A Probabilistic Algorithm for Computing
Hough Transforms

JAMES R. BERGEN AND HAIM SHVAYTser (SCHWEITzER)*

David Sarnoff Research Center, Subsidiary of SRI International,
Princeton, New Jersey 08543-5300

Received December 1989; revised December 1990

The Hough transform is a common technique in computer vision and pattern recognition for recognizing patterns of points. We describe an efficient probabilistic algorithm for a Monte-Carlo approximation to the Hough transform. Our algorithm requires substantially less computation and storage than the standard Hough transform when applied to patterns that are easily recognized by humans. The probabilistic steps involve randomly choosing small subsets of points that jointly vote for likely patterns. © 1991 Academic Press, Inc.

1. INTRODUCTION

The Hough transform is a technique for mapping points in digital pictures into a parameter space where patterns of points are identified as peaks. As an example consider the problem of detecting lines. Points \((x, y)\) on a line are constrained by the relation:

\[ y = ax + b. \]

Equivalently, the parameter pairs \((a, b)\) of lines that pass through a point \((x, y)\) are constrained by the relation:

\[ b = -xa + y. \]  

(This is a line in the parameter space.)

In order to detect lines in a set of points, each point \((x, y)\) "votes" for all pairs \((a, b)\) that are related according to Eq. (1). Parameter pairs that get a large number of votes are likely to represent lines. In Fig. 1, A has
FIG. 1. Detecting lines: A. a picture with 6 points; B. The parameter space of A.

six points, and B is the corresponding parameter space. The two parameter pairs that get the largest number of votes (3 votes each) are: \((a = 0, b = 3)\), and \((a = 0.5, b = 1)\).

The Hough transform was originally introduced by Hough as a United States patent [7]. It has become an important technique for many computer vision and pattern recognition applications. A recent survey [9] lists 136 references to papers and patents, covering a wide range of applications.

The standard Hough transform algorithm divides the continuous parameter space into rectangular cells and associates a counter with each cell. Each point in the image "votes" for all possible parameter combinations that describe a pattern which contains that point. The voting is manifested by incrementing cell counters. At the end of the voting phase the coordinates of cells that received a large number of votes are produced as output.

The space complexity of the standard algorithm is determined by the number of counters (cells), and the time complexity is determined by the total number of increments. Both the number of counters and the number of increments depend on the discretization of the parameter space (the cells size). In addition, the number of increments depends on the number of input points and the number of patterns that are associated with each point. Because of the large amount of computation and storage that is required for the implementation of the Hough transform, it is usually applied to detect patterns that depend on at most two parameters, such as straight lines or circles with a known radius.

Several techniques have been suggested for reducing the computational complexity of the standard Hough transform algorithm. These include...
gathering additional information from the picture (e.g., gradient direction) [9], and multi-resolution search in the parameter space [8]. Smart data structures sometimes allow reduction of the space complexity by keeping only a small number of counters in a hash table [2].

Recently, several authors suggested a probabilistic approach for reducing the complexity of Hough transform techniques. The basic idea behind these algorithms is that the voting can be replaced with a carefully designed poll. In [11], Kiryati et al. show that in order to detect straight lines it is usually enough to apply the Hough transform to a small number of randomly chosen picture points. A different approach was suggested by Xu et al. in [12]. Using ideas similar to [5] (see also [6]) they suggest randomly choosing \( n \)-tuples of points in order to detect patterns that are determined by \( n \) parameters. Thus, each \( n \)-tuple votes for a single pattern.

The main contribution of this paper is the observation that the voters can be arbitrary sets of points. A formal analysis of the complexity of the probabilistic techniques shows that the optimal size of these sets depends on the distribution of the values in the Hough transform representation and is usually proportional to the logarithm of the number of cells. The analysis shows that often, choosing small sets of two or three points enables a significant reduction in the complexity. Our algorithm is derived as a Monte-Carlo algorithm for a constraint satisfaction model. The generic Monte-Carlo approximation is similar to the algorithms described by Karp et al. in [10].

### 2. The Constraint Satisfaction Model

Our algorithms are derived as special cases of a constraint satisfaction problem that is described in this section.

Let \( \Theta^w \subset \mathbb{R}^w \) be a \( w \)-dimensional parameter space. A parameter vector \( \phi \in \Theta^w \) is a vector with \( w \) coordinates \( \phi = (\phi_1, \ldots, \phi_w) \), where each coordinate \( \phi_i, i = 1, \ldots, w \), is a parameter value. In the example of line detection (see Section 1) a parameter vector is the pair \((a, b)\) of line parameters.

A constraint \( c \) is a subset of \( \Theta^w \). All vectors \( \phi \in c \) satisfy the constraint \( c \), and all vectors \( \phi' \notin c \) do not satisfy \( c \). In the example of line detection (see Section 1) a constraint is associated with each picture point. Only parameter vectors of lines that pass through that point satisfy the constraint.

