1 Line Extraction Techniques

Basically, there are three main problems are being faced when performing line extraction and segmentation in an unknown environment [1].

1. How many lines are there?
2. Which points belong to in which line?
3. How to estimate the line model?

1.1 Hough Transformation

The Hough transform (HT) can be used to detect most of the parametric curves in which parametric equations are known such as lines, circles, etc. One of the main advantages of this is that it performs well under partial occlusion while having some noise components.

1. What is the equation for representing a line in cartesian coordinate space where a is the slope and b is the intercept?
2. What happens if we the change space x-y into a-b space?
3. Can you find any relation between what you define for line in cartesian coordinate space and polar coordinate space?

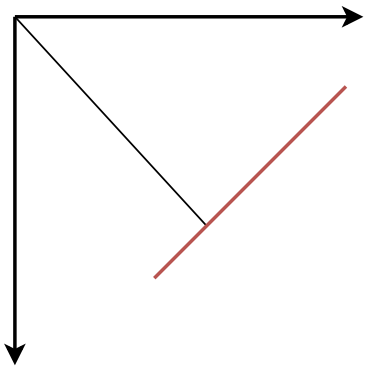

Figure 1: Polar coordinate representation of a line

Polar representation of a line can be defined as below.

$$xcos(\theta) + ysin(\theta) = \rho \tag{1}$$

where θ, ρ x, and y will be defined during your lab time in Fig. 1.

4. What can you say about the covariance matrix ($cov(\rho, \theta)$) of line in the polar coordinate space?
5. Let's try on this example? `https://docs.opencv.org/3.4.0/d9/db0/tutorial_hough_lines.html`
6. Can you try to sum up the basic steps of the Hough transformation?

1.2 Split-and-Merge Algorithm

Algorithm 1 Split-and-Merge Algorithm

1: **procedure** EXTRACTLINES(($input\_map$))
2: Initialize all the points (N) in the given matrix into one set (s)
3: Fit a line for set s
4: Find the most distance point to the line
5: If the distance to a line is higher than the predefined threshold split set s into two subsets.
6: Apply same logic for consecutive subsets.
7: Finally, merge all the collinear segments

As state in the Algo 01, for distance estimation form most distance point you may use any method such as perpendicular distance, etc.

1.3 Line Regression Algorithm

Algorithm 2 Line Regression Algorithm

1: **procedure** EXTRACTLINES(($input\_map$))
2: Initialize the $input\_map$ into sliding window of size N_f
3: Fit a line to every consecutive point
4: Compute a line fidelity array, each is the Mahalanobis distance between every 2 adjacent windows
5: If the distance to a line is less than the predefined threshold merges them into one line after fitting the whole segment
6: Apply same logic for consecutive windows.

In the Algo. 02, Mahalanobis distance can be defined as bellow,
Let's say line segment 1, $X_1 = [x_1, x_2, ..., x_n]$ and line segment 2, $Y_2 = [y_1, y_2, ..., y_m]$ where each line segment contains difference number of points n and m respectively for X_1 and Y_1. Thus, Mahalanobis distance between X_1 and Y_1 is given as

$$distance = \sqrt{(X_1 - \bar{X_1})^T C^{-1} (Y_1 - \bar{Y_1})} \tag{2}$$

where C is the cross-covariance matrix between X_1 and Y_1 which can be defined as

$$C = cov(X_1, Y_1) = E(X_1 - \bar{X_1})E(Y_1 - \bar{Y_1}) \tag{3}$$

2 Similarity Measurements

Let's say you have two feature set x and y as follow, $x = [2000, 2, 3456, 1, 0]$ and $y = [2000, 3, 3400, 3, 0]$. If it is required to get similarity between x and y as a percentage how you are going to calculate it?

$$distance = \frac{\Sigma_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma_{i=1}^{n}(x_i - \bar{x})^2 (y_i - \bar{y})^2}} \tag{4}$$

3 Edge Detection

Significant local changes of intensity in an image is called as edges. Geometric changes such as object or surface boundaries or non-geometric changes such as specularity, shadows and inter-reflection are the main causes for intensity changes. Mainly, there are two types of techniques are being used for edge detection either using derivative and/or the gradient.

3.1 Edge Detection Based on Derivative

First or the second derivative can use based on how pixel intensities are changed throughout the image. The first derivative is used to detect local maxima or minima whereas zero-crossing points are detected when using the second derivative.

