Introduction to Computer Vision: Local Features

Navasardyan Shant

October 11, 2019

Overview

Overview

What are edges?

A natural question arises:

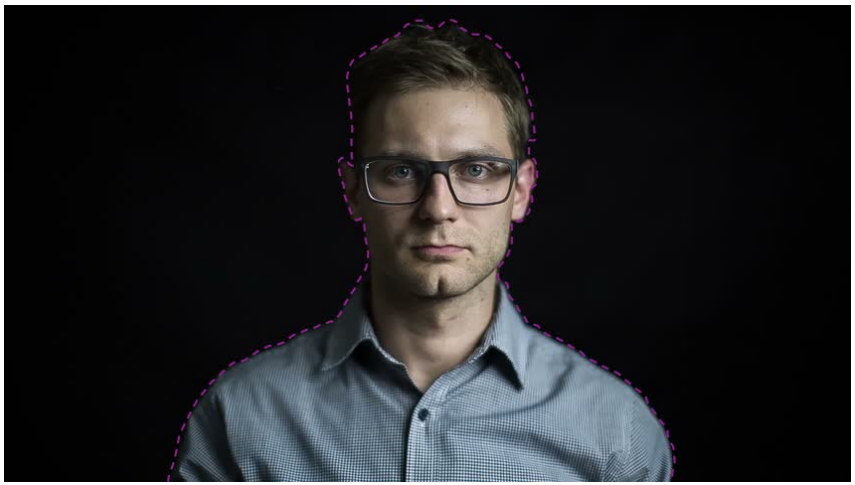
- What are edges in an image?
- What pixels in an image we, humans, treat as edges?
- Are edges in an image just intensity changes or there are some semantics in edge formations?

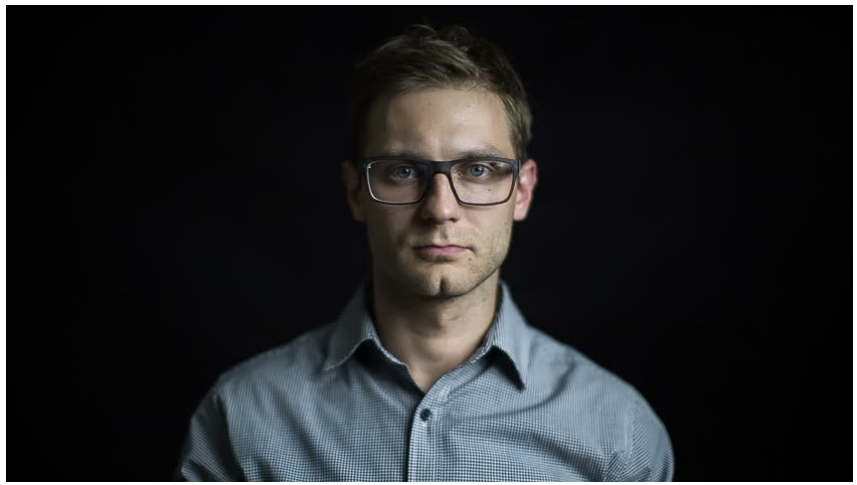
What are edges?

A natural question arises:

- What are edges in an image?
- What pixels in an image we, humans, treat as edges?
- Are edges in an image just intensity changes or there are some semantics in edge formations?

As you can see in the image below, some pixels of edges are far from being intensity-change pixels.



