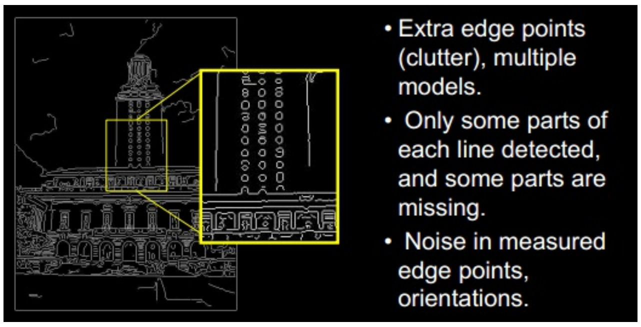
2B-L1 Hough transform - Lines

2017/11/12 16:50

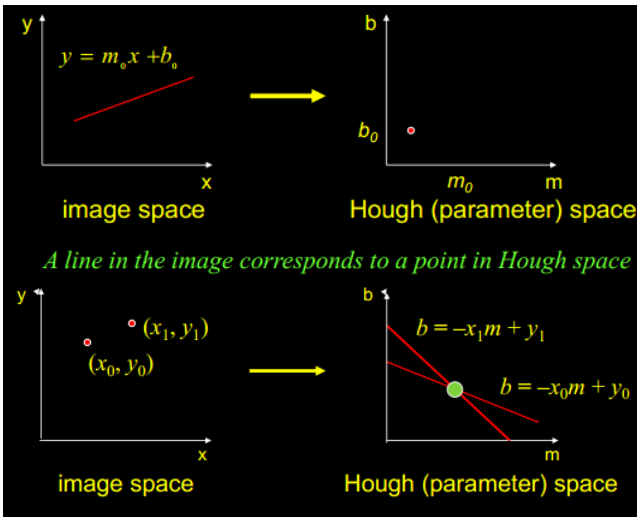
1. SUM
2. Intro
    a. previously, those are Image processing: F: I(x, y) --> I'(x,y)
    b. next, focus on real vision: F: I(x,y) --> good stuff
        i. e.g. Fitting a model
            1. line
            2. circle
            3. car
        ii. now to know how to find an simple obj in an image, later we'll learn how to find an arbitrary shape
3. Parametric Model
    a. Now, we're goint to find parametric models
    b. parametric model
        i. **A parametric model can represent a class of instances** where each is defined by a value of the parameters.
        ii. Examples include lines, or circles, or even a parameterized template.
    c. **Issues with Fitting a parametric model**
        i. how to choose a parametric model to represent a set of features
        ii. Membership criterion is not local:
            1. Can't tell whether a point in the image belongs to a given model just by looking at that point you have to look around or even the whole image
        iii. Computational complexity is important
            1. Not feasible to examine possible parameter setting
4. Line Fitting
    a. lines are important component to obj
    b. difficulty of line fitting



- Extra edge points (clutter), multiple models.
- Only some parts of each line detected, and some parts are missing.
- Noise in measured edge points, orientations.

5. Voting
    a. background

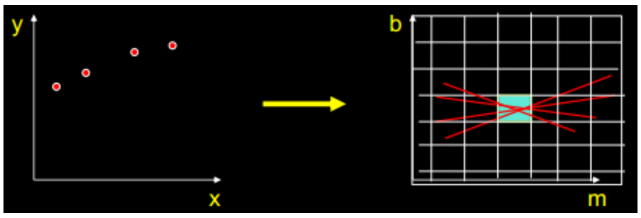
        i. It’ s not feasible to check all possible models or all combinations of features (e.g. edge pixels) by fitting a model to each possible subset.
    b. Voting is a general technique where we let the features vote for all models that are compatible with it.
        i. Cycle through features,e.g. edge pixels, each casting votes for model parameters.
        ii. Look for model parameters that receive a lot of votes
    c. why it works
        i. Noise & clutter features will cast votes too, but typically their votes should be inconsistent with the majority of “good” features.
            1. i.e. the voting is random which will be filter out
        ii. Ok if some features not observed, as model can span multiple fragments.
            1. it’s ok if part of obj is not observed, as long as the rest part is.
    d. Fitting lines
        i. To fit lines we need to answer a few questions:
            1. •Given points that belong to a line, what is the line?
                a. how to define a line in an image?
                b. the lecture solves this problem
            2. •How many lines are there?
            3. •Which points belong to which lines?
    e. the method -Hough Transform
        i. is a voting technique that can be used to answer all of these questions
        ii. Main idea
            1. Each edge point votes for compatible lines.
            2. Look for lines that get many votes.
6. Hough Space
    a. Hough space is based on m & b, while the ordinary space is based on x & y. so there’s a duality: a line in xy space is a point in Hough space while a point in xy space is line in hough space



b. from the image above, we can say if these two points define a line, then m&b must be in the intersection of the two lines in Hough space. So here comes the **Hough algorithm**

