is the lower bound of hysterisis thresholding. As we can see in the above figure, as we increase the tl value, lesser and lesser edgels are detected. This is because the Canny edge detector filters out any edgels that have magnitude smaller than tl.   
     Lower threshold values thus allow more edgels to contribute to an edge, leading to more complete edges, but they are unable to remove noise. Higher threshold values only let higher edgels contribute to edges, which decreases noise but also break up edges.

1. **Line Finding using Hough Transform**  
   1. **Reuse the edge image computed via the Canny algorithm with sigma=1.0.  
        
        
       A close up of a logo

      Description automatically generated  
        
       Fig 7 : macritchie image with canny edge detection with sigma = 1**
   2. **Explain why the Radon and Hough transforms are equivalent in this case. When are they different?**Radon and Hough transforms are equivalent in this case because they both transform points in the spatial domain to equivalent points in the theta domain.  
        
      The two transforms are different because the Radon transform takes into account the intensity of the pixel. This does not matter in this experiment because we are using a binary thresholded image, which have just two intensities. However, the output would be different if the image has different intensity pixels.  
        
      **Display H as an image. The Hough transform will have horizontal bins of angles corresponding to 0-179 degrees, and vertical bins of radial distance in pixels as captured in xp.  
        
       A picture containing computer

      Description automatically generated  
       Fig 8 : Radon transformed image**
   3. **Find the location of the maximum pixel intensity in the Hough image in the form of [theta, radius]. These are the parameters corresponding to the line in the image with the strongest edge support.  
        
      A picture containing graphical user interface

      Description automatically generated**
   4. **Derive the equations to convert the [theta, radius] line representation to the normal line equation form Ax + By = C in image coordinates.******
   5. **Based on the equation of the line Ax+By = C that you obtained, compute yl and yr values for corresponding xl = 0 and xr = width of image - 1.******
   6. **Display the original ‘macritchie.jpg’ image. Superimpose your estimated line by >> line([xl xr], [yl yr])  
        
      A vintage photo of a group of people in a field

      Description automatically generated A group of people on a field

      Description automatically generated  
        
       Fig 9 : Original image vs image with Image superimposed with estimated strongest edge  
      Does the line match up with the edge of the running path? What are, if any, sources of errors? Can you suggest ways of improving the estimation?**It can be seen form the above figure that the estimated strongest edge almost perfectly aligns with the line of the path. This means that the path line is the strongest edge in the image.   
        
      The line is still not perfect. A source of error could be due to noisy edgels, which could be removed by stricter thresholding. It is also possible that the running path is not strictly a straight line, so a simple linear function may not be able to compute the perfect strongest edge.
2. **3D Stereo**
   1. **Write the disparity map algorithm as a MATLAB function script******
   2. **Download the synthetic stereo pair images of ‘corridorl.jpg’ and ‘corridorr.jpg’.   
        
       A picture containing indoor, floor, sitting, room

      Description automatically generated A picture containing floor, computer, table, room

      Description automatically generated  
       Fig 10 : left and right corridor images**
   3. **Run your algorithm on the two images to obtain a disparity map D, and see the results  
        
       A picture containing jack, photo, sitting, person

      Description automatically generated A close up of a door

      Description automatically generated  
       Fig 11 : Actual disparity map vs experimental disparity map   
        
      Comment on how the quality of the disparities computed varies with the corresponding local image structure.**The quality of the disparity map for the corridor images is good. It is comparable to the original disparity map. It can be seen that as we go away in the image, the brightness becomes darker. The only problem with the obtained disparity map is the center of the image, which is the farthest point. It is supposed to be bright, but is dark instead. This could be because it is a large region with similar values in both the input images, so the algorithm is not able to differentiate.   
        
      The overall quality is good and SSD matching is working well because there are no large regions having similar colors, the contrast of the image is very high. The only region that had a large same colored region (the center) was mapped incorrectly by the algorithm.
   4. **Rerun your algorithm on the real images of ‘triclops-i2l.jpg’ and triclops-i2r.jpg’. Again you may refer to ‘triclops-id.jpg’ for expected quality. How does the image structure of the stereo images affect the accuracy of the estimated disparities?  
        
      A close up of a brick building

      Description automatically generated A large building in the background

      Description automatically generated  
       Fig 12 : left and right triclops images  
        
        
      A picture containing covered, person, cat, standing

      Description automatically generated A picture containing photo, table, covered, counter

      Description automatically generated  
       Fig 12 : Actual disparity map vs experimental disparity map**Since the contrast of this image is not that high, that is, the image contains large areas with similar pixel intensity, thus the disparity map is expected to have errors, which is true as it can be seen in the above images.
3. **[OPTIONAL] Implementing Bag of Features and Spatial Pyramidal Matching and comparing their classification accuracies**Done using anaconda, explanations and code included in the ipynb file submitted.