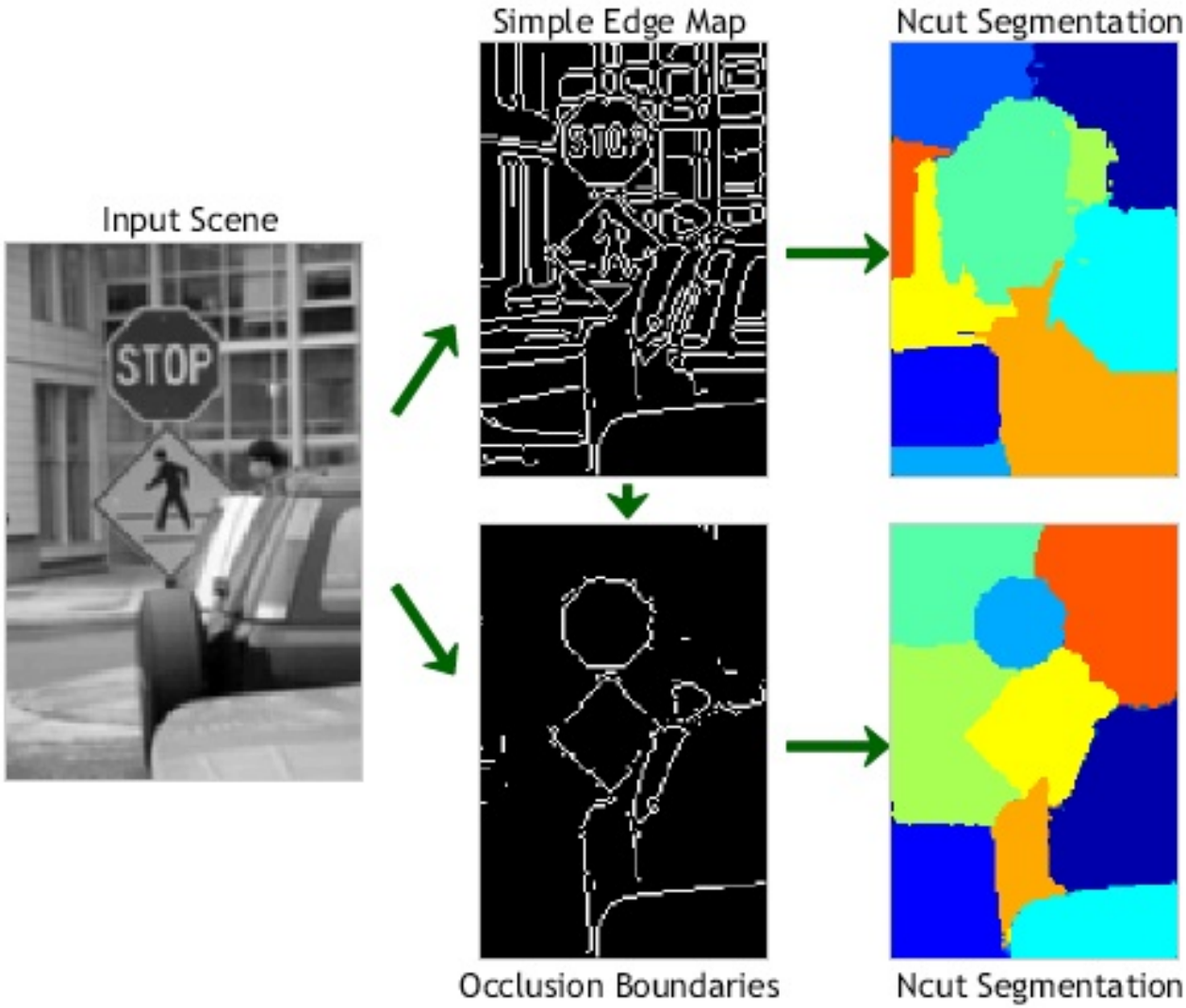


Artistic Sketch

Edge detection can be applied for image editing in effect like *artistic sketch*.

Segmentation

Edges in the image can be used as features for image segmentation:



Fingerprint Features

We also can find edges in a fingerprint image, in order to obtain features for detecting fingerprint similarity.

Overview

Edge Types: Step Edge

Basically in images we meet 3 types of edges: step edge, ramp edge, roof edge.

Edge Types: Step Edge

Basically in images we meet 3 types of edges: step edge, ramp edge, roof edge.