The notation \( | \cdot | \) is used for the cardinality (number of elements) of a set. The following is a definition of the continuous Hough transform. In the definition \( \mathbb{Z}^+ \) is the set of non-negative integers.
DEFINITION. The Hough transform\(^2\) of the set of constraints \(C\) is a function \(H: \Theta^w \to \mathbb{Z}^+\) such that

\[
\forall \phi \in \Theta^w, \quad H(\phi) = |\{c \in C : \phi \in c\}|.
\]

\((H(\phi)\) is the number of constraints in \(C\) that are satisfied by \(\phi\).

The continuous Hough transform can be approximated by a discrete Hough transform. This is done by dividing the continuous parameter space into a finite number of (rectangular) cells. Let \(t\) be the number of cells. We denote the cells by \(\Gamma' = \{\gamma_1, \ldots, \gamma_t\}\).

DEFINITION. The discrete Hough transform of the set of constraints \(C\) is a function \(H: \Gamma' \to \mathbb{Z}^+\) such that

\[
\forall \gamma \in \Gamma', \quad H(\gamma) = |\{c \in C : \exists \phi \in \gamma, \phi \in c\}|.
\]

\((H(\gamma)\) is the number of constraints in \(C\) that have satisfying parameter vectors inside the cell \(\gamma\).

Algorithm 1 is the "standard" algorithm for computing the discrete Hough transform.

**Algorithm 1.** Computing \(H\), the Hough transform of the set of constraints \(C\):

- start with zero counters \(H(\gamma) = 0, \forall \gamma \in \Gamma'\).
- for all \(c \in C\), increment \(H(\gamma)\) for all cells \(\gamma\) that contain a parameter vector which satisfies \(c\).
- \(\forall \gamma \in \Gamma', H(\gamma) = H(\gamma)\).

**2.1. The Monte-Carlo Approximation**

Let \(C\) be a set of \(m\) constraints. Consider a probability space in which an event is manifested as choosing a constraint \(c\) from \(C\) uniformly at random. We have

\[
\text{Prob}(c | c \in C) = \frac{1}{m}.
\]

From the definition of the Hough transform we have

\[
\forall \phi \in \Theta^w, \quad \text{Prob}(\phi \in c | c \in C) = \frac{H(\phi)}{m}.
\]

\(^2\)What we define here as the Hough transform is sometimes called "the accumulator space in the Hough transform representation."
Therefore, there is a simple probabilistic interpretation of $H(\phi)$:

$$\forall \phi \in \Theta^w, \quad H(\phi) = m \cdot \text{Prob}(\phi \in c \mid c \in C).$$

And for the discrete Hough transform,

$$\forall \gamma \in \Gamma', \quad H(\gamma) = m \cdot \text{Prob}(\exists \phi \in \gamma, \phi \in c \mid c \in C). \quad (2)$$

This simply states the fact that the value of the Hough transform for a cell $\gamma$ is proportional to the probability that a randomly chosen constraint from $C$ has a satisfying parameter vector inside the cell $\gamma$.

The basic idea behind our approximation to the Hough transform is that since only the location of peaks in the parameter space is relevant for recognizing patterns, it is unnecessary to get a good approximation where the value of the Hough transform is small. The three parameters that we use for defining the approximation are $\varepsilon$, $\delta$, and $\mu$. The parameter $\mu$ is the size parameter. It is the fraction of $m$ that is too small to be considered a "meaningful" value. The parameters $\delta$, $\varepsilon$ are the confidence and accuracy parameters, respectively. The approximation of meaningful values is required to have errors bounded by $\varepsilon$ with confidence of at least $1 - \delta$. The formal definition is:

**Definition.** Let $C$ be a set of $m$ constraints and let $H(\phi)$ be the Hough transform of the constraints in $C$. For $0 \leq \varepsilon, \delta, \mu \leq 1$ $\tilde{H}(\phi)$ is an $\varepsilon, \delta, \mu$ approximation of $H(\phi)$ if:

$$\forall \phi \in \Theta^w, \quad H(\phi) < \mu m \Rightarrow \text{Prob}(\tilde{H}(\phi) \geq (1 + \varepsilon)\mu m) \leq \delta \quad (3)$$

$$\forall \phi \in \Theta^w, \quad H(\phi) \geq \mu m \Rightarrow \text{Prob}(|\tilde{H}(\phi) - H(\phi)| \geq \varepsilon H(\phi)) \leq \delta \quad (4)$$

with $\varepsilon$-accuracy parameter, $\delta$-confidence parameter, $\mu$-size parameter.

Equation (3) guarantees (with confidence of $1 - \delta$) that there are no false peaks. Equation (4) guarantees (with confidence of $1 - \delta$) that peaks are accurately approximated. By replacing coefficient vectors with cells we obtain an analogous definition for the $\varepsilon, \delta, \mu$ approximation of the discrete Hough transform.

