Deep Learning for Computer Vision

Hough Transform

Vineeth N Balasubramanian

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Acknowledgements

- Most of this lecture's slides are based on lectures of **Deep Learning for Vision** course taught by Prof Yannis Avrithis at Inria Rennes-Bretagne Atlantique, as well as the **Computer Vision** course taught by Prof Mubarak Shah/Alper Yilmaz at the University of Central Florida

NPTEL

Line Fitting

- We have already seen a couple of line fitting algorithms: *Least squares fit* and *RANSAC*
- How do they perform when multiple lines are present?

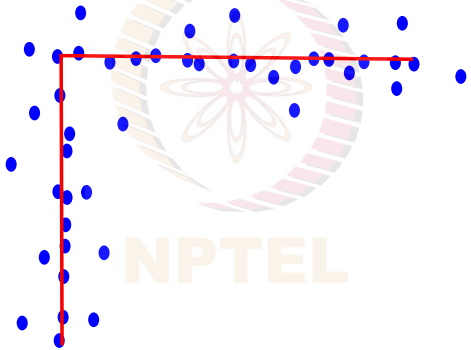


Figure 1: Example line configuration in an image.

Line Fitting: Hough Transform

- Hough, *Method and means for recognizing complex patterns*, U.S. Patent No. 3,069,654, Dec 1962
- Line equation in Cartesian co-ordinates is:
  - $y = mx + c$ m is slope, c is y-intercept
- Rearranging it slightly, we get:
  - $c = (-x)m + y$ which for a specific point (x_i, y_i) becomes $c = (-x_i)m + y_i$
- This can be thought of as the equation of line in parameter space; i.e in the (m, c) coordinate system with slope $-x_i$ and c-intercept y
- Each point in parameter space is a model

Source: Alper Yilmaz, Mubarak Shah, Fall 2011 UCF

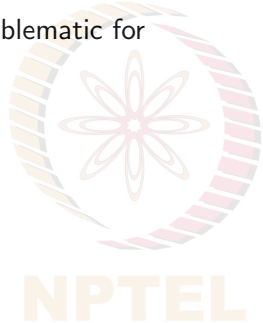
Line Fitting: Hough Transform

- N samples needed to fit a model (2 points to fit a line)
- But even one sample brings some information
- In the space of all possible models, vote for ones that satisfy a given sample
- Collect votes for all samples, and seek for consensus

Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

Hough Transform: Polar Parametrization

- The Cartesian formulation is problematic for vertical lines. Why?



Hough Transform: Polar Parametrization

- The Cartesian formulation is problematic for vertical lines. Why?
  - The slope is unbounded for vertical lines.
- Consider a polar parametrization of the line:
  - $\rho = x \cos\theta + y \sin\theta$
  - ρ is distance of line from origin and θ is angle made by normal to x-axis
  - For given line, $\rho \geq 0$ and $0 \leq \theta \leq 360^\circ$ are bounded

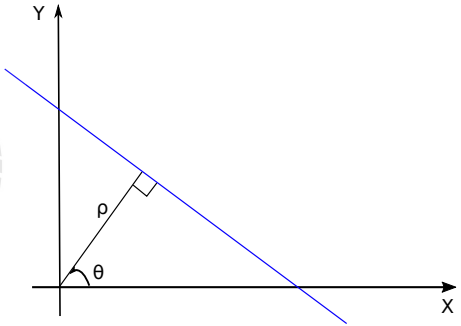


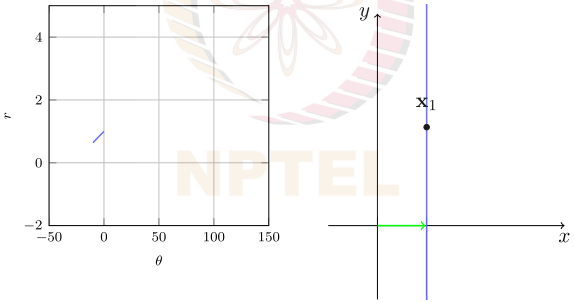
Figure 2: The ρ, θ parametrization.