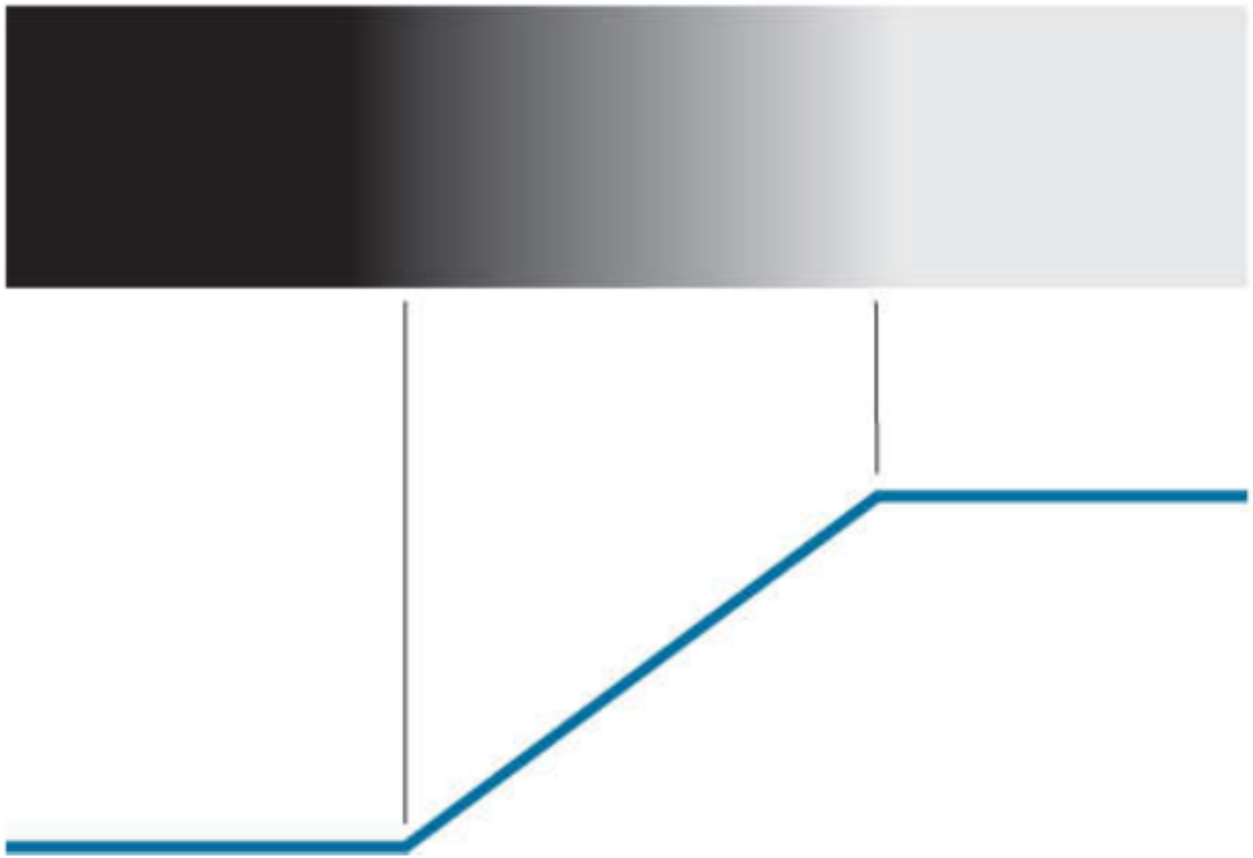
Step edge

A *step edge* is characterized by an abrupt difference between neighbor pixels:

Ramp Edge

Ramp edge

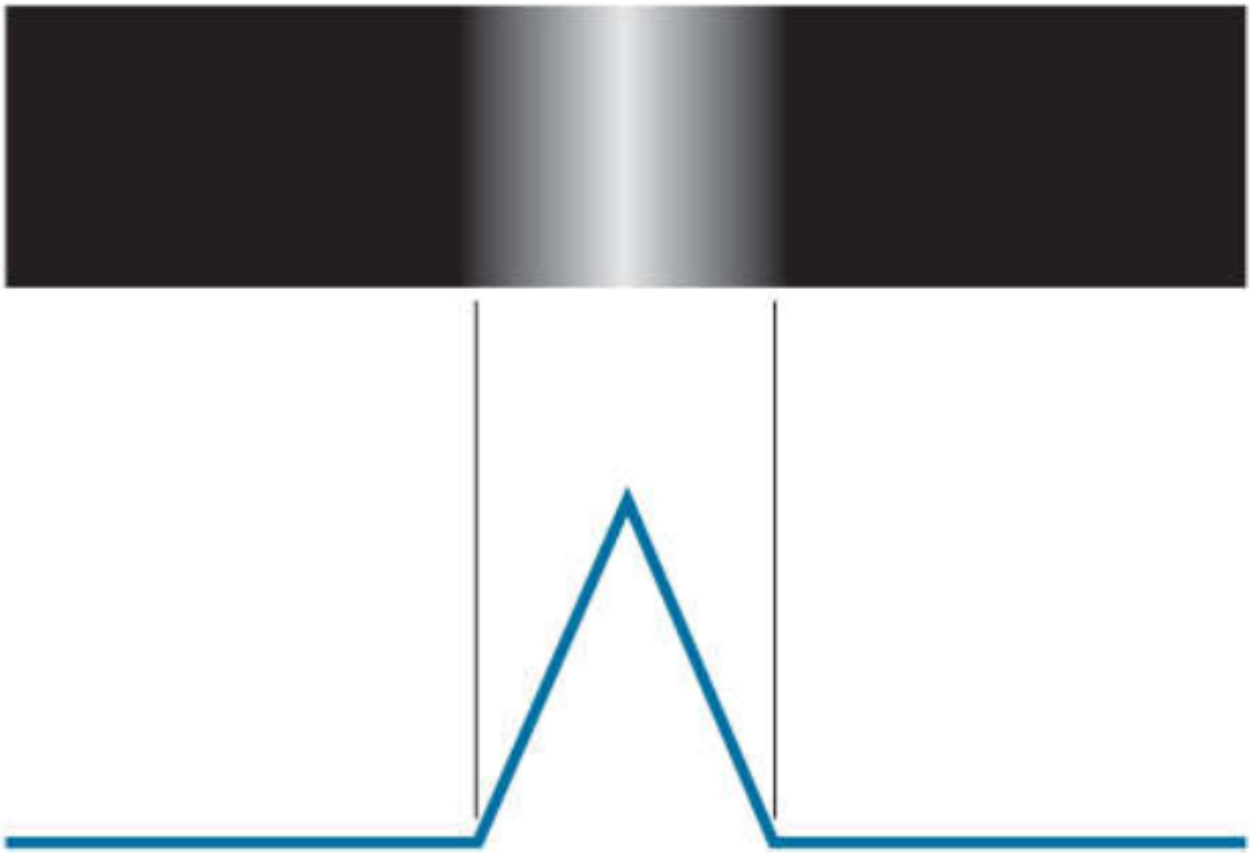
A *ramp edge* is characterized by a slow (linear) motion from a low-intensity part to a high-intensity part of an image:



Roof Edge

Roof edge

A *roof edge* is characterized by a line-like high (low) intensity on the low (high) intensity area :



