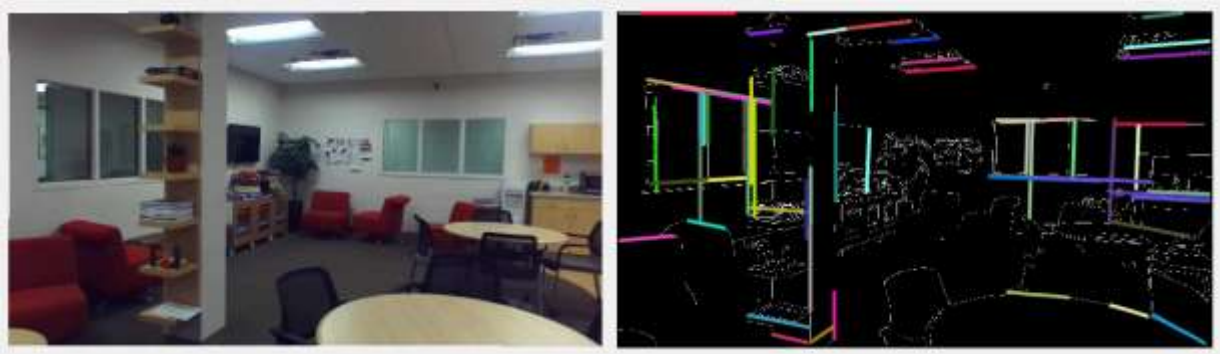
CS 6320 Computer Vision

Homework 1 (Due Date – January 18)

This assignment will consist of some basic image processing operations. We are also providing some sample outputs and pseudocode examples since this is your first homework.

1. Use canny edge detection (Try 'help edge' in Matlab) to extract edges (i.e., 2D points denoting changes in intensity values) from an image. Try to fit line segments to edge using RANSAC, a favorite technique for robust model fitting in computer vision applications. Using two 2D points (x1, y1) and (x2, y2), we can obtain a line equation of the form ax+by+c = 0. The basic idea is line fitting is simple: We randomly sample two edge pixels and compute a line equation that passes through these two points. For the chosen line equation, we check the total number of edge pixels that lie within a small threshold distance (say 1 pixel). Based on the number of points that lie on the chosen line equation, we decide to treat it as an image line or not. We iteratively find all the line segments in the image based on some threshold (i.e., each line should have at least 50 edge pixels). After finding every line, we remove all the edge points lying on this line before finding the next line. For this assignment, show the results on the three provided images. Please note that there are still research papers that are published on line detection. This is a naïve idea and will not work robustly for all images. For example, we show an image and the detected lines below:



Left: Original Image Right: A few detected line segments shown in different random colors.

Pseudocode

Input: Image

Output: Image with different line segments marked in different colors

1. Initialize LINE_SET = {}, ITERS = 0, TOTAL_NO_ITERS = 10000, MAX_PAIR_DISTANCE = 100, MIN_POINTLINE_DISTANCE = 1, MIN_LINE_PIXEL_NUM= 50.
2. Extract canny edge pixels from the given input image I. Let us denote the edge pixels by a set of 2D coordinates given by the EDGE_SET={(x1,y1),(x2,y2),…,(xn,yn)}.
3. ITERS = ITERS + 1; Select a random edge pixel (i.e. P(xi,yi)) from the EDGE_SET. Select another random edge pixel Q(xj,yj) that is within MAX_PAIR_DISTANCE from the point P. Fit a 2D line PQ passing through the points P and Q in the form ax+by+c = 0.
4. For all the points in the EDGE_SET, identify the edge points that are within MIN_POINTLINE_DISTANCE from the line PQ. Construct a set INLIER and insert all the edge points that satisfy the distance constraint. If INLIER has a sufficient number of edge pixels (say MIN_LINE_PIXEL_NUM), then you treat this as a line. Let the inlier set be given as follows: INLIER = {{xa,ya},{xb,yb},…{xm,ym}}. Now add the INLIER to the set LINE_SET. You can think of the LINE_SET as a "set of sets" consisting of several INLIER sets denoting individual lines. Now remove the INLIER from the EDGE_SET.
5. If (ITERS < TOTAL_NO_ITERS) GOTO step 3.
6. Display the line segments by plotting the points in each of the INLIER set from the LINE_SET using a different random color.

[30 Points]

2. Write a simple sky segmentation using six thresholds for specifying the upper and lower values for each of the 3 RGB components. If a pixel color values lie within the specified values, then it is considered to be a sky pixel. Note that this is a simple segmentation routine and the results are not expected to be perfect. In the following example, we define a range for Red to be between 0 and 100, Green to be between 1 and 150, and Blue to be between 100 and 255. Please show the sky segmentation for the provided images. Some of the images are provided from the Middlebury dataset: http://vision.middlebury.edu/stereo/data/

Left: Original Image, Right: Segmentation mask for the sky (White corresponds to Sky, and Black corresponds to non-sky region).

