

ADVANCED HOUGH TRANSFORM IMPLEMENTATIONS

C.M. Brown, M.B. Curtiss, and D.B. Sher
Computer Science Department, University of Rochester
Rochester, NY 14627 U.S.A.

Abstract

In the Hough transform, a feature in an input space I votes for parameters in a transform space P with which it is compatible. If it also uses negative votes to vote against parameters with which it is incompatible, background bias and inherent noise in P can be reduced and peaks sharpened.

In the usual Hough transform implementation, an array is used to accumulate votes, raising the problem that the space requirement increases exponentially with the number of parameters. Accumulation in a fixed-size content-addressable cache (or hash table) may provide a solution.

I. Definitions, Problems, Solutions

A. Hough Transform as Voting and Imaging

Considered as a voting process, the *Hough transform* maps features of a phenomenon in an input space I to sets of votes in the parameter space P . The parameter vectors receiving the most votes describe *instances* of the phenomenon explaining the most evidence. Ideally, the Hough transform maps a phenomenon into a set of votes that is a delta function in parameter space; the votes are unanimous. Usually the votes are more or less spread out, since several phenomena might generate a feature. Call the pattern of votes arising from a feature its *feature point spread function* (fpsf), and the superposition of all fpsfs from a phenomenon its *parameter point spread function* (ppsf) [Brown, 1983]. A ppsf usually consists of a peak of votes at the correct parameter vector surrounded by *sidelobes* of incorrect votes.

B. Inherent Noise and Bias? Complementary Hough

Problem: When the input space contains multiple instances close together, sidelobes from one instance can overlap the peak from another, adding both variation and bias to the peak height. Even in single-instance noiseless inputs, the near-peak votes can spread the peak. Solution: In the *Complementary Hough transform*, the fpsf contains

(negative) votes *against* parameters with which the feature is inconsistent, as well as (positive) votes *for* parameters with which it is consistent. The negative votes can cancel off-peak positive votes. Complementary Hough sharpens peaks and reduces sidelobe bias and variance.

C. Space Efficiency? Cache Hough

Problem: Multi-dimensional quantized parameter spaces are most straightforwardly implemented as arrays which grow impractically large as their dimension (the number of parameters) increases. Solutions: One solution is the use of dynamically quantized parameter space [O'Rourke, 1981; Sloan, 1981]. The solution explored here is *Cache Hough*: Accumulate vote tallies in a content-addressable cache (in hardware) or hash table (in software) [Brown and Sher, 1982a; 1982b]. The abstract problem is that of estimating the mode(s) of a distribution using fixed memory. Cache Hough is a promising technique suitable for hardware implementation.

II. Results: Synthetic Data

A. Complementary Hough Performance

What fpsf for complementary Hough maximizes peak detectability and minimizes bias? All the results of this section arise from synthesized versions of circle detection for a constant radius, but the results are suggestive of general principles. In every case, the circumferential feature points in the input "image" contain no directional information; it is as if they were generated by a rotationally symmetric feature detector. This means that the fpsf in the regular Hough parameter space of circle centers is a complete circle of unit weight votes (each edge feature votes for a set of centers at all locations a radius distance away). The fpsfs for complementary Hough are circles of unit-weight votes with neighboring concentric circles of negative votes. The first graph (Fig. 1) documents the search for the best complementary Hough fpsf. The data points correspond to fpsfs with the indicated negative surrounds. The fpsf with a circle of ones surrounded on both sides by $-1/2$'s was used for all the remaining work reported in this section.

Complementary Hough keeps the mean background near zero, with small biases magnified by the contributions of many noise features. Ideally, peak height is proportional to amount of evidence; complementary Hough achieve this. The graph of Fig. 2 shows one measure of regular Hough and complementary Hough performance in seven different imaging situations, getting

The preparation of this paper was supported in part by the Office of Naval Research under Grants N00014-80-C-0197 and N00014-82-K-0193, and the Defense Advanced Research Projects Agency under Grant N00014-78-C-0164.

progressively more complex and noisy. SNR3 indicates detectability by global thresholding (it is a multiple-instance version of the measure used below to quantify performance of cache Hough (Section IIB)).