Edge Types

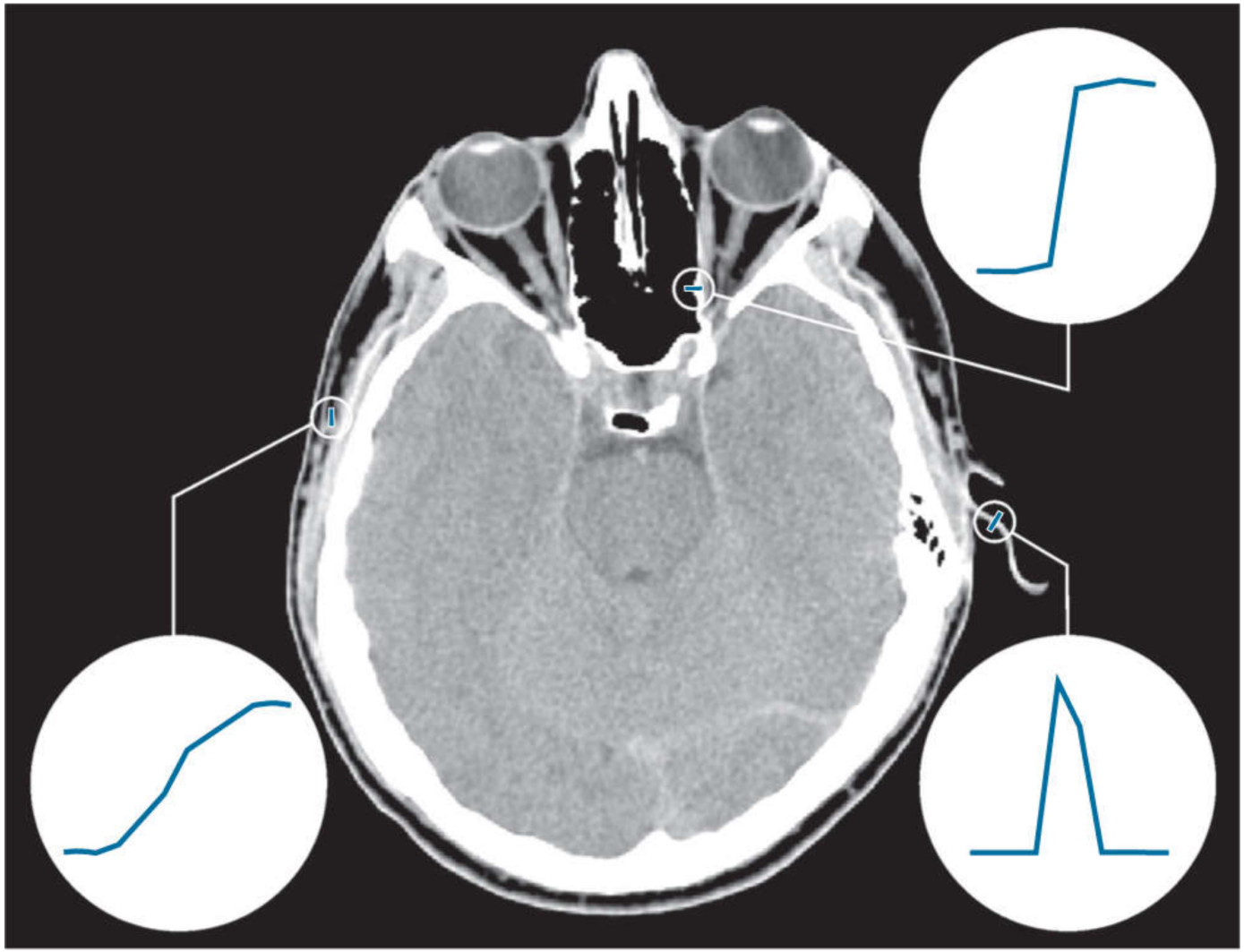


Figure: All 3 types of edges in one image.

Overview

Image Gradient Revisited

Let us recall the notion of the image gradient. We have that if the image is given with its functional form $f : [0, 1]^2 \to [0, 1]$, then its gradient we call image gradient. Now we can define the concept of digital image gradient as the gradient of an interpolation of this image.

Image Gradient Revisited

Let us recall the notion of the image gradient. We have that if the image is given with its functional form $f : [0,1]^2 \to [0,1]$, then its gradient we call image gradient. Now we can define the concept of digital image gradient as the gradient of an interpolation of this image.

Note

In this section we will discuss some approximations of image gradients, each of which with some convolution. Though we will not concentrate on why this approximations are image gradients in the sense above, it is interesting to try to find for each of them an interpolation with gradient equal to these approximations.

Image Gradient Revisited

Note

Recall that for image derivative along x-axis we have considered two approximations, namely convolutions with the kernels

$$\begin{pmatrix} -1 \\ 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

By the similar way for the derivative along y-axis.

Image Gradient Revisited

Note

Recall that for image derivative along x-axis we have considered two approximations, namely convolutions with the kernels

$$\begin{pmatrix} -1 \\ 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

By the similar way for the derivative along y-axis.

Now let's choose the second kernel for approximation of $\frac{\partial f}{\partial x}$ and consider the following example.

Edge and Gradient Directions: Example

Though we have not defined the notion of edge and its direction on an image, we can intuitively understand these meanings. Particularly, in the following example both the notions of edge and its direction are quite clear.

Edge and Gradient Directions: Example

Though we have not defined the notion of edge and its direction on an image, we can intuitively understand these meanings. Particularly, in the following example both the notions of edge and its direction are quite clear.