It is sometimes useful to view the Hough transform values as belonging to two categories: large values, which may indicate the presence of a pattern, and small values, which do not indicate the presence of a pattern. In these cases we would like to choose the values of $\varepsilon, \mu$ such that the two categories are separated with high probability in the $\varepsilon, \delta, \mu$ approximation.

**Proposition.** Let $H(\gamma)$ be the discrete Hough transform of a set $C$ of $m$ constraints. Let $m h_b$ be the smallest value of $H(\gamma)$ that may indicate the
presence of a pattern, and let $m h_\gamma$ be the largest value of $H(\gamma)$ that does not indicate the presence of a pattern, where $0 \leq h_\gamma < h_b \leq 1$. If the accuracy parameter $\varepsilon$ and the size parameter $\mu$ are related by

$$h_\gamma \leq \mu \leq \frac{1 - \varepsilon}{1 + \varepsilon} h_b,$$  \hspace{1cm} (5)

then with probability of at least $1 - \delta$ a value that indicates the presence of a pattern in an $\varepsilon, \delta, \mu$ approximation will be larger than a value that does not indicate the presence of a pattern.

**Proof.** Clearly, we must have $\mu \geq h_\gamma$. From Eq. (3) it follows that with probability of at least $1 - \delta$ a non-peak value is of size less than $(1 + \varepsilon)\mu m$. From Eq. (4) it follows that with probability of at least $1 - \delta$ the value $h_\gamma m$ can shrink to a size no smaller than $(1 - \varepsilon)h_b m$. The right-hand inequality in (5) is equivalent to the condition $(1 + \varepsilon)\mu m \geq (1 - \varepsilon)h_b m$.

Monte-Carlo algorithms are techniques that use random number generators (e.g., coin flipping) to approximate expected values. The interpretation of the Hough transform values as probabilities (in Eq. (2)) enables us to construct a Monte-Carlo algorithm for the Hough transform. Algorithm 2 is the generic Monte-Carlo algorithm for computing an $\varepsilon, \delta, \mu$ approximation of the discrete Hough transform.

**Algorithm 2.** Computing $\tilde{H}$, an $\varepsilon, \delta, \mu$ approximation to the Hough transform of a set $C$ of constraints:

- start with zero counters $\tilde{H}(\gamma) = 0, \forall \gamma \in \Gamma'$.
- repeat $n$ times steps (a), (b):
  - (a) choose at random a constraint $c \in C$.
  - (b) increment $\tilde{H}(\gamma)$ for all cells $\gamma$ that contain a parameter vector which satisfies $c$.
- normalize: $\forall \gamma \in \Gamma', \tilde{H}(\gamma) = m \cdot \tilde{H}(\gamma)/n$.

From Eq. (2) and elementary probability it follows that for all $\gamma \in \Gamma'$, $\tilde{H}(\gamma)$ approaches $H(\gamma)$ when $n \to \infty$. The following theorem gives a bound on the size of $n$ which guarantees that $\tilde{H}(\gamma)$ is an $\varepsilon, \delta, \mu$ approximation.

**Theorem.** Algorithm 2 gives $\varepsilon, \delta, \mu$ approximation with

$$n \geq \min\left\{ 3 \frac{\ln(1/\delta)}{\varepsilon^2 \mu}, \frac{1 - \mu}{\mu \varepsilon^2 \delta} \right\}.$$  \hspace{1cm} (6)

The proof is given in the Appendix.
2.2. Complexity

The major computation steps in Algorithm 2 are the \( n \) iterations of steps (a) and (b). Step (b) is

increment \( \hat{H}(\gamma) \) for all cells \( \gamma \) that contain a parameter vector which satisfies \( c \).

In most relevant cases it is unnecessary to explicitly go over all cells \( \gamma \in \Gamma' \) because the constraints are described as easy to follow patterns in the parameter space. (See the example of line detection in the Introduction.) We count as the number of operations in step (b) only the actual increments of cell counters. Let \( \bar{I} \) be the average number of increments in step (b). If \( \bar{I} \geq 1 \) then \( n \cdot \bar{I} \), the total number of increments, measures the average computational complexity of Algorithm 2. If \( \bar{I} < 1 \) we still count step (a) as a computation step in each iteration so that we must take \( n \bar{I} \) as a measure of the average computational complexity of Algorithm 2. Thus, the average complexity is given by \( \max(n, n\bar{I}) \).

In order to give a bound on the storage complexity of Algorithm 2 we assume an implementation that uses a hash table for keeping only the non-zero counters, so that space need not be allocated for the zero counters. Since the number of non-zero counters cannot exceed the number of increments, \( \max(n, n\bar{I}) \) is an upper bound on both the computational and the storage complexity of Algorithm 2. (Notice that the time needed to search for peaks in the Hough transform is determined by the storage complexity.)

