**Lab Review 03**

# Edge detection & Hough transform

**Image Analysis and Computer Vision**

**November 11, 2013**

## 1 Difference operators

Edge detection

The result of applying an edge detector to an image may lead to a set of connected curves that indicate the boundaries of objects, the boundaries of surface markings, as well as curves that correspond to discontinuities in surface orientation.

Preserve important structural properties and filter out less important information.

Create difference operator:

Simple difference operator

Central difference operator

Robert diagonal operator

Sobel operator : is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. The sobel operator uses two 3by3 kernels which are convolved with the srcinal image to calculate approximations of the derivatives – one for horizontal changes and one for vertical.

Since the sobel kernels can be decomposed as the products of an averaging ([1 0 -1]) and a differentiation kernel ([1 2 1]'), they compute the gradient with smoothing.

These operators are used for approximating first order partial derivatives in two orthogonal directions.

**Image gradient**

An image gradient is a directional change in the intensity or color in an image.

Each pixel of a gradient image measures the change in intensity of that same point in the srcinal image, in a given direction. To get the full range of direction, gradient images in the x and y directions are computed.

The pixels with largest gradient values in the direction of the gradient become edge pixels, and edges may be traced in the direction perpendicular to the gradient direction.

Mathematically, the gradient at each image points is a 2D vector with the components given by the derivatives in the horizontal and vertical directions. At each image points, the gradient vector points in the direction of largest possible intensity increase, and the length of the gradient vector corresponds to the rate of change in that direction.

Convolution

Convolution is a mathematical operation on two functions f and g, producing a third function that is typically viewed as a modified version of one of the srcinal functions, giving the area overlap between the two functions as a function of the amount that one of the functions is translated.

Convolution(valid, full, same)

Since convolution is a neighborhood operation, so to handle the pixels of the borders of the image, there are some suggestions , like 'valid', 'full', 'same'.

Valid region

Make the image big enough so that there is a guard ring around the region of interest. That is, use only the valid region of the convolution. The resulting image will be smaller by the width of the convolution operator.

Size = [ma-mb+1, na-nb+1]

A gradient image in the x direction measures horizontal change in intensity. A gradient image in the y direction measures vertical change in intensity.

Gray pixels have a smaller gradient; black and white pixels have a large gradient.

==========

**Question 1: What do you expect the results to look like and why? Compare the size of dxtools with the size of tools. Why are these sizes different?**

**tools = few256;**
**dxtools = lab3_1(tools, 'valid');**

Firstly I created two difference operators that approximate the first order partial derivatives in two orthogonal directions using 'sobel' operators and then compute derivatives using conv2.

**F=fspecial('sobel')**
returns a 3x3 horizontal edges sobel filter. If you want a vertical edges sobel filter, you can use transposition of F. F is [ 1 2 1; 0 0 0; -1 -2 -1].

## Edge

Edge pixels are pixels where the intensity changes sharply. The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image. The gradient is the change in gray level with direction. This can be calculated by taking the difference in value of neighboring pixels

**Vertical Edges**

Vertical edges can be detected by using a horizontal gradient operator followed by a threshold operation to detect the extreme values of the gradient. The gradient produces a doublet of extremes, positive-negative or negative-positive, depending on the direction of the transition.

**Horizontal edges**

Horizontal edges produce a vertical gradient in the image, and can be enhanced with a vertical gradient detector.

**'valid' - returns only those parts of the convolution that are computed without the zero-padded edges. size(C) = [ma-mb+1,na-nb+1] when all(size(A) >= size(B)), otherwise C is empty**.

size(tools);

size(tools(:));

size(dxtools) = [256-3+1,256-3+1] = [254,254]

==========

2 Point-wise thresholding of gradient magnitudes

The magnitude of the gradient tells us how quickly the image is changing i.e, length of the gradient vector tells us about the rate of change, while the direction of the gradient tells us the direction in which the image is changing most rapidly.

Image filtering

Modify the pixels in an image based on some function of a local neighborhood of the pixels.

Linear filtering

Replace each pixel by a linear combination of its neighbors. The prescription for the linear combination is called the convolution kernel.

Any linear shift invariant operator can be represented as a convolution.

Differentiation is convolution and convolution is associative.

