Machine vision
Summary # 7: Hough Transform

• The Hough transform was introduced by Hough in 1962 (patent) to detect straight lines in an image. It was later extended to more complex shapes such as circles and ellipses.
• In 1972, a new formulation was introduced by Duda and Hart. In the 1980s, the transform became very popular in the machine vision community.

Principle of the method
Consider a point in an image. How many lines pass through this point? The answer is an infinite number of lines. Each line is characterized by an equation of the form:

\[ y = mx + b \]  

where
- \((x, y)\): coordinates of the pixel.
- \(m\): is the slope
- \(b\): y-intercept

It is possible to write equation (1) as follows

\[ b = -xm + y \]  

If we consider the \((m, b)\) parameter space where \(x\) and \(y\) are constants, equation (2) represents an equation of a straight line (in the \((m, b)\) space). So, a single point in the image space \((x, y)\) is a straight line in the \((m, b)\) space. The \((m, b)\) parameter space is called the Hough space. The following result summarizes the Hough transform.

Result
If \(N\) straight lines in the Hough space that correspond to \(N\) given pixels in the image space intersect at a given point, then those \(N\) pixels reside on the same straight line in the image coordinates. The parameters of that line correspond to the Hough coordinates \((m, b)\) of the point of intersection.

Modified Hough transform
The Hough transform converts pixels into lines in the \((m, b)\) space. This slope-intercept representation of a line has a problem: \(m\) becomes very large and the slope tends to infinity for a vertical or near vertical line. In order to solve this problem, Duda and Hart suggested a new approach that is more practical. The modified approach uses polar normal form as shown in figure 1. The equation of a line in the polar normal form is given by

\[ \rho = x \cos \theta + y \sin \theta \]  

where
- \(\rho\): perpendicular distance from the origin to the line.

\[ \theta: \text{angle of the perpendicular line to the x-axis.} \]

Now the Hough space is the \((\rho, \theta)\) space. A comparison between the \((m, b)\) space and the \((\rho, \theta)\) space is shown in figure 2.

What are the limitations of the Hough transform? For a line, the coordinates space has two variables and the parameters space also has two variables \((m, b)\) or \((\theta, \rho)\). For more complicated shapes, the number of parameters increases. For more complicated models, the method is computationally expensive. The example of a circle is shown below.

Hough transform for detecting circles
The equation of a circle is given by

\[ (x - a)^2 + (y - b)^2 = r^2 \]  

Clearly, this equation has two variables and three parameters \((a, b, r)\) describing the circle center and radius. The parametric equations for the circle in polar coordinates are

\[ x = a + r \cos \theta \]  
\[ y = b + r \sin \theta \]  

which correspond to a cone when \(r\) is a variable. When the radius is known, we have two parameters and two variables. We can write:

\[ a = -r \cos \theta + x \]  
\[ b = -r \sin \theta + y \]

1) Each point \((x, y)\) on the circle corresponds to a circle in the \((a, b)\) space.
Different points in the image with the same slope \( m \) and y-intercept \( b \).

**Fig. 2. Illustration of the \((m, b)\) and \((\theta, \rho)\) parameter spaces**

2) Point \((x, y)\) is the center of the circle in the \((a, b)\) space.

3) If we sweep over every point in the circle, we will have many circles in the \((a, b)\) space.

4) The intersection of these circles corresponds to the center of the circle of the image.

This is illustrated in figure 3.

**A. Example**

For the image of figure 4

- Find the straight line representation in the \((m, b)\) space
- Find the straight line representation in the \((\theta, \rho)\) space

The points are given by

- \(A(1, 4)\)
- \(B(2, 3)\)
- \(C(3, 2)\)
- \(D(4, 1)\)

For point \(A\):

\[
b = -mx + y \quad (9)
\]
\[
b = -m1 + 4 \quad (10)
\]

For point \(A\), we have

- \(m = 0 \Rightarrow b = 4\)
- \(m = 4 \Rightarrow b = 0\).

Based on these calculations we can draw a line in the parameter space \((m, b)\) that represents point \(A\) as shown in figure 4. Same thing can be done with the other points. In the \((\theta, \rho)\) space:

\[
\rho = x \cos \theta + y \sin \theta \quad (11)
\]

The goal is to find \(\theta\) and \(\rho\). We take points \((0, 5)\) and \((5, 0)\).

\[
\rho = 5 \sin \theta \Rightarrow \text{(for point \((0, 5)\))} \quad (12)
\]
\[
\rho = 5 \cos \theta \Rightarrow \text{(for point \((5, 0)\))} \quad (13)
\]

The solution in the parameters space \((\theta, \rho)\) is \(\theta = 45, \rho = 3.53\). Figure 4 shows that \((m, b) = (-1, 5)\) and \((\theta, \rho) = (45, 3.5)\).

The Hough transform was originally introduced to detect lines. The Hough transform can be generalized to detect more complex shapes. Consider figure 5–left. In this figure, there is a straight line and three aligned points. The Hough transform is shown in figure 5–right. The three aligned points do not constitute a line, therefore, their intersection in the Hough space should not be taken for a straight line. In order to decide about which points constitute a line, we use a voting process.