Source: Alper Yilmaz, Mubarak Shah, Fall 2011 UCF

Hough Transform: Polar Parametrization

- A point (x_i, y_i) '*votes*' for many points in parameter space $\rightarrow$ **Hough Voting**
- Each line through a point (x_1, y_1) is a vote for a point in parameter space which satisfies $\rho = x_1 \cos\theta + y_1 \sin\theta$

voting in parameter space

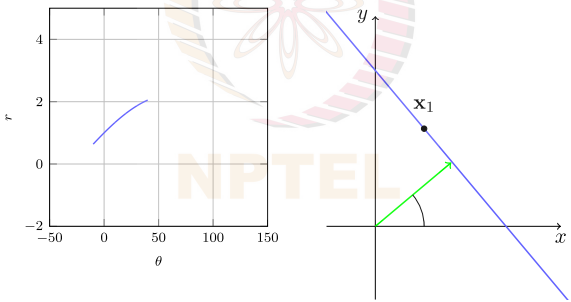


Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

Hough Transform: Polar Parametrization

- A point (x_i, y_i) '*votes*' for many points in parameter space $\rightarrow$ **Hough Voting**
- Each line through a point (x_1, y_1) is a vote for a point in parameter space which satisfies $\rho = x_1 \cos\theta + y_1 \sin\theta$

voting in parameter space

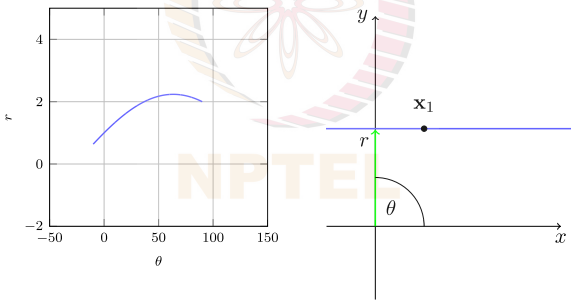


Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

Hough Transform: Polar Parametrization

- A point (x_i, y_i) '*votes*' for many points in parameter space $\rightarrow$ **Hough Voting**
- Each line through a point (x_1, y_1) is a vote for a point in parameter space which satisfies $\rho = x_1 \cos\theta + y_1 \sin\theta$

voting in parameter space

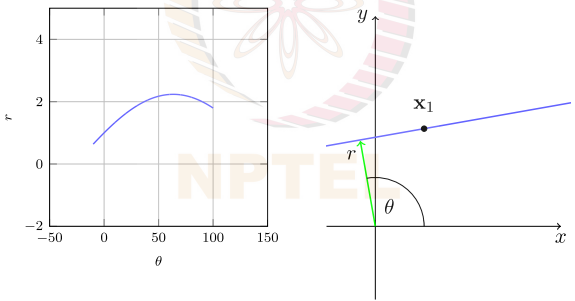


Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

Hough Transform: Polar Parametrization

- A point (x_i, y_i) '*votes*' for many points in parameter space $\rightarrow$ **Hough Voting**
- Each line through a point (x_1, y_1) is a vote for a point in parameter space which satisfies $\rho = x_1 \cos\theta + y_1 \sin\theta$

voting in parameter space

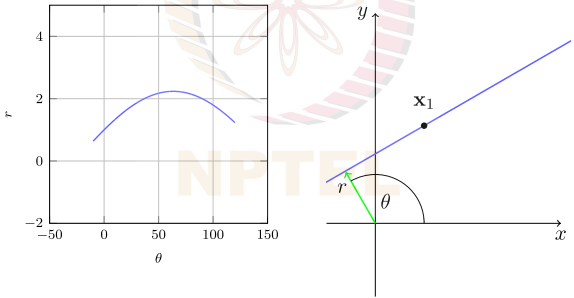


Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

Hough Transform: Polar Parametrization

- A point (x_i, y_i) '*votes*' for many points in parameter space $\rightarrow$ **Hough Voting**
- Each line through a point (x_1, y_1) is a vote for a point in parameter space which satisfies $\rho = x_1 \cos\theta + y_1 \sin\theta$

