Genetic Search of a Generalized Hough Transform Space

David J. Montana

Bolt Beranek and Newman Inc.
10 Moulton Street, Cambridge, MA 02138

ABSTRACT

We use a Generalized Hough transform (GHT) to detect and track instances of a class of sonar signals. This class consists of a four-dimensional set of curves and hence requires a four-dimensional transform space for the GHT. Many of the signals we need to detect are very weak. Such signals yield peaks in the transform space which are both very narrow and not too far above the random background variations. Finding such peaks is difficult. Exhaustive search over a predetermined discretization of the transform space will yield a nearly optimal point for a sufficiently fine discretization. However, even with an intelligently chosen discretization, exhaustive search requires searching over (and hence evaluating) many points in the transform space. We have therefore developed a genetic algorithm to more efficiently search the transform space. Designing the genetic algorithm to work properly has required experimentation with a number of its parameters. The most important of these are (i) the representation, (ii) the population size, and (iii) the number of runs.

1. THE DETECTION AND TRACKING ALGORITHM

1.1 Generalized Hough transforms (GHT's)

The Hough transform[1] maps a binary image into a gray-scale image as follows. Let x and y parametrize the axes of the input image, let r and s parametrize the axes of the output image, and let the intensities of the input and output images be $I_i(x, y)$ and $I_o(x, y)$ respectively. Furthermore, let $L(r, s) = \{(x, y)|x \sin(s) + y \cos(s) = r\}$, so that $L(r, s)$ consists of all points in the input image which lie along the line which has slope $\tan(s)$ and at its closest is a distance r from the origin. Then,

$$I_o(r, s) = \sum_{(x,y)\in L(r,s)} I_i(x, y) \tag{1}$$

A local peak of intensity at a point (r, s) in the output image whose intensity exceeds a certain threshold provides strong evidence of a line or line segment in the input image along the line $L(r, s)$. Hence, the Hough transform acts as a detector of lines and line segments in binary images. One way of thinking about the Hough transform is as a large bank of matched filters, each one matched to a particular line in the input image.

There are two (non-exclusive) ways to generalize the Hough transform. The first is to allow the input image to be gray-scale rather than binary. This requires no change in the definition of the transform and is hence a trivial generalization. (A Hough transform acting on gray-scale images is essentially equivalent to the Radon transform[2].) The second way to generalize the Hough transform is to recognize that we need not restrict ourselves to looking for lines but can use the same approach to look for instances from any low-dimensional set of curves[3]. For example, to look for circles in the input image, let a, b, and r parametrize the output space, and let $C(a,b,r) = \{(x,y)|(x-a)^2 + (y-b)^2 = r^2\}$. Then, the corresponding Generalized Hough transform is

$$I_o(a,b,r) = \sum_{(x,y)\in C(a,b,r)} I_i(x,y) \tag{2}$$

1.2 Using GHT's for signal detection and tracking

A spectogram is a two-dimensional image with one axis corresponding to frequency and the other to time. Narrowband signals appear in a spectogram as curves whose widths are often not more than a few pixels. Detecting the existence of narrowband signals and tracking how their center frequencies evolve in time is an important problem in many fields including sonar, radar, geophysics and astrophysics. In general, in one spectogram there can be an arbitrary number of signals whose center frequencies can follow arbitrary trajectories within certain broad constraints. We[4] and others[5,6] have developed algorithms to detect and track signals in this general case. A special case of this general problem is when we place stricter constraints on the possible trajectories of the signals. Algorithms have been developed for this case which utilize the knowledge of these constraints to accurately detect and track the signals at lower signal-to-noise (SNR) ratios than the more general algorithms can. If the constraints can be expressed locally, it is possible to incorporate them into a dynamic programming approach[7]. If the constraints are such as to restrict the possible signals to some low-dimensional set of curves in the image space, then a Hough transform[8] or Generalized Hough transform approach can be used.