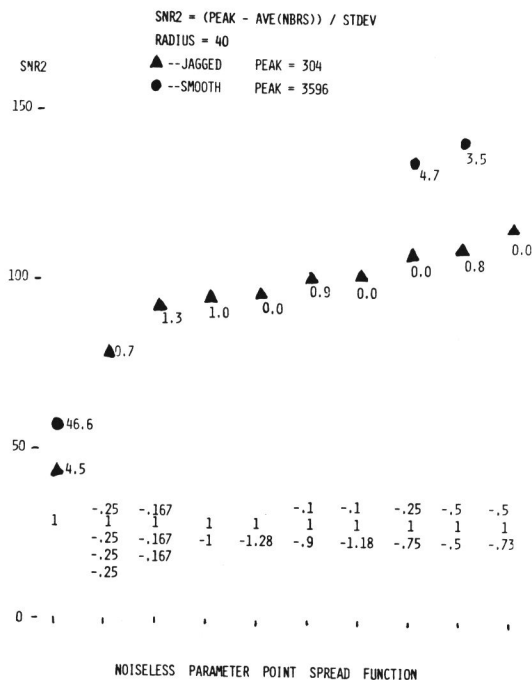


Figure 1. The numbers beside the data points are the mean sidelobe (off-peak value) of the ppsf. The ordinate, called here SNR2, is a signal/noise ratio defined by the local peak height (its absolute height minus the height of its nearest neighbors) divided by the standard deviation of the sidelobe (non-peak) heights. The regular Hough voting pattern for circle detection yields the leftmost data point. Its local vote weight cross-section (a single 1) is shown below it. The "jagged" and "smooth" data points refer to the idea of anti-aliasing the fpsfs (in the computer graphics sense). In smooth voting, votes are put in a cell in amount proportional to the amount of fpsf passing through the cell. In regular Hough, this multiplies the number of votes and makes for a higher peak and proportionally higher mean background. The other data points represent more complicated fpsfs. The fourth, for example, is a circle of 1's and an inner circle of -1's. The second from the right is a circle of 1's surrounded by circles of -1/2's.

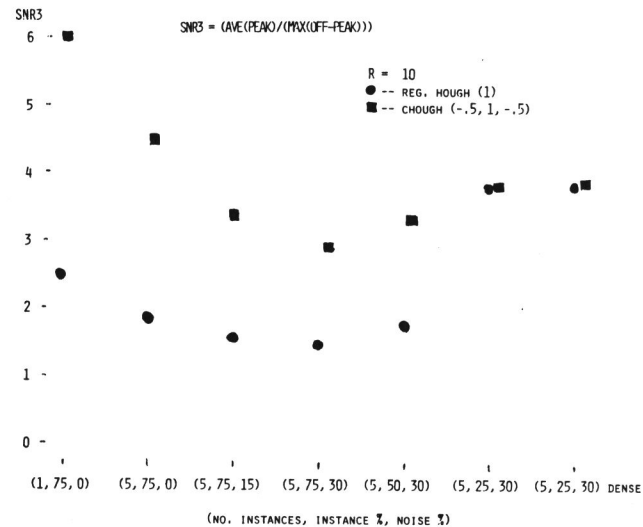


Figure 2. SNR3. SNR3 measures peak detectability when simple thresholding is the detection criterion. SNR3 is the average peak height divided by the maximum off-peak height. This is a multiple-instance version of the measure used for the cache-based Hough study of Section IIB.

