CS 565 Computer Vision

Nazar Khan
PUCIT

Lecture 11: Line Detection via Hough Transform

Note

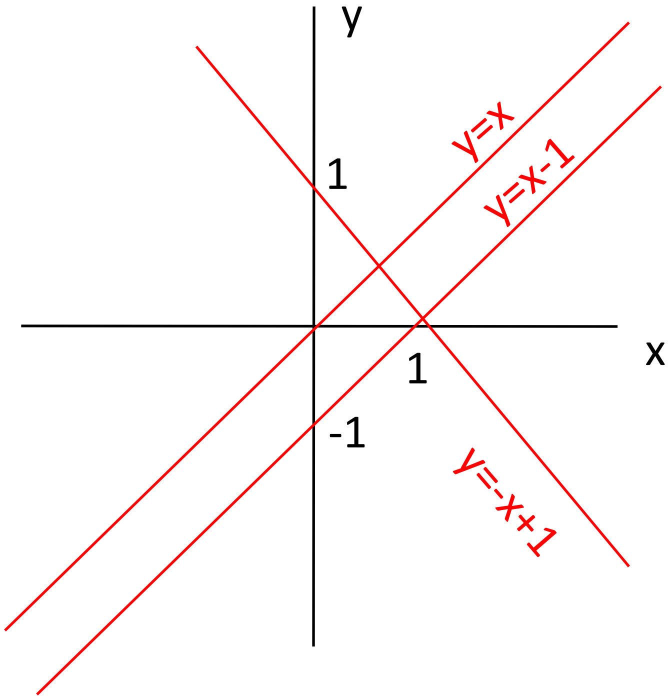
1. Missing classes/assignments/quizzes is unacceptable.
    - You will not be able to pass the exams.
2. Follow submission instructions carefully.
3. Computer Vision is a theory + practice based course. You will learn **only by implementing**.
    - Explore/verify/reject the ideas covered in class by writing small Matlab codes.
    - The lectures cover the basic ideas – implementation details are sometimes as important as the idea.
    - Some students are doing this. So don’t rationalise your laziness!

Hough Transform for Line Detection

- A powerful method for detecting curves from boundary information.
- Exploits the duality between points on a curve and parameters of the curve.
- Can detect analytic as well as non-analytic curves.

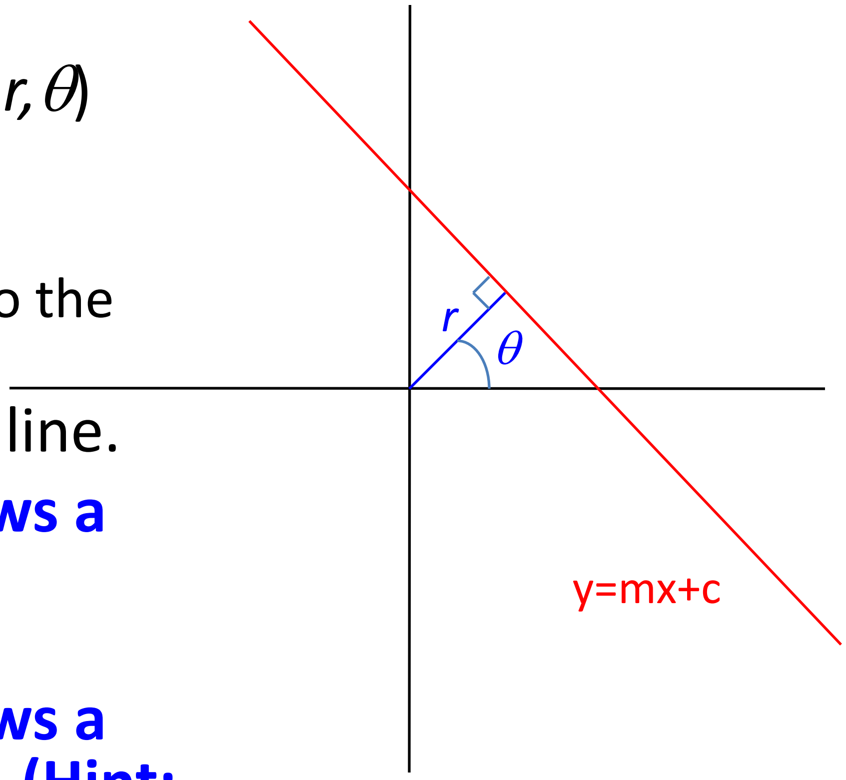
Analytic Representation of a Line

- Analytic Representation
  - Line: y=mx+c
- Every choice of parameters (*m*,*c*) represents a different line.
- This is known as the slope-intercept parameter space.
- Weakness: vertical lines have m=∞.