3. Hough Transforms of Patterns of Points

In this section we describe variants of Algorithm 2 that can be used to detect patterns of points. The algorithms get as input picture points and produce as output an approximation of the Hough transform of patterns.

3.1. Preliminary Definitions

**Input.** A set of \( d \) points, \( P = \{p_1, \ldots, p_d\} \). The points are given by their coordinates: \( p_i = (x_{i1}, \ldots, x_{iv}), \ i = 1, \ldots, d \). (In the case of 2D pictures \( v = 2 \).)

**Patterns.** Defined by a set of \( w \) parameters \( \phi = (\phi_1, \ldots, \phi_w) \), typically by an equation of the form:

\[
q(x_1, \ldots, x_v, \phi_1, \ldots, \phi_w) = 0.
\]  

We write the arguments of Eq. (7) in a vector form as \( q(p, \phi) = 0 \).

A pattern \( q_\phi \) is the set of points \( p \) such that \( q(p, \phi) = 0 \) for all \( p \in q_\phi \). The size of a pattern in the input data is the number of input points
belonging to the pattern. The size of \( q_\phi \) in \( P \) is denoted by \( |q_\phi|_P \), where \( |q_\phi|_P = |q_\phi \cap P| \). As an example, consider the simple picture of six points that was used in the Introduction. The constraint equation \( q \) is: \( ax + b - y = 0 \). The set \( P \) is the six picture points. The size of \( q_\phi \), for \( \phi = (0,3) \) (i.e., the line \( y = 3 \)) is 3. The size of \( q_\phi \), for \( \phi = (1,1) \) (i.e., the line \( y = x \)) is 1.

**Discretization.** The continuous parameter space is divided into \( t \) (rectangular) cells \( \Gamma' = \{\gamma_1, \ldots, \gamma_t\} \).

**Complexity.** The complexity of Hough transform algorithms depends on \( d \) the number of input points, and on \( t \) the number of cells in the discretization of the parameter space. The complexity of probabilistic algorithms for \( \varepsilon, \delta, \mu \) approximations depends also on \( \varepsilon, \delta, \mu \). We will show that the complexity of the probabilistic algorithms may depend also on the distribution of values in the Hough transform. In order to obtain an estimation of the complexity in these cases we will consider a uniform distribution, where the Hough transform is composed of \( u \) “large” values each of size \( dh_b \), and \( t - u \) “small” values each of size \( dh_s \), where \( h_s < h_b \).

3.2. **Individual Points as Constraints**

Each point \( p \in P \) can be associated with the constraint that is given by Eq. (7). The Hough transform \( H(\phi) \) of these constraints is the number of points \( p \in P \) in the pattern \( q_\phi \), i.e., the size of \( q_\phi \) in the input points:

\[
H(\phi) = |q_\phi|_P.
\]  
(8)

This is the standard interpretation of points as constraints. Algorithm 3 is the “standard” algorithm for the discrete Hough transform, which is the special case of Algorithm 1 when individual points are the constraints.

**Algorithm 3.** Computing \( H \), the Hough transform of the set of points \( P \):

- start with zero counters \( \hat{H}(\gamma) = 0 \), \( \forall \gamma \in \Gamma' \).
- for all points \( p \in P \), increment \( \hat{H}(\gamma) \) for all cells \( \gamma \) that contain a parameter vector \( \phi \) such that \( q(p, \phi) = 0 \).
- \( \forall \gamma \in \Gamma' \), \( H(\gamma) = \hat{H}(\gamma) \).

Algorithm 4 is the Monte-Carlo algorithm (a special case of Algorithm 2) for computing the Hough transform with individual points as constraints.
Algorithm 4. Computing $\tilde{H}$, a $\xi, \delta, \eta$ approximation to the Hough transform of the set $P$ of $d$ points:

- start with zero counters $\tilde{H}(\gamma) = 0$, $\forall \gamma \in \Gamma'$.
- repeat $n$ times steps (a), (b):
  (a) choose at random a point $p \in P$.
  (b) increment $\tilde{H}(\gamma)$ for all cells $\gamma$ that contain a parameter vector $\phi$ such that $q(p, \phi) = 0$.
- normalize: $\forall \gamma \in \Gamma'$, $\tilde{H}(\gamma) = d \cdot \tilde{H}(\gamma)/n$.

The value of $n$ is given by

$$n = \min\left(3 \frac{\ln(1/\delta)}{\xi^2 \eta}, \frac{1 - \eta}{\eta \xi^2 \delta}\right),$$

(9)

where $\eta$ is a predetermined fraction of the input points that is too small to be a peak, $\xi$ is the required accuracy in measuring a peak height, and $\delta$ is a tolerable probability of error.