We have been interested in detecting and tracking sonar signals whose tracks in a spectogram belong to a four-dimensionsal family of curves. We have devised a method for removing from the spectogram most signals not of this class based on the fact that the signals of interest vary much more rapidly in time. We assume that all the remaining signals in the filtered spectogram are of the class of interest and hence that we can apply one of the specialized algorithms. Since the constraints on the signals cannot be expressed locally, the dynamic programming approach is not applicable. This has led us to attempt a GHT approach as an alternative to our more general detection and tracking algorithm. In the next section, we discuss the one major problem we have encountered in applying the GHT to this detection problem.

1.3 Searching for peaks in the transform space

Searching for the best point in the transform space is inherently a difficult problem because (i) the space is relatively big, (ii) the peaks corresponding to signals are narrow, and (iii) the background has many local peaks. We now discuss the reason for each of these difficulties.

The reason for the local peaks in the background of the transform space is that the spectogram consists primarily of random noise (with possibly a signal or two superimposed). Since the sum of random variables is a random variable, the integral along the curve of interest will vary randomly as we vary the curve. The size of the local peaks depends on two factors, the integration time and the statistics of the noise. Integrating over a longer period of time means averaging more random variables and hence reducing the variance of this average. So, longer integration times yield smaller peaks in the background. (It may also yield a smaller peak for the signal because the signal may not persist for the entire time and because over a longer time the signal is more likely to vary from the assumed constraints. Optimal choice of integration time is an important issue.) The noise statistics affect the peak size as follows. If the noise is uncorrelated, then the variance rapidly diminishes with longer integration, and the peaks are small. However, if the noise is correlated, as it is for the sonar data we have used, the peaks are higher. Correlated noise arises in the sonar data from a few sources, one of which is the remnants of signals not totally removed by our filter.

The reason for the narrowness of peaks in transform space corresponding to true signals in the spectogram is that the signals are generally narrow. Hence, curves that track the true signal reasonably closely but remain a few pixels away from the signal's center frequency at most times will receive scores no better than curves that do not track the signal at all (see Figure 1). To receive a score significantly above that expected of a random curve, a curve must

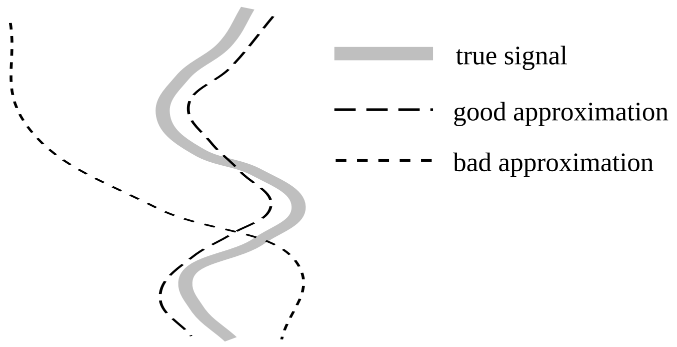

Figure 1: A bad approximation to the true signal may score as well as a good approximation.

track the signal precisely for a significant portion of the time interval. For a given signal, few curves will do this, and hence the peak is narrow.

The reason for the large size of the search space is not due to its high dimensionality. While four dimensions is larger than most GHT spaces, it is small by the general standards of search and optimization problems. Instead, the reason is the need to discretize the space finely along each dimension. For example, if each dimension is represented by 100 points, then the size of the space is 10^8, which is large. There are two reasons we require a fine discretization of the space. One is that we wish that the narrow peaks corresponding to signals do not get lost between samples. The second is that nomatter what parametrization we choose for the four-dimensional family of curves, a uniform sampling of parameter space does not yield a uniform sampling of the set of curves. To determine whether the set of curves is uniformly sampled, we need a concept of the distance between two curves. We use the L_2 metric, where if C_1 and C_2 are curves and I is the time interval of interest, then

$$L_2(C_1, C_2) = [\sum_{t \in I} (C_1(t) - C_2(t))^2]^{1/2} \tag{3}$$

For a few different parametrizations of the set of curves, we have found that there always exists a small change in the parameters $\Delta\vec{p}$ such that as the parameter values $\vec{p}$ vary over the parameter space, $L_2(C(\vec{p}), C(\vec{p} + \Delta\vec{p}))$ varies by at least an order of magnitude. This means that any uniform sampling of the parameter space which adequately samples all parts of the space of curves must grossly oversample in some places.

