- Canny Algorithm, Part One

So, the main difference between Canny Part One and Sobel is the smoothener (Canny uses a Gaussian Sobel uses the four one's.

To write code for canny, we will start with marrh.c and do these steps.

-- Marrh.c uses flexible size masks (which we need), we will keep this part of the code.

-- Marrh.c uses second derivatives, whereas we need only first derivatives (we need first x- and y- derivatives), so we will change the equation in the marrh.c line to be the first x-derivative.

-- Then, because we need two derivatives, we will double up on that line, i.e., make a copy of it to compute the y-derivative, finally ending up with two masks (xmask and ymask).

1

- Canny Algorithm, Part One

-- Then use the convolution code from marrh but remember to double up on it, to get two outputs.

-- Then delete the code in marrh that is below the convolution code.

-- Then bring in the sqrt (of squares) code from sobel. This will compute the magnitude, will scale it for output, and will print it out.

-- At this point, you are done with Canny part One, and your code should produce output very similar to the Sobel magnitude image.

2

Canny Part Two

Normally called Non-maximaSuppression
We will call it RidgePeaks of Magnitude Image

3

Canny Part Two

Consider the Magnitude image obtained in Part One

Think about it as a terrain map.
So, its numbers represent height.

4

Canny Part Two

The Magnitude image can be viewed as a relief map (on right)

Let us examine a ridge in this relief map (the magnitude pic)

5

Canny Part Two

Visualize ridges in this relief map (the magnitude image).
Two different ways to visualize are shown here.

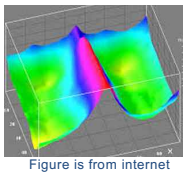


Figure is from internet

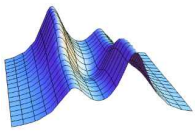


Figure is from internet

Ridges in the magnitude image represent edges.
The stronger the brightness jump in the original picture, the higher the ridge in the magnitude image.

6

Canny Part Two

Consider some ridges, and how to find their peaks.

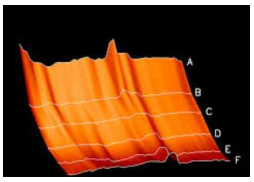


Figure is from internet

For the ridges shown, we would want to traverse a row, such as, say, B, C, D, or E, and ask if we are finding a peak as we encounter the values in that row. A Peak is simply a position whose magnitude value exceeds that of the neighbor to the left and to the right. This is MaxTest.

7

Canny Part Two

Direction for MaxTest:

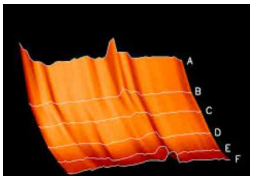


Figure is from internet

MaxTest runs across these rows. If we knew that the ridge is a vertical one (as these examples are), we would NOT want to run MaxTest in a vertical direction or in a diagonal direction (for these specific ridges). i.e., MaxTest runs in a direction perpendicular to the ridge.

8

Canny Part Two

MaxTest's Direction: Hmmm… Perpendicular to the ridge??

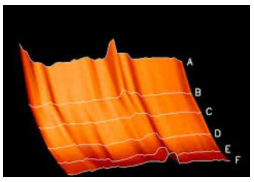


Figure is from internet

MaxTest runs in a direction perpendicular to the ridge.

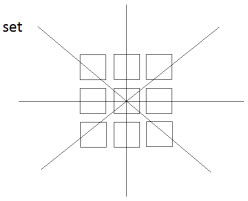
Well, that is simply IN THE DIRECTION OF THE GRADIENT!

The gradient was a vector. We used its Magnitude. Now we get to employ its Direction.

9

Canny Part Two

Consider a 3x3 set of pixels.

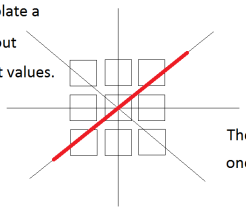


We need to examine triples along the directions shown to see if the center pixel is a peak (in magnitude) compared to its two neighbors.

10

Canny Part Two

So, we need to isolate a triple, and ask about the three adjacent values.

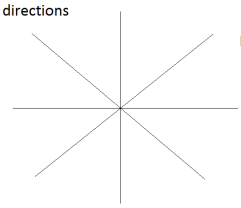


There are 4 such directions, one triple per direction.

11

Canny Part Two

So, consider the 4 directions

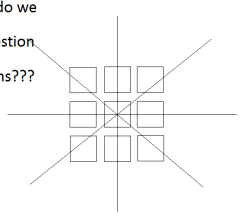


For each direction, we will ask whether the center mag value exceeds the mag value of neighbor on one side, and neighbor on other side; if yes, mark as peak.