Finter2 uses conv2 to do most of the work. 2D correlation is related to 2D convolution by a 180 degree rotation of the filter matrix.

Filter2 returns Y computed via 2D correlation with size specified by shape.

Thresholding

Thresholding is the simplest method of image segmentation. We expect to see a distinct peak in the histogram such that thresholds can be chosen to isolate this peak accordingly. Region with uniform intensity give rise to strong peaks in the histogram.

Thresholding creates binary images from grey level ones by the turning all pixels below some threshold to zero and all pixels about that threshold to one.

If g(x, y) is a threshold version of f(x, y) at some global threshold T, g is equal to 1 if f(x, y) >= T and zero otherwise.

Good threshold can be selected if the histogram peaks are tall, narrow, symmetric and separated by deep valleys.

==========

**Question 2: Is it easy to find a threshold that results in thin edges? Explain why or why not!**

```
gradtools = lab3_2(few);          using conv2 operator
or,
pixels = lab3_2a(few,'valid');    using filter2 operator
```

In this case, we expect to see a distinct peak in the histogram corresponding to foreground objects such that thresholds can be chosen to isolate this peak accordingly.

After computing edge strength (i.e. gradient magnitude), apply threshold, to decide whether edges are present or not at an image point.

The lower the threshold, the more edges will be detected, and the result will be increasingly susceptible to noise and detecting edges of irrelevant features in the image.

High threshold may miss subtle edges.

Edge thrsholding applied to gradient magnitude image, the resulting edges will be thick and sometype of edge thinning post processing is necessary.

Edge detected with non maximum suppression, the edge curves are thin by definition.

We approximate thresholds for thresholding by thresholding with hysteresis. This uses multiple thresholds to find edges.

Edge thinning

-To remove spurious points on edge

- Employed after filtered for noise, the edge operator has been applied and after that the edges have been smoothed using an approximate threshold.

So it is difficult to find a threshold that results in thin edges.

==========

Four steps of an edge detection:

Smoothing

Enhancement (sharpening)

Detection (thresholding)

Localization (Determine the exact location of an edge)

Non maxima suppression

- Find local maxima of the gradient magnitude
- All values along the direction of the gradient that are not peak values of a ridge are suppressed

The output of non maxima suppression still contains the local maxima created by noise. So hysteresis works by edge linking.

==========

**Question 3: Does smoothing the image help to find edges?**
**pixels = lab3_2ab(few);** **gaussian smoothing + filter2**
**or**
**pixels = lab3_2b(few);**
**Lvvtild = lab3_final(few, .001, 'same');**
Yes, smooting helps to find the edges.
The differential masks act as high pass filters which tend to amplify the noise. To reduce the noise, smooth first with a low pass filter.

- A larger filter reduces noise but worsens localization i.e. add uncertainty to the location of the edge
- Edge thinning and linking are required
- Good detection : the optimal detection must minimize the probability of false positives (detecting spurious edges caused by noise) as well as that of false negatives(missing real edges)
- Good localization: a larger filter reduces noise but worsens localization(i.e. it adds uncertainty to the location of the edge)

Smoothing operation cannot remove the false edge due to the non meaningful discontinuities. So a global histogram (gradient histogram) of the gradient magnitudes at all pixels in the image can be constructed

and then a thresholding algorithm can be used to determine a threshold in order to remove the unwanted false edge.

==========

Differential geometry based edge detection

- Point for which the gradient magnitudes reaches local maxima in gradient direction
- 2nd order derivative equal to zero
- 3rd order derivative negative

Mask test – lxxx = lab_mask_test()

Histogram - hist = lab_try_3(few)

==========

**Question 4: What can you observe? Provide explanation based on the generated images.**
**Lvvtild = lab3_4final(few,0.001,'same');**

In differential geometric terms edge points can be characterized as poins at which the second order directional derivative in the gradient direction is zero.

We get the zero crossings of the edge and changing the scale from 0.1 to 64 we observe that within scale increasing the we get the less zero crossing boundaries.
The result of applying an edge detector to an image may lead to a set of connected curves that indicate the boundaries of the objects , the boundaries of surface markings as well as curves that corresponds to discontinuities in surface orientation.

Observe how different types of edge structures are extracted at different scales, and specifically how certain diffuse edge structures fail to give rise to connected edge curves at the finest levels of scale.

