Hough Transform

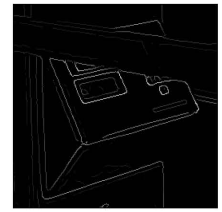
Feature Engineering

Xiaohui Yuan

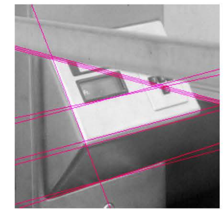
Associate Professor
Department of Computer Science and Engineering
University of North Texas
xiaohui.yuan@unt.edu

Line Detection

Original



Edge Detection

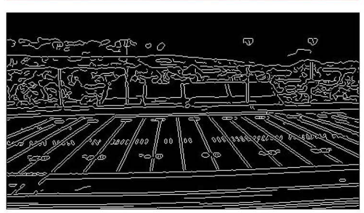


Detected Lines

Parameter Space

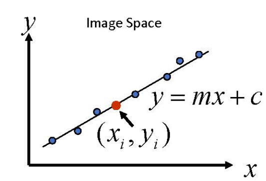
Hough Transform

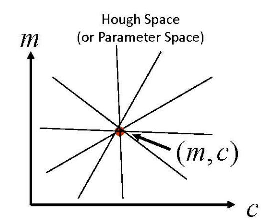
- Detect objects with a shape that has explicit mathematical expression
  - Disconnected boundaries
  - Partially visible objects
  - Distortions (noise, overlaps, etc.)
- Hough transform maps an edge image into a parameter space and uses voting to identify plausible objects/models
  - It is infeasible to check all combinations of features by fitting a model to each possible subset.
  - Hough transform let features vote for models and look for models (parameters) that receive many votes.
  - Noise and clutters also cast votes. But their votes are mostly inconsistent with the majority of the "good" features.

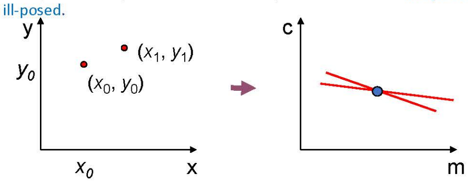


Line Detection using Hough Transform

- A line can be expressed as a function of (x, y): $y = mx + c$
- Given a set of points (x_i, y_i) sampled from this line, the problem becomes estimating parameters m and c.
- For a point (x_i, y_i), the lines go through it are $c = -x_i m + y_i$
  - An infinite number of lines go through this point (x_i, y_i)
    - A point in the image space corresponds to a line in the **Hough space**
  - We just need two such equations to decide the parameters m and c
    - Unfortunately, we usually have more than two points and, hence, the problem is usually ill-posed.





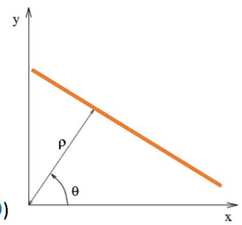


Hough Transform Algorithm

- Problems with the (m, c) space (i.e., using $y = mx + c$ as a line model):
  - Unbounded parameter space
  - Vertical lines require infinite m
- An alternative formulation is the polar representation

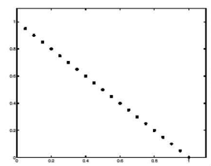
$$x\cos\theta + y\sin\theta = \rho$$

- Each point on this line is represented with a pair of polar coordinates (ρ, θ)
  - The parameter space: $\theta \in [0, \pi)$ and $\rho < k$, k is the half length of the diagonal line of the image
- Hough transform algorithm
  1. Initialize an accumulator matrix H with zeros
  2. For each point (x, y) in the image
     For θ = 0 to 180
       compute $\rho = x\cos\theta + y\sin\theta$
       $H(\theta,\rho) = H(\theta,\rho) + 1$
     end
     end
  3. Find the value(s) of (θ, ρ), where $H(\theta,\rho)$ is a local maximum
  4. The detected line is given by $\rho = x\cos\theta + y\sin\theta$

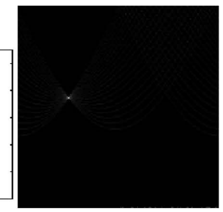


	1						1		
		1				1			
			1		1				
				2					
			1		1				
		1				1			
	1						1		

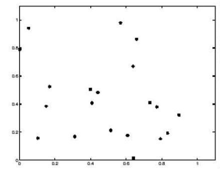
Examples of Hough Transform Results



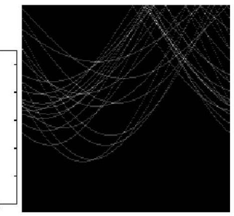
Input space



Hough space



Input space

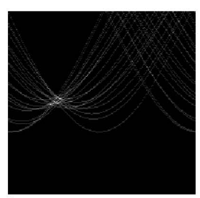


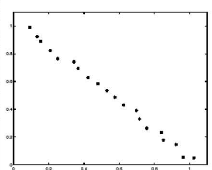
Hough space

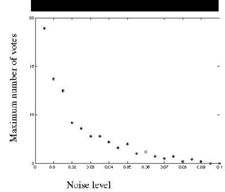
Horizontal axis is θ and vertical axis is ρ