B. Cache Hough Performance

We created synthetic phenomenon instances ("shapes") of controllable size, number, and location, embedded them in controllable amounts of noise features, and scanned the resulting images in several ways. Each time a feature was encountered, a vote burst (fpsf) of controllable size was emitted sequentially into a cache whose length and cache-flushing strategy were also controllable. The hope is that the parameter vector corresponding to the instance survives in the cache and accumulates the most votes. For technical details see [Brown and Sher, 1982b]. Comprehensive results appear in [Brown and Sher, 1982a].

The form of synthetic images and fpsfs is dictated by theory in [Brown, 1983]. The size of the instances and psfs and number of noise features in the image determine how many votes are sent to the cache. The scanning methods determine the order of sending. The cache length and flushing algorithm determine what vote tallies survive. We used four cache lengths (32, 64, 128, infinite). The infinite cache is equivalent to an accumulator array. Our instances had horizontal extent of 1, 2, or 3 and vertical extent of 11. The fpsfs were of 3, 9, and 15 votes. There was only one instance per image. Regular and Complementary Hough were used. Images were scanned five ways (Left to Right within Top to Bottom, Top to Bottom within Left to Right, Random with Replacement, Random without Replacement, and Left to Right within Random Scanline). One flushing method eliminated tallies below a threshold, the other eliminated a fixed fraction of tallies, chosen at random, below a larger threshold.

The results of these experiments yielded statistics such as the histograms of Figs. 3 and 4, which display a ratio

$$\text{SNR3} = \frac{(\text{votes for vector describing instance})}{\max(\text{votes for other vectors in cache})}$$

If $\text{SNR3} > 1$, the correct parameter vector tallied a plurality of votes and would be detected as the largest entry in the cache. Otherwise, an incorrect parameter tallied at least as many (see also Fig. 2).

III. Results: Real Data

Our Hough Transform software laboratory allows flexible experimentation and comparison between methods. The cache accumulator module can be parameterized to provide finite or "infinite" length and various flushing methods. Fig. 5 shows sample output.

IV. Discussion

Complementary Hough reduces inherent noise and bias and sharpens peaks at the cost of increased voting effort, depending on the implementation. In a naive implementation the cost might be strictly proportional to the number of votes, but in an implementation with hard-wired voting patterns there is no extra cost. Antialiasing is of some benefit, but also may affect voting cost.

Complementary Hough is most effective with dense voting that maximizes the superimposition of cancelling votes. If evidence is sparse, fpsfs and votes are sparse in the accumulator and little cancellation takes place; similarly if fpsfs are sparse. In the example of Section III, positive votes occur in lines through 3-space, and each positive vote is surrounded by only six (of a possible 26 in 3-D) nearest-neighbor negative votes, resulting in less than optimal cancellation. If each feature votes for exactly one parameter vector (the fpsf is a single point), then complementary Hough degenerates to a linear enhancement operation: a central positive vote surrounded by a cloud of complementary ones is like a Laplacian operator.

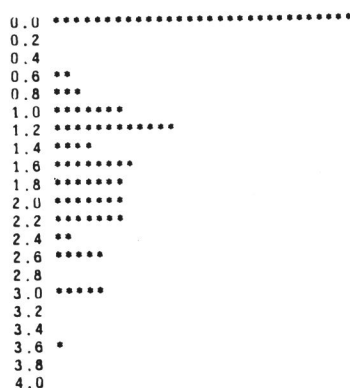
In complementary Hough, sidelobes should cancel while the central peak accumulates positive votes. Feature perturbation error arises from quantization effects [Shapiro and Iannino, 1979]; e.g., the angle computed for an edge element is only approximate. Such error spreads the locus of the peak and superimposes complementary votes on it, degrading complementary Hough performance. Thus the results of Section III represent a severe test for the complementary Hough, and it is gratifying that it performs slightly better than regular Hough.