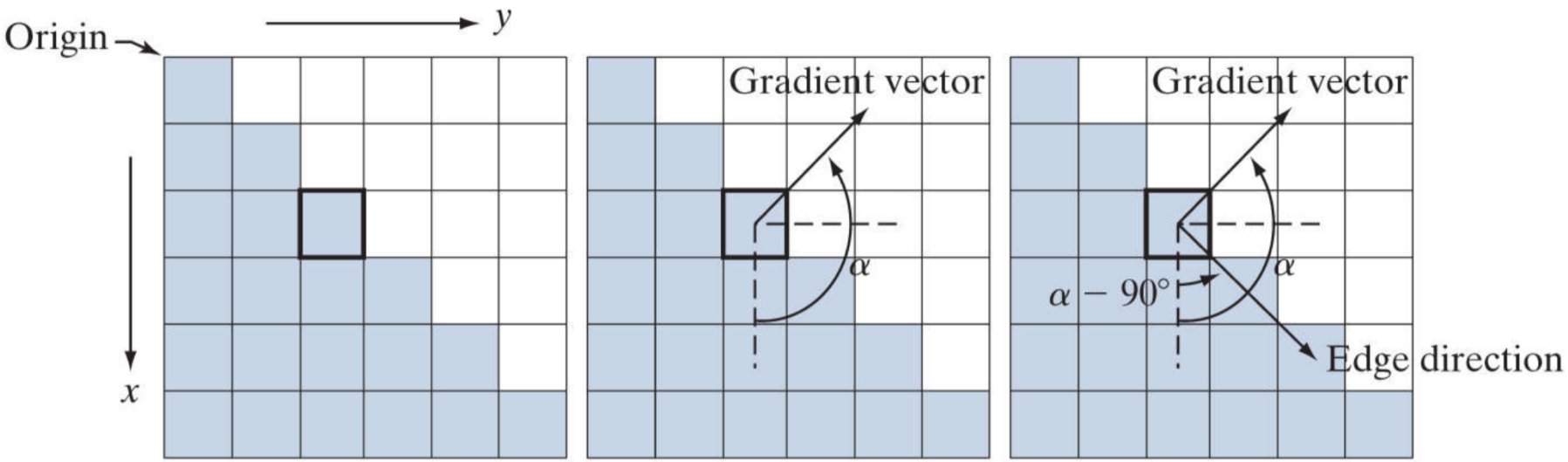


Figure: Here the shaded pixels are 0-valued, and white pixels are 1-valued. So, we get that the gradient is equal to $\begin{pmatrix} -1 \\ 1 \end{pmatrix}$, which is perpendicular to the edge direction.

Edge and Gradient Directions

Note

In the example above we point out the fact of perpendicularity of edge and gradient directions in order to determine the edge direction in more complicated cases. This will be clear within non-maximum suppression algorithm.

Edge and Gradient Directions

Note

In the example above we point out the fact of perpendicularity of edge and gradient directions in order to determine the edge direction in more complicated cases. This will be clear within non-maximum suppression algorithm.

Note

For edge detection we can use the magnitude of gradient in order to find out in which pixels the transition from low (high) to high (low) intensity is large. But this is the same as we use the image derivative in the direction of the image gradient.

In some cases we need to fix only the edges of particular direction. For this we can use the notion of image derivative in the particular direction. Recall that this direction need to be perpendicular to the direction for which we want to detect edges.

Edge and Gradient Directions

Recall

Let's recall the definition of directional derivative. Let $u = \begin{pmatrix} u_x \\ u_y \end{pmatrix} \in \mathbb{R}^2$ be a direction, i.e. $\|u\| = 1$; and let $f : \mathbb{R}^2 \to \mathbb{R}$ be a differentiable function. Then the **directional derivative** of f in the direction u is the following:

$$\nabla_u f(x, y) = \lim_{\varepsilon \to 0} \frac{f(x + \varepsilon u_1, y + \varepsilon u_2) - f(x, y)}{\varepsilon} = \nabla f^T \cdot u.$$

Edge and Gradient Directions

Recall

Let's recall the definition of directional derivative. Let $u = \begin{pmatrix} u_x \\ u_y \end{pmatrix} \in \mathbb{R}^2$ be a direction, i.e. $\|u\| = 1$; and let $f : \mathbb{R}^2 \to \mathbb{R}$ be a differentiable function. Then the **directional derivative** of f in the direction u is the following:

$$\nabla_u f(x, y) = \lim_{\varepsilon \to 0} \frac{f(x + \varepsilon u_1, y + \varepsilon u_2) - f(x, y)}{\varepsilon} = \nabla f^T \cdot u.$$

So, if we need to detect the edges which have the direction u, we need to find a direction v perpendicular to the direction u and approximate $\nabla_v f$. Examples of this approximations are the convolutions with the following kernels, where v is a diagonal direction:

$$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

More Kernels for Gradient Operation

The image gradient can also be estimated with convolutions with the following kernels:

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Figure: Prewitt kernel

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Figure: Sobel kernel

More Kernels for Gradient Operation

The image gradient can also be estimated with convolutions with the following kernels:

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Figure: Prewitt kernel

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Figure: Sobel kernel

Sobel kernels have better noise-suppression characteristics.

More Kernels for Gradient Operation

The image gradient can also be estimated with convolutions with the following kernels:

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

Figure: Prewitt kernel

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Figure: Sobel kernel

Sobel kernels have better noise-suppression characteristics.