One way to greatly reduce the size of the search space is to sample uniformly in the space of curves rather than in the parameter space. Therefore, we have devised a simple method for constructing probably approximately uniform samplings of the space of curves. It works as follows:

1. Start with an empty subset of curves.
2. Randomly choose values for the parameters and hence generate a new curve.
3. If the curve is not within distance ϵ of any of the curves in the current subset, add this curve to the subset.
4. If we have generated at least N curves and at most k of the last N curves generated have been added to the subset, stop; otherwise, repeat from step 2.

Note that this procedure generates a sampling such that no two curves in the sampling are within distance ϵ of each other but such that most curves in the space are within distance ϵ of at least one curve in the sampling.

However, even with a nearly uniform sampling of the space of curves, an exhaustive search of the transform space requires evaluating a large number of points a sum anlogous to the ones in Equations 1 and 2. To reduce the required number of such evaluations, we have investigated genetic search as an alternative to exhaustive search.

$$(\,19.3,\ 0.05,\ -1.2,\ 345,\ 2.0,\ 7.7,\ 68.0\,)\ \xrightarrow{\text{mutation}}\ (\,19.3,\ 0.05,\ -1.2,\ 345,\ 2.0,\ 8.2,\ 68.0\,)$$

$$\begin{array}{l}(\,19.3,\ 0.05,\ -1.2,\ 345,\ 2.0,\ 7.7,\ 68.0\,)\\[1em](\,17.6,\ 0.05,\ -3.5,\ 250,\ 3.0,\ 7.7,\ 70.0\,)\end{array}\ \xrightarrow{\text{crossover}}\ (\,17.6,\ 0.05,\ -1.2,\ 345,\ 3.0,\ 7.7,\ 68.0\,)$$

Figure 2: The mutation and crossover operators.

2. THE GENETIC SEARCH ALGORITHM

2.1 An overview of genetic algorithms

Genetic algorithms are algorithms for optimization and machine learning based loosely on several features of biological evolution[9]. They require five components[10]:

1. A way of encoding solutions to the problem on chromosomes.
2. An evaluation function which returns a rating for each chromosome given to it.
3. A way of initializing the population of chromosomes.
4. Operators that may be applied to parents when they reproduce to alter their genetic composition. Standard operators are mutation and crossover (see Figure 2).
5. Parameter settings for the algorithm, the operators, and so forth.

Given these five components, a genetic algorithm operates according to the following steps:

1. Initialize the population using the initialization procedure, and evaluate each member of the initial population.
2. Reproduce until a stopping criterion is met. Reproduction consists of iterations of the following steps:
   (a) Choose one or more parents to reproduce. Selection is stochastic, but the individuals with the hightest evaluations are favored in the selection.
   (b) Choose a genetic operator and apply it to the parents.
   (c) Evaluate the children and accumulate them into a generation. After accumulating enough individuals, insert them into the population, replacing the worst current members of the population.

When the components of the genetic algorithm are chosen appropriately, the reproduction process will continually improve the population, converging finally on solutions close to a global optimum. Choosing these components correctly is often a non-trivial exercise, as we will see in Sections 2.3–2.5. Genetic algorithms can efficiently search large and complex (i.e., possessing many local optima) spaces to find nearly global optima. We have therefore designed a genetic algorithm for the search problem described in Section 1.3.