==========

We can alter the number of extracted edges either by changing the width of the Gaussian filter or by introducing some sort of thresholding technique (although this may not produce closed contours).

A line may be viewed as an edge segment in which the intensity of the background on either side of the line is either much higher or much lower than the intensity of the line pixels.

- Zero crossings for better localization when the edges are not very sharp
- To reduce the noise effect, the image is first smoothed with a low pass filter, Gaussian filter, and sigma in to the Gaussian determines the degree of smoothing, mask size increases with sigma
- It determines which edges are most significant in terms of the range of scale over which they are observed to occur
- Zero crossings lie on closed contours and so the output from the zero crossing detector is usually a binary image with single pixel thickness lines showing the positions of the zero crossing points

- Zeros of second derivatives to indicate maxima of gradient magnitude, but the problem is that this doesn't include any notion of how strong that maximum is
- Zero crossings edge points are very sensitive to noise
- The magnitude array will have large values where the image gradient is large , but that is not sufficient to localize the edges. The broad ridges in the magnitude array must be thinned so that only the magnitudes at the points of greatest local change remain.

==========

**Question 5: Assemble the results of the experiment above into an illustrative collage with the subplot command. Which are your observations and conclusions?**

```
Lvvvtild = lab4_final(few,0.001,'same');
Lvvvtild = lab4_final(few,1,'same');
Lvvvtild = lab4_final(few,16,'same');
Lvvvtild = lab4_final(few,64,'same');

Lv = lab3_3b(pic, shape)
Lvvvtild = lab4_ruf(inpic, shape)
```

In differential geometric terms edge points can be characterized as poins at which the third order directional derivative in the gradient direction is negative.

We observe that with the scale the image is smoothing more and more and with that the change in the directional derivatives also becoming unclear with the srcinal picture.

An intuitive explanation of why the edge strength measures differ in this respect, is that the third order derivative operator has more narrow response properties to edges. Therefore, the magnitude of this response will start to decrease earlier with scale, when interference effects between neighboring edges start affecting the edge responses.

The third order derivative operator has more narrow response properties to edges. Therefore, the magnitude of this response will start to decrease earlier with scale.

- Any single choice of sigma into Gaussian filters does not produce a good edge map
- A large sigma will produce edges from only the largest objects and they will not accurately delineate the object because the smooting reduces shape detail
- A small sigma will produce many edges and very jagged boundaries of many objects

==========

**Question 6: How can you use the response from Lvv_tilda todetect edges, and how can you improve the result by using Lvvv_tilda?**

**edgecurves = njetedge(inpic, scale, threshold, shape)**

**edgecurves = njetedge(few, 0.001);**

Using Lvv_tilda we will get the edge points of zero crossings and then from that zero crossings we can find out the maximum gradient magnitudes using Lvvv_tilda we will find out the points where Lvvv_tilda is negative.

- No need for any explicit estimate of the gradient direction.
- No specific assumptions about the shape of the intensity profile perpendicular to the edge.
- Works as a non maxima suppression.

==========

## Hough transform

- **For each edge point, find xcostheta+ysintheta**
- **For each theta, find xcostheta+ysintheta => r**
- **Accumulator array contains the number of times each value (r, theta) appears in the above step**
- **Find the largest value of acc**
- **Then these (r, theta) makes the line, using xcostheta + ysintheta = r**

==========

**Question 7: Identify the correspondences between the strongest peaks in the accumulator and line segments in the output image. Doing so, convince yourself that the implementation is correct. Summarize the results of this study and print out on paper.**

**hough_acc_1(test);**
**houghtr(test)**

**hough_all(test)**
**[ accu ] = hough_tran(test);**

**houghnote(test)**

For each pixel in the curve, a number of solid lines can be plotted all at different angles. Each solid line is represented by its distance with the center and the angle theta to the perpendicular line to it to the center. The point where the curves intersect gives a distance and angle. This distance and angle indicate the line which intersects the points being tested.

==========

==========

**Question 8: How do the results and computational time depend on the number of cells in the accumulator?**

When ntheta increases, the computation needs more time as it is in the innermost loop.

==========

**Question 9: How do you propose to do this? Relate your answer to the different parts of the images.**

==========