Canny Part Three

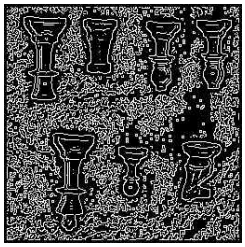
Normally called Hysteresis Thresholding
We call it Double Thresholding

1

Double Thresholds

Let us first review what we have produced so far:

A Magnitude Image



A Peaks Image

2

Double Thresholds

- Two thresholds will be used
- Will be applied to Magnitude image, but only to places that have shown up as peaks
- Two thresholds, a HIGH and a LOW
- If Mag exceeds HI, definitely pass pixel to Final
- If Mag is between HI and LO, then check if geographically adjacent to a position (pixel) that has made it in to Final; if yes, then pass pixel to Final
- If Mag lower than LO, definitely never be in final

3

Double Thresholds

- Will be applied to Magnitude image, but only to places that have shown up as peaks
- Will have two thresholds, a HIGH and a LOW
- If Mag exceeds HI, definitely pass pixel to Final
- If Mag is between HI and LO, then check if geographically adjacent to a position (pixel) that has made it to in Final; if yes, then pass pixel to Final
- If Mag lower than LO, definitely never be in final

4

Double Thresholds

- up as peaks
- WWill be applied to Magnitude image, but only to places that have shown ill have two thresholds, a HIGH and a LOW
- If Mag exceeds HI, definitely pass pixel to Final
- If Mag is between HI and LO, then check if geographically adjacent to a position (pixel) that has made it to in Final; if yes, then pass pixel to Final
- If Mag lower than LO, definitely never be in final

5

Double Thresholds

- The typical way to write this is using Recursion.
- Simply scan the image, looking only at Peaks, and at each Peak, ask if Mag exceeds HI; if No, do nothing (go on to next peak); if Yes, then call a recursive procedure on each of the 8 neighbors
- The recursive procedure must use LO to determine if it should call itself again on the 8 neighbors of the peak it was given. If exceeds LO, call recursion.

6

Double Thresholds

- Since we do NOT assume that all students in class know how to write recursion, here is an iterative, simple-to-follow, but inefficient procedure:

```
For i, For j
    if  peaks(ij) == ON
        if  mag(ij)> HI
              peaks(ij) = OFF, final(ij) = ON
         else if mag(ij)< LO
              peaks(ij)=final(ij)= OFF.
```

Then, do the WHILE-LOOP from next slide.

7

Simple, inefficient cont'd

```
moretodo=ON
While moretodo==ON
    moretodo= OFF
    For i, For j
       if  peaks(ij) == ON
           For p (-1 to +1), For q (-1 to +1)
              if  final(i+p,j+q) == ON
                peaks(ij) = OFF, final(ij) = ON, moretodo=ON

ALL DONE
```

8

Simple, inefficient : All on One slide

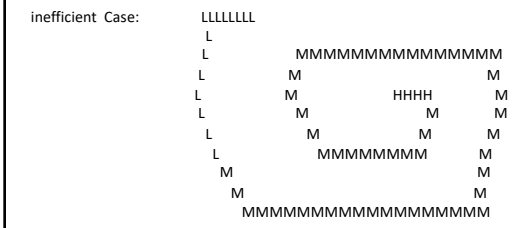
```
For i, For j
    if  peaks(ij) == ON
        if  mag(ij)> HI
              peaks(ij) = OFF, final(ij) = ON
         else if mag(ij)< LO
              peaks(ij)=final(ij)= OFF.
moretodo=ON
While moretodo==ON
    moretodo= OFF
    For i, For j
       if  peaks(ij) == ON
           For p (-1 to +1), For q (-1 to +1)
              if  final(i+p,j+q) == ON
                peaks(ij) = OFF, final(ij) = ON, moretodo=ON

ALL DONE
```

9

Simple, inefficient cont'd

inefficient Case:



Thankfully, most M-chains are small.

10

Canny Part Four

Automatically get HI (and hence LO)

11

Automatically Get HI

Use Percent as input

Then apply it to histogram of scaled mags

In the histogram of scaled mags, find the Point that exceeds Percent of all, mark that as HI. Then, LO is 0.35 of HI

12

Details of Automatically Get HI

Read Percent as input
Compute Histogram of scaled magnitudes
CutOff = Percent*Rows*Cols
for (i= HistogramSize downto 1, i--)
  AreaOfTops += Histogram[i]
  if (AreaOfTops>CutOff)
    Break out of for-loop
HI=I
LO= .35*HI

Histogram of scaled magnitudes obtained by:
for i, for j
  (Histogram[Magnitude[I,j]])++

13

Details of Automatically Get HI

In the histogram of scaled mags, find the Point that exceeds Percent of all, mark that as HI.

14

Complete Canny Algorithm

Part One: Compute Gradient Magnitude
Part Two: Compute Peaks
Part Four: Automatically Computer HI and LO
Part Three: Double Threshold

15