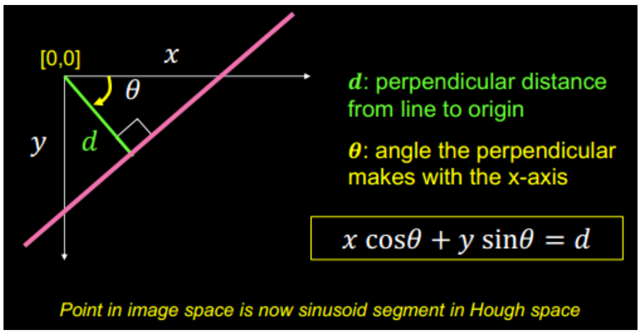
i. • Let each edge point in image space **vote for a set of possible parameters in Hough space**

ii. **Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image space.**



c. But this model has a problem with the vertical line, where m is infinite. So we let polar representation which is more robust to address this problem.

1. Polar Representation for Lines


[0,0]
x
θ
y
d
d: perpendicular distance from line to origin
θ: angle the perpendicular makes with the x-axis
$x\cos\theta + y\sin\theta = d$
Point in image space is now sinusoid segment in Hough space

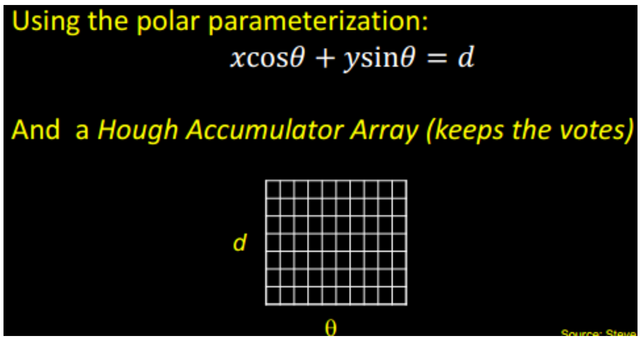
a. about the theta and d range, all possible

i. if d is non-negative, then theta shoud be[0, 360]

ii. if d is R, then theta is [0, 180]

iii. for specfic case that the line has to been in the image where the top left corner is the origin, then theta is [0, 90]

1. Basic Hough Transform Algorithm

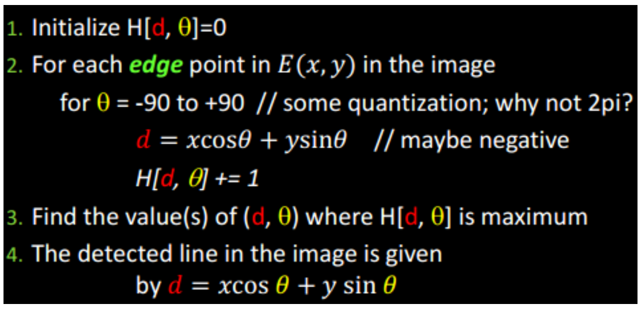


Using the polar parameterization:

$$x\cos\theta + y\sin\theta = d$$

And a *Hough Accumulator Array (keeps the votes)*

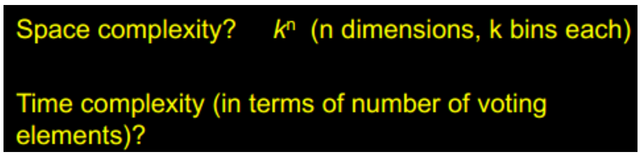
a. the thing you have to decide is the #bins



```
1. Initialize H[d, θ]=0
2. For each edge point in $E(x, y)$ in the image
       for θ = -90 to +90  // some quantization; why not 2pi?
            $d = x\cos\theta + y\sin\theta$   // maybe negative
            $H[d, \theta]$ += 1
3. Find the value(s) of (d, θ) where H[d, θ] is maximum
4. The detected line in the image is given
            by $d = x\cos\theta + y\sin\theta$
```

b. there should be another loop over d???

1. Complexity of the Hough Transform

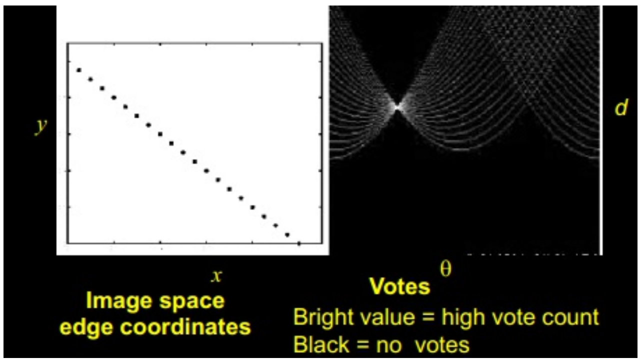


Space complexity? k^n (n dimensions, k bins each)

Time complexity (in terms of number of voting elements)?