3.2.1. Correctness of Algorithm 4. The value of $n$ in Eq. (9) is obtained from Eq. (6) with $\mu = \eta$, and $\varepsilon = \xi$.

3.2.2. Complexity of Algorithms 3 and 4. Let $\bar{I}$ be the average number of increments for a point $p \in P$. The value of $\bar{I}$ is given by the total number of increments divided by the number of points:

$$\bar{I} = \sum_{\gamma \in \Gamma'} H(\gamma)/d.$$

The complexity of Algorithm 3 is $d \cdot \bar{I}$. The average complexity of Algorithm 4 is $\max(n, n\bar{I})$, independent of $d$. The value of $\bar{I}$ is usually a function of $t$, the number of cells. For example, in line detection we have $\bar{I} \approx \sqrt{t}$.

Using the first bound on $n$ from Eq. (9) we have

$$n \cdot \bar{I} = \frac{3\ln(1/\delta)}{\xi^2 \eta} \sum_{\gamma \in \Gamma'} H(\gamma)/d.$$

Now if $\bar{I} \geq 1$ and the Hough transform is composed of $u$ large values of size $dh_b$ and $t - u$ small values of size $dh_s$, then

$$n \cdot \bar{I} = \frac{3\ln(1/\delta)}{\xi^2 \eta}(u \cdot h_b + (t - u) \cdot h_s) = O(\ln(1/\delta) \cdot t/\xi^2 \eta).$$

This shows that Algorithm 4 can produce approximations with a high degree of confidence, since the term $\delta$ appears inside a log. Algorithm 4 is
more efficient than Algorithm 3 when the number of points is large, and the values of $\xi$ and $\eta$ need not be too small. See the discussion in Section 3.4.

3.3. Multiple Points as Constraints

The algorithm described in this section is a generalization of Algorithm 4. Here we choose small samples of (not necessarily distinct) points $p_{i1}, \ldots, p_{ir}$ from $P$ as the constraints. The parameter vector $\phi$ is satisfied by the constraint that is imposed by the points $p_{i1}, \ldots, p_{ir}$ if and only if

$$for \ j = 1, \ldots, r, \quad q(p_{ij}, \phi) = 0,$$

that is, if and only if all $r$ points belong to the pattern $q\phi$. The number of constraints of this type is $d'r$, and exhaustive computations, that is implementations of Algorithm 1 are usually impractical even for $r = 2$. The Hough transform of these constraints is given by

$$H_r(\phi) = (|q\phi|_P)^r,$$

because the pattern $q\phi$ has $(|q\phi|_P)^r$ $r$-tuples of points in $P$. Therefore, the value of $H(\phi)$ that measures the pattern size (as given by Eq. (8)) is related to $H_r(\phi)$ by

$$H(\phi) = (H_r(\phi))^{1/r}.$$

Algorithm 5 is the Monte-Carlo algorithm (a special case of Algorithm 2) applied to constraints of this type.

Algorithm 5. Computing $\tilde{H}$, a $\xi, \delta, \eta$ approximation to the Hough transform of the set $P$ of $d$ points:

- start with zero counters $\tilde{H}(\gamma) = 0, \forall \gamma \in \Gamma'$.
- repeat $n$ times (for $i = 1, \ldots, n$) steps (a), (b):
  (a) choose at random (with replacement) $r$ points $p_{i1}, \ldots, p_{ir}$ from $P$.
  (b) increment $\tilde{H}(\gamma)$ for all cells $\gamma$ that contain a parameter vector $\phi$ such that $q(p_{ij}, \phi) = 0$ for $j = 1, \ldots, r$.
- normalize: $\forall \gamma \in \Gamma', \ H(\gamma) = d' \cdot H_r(\gamma)/n$.
- estimate the standard Hough transform: $\forall \gamma \in \Gamma', \ H(\gamma) = (\tilde{H}(\gamma))^{1/r}$.
The value of $n$ is given by

$$n = \min \left\{ \frac{3 \ln(1/\delta)}{(\varepsilon^2 \eta)^r}, \frac{1 - \eta'}{(\delta^2 \eta)^r} \right\},$$

(10)

where $\eta$ is a predetermined fraction of the input points that is too small to be a peak of $H(\gamma)$, $\varepsilon$ is the required accuracy in measuring a peak of $H(\gamma)$, and $\delta$ is a tolerable probability of error.