If the number of votes for a given point in the \((\theta, \rho)\) space is greater than a threshold \(T_h\), then this point is considered to represent a line. The algorithm forms an accumulator array to hold the locations of the coordinates in the Hough space. This is summarized in the following algorithm.

**The accumulator algorithm**

Line detection can be carried out by the following steps

1) Create a 2D array \(H(\rho, \theta)\) for the parameter space

2) Find the gradient of the image

\[
G(x, y) = |G(x, y)| \angle G(x, y) \quad (14)
\]

The gradient is used to detect edges.

3) For any pixel satisfying \(|G(x, y)| > T_s\) for all \(\theta\)

\[
\rho = x \cos \theta + y \sin \theta \quad (15)
\]
\[
H(\rho, \theta) = H(\rho, \theta) + 1 \quad (16)
\]

4) In the parameter space, any element \(H(\rho, \theta) > T_h\) represents a straight line in the image.

**Parametric equation of a shape**

The equation of a line under normal form is

\[
\rho = x \cos \theta + y \sin \theta \quad (17)
\]

This equation can be written under the general form as follows

\[
f(x, y, \rho, \theta) = 0 \quad (18)
\]

A generalization of this equation to any shape can be written as follows

\[
f(x, y, \alpha_1, \alpha_2, ..., \alpha_n) = 0 \quad (19)
\]
Fig. 3. Each point on the circle in the \((x, y)\) space corresponds to circle in the parameter space. The intersection of the circles in the parameter space \((a, b)\) correspond to the center in the original image. In this case \((a, b) = (2, 3)\)

The gradient vector is perpendicular to the direction of the tangent. Now, using the gradient direction we can write two equations

\[
\begin{align*}
 f(x, y, \alpha_1, \alpha_2, \ldots, \alpha_n) &= 0 \\
 \tan \angle G(x,y) &= \frac{\nabla_y f(x, y, \alpha_1, \alpha_2, \ldots, \alpha_n)}{\nabla_x f(x, y, \alpha_1, \alpha_2, \ldots, \alpha_n)}
\end{align*}
\]  

(20)  
(21)

1) The shape has to be described analytically by an equation from which the relationship between pixel coordinates and parameter space is deduced.

2) The number of parameters describing the shape may be higher.

The generalized Hough transform was introduced in 1981 by Ballard to solve these problems and detect arbitrary shapes. GHT consists of two steps: preparation and detection. The preparation phase is a training phase where a look up table is constructed using a prototype shape. The lookup table defines the relationship between the boundary coordinates and
the orientation, and the Hough parameters. An illustration of the parameters is shown in figure 6. The table replaces the analytical equation describing the boundary.

- **Step 1**: preparation: this is a learning phase.
  1) Pick a reference point inside the object
  2) Draw a line from the reference point to the boundary
  3) Compute $\phi_i$ (perpendicular to the gradient direction) along the object boundary
  4) For each point on the boundary of the shape, find
    \begin{align*}
    r &= \sqrt{(x-x_c)^2 + (y-y_c)^2} \\
    \beta &= \tan^{-1}\left(\frac{y-y_c}{x-x_c}\right)
    \end{align*}
  5) Build the A-table as a function of $\phi$ (table indexed by $\phi$) Each object has its own A-table.
  6) Prepare a 2D Hough array $H(x_c, y_c)$ initialized to zero.

- **Step 2**: detection of the shape and its location in the image.
  The objective is to find the object reference point given the boundary points.
  1) For each point in the image $(x, y)$ with $|G(x, y)| > T_a$ (these points correspond to the boundary points), find the table entry and its corresponding $\phi$
  2) For each pair $(r, \beta)$ in the A-table find
    \begin{align*}
    x_c &= x + r \cos \beta \\
    y_c &= y + r \sin \beta
    \end{align*}
  and increment the corresponding element in $H$ array
  \begin{equation}
  H(x_c, y_c) = H(x_c, y_c) + 1
  \end{equation}
  All elements in the H-table satisfying $H(x_c, y_c) > T_h$ represent the locations of the shape in the image.

**Example**
This example illustrates the principle of the accumulator. The original image is shown in figure 7, the algorithm is used to detect circles. The results of the application of the generalized Hough transform in figure 7.

**Scale and rotation**
We want the shape detector to be insensitive to scale and rotation. We define a new accumulator in this case:
\begin{equation}
H(x_c, y_c, S, \theta)
\end{equation}
where $S$ is the scaling factor and $\theta$ is the rotation angle as illustrated in figure 8. The detection algorithm consists of the following steps: For each image point $(x, y)$ with $|G(x, y)| > T_s$, do the following

1) For all $S$ and $\theta$ find

$$x_c = x + rS \cos(\theta + \beta)$$

$$y_c = y + rS \sin(\theta + \beta)$$

and increment

$$H(x_c, y_c, S, \theta) = H(x_c, y_c, S, \theta) + 1$$

2) All elements in H-table that satisfy

$$H(x_c, y_c, S, \theta) > T_h$$

represent the shape location in the image with scaling factor $S$ and orientation $\theta$. 

Fig. 7. Original image, circle detection and number of votes (accumulator)

Fig. 8. Illustration of scaling and rotation