Polar Representation

- Solution: Polar representation (r, θ) where
  - r = distance of line from origin
  - θ = angle of vector orthogonal to the line
- Every (r, θ) pair represents a 2D line.
- **H.W. Write a function that draws a line given in Cartesian representation y=mx+c.**
- **H.W. Write a function that draws a line given in polar coordinates. (Hint: convert to Cartesian representation first.)**



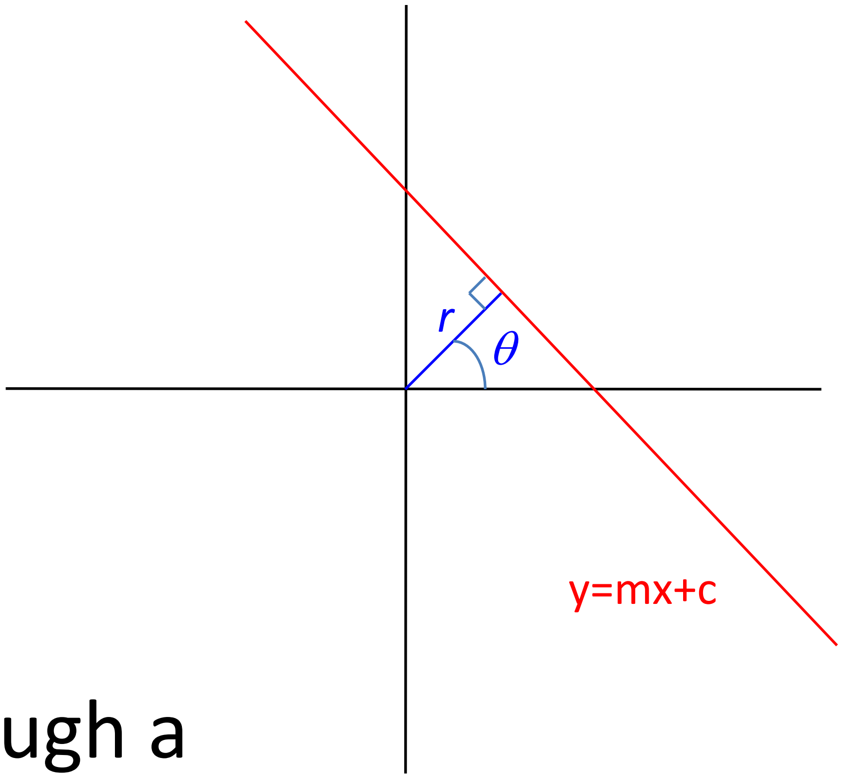
Polar Representation

- Cartesian to Polar

$$y = mx + c$$

$$y = -\frac{\cos(\theta)}{\sin(\theta)}x + \frac{r}{\sin(\theta)}$$

$$r = x\cos(\theta) + y\sin(\theta)$$



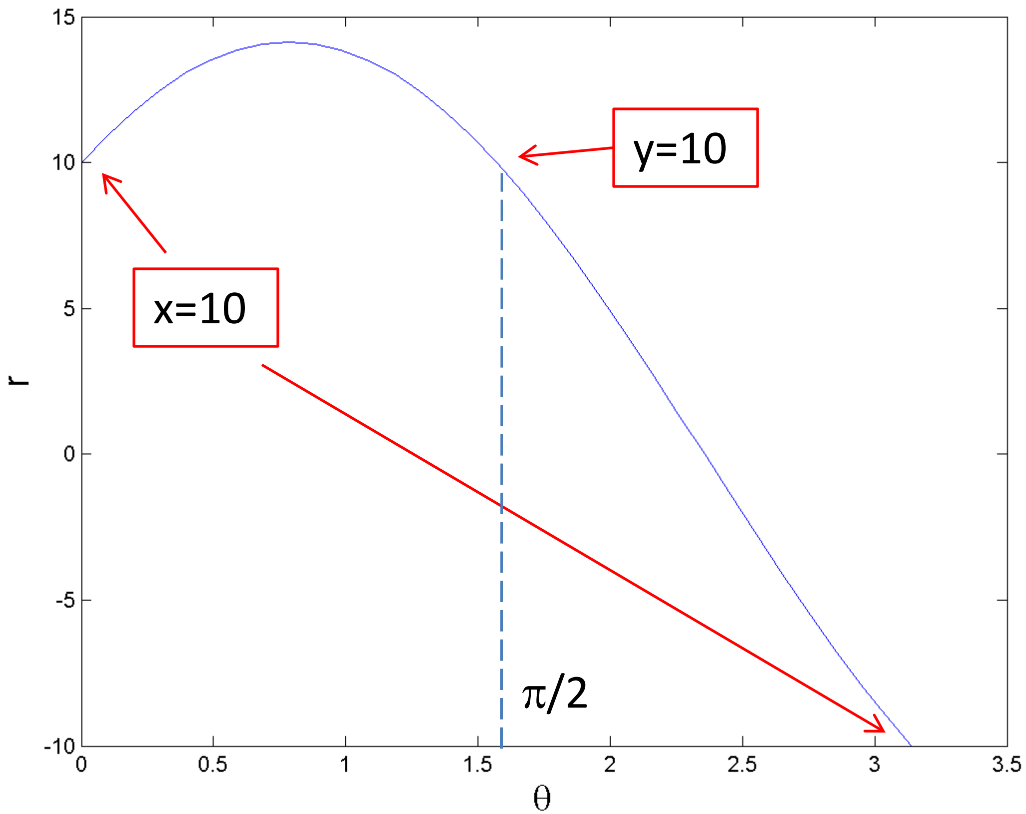
- **Key insight**: If a line through a known point (*x*,*y*) has angle θ, how can we find *r*?