Implementation Issues

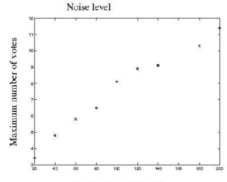
- How many lines?
  - Count the peaks (local maxima) in the Hough space
  - Treat adjacent peaks as a single one
- How to decide the quantization bin of the accumulator?
  - big bin merges multiple lines into one
  - small bin leads to less robustness to noise; the votes are not strong enough to pass the threshold





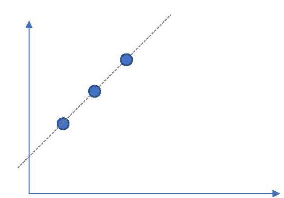


When noise increases, fewer votes land in a single bin



Having more clutters/noise points increases the number of bins with false peaks.

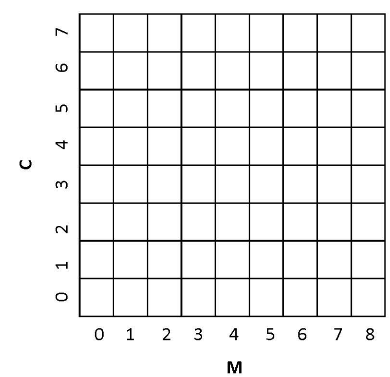
Practice



x	y
1	2
2	3
3	4

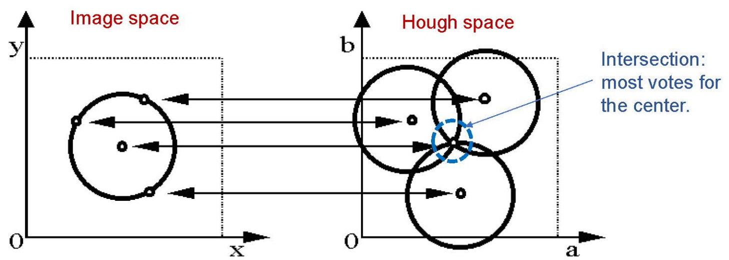
$$c = -1m + 2$$
$$c = -2m + 3$$
$$c = -3m + 4$$

$$c = -x_i\, m + y_i$$

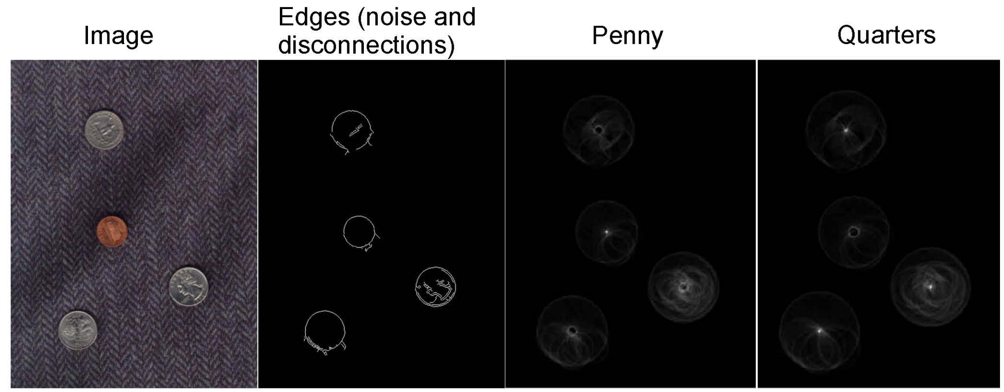


What About Detecting Circles?

- A circle is represented as $(x-a)^2+(y-b)^2=r^2$
  - Parameters a, b, and r.
- For a fixed (known) radius r, we have



Example: Finding Coins

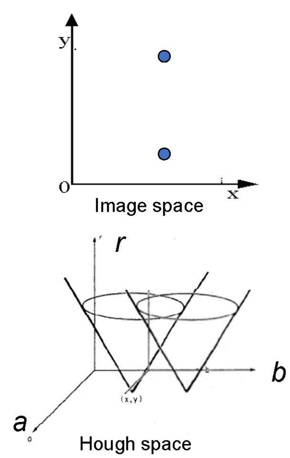


Detecting Circles with an Unknown Radius

$$(x-{\color{blue}a})^2+(y-{\color{blue}b})^2={\color{blue}r}^2$$

For circles with an unknown radius r, the Hough transform algorithm is

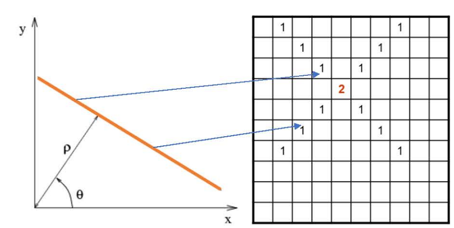
1. For every edge pixel (x, y):
2. For each radius r:
3. For each possible gradient direction θ:
4. a = x – r cos(θ) // column
5. b = y + r sin(θ) // row
6. H[a, b, r] = H[a, b, r] + 1