Note

Remember, that a noise in an image can result to the fake responses of gradient operators. So in many cases before applying a gradient operation we smooth the image while preserving edges. It can be done for example with gaussian blur or guided image filtering.

Edge Detection by Thresholdoing Magnitude

As we have discussed earlier, edges in an image can be detected just by thresholding the gradient magnitude image.

Edge Detection by Thresholdoing Magnitude

As we have discussed earlier, edges in an image can be detected just by thresholding the gradient magnitude image.

Note

Sometimes we compute the magnitude as $Mag = |G_x| + |G_y|$, where G_x and G_y are partial derivatives of the image.

Edge Detection by Thresholding Magnitude

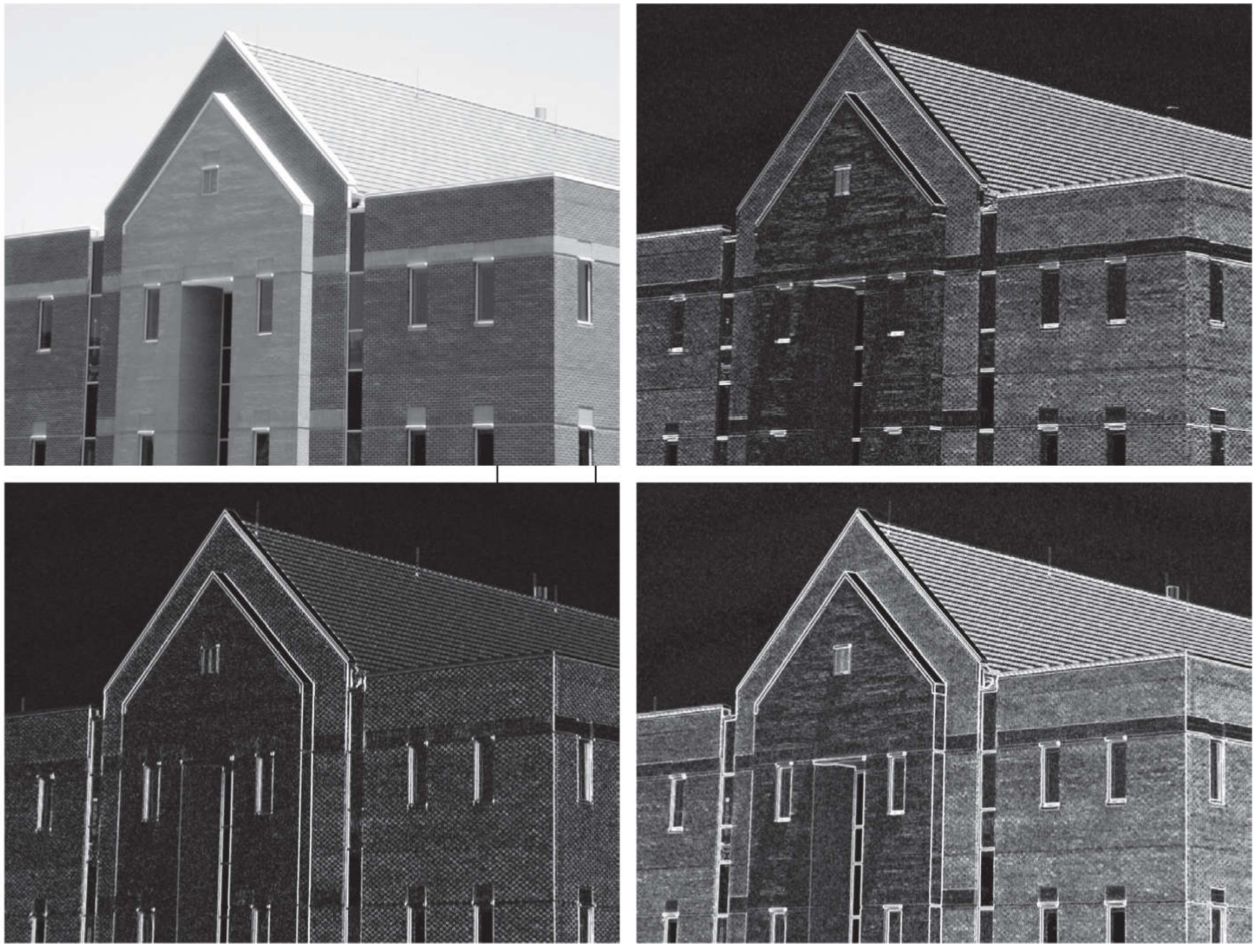


Figure: From left to right, top to bottom: the *original image*, *partial derivative* G_x in direction x, *partial derivative* G_y in direction y and the *magnitude* $|G_x| + |G_y|$