3.3.1. Correctness of Algorithm 5. The value of $n$ in Eq. (10) is obtained from Eq. (6) with $\mu = \eta'$, and with $\varepsilon = \varepsilon'$. The value of $\mu$ is the probability that a randomly chosen constraint is satisfied by a pattern of size $\eta d$, and this is given by $(\eta d')/d' - \eta'$. The value of $\varepsilon$ is the required accuracy in measuring a peak of $H_r(\phi)$. With probability of at least $1 - \delta$ we have (see Eq. (4))

$$(1 - \varepsilon) H' \leq H_r(\phi) \leq (1 + \varepsilon) H_r(\phi)$$

which implies that

$$(1 - \varepsilon)^{1/r} H(\phi) \leq (H_r(\phi))^{1/r} \leq (1 + \varepsilon)^{1/r} H(\phi)$$

and the relation between $\varepsilon$ and $\varepsilon'$ is obtained from the two inequalities

$$(1 - \varepsilon)^{1/r} \geq 1 - \varepsilon^{1/r} = 1 - \xi, \quad (1 + \varepsilon)^{1/r} \leq 1 + \varepsilon^{1/r} = 1 + \xi.$$

3.3.2. Complexity of Algorithm 5. Let $\bar{I}_r$ be the average number of increments for a constraint in Algorithm 5. The value of $\bar{I}_r$ is given by the total number of increments divided by the number of $r$-tuples:

$$\bar{I}_r = \sum_{\gamma \in \Gamma^r} H_r(\gamma) / d^r = \sum_{\gamma \in \Gamma^r} (H(\gamma) / d)^r.$$

Using the first bound on $n$ from Eq. (10) we have

$$n = 3 \ln(1/\delta) (\varepsilon^2 \eta)^{-r}$$

and

$$\text{Complexity} (\text{Algorithm 5}) \leq n \cdot \max(1, \bar{I}_r).$$

(11)

By increasing the value of $r$ the value of $n$ becomes larger, but $\bar{I}_r$ becomes smaller. The optimal value of $r$ depends on the distribution of
values in the Hough transform. In order to obtain an approximate value for \( r \) we consider the case in which the Hough transform is composed of \( u \) large values of size \( dh_b \) and \( t - u \) small values of size \( dh_s \). In this case,

\[
\tilde{I}_r = uh_b^r + (t - u)h_s^r 
\]

and the total number of increments is given by

\[
n \cdot \tilde{I}_r = 3 \ln(1/\delta) \left( u \left( \frac{h_b}{\xi^2 \eta} \right)^r + (t - u) \left( \frac{h_s}{\xi^2 \eta} \right)^r \right). 
\]

Therefore, if \( \xi^2 \eta \leq h_s \) then the number of increments grows with \( r \), and \( r \) should be taken to be 1. If \( h_s < \xi^2 \eta \) then the value of \( r \) that minimizes the number of increments can be found by equating to zero the derivative with respect to \( r \) of the above equation and then solving for \( r \). This gives

\[
r = \frac{\ln(t/u - 1) + \ln\left( \frac{\xi^2 \eta}{h_s} \right) - \ln\left( \frac{h_b}{\xi^2 \eta} \right)}{\ln(h_b/h_s)}. 
\]

For large \( t \) we have

\[
r \approx \frac{\ln t}{\ln(h_b/h_s)}. 
\]

Substituting this value of \( r \) in Eq. (12) it is easy to see that if \( h_b < 1 \) then \( \tilde{I}_r \to 0 \) when \( t \to \infty \). This means that the complexity is dominated by the value of \( n \) and not by the value of \( n \tilde{I}_r \). However, a value of \( r \) that is proportional to \( \ln t \) can still be used. Choosing \( r \) as

\[
r = \frac{\ln t}{\ln(1/h_s)} 
\]

and substituting in Eq. (12) it is easy to see that \( \tilde{I}_r \to 1 \) when \( t \to \infty \).

By choosing \( r \) according to Eq. (13) we have

\[
n = 3 \ln(1/\delta) \left( \frac{1}{\xi^2 \eta} \right)^r = 3 \ln(1/\delta) t^{\ln(\xi^2 \eta)/\ln(h_s)} 
\]

and, since \( \tilde{I}_r \to 1 \) as \( t \to \infty \),

\[
\text{Complexity(Algorithm 5)} = O\left( \ln(1/\delta) t^{\ln(\xi^2 \eta)/\ln(h_s)} \right). 
\]

This is a significant improvement over the complexity of Algorithm 4, especially when \( \xi, \eta \) can be chosen such that \( h_s \ll \xi^2 \eta \).
3.4. Recognizing Patterns in (Almost) Noise-Free Pictures

The complexity of the probabilistic algorithms depends on the values of \( \xi \) and \( \eta \). In this section we show how to determine these values and minimize the amount of computation for recognizing patterns in pictures that are almost noise-free.

We consider \( u \) patterns \( q_1, \ldots, q_u \) such that each input point is on at least one of the \( u \) patterns. In this case the Hough transform has only \( u \) large peaks. Let \( d \) be the number of input points, \( dh_b \) be a lower bound on the smallest size of a peak, and let \( dh_B \) be an upper bound on the largest size of a peak \( (h_b \leq |q_i|_p/d \leq h_B, i = 1, \ldots, u) \). Let \( dh_S \) be an upper bound on the largest non-peak value.