Generating all possible lines through a point (x,y)

```
x=10;
y=10;
theta=0:pi/32:pi;
r=x*cos(theta)+y*sin(theta);
plot(theta,r);
```

In the space (r,θ) of polar parameters, the light blue curve represents **all lines** that can pass through the point (10,10).

We can generate lines through (x,y) by varying θ and computing the corresponding r-value.

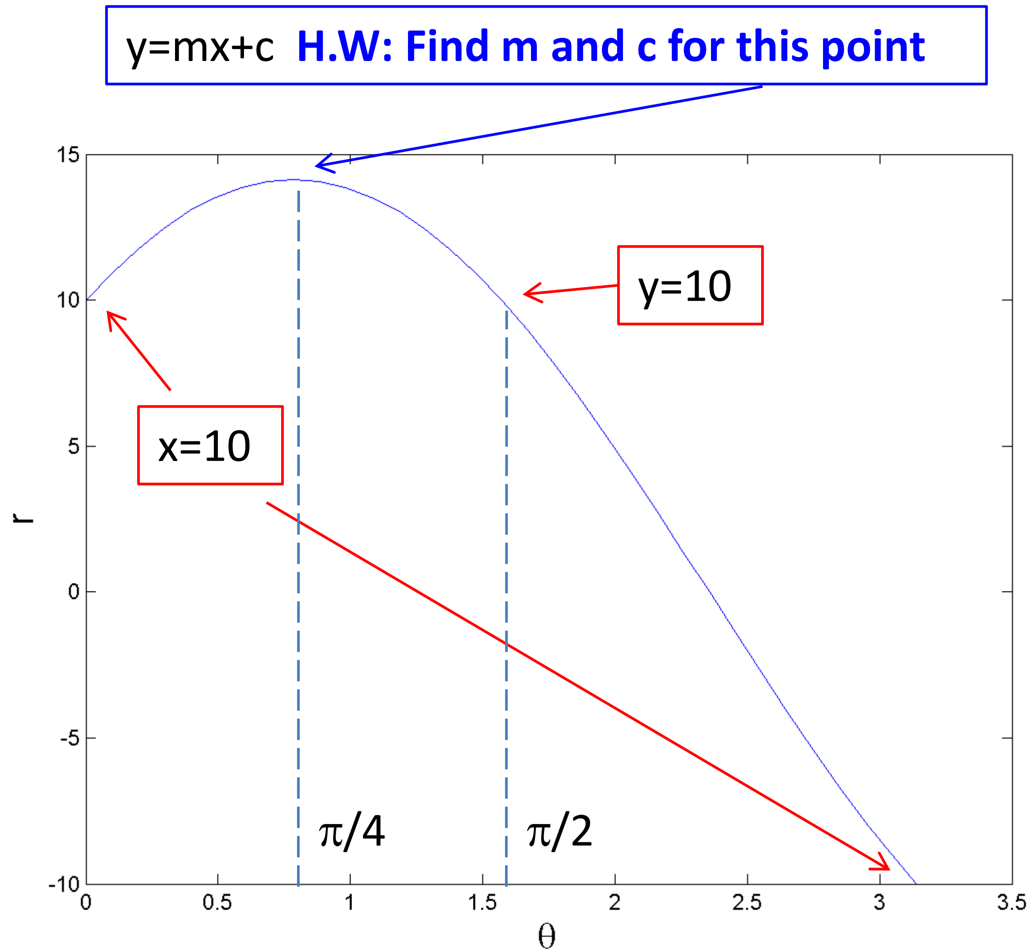


Generating all possible lines through a point (x,y)

```
x=10;
y=10;
theta=0:pi/32:pi;
r=x*cos(theta)+y*sin(theta);
plot(theta,r);
```

In the space (r,φ) of polar parameters, the light blue curve represents **all lines** that can pass through the point (10,10).