The remainder of the discussion concerns cache Hough. Finite caches of adequate size seem to be practical

Cache length: 32. Scan: ltoB, ltoR.
Noise: 15%. Instances: 1.
Parts: 1. Fpsf length: 9.
Part length: 11. N: 100

Flush: Threshold

Mean of Ratios: 1.212250
Std. Dev. of Ratios: 0.972063

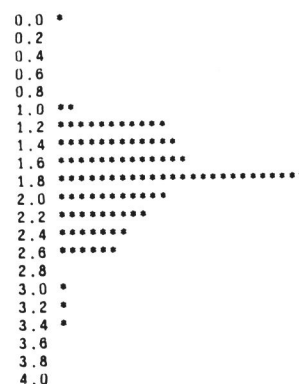


(a)

Cache Length: 64. Scan: ltoB, ltoR.
Noise: 15%. Instances: 1.
Parts: 1. Fpsf length: 9.
Part Length: 11. N: 100

Flush: Threshold

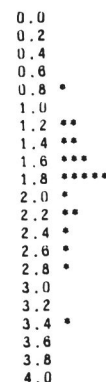
Mean of Ratios: 1.805774
Std. Dev. of Ratios: 0.501593



(b)

Cache length: 64.
Scan: ltoB, ltoR.
Noise: 15%

fpsf length: 9



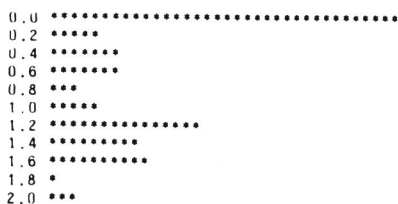
(c)

Figure 3. (a,b) Performance varying with cache length for regular Hough. The mean number of votes per run was 639 and the histogram records 100 runs. (c) Complementary Hough. There were twenty runs and the mean number of votes was 1917.

Cache length: 64. Scan: ltoB, ltoR.
Noise: 30%. Instances: 1.
Parts: 1. tpsf length: 9.
Part length: 11. N: 100

Flush: Threshold

Mean of Ratios: 0.695691
Std. Dev. of Ratios: 0.645508



Flush: Random Below Threshold

Mean of Ratios: 0.893294
Std. Dev. of Ratios: 0.611755

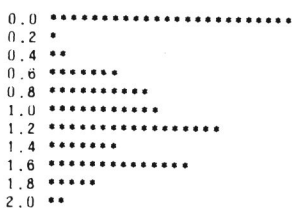
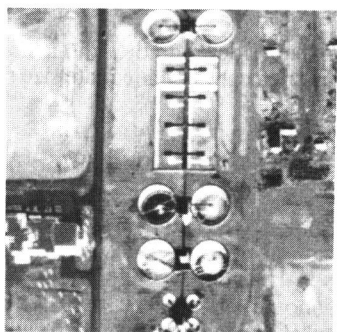
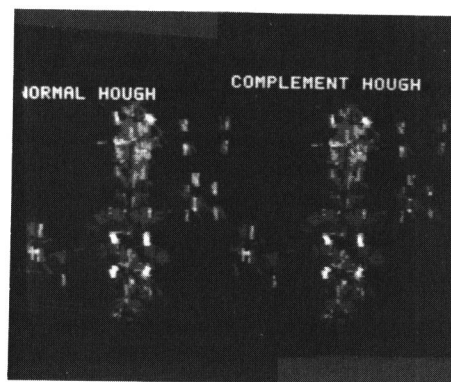


Figure 4. Performance varying with flushing algorithm. With 30% noise, the mean number of votes cast per run is 1179.