Edge Detection by Thresholding Magnitude

As you can notice, the results are noisy

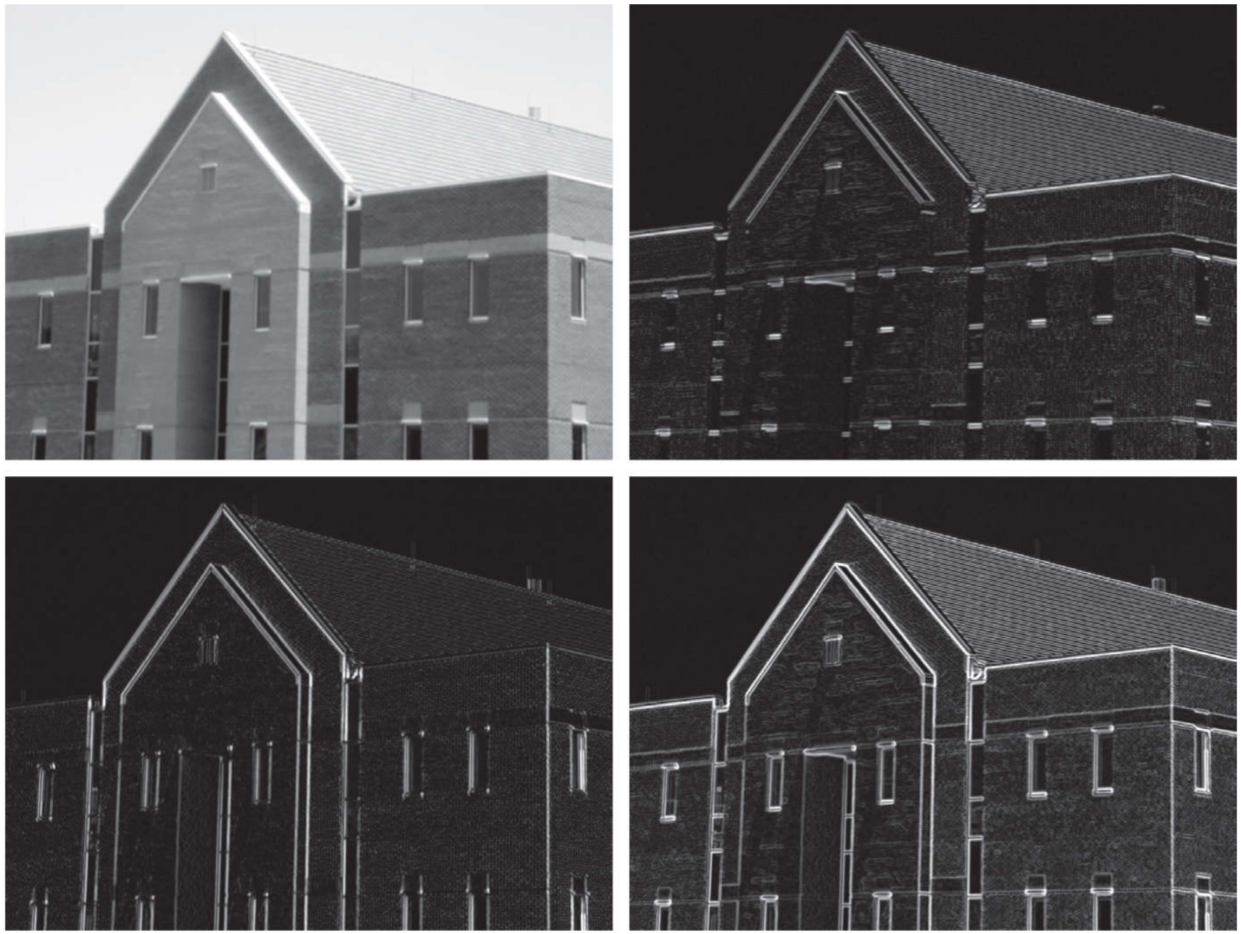


Figure: Same sequence as in figure above, but with the original image smoothed

Edge Detection by Thresholding Magnitude

Here is the result of thresholding magnitudes of the original image and the smoothed image:

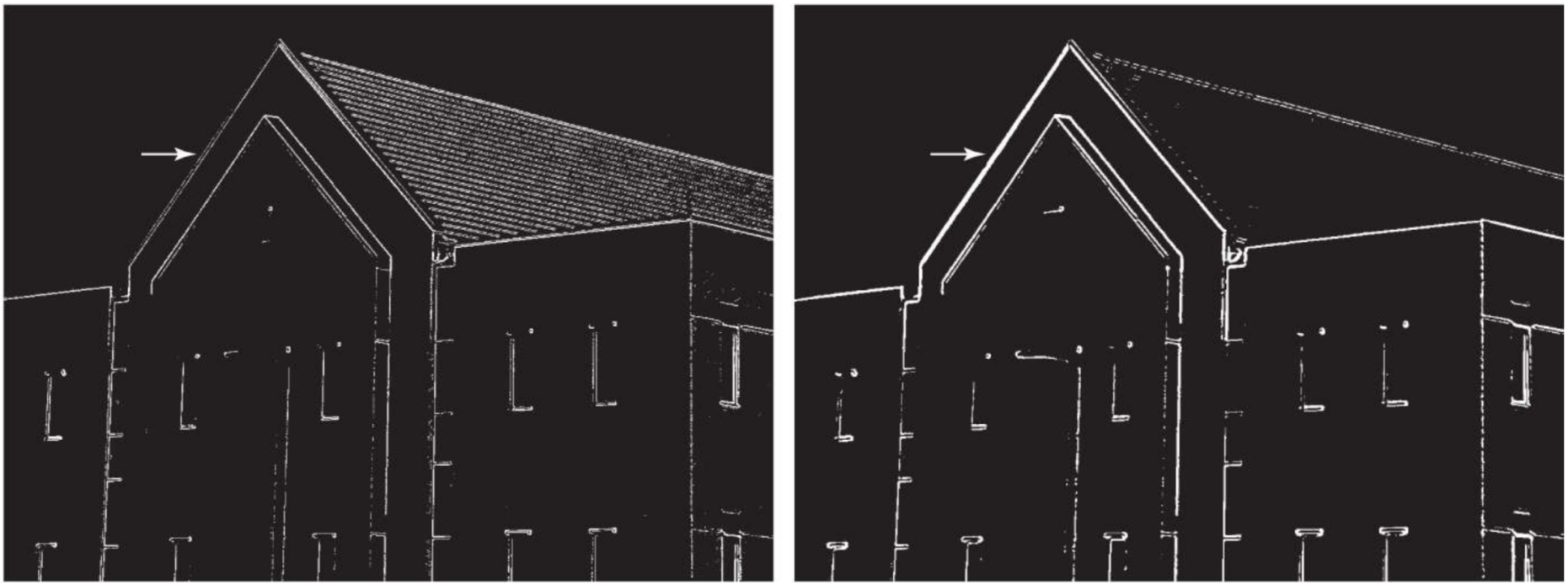


Figure: As you can see, noise results some lines, which are broken.

Overview

Non-Maximum Suppression

Question

What is wrong with thresholding the magnitude image? In which cases this is completely not what we want?

Non-Maximum Suppression

Question

What is wrong with thresholding the magnitude image? In which cases this is completely not what we want?

Though there can be many issues concerning the thresholding approach, one of them, as you can see in the image above, is that some edges are not connected with their true continuations (for example windows in the image) or some edges are thicker than one pixel (for example on the roof). In order to solve this issues we introduce the **Canny edge detector**. The Canny edge detector algorithm is composed of

- non-maximum suppression,
- thresholding by two thresholds.

Non-Maximum Suppression

Non-maximum suppression aims to deal with thick edges and suppress them in gradient direction in order to get edges of thickness equal to one pixel.

Non-Maximum Suppression

Non-maximum suppression aims to deal with thick edges and suppress them in gradient direction in order to get edges of thickness equal to one pixel.

Orientations

Recall that we also have the orientations of gradients at every pixel:

$$Ori(x, y) = arctan\left(\frac{G_y}{G_x}\right).$$

Non-Maximum Suppression: The Algorithm

For non-maximum suppression algorithm we use the following directions:

- vertical direction - $d_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
- first diagonal direction - $d_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$
- horizontal direction - $d_3 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$
- second diagonal direction - $d_4 = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix}$

Non-Maximum Suppression: The Algorithm

The algorithm is the following. Let M be the magnitude image, O be the orientation image. We construct an image B by the following way: for every point (i,j) $(i \in \{0,\dots,H-1\}, j \in \{0,\dots,W-1\})$ we find the direction d_k $(k = 1,2,3,4)$ which is the closest to the orientation O_{ij} (similarity is measured as the angle between the lines containing these directions). Then we take two neighbors of (i,j) along the direction d_k, let's denote them $(i_1,j_1),(i_2,j_2)$. Then the output of non-maximum suppression algorithm is the image B for which

$$B_{ij} = \begin{cases} 0 \text{ if } M_{ij} \leq max\{M_{i_1j_1}, M_{i_2j_2}\} \\ M_{ij} \text{ otherwise} \end{cases}.$$

Overview

Thresholding

Let we have obtained an image M after non-maximum suppression. Now we want to keep some pixels as edges of the initial image. If we use a threshold and set to 0 all the values below this threshold, we will end up with

- many false edges (false positives) in case of low threshold,
- less real edge points in case of high threshold.

Thresholding

Let we have obtained an image M after non-maximum suppression. Now we want to keep some pixels as edges of the initial image. If we use a threshold and set to 0 all the values below this threshold, we will end up with

- many false edges (false positives) in case of low threshold,
- less real edge points in case of high threshold.

So Canny edge detector algorithm uses two thresholds: high and low thresholds. This thresholding method is called *hysteresis thresholding*.

Hysteresis Thresholding

Let M be the image after non-maximum suppression. Let $T_L \leq T_H$ be two real numbers, we call *low and high* thresholds.

The algorithm of hysteresis thresholding is the following.

Hysteresis Thresholding

Let M be the image after non-maximum suppression. Let $T_L \leq T_H$ be two real numbers, we call *low and high* thresholds.

The algorithm of hysteresis thresholding is the following.

The Algorithm

Let's denote by M_H the image thresholded with the high threshold T_H and by M_L the image thresholded with the low threshold T_L, i.e.

$$M_H[i,j] = \begin{cases} 1 & M[i,j] \geq T_H \\ 0 & \text{otherwise} \end{cases} \qquad M_L[i,j] = \begin{cases} 1 & M[i,j] \geq T_L \\ 0 & \text{otherwise} \end{cases}$$

Then for every 1-valued pixel of the image M_L we set it as edge pixel, if it has a 8-connected neighbor, which is 1-valued in the image M_H (in this context we assume that any pixel is a neighbor of itself).

Introduction to Computer Vision: Local Features

Navasardyan Shant

October 11, 2019