In the noise-free model we assume that the values of \( h_b, h_B, h_S, \) and \( u \) are known. (The value of \( h_S \) can sometimes be computed from the parametric description of the patterns. For example, if the patterns are lines then \( h_S < u/d \).) Our goal is to recognize the \( u \) patterns. This can be done by computing the Hough transform and choosing the \( u \) cells with the largest values. Since the actual values of the peaks is irrelevant here, we can choose \( \eta \) as large as allowed by the bound in Eq. (5):

\[
\eta = h_b \cdot \frac{1 - \xi}{1 + \xi}.
\]

The complexity of the probabilistic algorithms depends on \( 1/\xi^2 \eta \), so that the value of \( \xi \) can be chosen to minimize the expression \( \xi^2(1 - \xi)/(1 + \xi) \), and this gives \( \xi = (\sqrt{5} - 1)/2 \approx 0.61 \). The corresponding value of \( \eta \) is 0.242\(h_b\), and \( 1/\xi^2 \eta = 11.1/h_S \).

We can now use these values of \( \xi \) and \( \eta \) to get an upper bound on the complexity of Algorithms 4 and 5 in the noise-free case. The number of increments is

\[
n \cdot \bar{T}_r \leq 3 \ln(1/\delta) \left( u \left( 11.1 \frac{h_B}{h_b} \right) + (t - u)\left( 11.1 \frac{h_S}{h_b} \right) \right), \quad (14)
\]

and the value of \( n \) is given by

\[
n = 3 \ln(1/\delta) \cdot \left( \frac{11.1}{h_b} \right) \cdot \left( 11.1 \frac{h_B}{h_b} \right).
\]

3.4.1. A Numeric Example. As a numeric example we consider the problem of recognizing patterns determined by four parameters, such as
ellipses as given by the equation:

\[ \frac{(x_1 - \phi_1)^2}{\phi_3^2} + \frac{(x_2 - \phi_2)^2}{\phi_4^2} = 1. \]

We take the number of input points \( d = 10,000 \), and \( u = 5 \), \( h_b = 1900/d \), \( h_a = 2100/d \), and \( h_5 = 10/d \), i.e., five ellipses, each one with approximately 2000 points. An illustration of this case (with fewer points) is shown in Fig. 2. Dividing each of the four parameters into 100 intervals gives \( \tau = 100^4 = 10^8 \).

If the standard Hough transform (Algorithm 3) is used for this problem then each one of the 10,000 points votes for \( 100^3 = 10^6 \) cells, giving a total of \( 10^8 \) increments.

For Algorithms 4 and 5 we take \( \delta = 0.001 \), which guarantees a success rate of 99.9%. The complexity of Algorithm 4 is obtained from Eq. (14) with \( r = 1 \), giving \( 1.2 \times 10^8 \) increments on the average. This requires sampling \( n = 1211 \) points.

The optimal value of \( r \) is 3 in this case, and Eq. (14) gives \( 6 \times 10^5 \) as the average number of increments for Algorithm 5 with \( r = 3 \). This requires sampling \( n = 3 \times 10^6 \) triplets of points. The complexity here is better than the standard algorithm by a factor of \( 10^4 \). Thus, in this case a
computation that takes minutes for Algorithm 5 may take months for the standard Hough transform!

4. Concluding Remarks

We have shown that an approximation to the Hough transform can be computed by efficient probabilistic, Monte-Carlo type algorithms. Unlike the majority of the algorithms in computer vision and pattern recognition, these probabilistic algorithms have the nice property that their complexity is independent of the input size. They are especially efficient when handling "easy" cases for humans, where the input is composed of a few high resolution patterns.

The major disadvantage of our algorithms compared to the standard Hough transform is that they may fail with a small probability. However, it can be guaranteed that the probability of failure is no larger than can be tolerated by most applications. Notice that this probability is from a probability space that is created by a random number generator and is independent of the picture content.

Since the complexity of our algorithms is independent of the input size, they are usually much more efficient than the standard Hough transform when the input size is large. We have shown that the complexity can be further reduced by sampling small subsets of points that jointly vote for likely patterns. In general, the number of points $r$ depends on the distribution of the Hough transform values. Notice, however, that an $\epsilon, \delta, \eta$ approximation can be obtained for any value of $r$. A non-optimal value of $r$ will only cause the algorithm to run more slowly.

Various techniques have been suggested by computer vision researchers to enhance the performance of the classical Hough transform. It appears that many of these techniques can also be applied to further improve the performance of the probabilistic algorithms. Many of them can be viewed as methods for effectively increasing the value of $\eta$ in our algorithm.