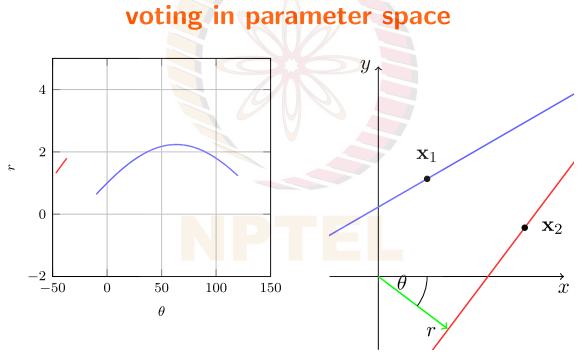
voting in parameter space



Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

Hough Transform: Polar Parametrization

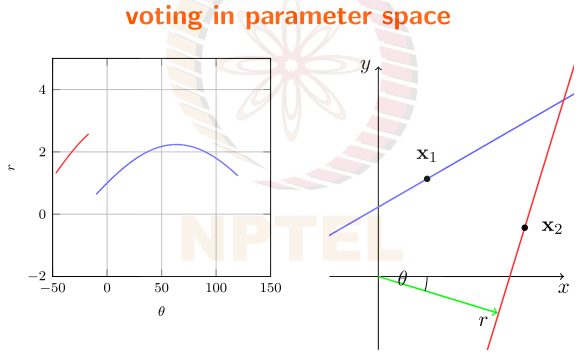
- Each line through a point (x_2, y_2) is a vote for a point in parameter space which satisfies $\rho = x_2 \cos\theta + y_2 \sin\theta$



Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

Hough Transform: Polar Parametrization

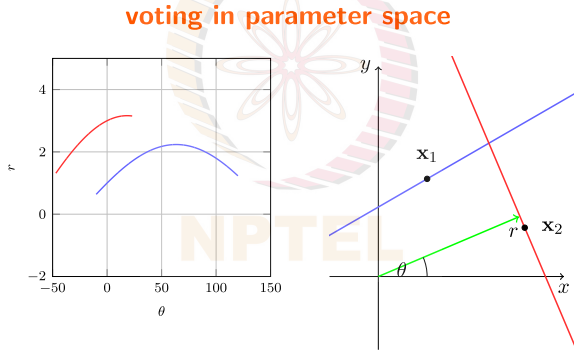
- Each line through a point (x_2, y_2) is a vote for a point in parameter space which satisfies $\rho = x_2 \cos\theta + y_2 \sin\theta$



Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

Hough Transform: Polar Parametrization

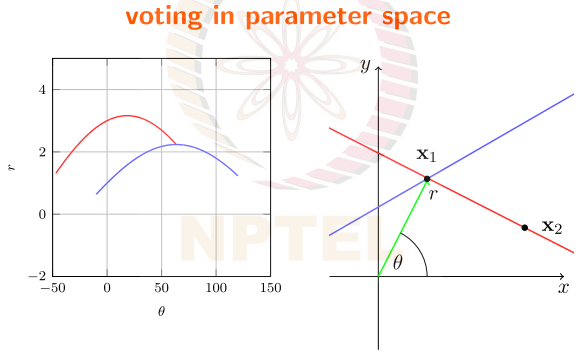
- Each line through a point (x_2, y_2) is a vote for a point in parameter space which satisfies $\rho = x_2 \cos\theta + y_2 \sin\theta$



Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

Hough Transform: Polar Parametrization

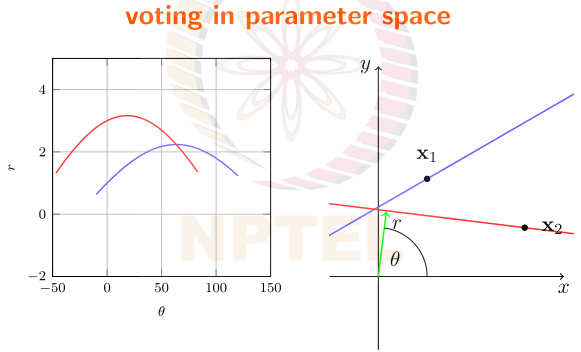
- Each line through a point (x_2, y_2) is a vote for a point in parameter space which satisfies $\rho = x_2 \cos\theta + y_2 \sin\theta$



Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

Hough Transform: Polar Parametrization

- Each line through a point (x_2, y_2) is a vote for a point in parameter space which satisfies $\rho = x_2 \cos\theta + y_2 \sin\theta$



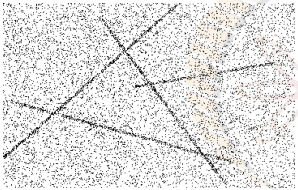
Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

Hough Voting

procedure HOUGH VOTING(X, Θ)
    X: data Θ: quantized parameter $\theta_{min}, ..., \theta_{max}$
    A: accumulator array, initially zero
    for $(x, y) \in X$ **do**
        for $\theta \in \Theta$ **do**
            $\rho = x \cos\theta + y \sin\theta$
            ▷ for each set of model parameters consistent with a sample, increment A
            $A[\theta, \rho] = A[\theta, \rho] + 1$
        end for
    end for
    Non-maximum Suppression: detect local maxima in A
end procedure

Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

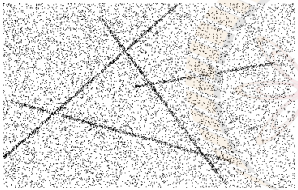
Line Detection



points

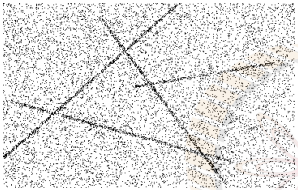
Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

Line Detection

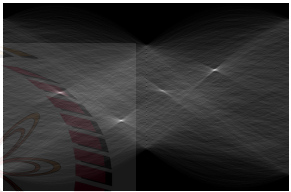


Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

Line Detection



points

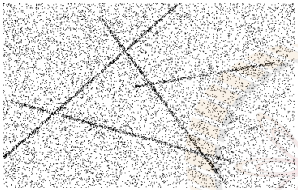


accumulator

local maxima

Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

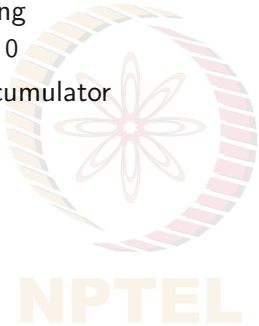
Line Detection



Source: Yannis Avrithis, Deep Learning for Vision, Spring 2019 SIF

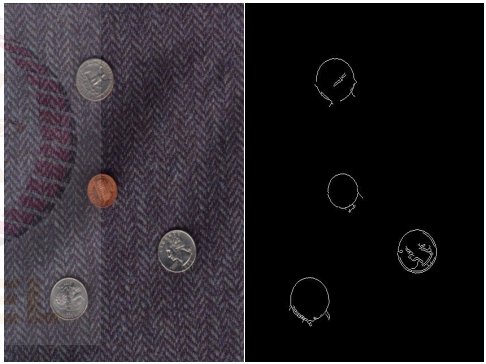
Hough Transform: Finding Circles

- Circle fitting similar to line fitting
  - $(x - x_0)^2 + (y - y_0)^2 - r^2 = 0$
- What are the dimensions of accumulator A for circle fitting?



Hough Transform: Finding Circles

- Circle fitting similar to line fitting
  - $(x - x_0)^2 + (y - y_0)^2 - r^2 = 0$
- What are the dimensions of accumulator A for circle fitting?
  - 3D accumulator with dimensions x_0, y_0, r
- Fix one of the parameters (generally radius is fixed) and loop for the rest
- Increment accumulator A
- Find local maxima in A



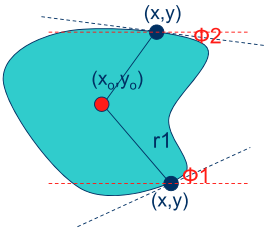
Source: Alper Yilmaz, Mubarak Shah, Fall 2011 UCF

Source: Ioannis Gkioulekas, 16-385 Computer Vision, Spring 2020, CMU

Generalized Hough Transform

- Used for shapes with no analytical expression
- Involves a training phase where R-table is computed
- Given object of interest, compute R-table as follows:
  - Compute centroid (x_c, y_c).
  - For each edge point (x_i, y_i), compute distance to centroid r_i and find edge orientation ϕ_i.
  - Construct a table of angles and r-values