12

Canny Part Two

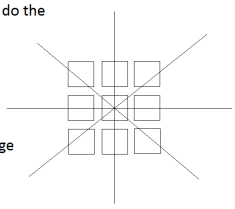
For a center pixel, do we ask the maxing question for all four directions???



13

Canny Part Two

No, we should only do the maxing test for the direction that is dictated by the direction of the edge

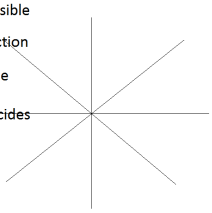


We always want to max-test in the direction perpendicular to the edge, i.e., across the edge. This means in the direction of the gradient vector.

14

Canny Part Two

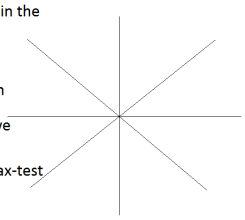
So, given these 4 possible directions, each direction will be called up, if the gradient vector coincides with the max-test direction.



15

Canny Part Two

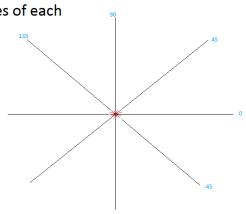
Hence, we ascertain the gradient vector's direction, and then depending on it, we proceed to the max-test



16

Canny Part Two

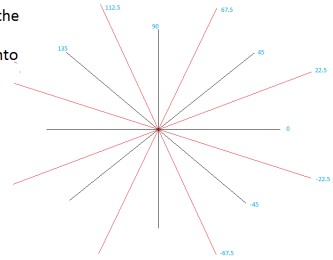
We know the angles of each direction.



17

Canny Part Two

We break up the angle space into 4 cones of directions

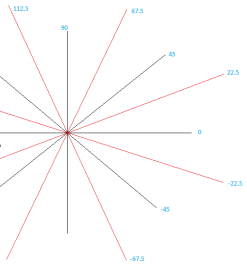


18

Canny Part Two

For example, if it is found that the gradient at the center pixel is between -22.5 and +22.5,

Which max-test should we employ?



19

Canny Part Two

For each pixel (i.e., double-for loop)
Get pixel's gradient direction
If - 22.5 < Dir < 22.5
Employ Horizontal Max-Test
else if +22.5 < Dir < +67.5
Employ Test involving SW and NE pixels
else if -67.5 < Dir < -22.5
Employ Test involving SE and NW pixels
else Test for Vertical cone, and then Employ Vertical Test

20

Canny Part Two

For each pixel (i.e., double-for loop)
Get pixel's gradient direction , Dir
If - 22.5 < Dir < 22.5
Employ Horizontal Max-Test
else if +22.5 < Dir < +67.5
Employ Test involving SW and NE pixels
else if -67.5 < Dir < -22.5
Employ Test involving SE and NW pixels
else Test for Vertical cone, and then Employ Vertical Test

Two problems:
-- Vertical cone is ...
-- Need to include end-points of cones

21

Canny Part Two

So, put = sign in

And, remove Vertical cone test, use "otherwise"

For each pixel (i.e., double-for loop)
Get pixel's gradient direction , Dir
If - 22.5 < Dir <=22.5
Employ Horizontal Max-Test
else if +22.5 < Dir <=+67.5
Employ Test involving SW and NE pixels
else if -67.5 < Dir <=-22.5
Employ Test involving SE and NW pixels
else Employ Vertical test

22

Canny Part Two

Lastly, avoid the atan call , to get direction, take Tan of everything

Becomes: tandir= convY/convX, instead of Dir= atan(convY/convX)

and if ((tan (-22.5)) < tandir <= (tan(22.5)))

etc.

23

Actual code for Peaks

```
for(i=MR;i<256-MR;i++){
  for(j=MR;j<256-MR;j++){

    if((xconv[i][j]) == 0.0) {
      xconv[i][j] = .00001;
    }
    slope = yconv[i][j]/xconv[i][j];
    if( (slope <= .4142)&&(slope > -.4142)){
      if((mag[i][j] > mag[i][j-1])&&(mag[i][j] > mag[i][j+1])){
        cand[i][j] = 255;
      }
    }
    else if( (slope <= 2.4142)&&(slope > .4142)){
      if((mag[i][j] > mag[i-1][j-1])&&(mag[i][j] > mag[i+1][j+1])){
        cand[i][j] = 255;
      }
    }
    else if( (slope <= -.4142)&&(slope > -2.4142)){
      if((mag[i][j] > mag[i+1][j-1])&&(mag[i][j] > mag[i-1][j+1])){
        cand[i][j] = 255;
      }
    }else{
      if((mag[i][j] > mag[i-1][j])&&(mag[i][j] > mag[i+1][j])){
        cand[i][j] = 255;
      }
    }
  }
}
```

24

Hysteresis (Double) Threshold

25