3.1.1 Computing the First Derivative

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{5}$$

3.1.2 Computing the Second Derivative

$$f''(x) = \lim_{h \to 0} \frac{f'(x+h) - f'(x)}{h} \simeq f'(x+h) - f'(x) \tag{6}$$

3.2 Edge Detection Based on Gradient

The gradient vector can be defined as,

$$\bigtriangledown f = [\frac{\partial f}{\partial x} \frac{\partial f}{\partial y}]^T \tag{7}$$

where $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial x}$ can be approximated for finite difference as

$$\frac{\partial f}{\partial x} = f(x+1, y) - f(x, y) \tag{8}$$

and

$$\frac{\partial f}{\partial y} = f(x, y+1) - f(x, y) \tag{9}$$

respectively. The magnitude can be calculated as,

$$|\bigtriangledown f| = \sqrt{((\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2)} = \sqrt{M_x^2 + M_y^2} \tag{10}$$

the direction of the vector can be derived as,

$$dir(\bigtriangledown f) = \tan^{-1}(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}) \tag{11}$$

Try to find $|\bigtriangledown f|$ of this image (Fig. 2), if M_x and M_y are defined as bellow,

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{12}$$

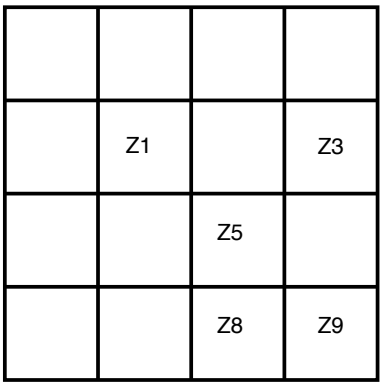

Figure 2: Part of an image where pixels are defined from $z1$ to $z9$

and

$$M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{13}$$

Let define,

$$\frac{\partial f}{\partial x} = f(x,y) - f(x+1,y+1) \tag{14}$$

and

$$\frac{\partial f}{\partial y} = f(x,y+1) - f(x+1,y) \tag{15}$$

Try to find out what are the values of M_x and M_y?

4 Corner Detection

The intensity changes over an image for a given direction is defined by the sum of squared root difference (SSD).

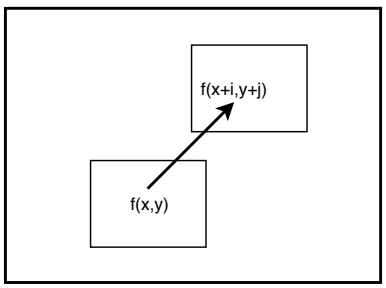

Figure 3: Change of the intensity of an image

$$Dis(i,j) = \Sigma_{x,y}(f(x+i,y+j) - f(x,y))^2 \tag{16}$$

With the assumption i and j are small, by using the Taylor theorem:

$$f(x+i,y+j) \simeq f(x,y) + \frac{\partial f}{\partial x} * i + \frac{\partial f}{\partial y} * j = f_x * i + f_y * j + f(x,y) \tag{17}$$

Thus, $(f(x+i,y+j)-f(x,y))^2$ can be quantified as,

$$(f(x+i,y+j)-f(x,y))^2 = \begin{bmatrix} i & j \end{bmatrix} \begin{bmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} \tag{18}$$

Therefor,

$$Dis(i,j) = \begin{bmatrix} i & j \end{bmatrix} \begin{bmatrix} \Sigma f_x^2 & \Sigma f_x f_y \\ \Sigma f_x f_y & \Sigma f_y^2 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} i & j \end{bmatrix} C \begin{bmatrix} i \\ j \end{bmatrix} \tag{19}$$

Based on intensity change, it is required to decide whether it is an edge or corner or flat surface. To achieve this we need to find out the direction and strength of eigenvectors and values of C after finding the characteristic equation. Also, Dis(i,j) represents the function of an ellipse. The characteristic equation is the equation which uses to find a matrix's eigenvalues. For a general nn matrix A, the characteristic equation in variable λ is defined by $det(A-\lambda I)=0$. In our case, C is the A. Thus, need to analyze the eigenvalues to make the final decision. Since C is symmetric, and can be diagonalized by the rotation of the coordinate axes; C can think of as bellow,

$$C = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{20}$$

1. Do the eigenvalues represent the edges?
2. Do the eigenvectors represent the edges?
3. What happens when both eigenvalues are very small?
4. If there is an edge what can you say about eigenvalues?
5. If it contains corners what can you say about corresponding eigenvalues and its directions?