Φ1	r1, r2, r3 …
Φ2	r14, r21, r23 …
Φ3	r41, r42, r33 …
Φ4	r10, r12, r13 …



Source: Alper Yilmaz, Mubarak Shah, Fall 2011 UCF

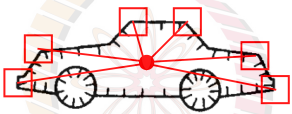
Generalized Hough Transform

procedure GENERALIZED HOUGH TRANSFORM($X, R, A[x_c, y_c]$)
    X: data R: R-table $A[x_c, y_c]$: Accumulator array with quantization $x_{c_{\min}}, \cdots, x_{c_{\max}}$ and $y_{c_{\min}}, \cdots, y_{c_{\max}}$
    A: accumulator array, initially zero
    for $(x, y) \in X$ **do**
        for each $(r_i, \phi_i) \in R$ **do**
            $x_c = x + r_i \cos \phi_i$
            $y_c = y + r_i \sin \phi_i$
            $A[x_c, y_c] = A[x_c, y_c] + 1$
        end for
    end for
    Non-maximum Suppression: detect local maxima in A
end procedure

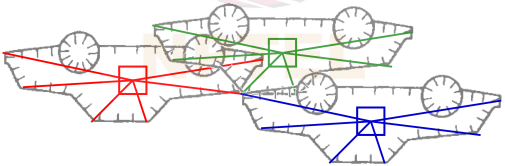
NPTEL

Generalized Hough Transform: Example

- **Build model:** Record coordinates relative to reference point
- **Test phase:** Each point votes for all possible coordinates of reference point



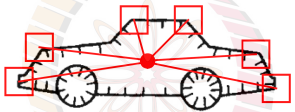
Model image



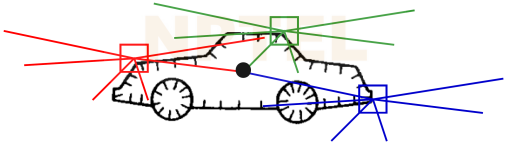
Test image

Generalized Hough Transform: Example

- **Build model:** Record coordinates relative to reference point
- **Test phase:** Each point votes for all possible coordinates of reference point

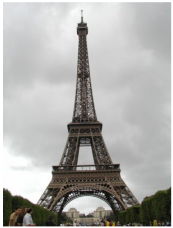


Model image



Test image

Generalized Hough Transform: Example



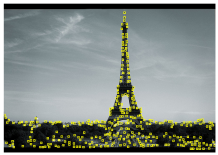
Model image

Test image

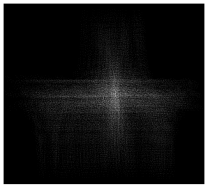
NPTEL

Generalized Hough Transform: Example

Model image points



Test image points



Accumulator

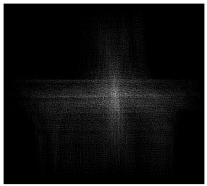
Local Maxima

NPTEL

Generalized Hough Transform: Example

Model image points

Test image with location



Accumulator

Local Maxima

NPTEL

Hough Transform

- Can be used for detection of:
  - shapes
  - objects, including multiple instances
- Advantages
  - Deals with occlusion well
  - Robust to noise
- Disadvantages
  - Can be computationally expensive
  - Setting parameters is not easy

Source: Ioannis Gkioulekas, 16-385 Computer Vision, Spring 2020, CMU

Homework

Readings

- Chapter 4.3, Szeliski, *Computer Vision: Algorithms and Applications*

Questions

- How would you use Hough transform to detect ellipses, squares and rectangles?
- Your friend working in a diagnostics startup asks you how to use Hough transform to automatically count Red Blood Cells in a blood sample. What would you advise your friend?

References


Richard O. Duda and Peter E. Hart. "Use of the Hough Transformation to Detect Lines and Curves in Pictures". In: *Commun. ACM* 15.1 (Jan. 1972), 11–15.


John Illingworth and Josef Kittler. "A survey of the Hough transform". In: *Computer vision, graphics, and image processing* 44.1 (1988), pp. 87–116.


Richard Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. London: Springer-Verlag, 2011.