Appendix

In this appendix we give the proof of the theorem of Section 2.1. We first show in a lemma that in our case the condition in Eq. (4) implies the condition in Eq. (3) and, therefore, is sufficient to guarantee an $\epsilon, \delta, \mu$ approximation. Karp et al. considered approximations as defined by Eq. (4) for $0 < \epsilon \leq 2$, and give $n = 4 \ln(2/\delta)/(\epsilon^2 \mu)$ as an upper bound on $n$, optimal with respect to $\epsilon, \delta$ up to a constant [10]. A slightly better
constant can be obtained from Chernoff's tail bounds for $0 < \varepsilon \leq 1$, and for large values of $\mu$ ($\mu$ close to 1) it is possible to do even better.

**Theorem.** Algorithm 2 gives $\varepsilon, \delta, \mu$ approximation with

$$n \geq \min \left\{ \frac{3 \ln(1/\delta)}{\varepsilon^2 \mu}, \frac{1 - \mu}{\mu \varepsilon^2 \delta} \right\}.$$

**Proof.** For each cell $\gamma \in \Gamma'$, Algorithm 2 can be viewed as a series of $n$ independent Bernoulli trials, where success is manifested as choosing a constraint $c$ such that $c$ is satisfied by a parameter vector from the cell (i.e., $\exists \phi \in \gamma, \phi \in c$). The probability of success is, therefore, $H(\gamma)/m$, and $H(\gamma)$ is distributed according to the binomial distribution. The following lemma shows that in this case the condition in Eq. (4) is sufficient to guarantee an $\varepsilon, \delta, \mu$ approximation:

**Lemma.** If $\forall \phi \in \Theta^w$ the value of $\tilde{H}(\phi)$ is distributed according to the binomial distribution with $p = H(\phi)/m$ then the condition in Eq. (4) implies the condition in Eq. (3).

**Proof.** $\forall \phi \in \Theta^w$,

$$\begin{align*}
\text{Prob}(\tilde{H}(\phi) \geq (1 + \varepsilon)\mu m | H(\phi) < \mu m) &
\leq \text{Prob}(\tilde{H}(\phi) \geq (1 + \varepsilon)\mu m | H(\phi) = \mu m) \\
&= \text{Prob}(\tilde{H}(\phi) \geq (1 + \varepsilon)H(\phi) | H(\phi) = \mu m) \\
&\leq \text{Prob}(|\tilde{H}(\phi) - H(\phi)| \geq \varepsilon H(\phi) | H(\phi) = \mu m) \\
&\leq \text{Prob}(|\tilde{H}(\phi) - H(\phi)| \geq \varepsilon H(\phi) | H(\phi) \geq \mu m) \\
&\leq \delta.
\end{align*}$$

From the lemma it follows that in order to prove the theorem it is enough to show that the bounds on $n$ in the theorem imply the condition in Eq. (4). The two bounds are obtained from the Chernoff and the Chebichev inequalities respectively:

1. $n \geq 3(\ln(1/\delta)/\varepsilon^2 \mu) \Leftrightarrow \delta \geq e^{-\varepsilon^2 \mu n}/3$. From Chernoff's tail bounds (see [1]) for having $x$ successes in $n$ independent Bernoulli trials with probability of success $p$:

$$\text{Prob}(|x - np| \geq \varepsilon np) \leq e^{-\varepsilon^2 np}/3.$$
Therefore,

\[
\Pr(|\hat{H}(\gamma) - H(\gamma)| \geq \epsilon H(\gamma)) = \Pr\left(\left|\frac{\hat{H}(\gamma)}{H(\gamma)} - \frac{n}{m}\right| \geq \frac{n}{m}\epsilon H(\gamma)\right) 
\leq e^{-\epsilon^2(n/m)H(\gamma)/3}.
\]

And for \(H(\gamma) \geq \mu m\),

\[
e^{-\epsilon^2(n/m)H(\gamma)/3} \leq e^{-\epsilon^2n\mu/3} \leq \delta.
\]

II. \(n \geq (1 - \mu/\mu\epsilon^2\delta \Leftrightarrow \delta \geq (1 - \mu)/\mu\epsilon^2n\). From the Chebichev inequality (see, e.g., [4]),

\[
\Pr(|\hat{H}(\gamma) - H(\gamma)| \geq \epsilon H(\gamma)) = \Pr\left(\left|\frac{\hat{H}(\gamma)}{H(\gamma)} - \frac{n}{m}\right| \geq \frac{n}{m}\epsilon H(\gamma)\right) 
\leq \frac{m - H(\gamma)}{H(\gamma)} \cdot \frac{1}{n\epsilon^2}.
\]

To prove the claim it is enough to show that \((m - H(\gamma))/H(\gamma) \leq (1 - \mu)/\mu\). This follows because the left-hand side is monotonic decreasing as a function of \(H(\gamma)\), and \(H(\gamma) \geq \mu m\). □

REFERENCES