5 The Laplace Operator

The Laplace operator is defined as two gradient vectors operators.

$$\Delta = \bigtriangledown . \bigtriangledown = \begin{bmatrix} \frac{\partial f}{\partial x_1} & . & . & . & \frac{\partial f}{\partial x_n} \end{bmatrix} \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ . \\ . \\ . \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \tag{21}$$

So when n becomes 2 and $\bigtriangledown = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix}$, what will be the Laplace operator? And what can you say about this result?

Let define the first order difference as $\bigtriangledown f(x,y) = f(x+1,y) - f(x,y)$, try to derive an expression for the second order differences $(\bigtriangledown(\bigtriangledown f(x,y)))$?

If you have noticed we were trying to find the second order differences on which direction?

Let's try to catch up with the 2d case as bellow,

$$\Delta f(x,y) = \Delta_x f(x,y) + \Delta_y f(x,y) \tag{22}$$

Based on the way we defined the 1st order difference and carry out the $\Delta f(x, y)$ operation you may be ended up somewhere right here.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} or \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{23}$$

5.1 Laplacian of Gaussian (LoG)

The Laplace operation does not know whether it detects edges or noise. Thus, to reduce the false positives it may adhere to use some smoothing techniques such as Gaussian filtering and etc. The Gaussian kernel can be defined for 2d case as,

$$G_\sigma(x, y) = \frac{1}{2 * \pi * \sigma^2} \exp -\frac{x^2 + y^2}{2 * \sigma^2} \tag{24}$$

Thus, to suppress the noise before apply the Laplace, Laplace operator can be modified as bellow,

$$\Delta(G_\sigma(x, y) * f(x, y)) = \Delta G_\sigma(x, y) * f(x, y) = LoG * f(x, y) \tag{25}$$

Let $\frac{\partial^2 G_\sigma(x,y)}{\partial^2 x} = \frac{x^2-\sigma^2}{\sigma^4} \exp^{-\frac{x^2+y^2}{2\sigma^2}}$, then can you find an expression for the LoG ?

1. So what will be the 3 x 3 kernel representation of 2d LoG?
2. What will be the effect when we apply -LoG?

Finally, to detect the edges form LoG, it is required to apply LOG to the image followed by detecting zero-crossings and keep in which stronger than a predefined threshold value.

5.2 Difference of Gaussian (DoG)

DoG is a kind of an approximation for the LoG which is defined as,

$$DoG = G_{\sigma 1}(x, y) - G_{\sigma 2}(x, y) \tag{26}$$

So what will be the 3 x 3 kernel representation of 2d DoG? And what can you say about the edge detection steps with DoG?

6 Gaussian and Laplacian Pyramids

The Gaussian pyramids are made of convoluting set of Gaussian kernels with an original image where each layer is constructed by downsampling previous layered image whereas the Laplacian is constructed as a difference between original and after applying a low pass filter (Gaussian filter).

7 Scale Invariant Feature Transform (SIFT)

7.1 Scale-space Extrema Detection

SIFT uses DoG instead of LoG because its little costly. The first octave can be made of n number of filtered images with difference σ values. As stated in the original paper, it is sufficient to use 5 images at the initial stage which is shown in Fig. 5. In order to detect the extrema (minima or maxima), as shown in the Fig. 6, each pixel compares with its eight neighbours and 9 pixels in the next and previous scales. If it is a local extrema it will be considered as a potential key point.

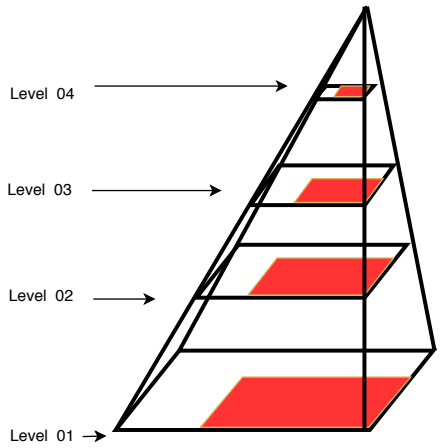

Figure 4: Level 01 is the filtered original image and followed layers from 2 to 4 are stacked after applying a selected filter and scale down with pre-defined factor e.g., 2, 4