2.2 Our genetic algorithm

The core of our genetic algorithm is a software package called OOGA (Object-Oriented Genetic Algorithm)[10]. It implements the standard operation of a genetic algorithm described above along with some non-standard features such as exponential normalization of evaluations and adaptive operator probabilities. The user selects the five components described above appropriately for the problem of interest. We have chosen the five components as follows:

1. We represent a member of the set of possible curves as a string of four real-valued parameters. (As we discuss in Section 2.3, the particular parametrization we choose for the set of curves is crucial.) For each parameter, we select a range and step size, thus restricting it to a finite number of possible values. We choose the step sizes so that the space of curves is sufficiently well sampled.

2. The evaluation function is the sum (analogous to those in Equations 1 and 2) of the image intensity along the curve corresponding to the four parameter values.

3. We initialize the population with individuals chosen randomly with uniform probability across the set of possible parameter values.

4. The genetic operators are mutation and uniform crossover (see Figure 2). Mutation creates a child by changing one of the four parameter values of the parent to a different value selected randomly from the possible values for that parameter. Uniform crossover selects for the value of each parameter of the child the value of the same parameter from one of the two parents; which parent to inherit from is randomly chosen for each parameter.

5. Some key parameters are population size and generation size, whose values are 1200 and 1 respectively. Section 2.4 explains the need for such a large population. When evaluating all the individuals on a single machine, a generation size of one is usually optimal[4].

We now discuss some of the key issues in getting the genetic algorithm to work properly.

2.3 Choosing a parametrization

We successively tried three different encodings using three different parametrizations of the signals. Changing from the first representation to the second was done by transforming two Cartesian parameters into polar parameters. Changing from the second representation to the third involved a similar transformation of two parameters. Each encoding significantly improved (i.e. found global optima more reliably) over its predecessor. Indeed, with the first representation the genetic algorithm never found a nearly global optimum even for high-SNR signals, while with the third representation the genetic algorithm usually found a nearly global optimum for all signals (see Section 3). This is not surprising given the critical role that representation is known to play in the success of genetic algorithms. In fact, it has been shown that the only difference between a problem which is hard for a genetic algorithm and one which is easy is the choice of represesentation; a change of representation can convert any problem into one which is easy[11].

We now examine a particular example which illustrates how and why a change from Cartesian to polar coordinates has the potential to greatly improve (or worsen) genetic algorithm performance. Consider the problem of optimizing the function $f(r,\theta) = 10(\theta - \theta_0)^2 + (r - r_0)^2$ over the unit disk. Because the term involving θ is the dominant term, the best individuals from the initial population will have $\theta \approx \theta_0$. Now consider the problem of generating children with $\theta \approx \theta_0$ and $r \approx r_0$. If we encode points in the plane using Cartesian coordinates, the standard genetic operators (mutation and crossover) will have a very difficult time generating such children. First, consider crossing (x_1, y_1) with (x_2, y_2), where $\theta_1 = \theta_2 = \theta_0$, $x_1 \neq x_2$ and $y_1 \neq y_2$. The child would be either (x_1, y_2) or (x_2, y_1), both of which have $\theta \neq \theta_0$ and which are therefore worse than either parent. Second, consider mutating (x_1, y_1), where $\theta_1 = \theta_0$. If only one value is changed (the likely scenario), the child will have $\theta \neq \theta_0$ and will thus be worse than the parent. If both values are changed (an unlikely occurrence), the new values (x_2, y_2) will have to be such that $\theta_2 = \theta_0$ and $\|r_2 - r_0\| < \|r_1 - r_0\|$ (an extremely unlikely occurrence). If we instead encode points using polar coordinates, the standard operators work fine. For any point with $\theta = \theta_0$, mutation will change r to be a better value with a relatively high probability. For points (θ_1, r_1) and (θ_2, r_2) such that $\|\theta_1 - \theta_0\| < \|\theta_2 - \theta_0\|$ and $\|r_2 - r_0\| < \|r_1 - r_0\|$, crossover can create the child (θ_1, r_2), which is better than either parent.