Pseudocode

Input: original image

Output: Segmented image showing sky and non-sky region

1. Let the input image be given by I. Initialize the color range for each component: R_MIN = 0, R_MAX = 100, G_MIN = 1, G_MAX = 150, B_MIN = 100, and B_MAX = 255;
2. Use a new output image SEGMENT that has the same dimensions as the input image.
3. Loop over every pixel in the image and if the three components (R, G, B) lie within the specified range, treat it as the sky and give White color (255,255,255). Otherwise, treat it as non-sky and give black color (0,0,0).
4. Output the SEGMENT image.

[20 Points]

3. Write a simple stereo matching program to compute the disparity map. The two images in a stereo pair are usually referred to as "left" and "right" images. The images are said to be rectified if every pixel in the left image has a matching pixel in the right image at the same height. In a typical stereo pair, a pixel p(x,y) in the left image usually has a matching pixel q(x',y) in the right image. Note that the 'y' coordinate is the same in both the left and right images. Every pixel in the left image gets shifted by a small value 'd' on the right image in the following manner: x' = x – d. Here 'd' is the positive disparity value for the pixel p(x,y) in the left image. Note that a pixel that is further away from the camera will have a small disparity value. A pixel that is very close to the camera will have a high disparity value. In other words, the depth "D" of a pixel is inversely proportional to the disparity 'd'. To match a pixel p(x,y) with another pixel q(x',y) we use normalized cross-correlation (NCC). The expression for NCC is given below for two square matrices A and B with elements a_{ij} and b_{ij}.

$$NCC(A,B) = \frac{\sum_{ij} a_{ij}.b_{ij}}{\sqrt{\sum a_{ij}^2}\sqrt{\sum b_{ij}^2}}$$

Note that two pixels p(x,y) and q(x',y) can be considered as a perfect match if the NCC score is 1. If the NCC score is 0, then the two pixels form the worst possible match. For every pixel p(x,y) we take a window of size, say 5x5, centered at p(x,y) and treat this as the matrix A. For every pixel q(x',y) we do the same and obtain a window centered at q and call it B. We iterate over different values of x'=x-d, where d = {1,2,3,…,50} and compute the d that gives the highest NCC score for matrices A and B. The goal is to compute a disparity image where we have the best disparity 'd' value for every pixel in the left image. You need to ignore disparity computation at the boundary so that your matrices A and B do not cross the boundaries. Show the stereo reconstruction for the stereo pairs that are provided. Note that this is a simple and naïve stereo matching method and the results are not expected to be perfect.

Left: left image, Middle: right image, Right: Disparity map (white corresponds to large disparity and darker gray regions correspond to small disparity values). We searched for disparity values ranging from 1 to 50 for each pixel.

Pseudocode

Input: Left and Right images
Output: Disparity image

1. DISPARITY_RANGE = 50, WIN_SIZE = 5, EXTEND = (WIN_SIZE-1)/2;
2. Let left and right be the rectified input images with the same dimensions. Ny = y-dimension of the image, Nx = x-dimension of the image.
3. Convert the two images to grayscale images: Ileft = rgb2gray (left), Iright = rgb2gray(right).
4. Initialize a disparity image DISPARITY that has the same dimensions as Ileft or Iright.
5. For y = 1 to Ny

```
{
        For x = 1 to Nx
        {
                bestDisparity = 0;
                bestNCC = 0; // lowest NCC score
                For disp = 1 to DISPARITY_RANGE
                {
                        Patch1 = Ileft(y-EXTEND:y+EXTEND,x-EXTEND:x+EXTEND)
                        Patch2 = Iright((y-EXTEND:y+EXTEND,x-disp-EXTEND:x-disp +EXTEND)
                        currNCC = NCC(Patch1,Patch2)
                        if (currNCC > bestNCC)
                        {
                                bestNCC = currNCC;
                                bestDisparity = disp;
                        }
                }
        DISPARITY(y,x) = bestDisparity;
        }
}
```

6. Output the disparity image DISPARITY.

[35 Points]

4. Download the visual SFM software (http://ccwu.me/vsfm/). Create a 3D model using at least 25 images. The images can be obtained from the following sources:

   • The images can be captured using your phone or any digital camera.
   • The images can also be downloaded from the web (Flickr, Google images).

   You can search for “visualsfm models” in YouTube to see some examples of 3D models. Try to be creative and submit models of interesting objects. For example, you can try reconstructing Submit the 3D model in PLY format, which can be visualized using meshlab (http://www.meshlab.net/) [20 points].

   [15 Points]