Computer Vision -- Lab 2

1. OBJECTIVES

This laboratory aims to provide further exposure to advanced image processing and computer vision techniques in MATLAB. In this laboratory you will:

(a) Explore different edge detection methods;

(b) Employ the Hough Transform to recover strong lines in the image;

(c) Experiment with pixel sum-of-squares difference (SSD) to find a template match within a larger image. Estimate disparity maps via SSD computation;

(d) Optionally, compare the bag-of-words method with spatial parymid matching (SPM) on the benchmark Caltech-101 dataset.

2. INTRODUCTION

2.1 Edge Detection

Edge detection is used to extract discontinuities or sharp gradients in the image. There are many different methods available, ranging from the basic Sobel operators with filter masks of

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

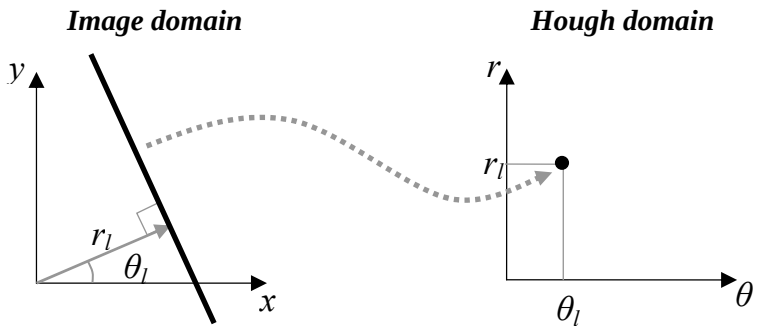
to the more complex Canny algorithm which involves separate steps of Gaussian derivative filtering, non-maximal suppression and hysteresis thresholding. Each edge point extracted is also known as an *edge element*, or “edgel”.

2.2 Hough Transform

Edge detection may not give you all the information that you want. While the above methods extract local edges, they are unable to determine the presence of long, consistent lines which have broken local edges.

Hough transform is a transformation which maps a straight line in the image domain to a single point in the Hough domain, based on the parameter mapping shown in the diagram below. Hence it can also be shown that a single point in image domain is mapped to a *sinusoidal curve* in Hough domain.

When all edgels in image space are mapped to Hough space, edgels that lie on a straight line will each map to sinusoidal curves that *intersect* at a single point in Hough space. These intersection points will have large values after the Hough transform, and can be easily detected. The coordinates of an intersection point are parameters that define the line in image space.



2.3 Pixel Intensity Sum-of-Squares Difference (SSD) and 3D Stereo Vision

The pixel intensity sum-of-squares difference (SSD) between a small image patch *T* and a large image *I* at the (*x*,*y*) location given by

$$S(x,y)=\sum_{j=0}^{M}\sum_{k=0}^{N}\left(I(x+j,y+k)-T(j,k)\right)^2$$

By computing the SSD at *all locations* of the image *I*, we can generate a SSD image *S*(*x,y*). This allows the location in the image that best matches the image patch *T* to be found.

Evaluating the above equation directly can be very time consuming. An alternative approach is to decompose the equation into 3 parts

$$S(x,y)=\sum_{j=0}^{M}\sum_{k=0}^{N}I^2(x+j,y+k).1+\sum_{j=0}^{M}\sum_{k=0}^{N}T^2(j,k)-2\sum_{j=0}^{M}\sum_{k=0}^{N}I(x+j,y+k)T(j,k)$$

The second term is constant with respect to different parts of the image, while the first and third terms can be expressed as convolution. Convolution on large images can be efficiently computed via FFTs.

In stereo vision, the goal is to compute the relative depth of a 3D point from the stereo cameras. If rectified images (i.e. where projected 2D points from the same 3D point have the same *y* coordinate in different images -- they lie on the same scanline) are used, we can express depth in terms of disparity. Disparity is the difference between the *x* coordinate of the projections in two images, and is inversely proportional to depth.

Given two images, stereo vision allows the computation of a *disparity map*. A disparity map is simply an array containing the disparities or depths that are associated with every pixel in one image (typically the left image).

3. EXPERIMENT

3.1 Edge Detection

a) Download `macritchie.jpg' from edveNTUre and convert the image to grayscale. Display the image.

b) 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?

c) 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.

d) 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?

e) Recompute the edge image using the more advanced Canny edge detection algorithm with tl=0.04, th=0.1, sigma=1.0

```
>> E = edge(I,'canny',[tl th],sigma);
```

This generates a binary image without the need for thresholding.

(i) 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.

(ii) Try raising and lowering the value of tl. What does this do? How does this relate to your knowledge of the Canny algorithm?