Here $\theta = \theta_0$ is an example of a "building block"[9]. A building block is a set of values for a subset of the parameters of the representation which yield better evaluations than other values for these parameters nomatter what the values for the parameters not in the subset. The existence of small building blocks makes a problem easy for a genetic algorithm. For real-world problems, knowing which representation will yield small building blocks is difficult. However, from the results of runs with bad representations, we found clues which pointed to better representations. These clues were similarities between all solutions from the genetic algorithm and the actual global optimum. In the above hypothetical example, for the Cartesian encoding we would always find the ratio of x and y for the genetic algorithm solution to be the same as the ratio of x and y for the global optimum, thus directing us towards using θ as a parameter. Both improvements in our encoding of the actual problem were the results of similar reasoning.

2.4 Choosing a population size

When we used a population size of 50 (a standard size) for our genetic algorithm, it always failed. It rarely even generated an individual significantly better than the best individual of the initial population. Increasing the population size to 300 made the genetic algorithm perform better for two observable reasons. First, the best individual of the initial population was usually better (because a random search of 300 points is likely to do better than a random search of 50 points). Second, reproduction was usually able to significantly improve upon the initial best individual. However, even with the best parametrization and a population size of 300, the genetic algorithm usually did not find a nearly global optimum, especially when working with signals of low SNR. Increasing the population size to 600 and then to 1200 greatly improved the performance to the point described in Section 3.

We needed such a large population due to the narrowness of the peak in the search space[12]. The probability of a random individual lying within the region of the search space where there is partial information is very small. However, genetic algorithms rely on recombining the genetic material of a pool of individuals which contain partial information. Increasing the population size increases the expected number of individuals to fall in this region of the space. The genetic algorithm starts to function well when the population size is large enough that the number of initial individual in this region is around some threshold.

There is also a good reason to make the population size as small as possible: a smaller population size means less evaluations before convergence. Increasing the population size beyond 1200 resulted in increased ability to find nearly global optima too minor to justify a larger number of evaluations. If we have sufficient time/resources for more evaluations, we should instead use these evaluations in a manner described in the next section.

2.5 Handling local optima

Even with a large population and a good parametrization, the probability can be high that the genetic algorithm will not find a point near the global optimum. In Section 3 we describe a case for which this probability is around 60%. This is clearly unacceptable, but there is an easy remedy. Each run of the genetic algorithm is independent; hence, if we run it n times, the probability is $(0.6)^n$ of not finding the global optimum. Observing that $(0.6)^4 \approx 13\%$, $(0.6)^6 < 5\%$, and $(0.6)^9 \approx 1\%$, it is clear that taking the best result from multiple runs can make the chance of finding a global optimum (and thus detecting the signal) very good. Note that varying the number of runs allows an explicit tradeoff between the number of evaluations and the probability of finding a global optimum. A similar tradeoff exists for the exhaustive search algorithm described in Section 1.3 by varying the value of the parameter ϵ.

3. RESULTS

We have done some very preliminary performance comparisons of the exhaustive search algorithm and the genetic algorithm using some actual sonar signals as well as some simulated data. To evaluate the exhaustive search algorithm, we generated two samplings of the space, one for $\epsilon = 2.0$ and the other for $\epsilon = 1.0$. The sampling for $\epsilon = 2.0$ contained approximately 600 points and took around three hours to generate. The sampling for $\epsilon = 1.0$ contained approximately 4700 points and took around a week to generate. We would like to have generated even finer samplings but could not do so without first moving to a more powerful computing environment. During experimentation, we quickly decided that the $\epsilon = 2.0$ sampling was far too coarse and concentrated on the $\epsilon = 1.0$ sampling. This too turned out to be too coarse. With $\epsilon = 1.0$, exhaustive search could detect high-SNR signals but could not localize them well in parameter space. Additionally, it could not detect low-SNR signals. This is the exact behavior one would expect from an undersampled exhaustive search, finding the sides of the large peaks and missing the small peaks. The Generalized Hough transform with this search algorithm performed worse at both detection and tracking than a general-purpose detection and tracking algorithm we have developed[4] running on the same data. We estimate that before the GHT with exhaustive search can outperform the general detection and tracking algorithm we need to lower ϵ at least to 0.5, which implies a sampling of at least 40,000 points.