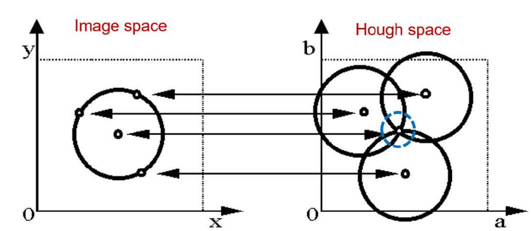


Generalized Hough Transform

– for Detecting Arbitrary Shapes!

Let's take another look at the detection of lines and circles





Generalized Hough Transform

- Describe a shape using a reference point **a**.

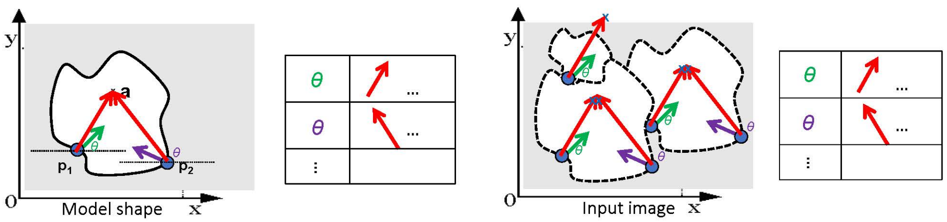
Build a Look Up Table (LUT)

- At each boundary point, compute the displacement vector: $r = a - p_i$
- Store these vectors in a table indexed by the gradient direction θ

Apply This LUT for Shape Detection

For each edge point:

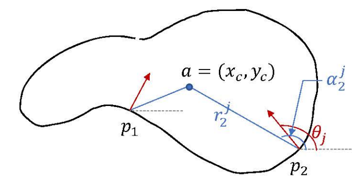
- Use its gradient θ to get the stored vector
- Use the retrieved r to compute and vote for the reference point



Generalized Shape Model

- The parameters are the angle and magnitude of the vectors
- Points of a shape model are ordered based on the gradient direction
- More than one possible vectors at each gradient direction
- Each vector v_i consists of α_i^j and r_i^j, where i is the index of the points, j is the index of the gradient direction

Direction	Vector $v_i = (r_i, \alpha_i)$
θ_1	v_1, v_7, v_{23}
...	...
θ_j	v_2
...	...
θ_J	v_{11}, v_{20}



Generalized Hough Transform Algorithm

1. Given a shape model $LUT(\theta, v)$
2. Create an accumulator $H(x_c, y_c;\ r, \alpha) = 0$
3. For each point on the edge $(x_i, y_i; \theta)$
4. For each entry $< \theta; v >= \{r_i^1, r_i^2, \ldots, r_i^k\}$ in table, compute:
$$x_c = x_i + r_i^k \cos(\alpha_i^k)$$
$$y_c = y_i + r_i^k \sin(\alpha_i^k)$$
5. Increment $H(r, \alpha) = H(r, \alpha) + 1$
6. Find local maxima in H

Generalized Hough Transform Algorithm

Unknown scale and rotation of the target shape

1. Given a shape model $LUT(\theta, R)$
2. Create an accumulator $H(r, \alpha, s, \beta) = 0$
3. For each point on the edge $(x_i, y_i; \theta)$
4. For each entry $< \theta; R >= \{r_i^1, r_i^2, \ldots, r_i^k\}$ in table, compute for all s and β:
$$x_c = x_i + s\, r_i^k \cos(\alpha_i^k + \beta)$$
$$y_c = y_i + s\, r_i^k \sin(\alpha_i^k + \beta)$$
5. Increment $H(r, \alpha, s, \beta) = H(r, \alpha, s, \beta) + 1$
6. Find Local Maxima in H

Hough Transform: Pros and Cons

- All points are processed independently
  - It is robust to occlusion and gaps
- Robustness to noise (to some extent)
  - noise points are unlikely to contribute *consistently* to any single shape (cell)
- It detects multiple instances in a single pass

- Complexity increases exponentially with the number of model parameters
- Non-target shapes can produce spurious peaks in the parameter space
- Quantization bin
  - It can be tricky to pick a good bin size