3.2 Line Finding using Hough Transform

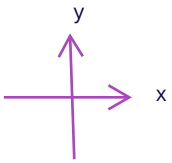
In the section, the goal is to extract the long edge of the path in the `macritchie.jpg' image as a consistent line using the Hough transform.

a) Reuse the edge image computed via the Canny algorithm with sigma=1.0.

b) As there is no function available to compute the Hough transform in MATLAB, we will use the Radon transform, which for binary images is equivalent to the Hough transform. Read the help manual on Radon transform, and explain why the transforms are equivalent in this case. When are they different?

```
>> [H, xp] = radon(E);
```

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. The transform is taken with respect to a Cartesian coordinate system where the origin is located at the centre of the image, and the *x*-axis pointing right and the *y*-axis pointing up.



c) 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.

d) Derive the equations to convert the [theta, radius] line representation to the normal line equation form $Ax + By = C$ *in image coordinates*. Show that A and B can be obtained via

```
>> [A, B] = pol2cart(theta*pi/180, radius);
>> B = -B;
```

B needs to be negated because the y-axis is pointing downwards for image coordinates.

Find C. Reminder: the Hough transform is done with respect to an origin at the centre of the image, and you will need to convert back to image coordinates where the origin is in the top-left corner of the image.

e) 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.

f) Display the original 'macritchie.jpg' image. Superimpose your estimated line by

>> line([xl xr], [yl yr]);

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?

3.3 **3D Stereo**

This is a fairly substantial section as you will need to write a MATLAB function script to compute disparity images (or *maps*) for pairs of rectified stereo images Pl and Pr. The disparity map is inversely proportional to the depth map which gives the distance of various points in the scene from the camera.

Estimating Disparity Maps

The overview of the algorithm is:

for *each* pixel in Pl,

i. Extract a template comprising the 11x11 neighbourhood region around that pixel.

ii. Using the template, carry out SSD matching in Pr, but *only along the same scanline*. The disparity is given by

$$d(x_l, y_l) = x_l - \hat{x}_r$$

where x_l and y_l are the relevant pixel coordinates in Pl, and $\hat{x}_r$ is the SSD matched pixel's *x*-coordinate in Pr. You should also constrain your horizontal search to small values of disparity (<15).
Noted that you may use **conv2**, **ones** and **rot90** functions (may be more than once) to compute the SSD matching between the template and the input image. Refer to the equation in section 2.3 for help.

iii. Input the disparity into the disparity map with the same Pl pixel coordinates.

a) Write the disparity map algorithm as a MATLAB function script which takes two arguments of left and right images, and 2 arguments specifying the template dimensions. It should return the disparity map. Try and minimize the use of **for** loops, instead relying on the vector / matrix processing functions.

b) Download the synthetic stereo pair images of 'corridorl.jpg' and 'corridorr.jpg', converting both to grayscale.

c) Run your algorithm on the two images to obtain a disparity map D, and see the results via

>> imshow(-D,[-15 15]);

The results should show the nearer points as bright and the further points as dark. The expected quality of the image should be similar to `corridor_disp.jpg' which you can view for reference.

Comment on how the quality of the disparities computed varies with the corresponding local image structure.

d) 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?

3.4 OPTIONAL

Attempting the optional section will earn you extra credit. However, this is not necessary to achieve a decent coursework grade.

You will need to implement the algorithm in the CVPR 2006 paper entiled "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories". You will need to use the benchmark Caltech-101 dataset and compare the classification results of Spatial Pyramid Matching (SPM) and the bag-of-words (BoW) method as in Table 2 of the paper by following the experimental setting in Section 5.2 of the paper.

4 REPORT

Please complete all questions and write / print all source code. You are also encouraged to print the relevant images and graphs for inclusion in your logbook.

5 REFERENCES

1. Gonzalez, R. C. and Woods, R. E., Digital Image Processing, Addison-Wesley, 1992.

2. K. R. Castleman, Digital Image Processing, Prentice-Hall, 1996

3. S. E. Umbaugh, Computer Vision and Image Processing, Prentice-Hall, 1998

4. Anil K Jain, Fundamentals of Digital Image Processing, Prentice-Hall, 1989

5. Getting Started with MATLAB. The MathWork, Inc., Natick, MA, 1984-1997.

6. Using MATLAB. The MathWork, Inc., Natick, MA, 1984-1998.

7. Using MATLAB Graphics. The MathWork, Inc., Natick, MA, 1984-1996.

8. Image Processing Toolbox User's Guide, The MathWork, Inc., Natick, MA, 1993-1997.

9. Signal Processing Toolbox User's Guide, The MathWork, Inc., Natick, MA, 1988-1996.