a. space O() : i'm not clear with it
   i. the space is the place to store the voting, so it's #bins, if k bins in each dim, then it's k ˆ #dim
   ii. so for grayscaled images it's k ˆ2
b. time O(): the same as #edge pixels

1. Hough Example



11. Hough Demo Intro
   a. try to implement the Hough Algorithm all by yourself. Things will definitely go wrong and then you have to fix it. Through it, you learn the real essence of the algorithm
   b. don't use the pre-defined function or some code from the web
12. Hough Demo
   a. procedure
      i. get the edge pixels: use canny on grayscale image
      ii. get the lines: use hough(matlab)/ houghtf(octave) to get the lines

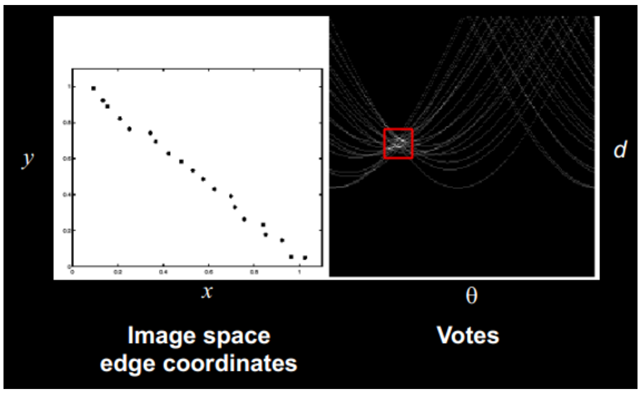
iii. find the peaks of the accumulation: houghpeaks(accum, 100) / immaximas() (octave)

iv. to get better results

1. increase the threshold for houghpeaks()
2. parameter of houghlines()

13. Impact of Noise on Hough

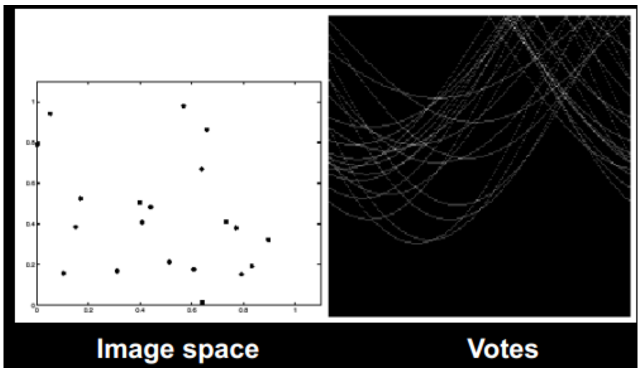
a. some noise



i. in this case, there're many peaks around the red box, which is the problem

1. solution 1: use finer bins, talk later
2. solution2: smooth the left image and find the peaks

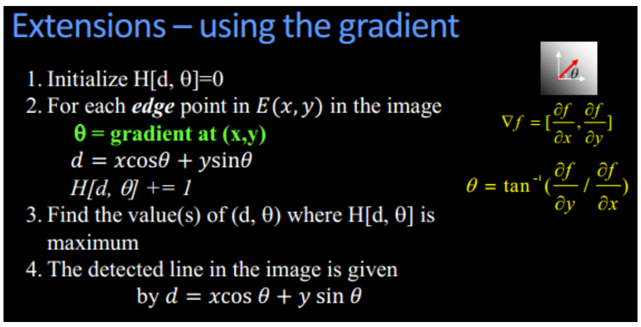
a. the image is at random



i. no real lines in fact. So the thing that really matters is to determine whether a peak is a real peak or just the accidental alignment of votes.

1. Extensions

a. Extension 1: narrow the theta range by using gradient



Extensions – using the gradient

```
1. Initialize H[d, θ]=0
2. For each edge point in $E(x,y)$ in the image
        θ = gradient at (x,y)
        $d = x\cos\theta + y\sin\theta$
        $H[d, \theta] += 1$
3. Find the value(s) of (d, θ) where H[d, θ] is maximum
4. The detected line in the image is given by $d = x\cos\theta + y\sin\theta$
```

$$\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]$$

$$\theta = \tan^{-1}(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x})$$

i. originaly theta is [-90, 90]. Now it can be a certain value or a range.

ii. the rationale

1. gradient is the direction of sharp increase of pixel value. a line should be different from the neighborhood. this direction is also w.r.t. the origin as theta of the line. so it makes sense

a. Extension 2: Give more votes for stronger edges

i. make the stronger edge pixels value more

b. Extension 3 : change the sampling of (d, θ) to give more/less resolution

i. change the bin size

ii. from corase to fine: find a region with peaks and hough this region with finer bins

iii. big bins make it easy and fast to find the peak but may omit some lines; small bins make it vulnerable to nosie but get more precise lines

c. Extension 4

i. The same procedure can be used with circles, squares, or any other shape