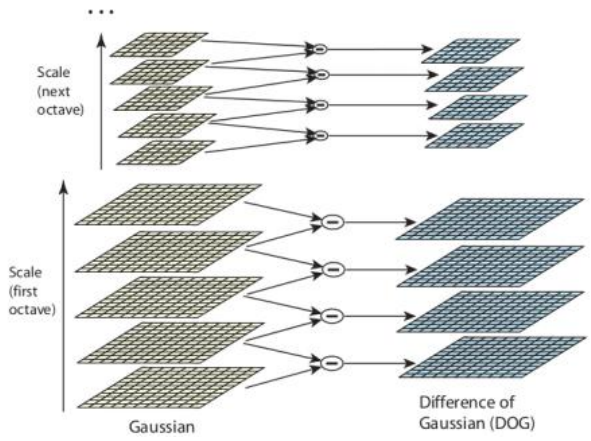

Figure 5: Construction of each layer of the pyramid where the first layer stacked with 5 images with a difference σ. Then it uses DoG to construct successive layers of the pyramid. `https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html`

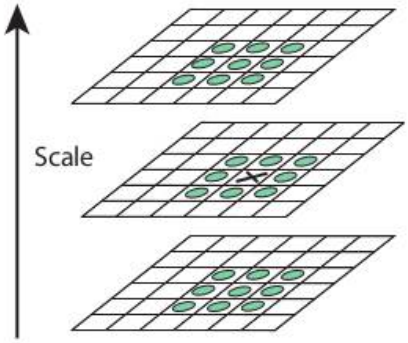

Figure 6: Local extrema calculation

7.2 Keypoint Localization

Let define the DoG for a given scale as bellow,

$$D(x, y, \sigma)_{n+1\,,n} =? \tag{27}$$

where n may goes up to ...?

In order to do the extrema localization as stated in the original paper [2], it is required to calculate a quadratic approximation of Taylor expansion of the scale-space ($D(x) + \frac{\delta D^T}{\delta x}x + \frac{1}{2}x^T\frac{\delta^2 D}{\delta x^2}x$ where ($D(x, y, \sigma) \leftarrow D(x)$) and remove all the key points which gives less than 0.03. Let's try to put this on in a different way. Let define the Hessian matrix of $D(x)$ as bellow,

$$H = \begin{bmatrix} \delta^2 D/\delta x^2 & \delta^2 D/\delta x \delta y & \delta^2 D/\delta x \delta \sigma \\ \delta^2 D/\delta x \delta y & \delta^2 D/\delta y^2 & \delta^2 D/\delta y \delta \sigma \\ \delta^2 D/\delta x \delta \sigma & \delta^2 D/\delta y \delta \sigma & \delta^2 D/\delta \sigma^2 \end{bmatrix} \tag{28}$$

Thus, strength of the extrema points can be calculated as $D(x, y, \sigma) + \hat{x}$ where $\hat{x}$ can be quantified as,

$$\hat{x} = -H^{-1} \begin{bmatrix} \delta D/\delta x \\ \delta D/\delta y \\ \delta D/\delta \sigma \end{bmatrix} \tag{29}$$

Hence, low contrast points can be pruning,

$$0.03 > |D(x, y, \sigma) + \frac{1}{2}\begin{bmatrix} \delta D/\delta x & \delta D/\delta y & \delta D/\delta \sigma \end{bmatrix}\hat{x}| \tag{30}$$

Furthermore, unstable points pruning are achieved by using this condition,

$$M = \begin{bmatrix} \delta^2 D/\delta x^2 & \delta^2 D/\delta x \delta y \\ \delta^2 D/\delta x \delta y & \delta^2 D/\delta y^2 \end{bmatrix} \tag{31}$$

If points are rejected when $det(M) < 0$.

7.3 Orientation Assignment

The basic idea is the assign the constant orientation to each key points to obtain the rotation invariance. This is how it is being achieved,

Let say magnitude ($m(x, y)$), then

$$m(x, y) = \sqrt{(I(x+1, y) - I(x-1, y))^2 + (I(x, y+1) - I(x, y-1))^2} \tag{32}$$

And angle ($\theta(x, y)$),

$$\theta(x, y) = tan^{-1}\frac{I(x, y+1) - I(x, y-1)}{I(x+1, y) - I(x-1, y)} \tag{33}$$

Similar to this, after calculating magnitude and angle for all points; construct an orientation histogram.

7.4 Keypoint Descriptor

To create a keypoint descriptor, it is taken 16x16 block around the key point in the direction of the dominant orientation. Divide the region into 4x4 subregions. Then create 8 bin gradient histograms for each subregion.

References

[1] DA Forsyth and J Ponce. “Computer vision: A modern approach . Prentice Hall series in artificial intelligence”. In: (2003).

[2] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2 (2004), pp. 91–110.