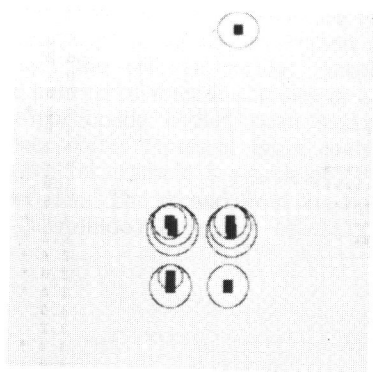
a)



b)



c)



d)

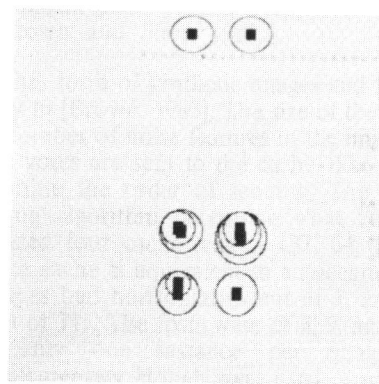


Figure 5. (a) An input image. (b) A projection of the 3-D Hough and complementary Hough accumulator arrays with parameters of circle location and radius. (c) The ten circles with the most votes with regular Hough. (d) As in (c) with complementary Hough.

Hough accumulators, not significantly worse than accumulator arrays. The results show a graceful and qualitatively predictable degradation of performance as noise and fpsf length increase and cache length decreases. Cache lengths of a few percent of the total number of votes seem adequate, but we do not yet know how this figure grows with parameter dimensionality.

The random (with replacement) scan was uniformly significantly worse in all cases than other methods. The random (without replacement) does not seem significantly better than nonrandom scans. Even in 30% noise, the Top to Bottom within Left to Right scanning method was not better than the Left to Right within Top to Bottom. This indicates (with our vertical instances) that successive bursts of votes (fpsfs) containing votes for the "true parameters" are not enough to help the peak establish itself. Probably the noise that arrives before the instance and the flush algorithm dominate the peak-establishment process. Thus the results are fairly insensitive to scanning method, as long as each input space point is considered once.

In good conditions (short fpsf, low noise, large instances), average cache complementary Hough performance is reliably markedly better than cache regular Hough. However, as conditions get worse, their average performance is approximately the same. Several of the histograms show a significant bimodality with one peak at 0. This indicates that the "true peak" never gets established in a significant number of cases, and that some pre-processing might help dramatically in establishing peaks. One idea is "pair Houghing," in which only parameter vectors that appear in two fpsfs are allowed into the cache. Clearly without such preprocessing the issue is whether peaks are able to survive in the cache, and this depends to a large extent on the Flush algorithm. The Random Below Threshold Flush makes for a significant improvement in (mean peak)/(maximum background) ratio, and reduces the frequency of occasions on which the correct parameters end up with zero vote weight.

Systematic distortion, occlusion, and feature perturbation pose thorny problems for cache Hough. An important topic of current work is to assess the performance of cache Hough schemes in more stringent noise conditions and to develop means of improving such performance.

Hierarchical caches can incorporate a form of "geometric locality" that is lost in simple content-addressable caches. Natural flush algorithms for a single-level cache are based strictly on tallies. Multiple caches in a resolution pyramid, maintained in parallel, can be used to restrict flushing to coherent localities in the input space. This idea is another topic of current research.

REFERENCES

- Brown, C.M., "Inherent Bias and Noise in the Hough Transform." To appear, *IEEE PAMI* (1983).
- Brown, C.M. and D.B. Sher, "Hough Transformation into Cache Accumulators: Considerations and Simulations." TR 114, Computer Science Dept., U. Rochester, August 1982(a).
- Brown, C.M. and D.B. Sher, "Modeling the Sequential Behavior of Hough Transform Schemes." In *Proceedings. DARPA Image Understanding Workshop*. Palo Alto, CA, September 1982(b), pp. 115-123.
- O'Rourke, J., "Dynamically Quantized Spaces for Focusing the Hough Transform." *Proceedings. 7th IJCAI*. Vancouver, B.C., August 1981, pp. 737-739.
- Shapiro, S.D. and Iannino, A., "Geometric Constructions for Predicting Hough Transform Performance." *IEEE Trans. PAMI*, 1:3, July 1979.
- Sloan, K.R., Jr., "Dynamically Quantized Pyramids." *Proceedings. 7th IJCAI*. Vancouver, B.C., August 1981, pp. 734-736.