The GHT with genetic search required far fewer evaluations to outperform the general-purpose algorithm. One run of the genetic algorithm consisted of 2300 evaluations, 1200 for the initial population and 1100 during reproduction. The genetic algorithm would find a nearly global optimum around 75% of its runs for high-SNR signals and around 40% of its runs for low-SNR signals. Taking the best of six runs would yield a detection rate of $> 99.9\%$ for high-SNR signals and $> 95\%$ for low-SNR signals. This is a big improvement over the general-purpose algorithm, which usually cannot detect the low-SNR signals. Additionally, the genetic search yields very precise tracks; in simulated data it finds parameter values very close to those used to generate the tracks. Since six runs requires only 13800 evaluations, we conclude that the genetic search is at least three times faster than the exhaustive search.

4. CONCLUSION

When the set of possible signals is a low-dimensional set of curves, a Generalized Hough transform (GHT) can yield detection and tracking performance superior to that of a general-purpose algorithm. The problem with the GHT approach is that searching the transform space to find the peaks corresponding to signals can be a very computationally expensive procedure. We have developed two algorithms for performing this search: (1) an exhaustive search using a discretization of the space which is approximately uniformly distributed in the space of curves and (2) a genetic algorithm. To get the genetic algorithm to run properly required tuning some of its parameters, the most critical being (1) the encoding, (ii) the population size, and (iii) the number of runs. In preliminary tests, the genetic algorithm outperformed the exhaustive search algorithm, providing at least a three-fold increase in speed.

5. ACKNOWLEDGMENTS

Thanks are due to the following people: Steve Milligan, for his suggesting this problem and for his useful discussions; and Dave Davis, for the use of his OOGA code.

6. REFERENCES

1. P. V. C. Hough, "Method and Means for Recognizing Complex Patterns", U.S. Patent 3,069,654; 1962.

2. S. R. Deans, *The Radon Transform and Some of Its Applications,* Wiley, New York, 1983.

3. D. H. Ballard and C. M. Brown, *Computer Vision,* Prentice-Hall, Englewood Cliffs, NJ, 1982.

4. D. J. Montana, "Genetic Optimization of the Parameters of a Track-While-Detect Algorithm," *in these proceedings.*

5. S. S. Blackman, *Multiple Target Tracking with Radar Applications,* Artech House, Norwood, MA, 1986.

6. Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association,* Academic, New York, 1988.

7. Y. Barniv, "Dynamic Programming Solution for Detecting Dim Moving Targets," *IEEE Trans. Aerospace and Electrical Systems,* vol. 21, no. 1, pp. 144–156, 1985.

8. M. C. Smith and E. M. Winter, "On the Detection of Target Trajectories in a Multi-Target Environment," *Proc. IEEE Conf. Decision and Control,* pp. 1189–1194, 1978.

9. D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning,* Addison-Wesley, Redwood City, CA, 1988.

10. L. Davis, *Handbook of Genetic Algorithms,* Von Nostrand Reinhold, New York, 1991.

11. M. D. Vose and G. E. Liepins, "Schema Disruption," *Proc. Fourth International Conference on Genetic Algorithms,* pp. 237–242, 1991.

12. D. J. Montana, "A Weighted Probabilistic Neural Network," *Advances in Neural Information Processing Systems 4,* J.E. Moody, S.J. Hanson, and R.P. Lippman (eds.), Morgan Kaufmann, San Mateo, CA, 1992.