We can generate lines through (x,y) by varying φ and computing the corresponding r-value.



Hough Transform for Line Detection

- All lines going through a point (x,y) can be generated by iterating over $\theta=[0,\pi]$ and computing the corresponding $r(\theta)$.
  - That is, all lines going through a point (x,y) satisfy $r(\theta) = x.\cos(\theta) + y.\sin(\theta)$.
- So given any edge point (x,y), iterate over $\theta=[0,\pi]$ and generate the pair $(r(\theta), \theta)$.
  - The point (x,y) votes for all lines $(r(\theta),\theta)$ that pass through it.
- Valid lines can be detected by thresholding the votes.

Hough Transform for Line Detection

Pseudocode

1. initialise 2D (vote) **accumulator array** *A* to all zeros.
2. for every edge point (x,y)
3. for $\theta = 0$ to π
    1. compute $r = x.\cos(\theta) + y.\sin(\theta)$
    2. increment $A(r, \theta)$ by 1 $\leftarrow$ vote of point (x,y) for line (r,ϕ)
4. valid lines are where $A >$ threshold

Hough Transform for Line Detection

Detailed Pseudocode

1. range_θ = 360 degrees
2. binsize_θ = 1 degree (for example)
3. size_θ = ceil(range_θ / binsize_θ)
4. range_r = 2 * maximum possible r value in image + 1
5. binsize_r = 1 pixel (usually)
6. size_r = ceil(range_r / binsize_r)
7. initialise 2D **accumulator array** *A* of size (size_r, size_θ) to all zeros.
8. for every edge point (x,y)
    a) for $\theta = -\pi$ to π
        i. compute $r = x.\cos(\theta) + y.\sin(\theta)$
        ii. *r_ind* ← array index corresponding to r
        iii. θ*_ind* ← array index corresponding to θ
        iv. increment *A*(*r_ind*, θ*_ind*) by 1 ← vote of point (x,y) for line (r,ϕ)
9. valid lines are local maxima of *A* and where *A* > threshold

Hough Transform

- **Improvement 1**: After edge detection, we already know the gradient direction at (*x*,*y*).
  - So there is no need to iterate over all possible $\theta=[0,\pi]$. Use the correct θ from the gradient direction.
- **Improvement 2**: Smooth the accumulator array *A* to account for uncertainties in the gradient direction.

Hough Transform for Circle Detection

- Analytic representation of circle of radius r centered at (a,b) is $(x-a)^2+(y-b)^2-r^2=0$
- Hough space has 3 parameters (a,b,r)

For every boundary point (x,y)

    For every (a,b) in image plane

        Compute r(a,b)

        Increment A(a,b,r) by 1

A>threshold represents valid circles.

> What if we know the gradient direction at (x,y)?

Hough Transform for Circle Detection

- If we know the gradient direction g(x,y) at point (x,y), then we also know that the center (a,b) can only lie along this line
- Hough space still has 3 parameters (a,b,r) but we search for r over a 1D space instead of a 2D plane.

For every boundary point (x,y)

    For every (a,b) along gradient direction g(x,y)

        Compute r

        Increment A(a,b,r) by 1

A>threshold represents valid circles.

Hough Transform

- Any analytic curve (represented in the form f(x)=0) can be detected using the Hough transform.
  - LINE: $r = x\cos\theta + y\sin\theta$
  - CIRCLE: $x_0 = x - r\cos\theta$ where θ is gradient direction
    $y_0 = y - r\sin\theta$
  - ELLIPSE: $x_0 = x - a\cos\theta$ where θ is gradient direction
    $y_0 = y - b\sin\theta$
  - GENERAL: f(**x**, **params**) = 0

Hough Transform

- Hough space $param_1$ x $param_2$ x … x $param_N$ becomes very large when number of parameters N is increased.
- Using orientation information g(x,y) in addition to positional information (x,y) leads to a smaller search space.