Generation of Convex Hull Using Neural Network Technique

Ashish Kumar Nayyar*
Varun Sapra**

Abstract

The convex hull problem has had a long history going back to the beginning of computational geometry and has been an intensively studied subject even up to the present day. The computation of the convex-hull of a finite set of points, particularly on the plane, has been studied extensively and has wide applications in pattern recognition, image processing, cluster analysis, statistics, robust estimation, operations research, computer graphics, robotics, shape analysis, and several other fields. A convex-hull based shape representation is suitable for classification and recognition of irregular objects because it is invariant with respect to coordinate rotation, translation, and scaling.

Since 1970s, the problem of convex-hull computation has been an interesting area of research. As a result, a wide variety of algorithms are available in the literature to solve this problem. Early papers dealt primarily with the planar case d = 2. These wide varieties of algorithms were classified as (a) computing exact convex hull and (b) computing approximate convex hull. They were also known as sequential and parallel. Sequential means using a single processor for the computation and parallel means using multiple processors for the purpose of computation.

This research aims in finding the solution of the convex hull problem using a neural network technique. The convex hull problem is the problem of computing the convex hull of S and reporting the points on the convex hull in the order in which they appear on the hull where S = {S[0], . . . , S[n – 1]} be a set of n distinct points in the Euclidean plane. This problem has been solved efficiently using standard methods but our research is confined to find a solution of the problem using neural network. Because convex hull is basically a pattern recognition problem and neural networks are good at pattern recognition.

Keywords: Neural, Convex, Hull, Euclidean, Computation, etc.

Introduction

Computational Geometry

The term "computational geometry" was coined in the mid-1970's, geometry is one of the longest studied mathematical areas because of its enormous number of applications. With the advent of computers, which could perform millions of mathematical calculations per second, new topics and problems in geometry began to emerge and the field of computational geometry was born. As it evolved, a number of new applications became apparent, ranging from computer vision and geographical data analysis, to collision detection for robotics and molecular biology. Computational geometry studies the design and analysis of algorithms for solving geometric problems. One central problem that has received considerable attention in the area is the problem of constructing convex hulls. The importance of the problem stems not only from its many applications (such as to pattern recognition, statistics, and image processing) but also from its usefulness as a tool for solving a variety of problems in computational geometry. The concept of convex hulls is well-studied in mathematics and is appealing both mathematically and intuitively.

The Convex Hull Problem

Given a set of planar points, the two-dimensional convex hull problem is to find the convex polygon with the smallest possible area which completely contains all of the points. Let S = {S[0], . . . , S[n – 1]}

Ashish Kumar Nayyar*
IITM, New Delhi

Varun Sapra**
JIMS, New Delhi

be a set of n distinct points in the Euclidean plane. The convex hull of S is the minimal convex region that contains every point of S. From this definition, it follows that the convex hull of S is a convex polygon whose vertices are points of S. For convenience, we say that a point p to S is "on the convex hull of S" if p is a vertex of the convex hull of S [3].

The convex hull of a finite point set S = {P} is the smallest 2D polygon Ω (or polyhedron in 3D) that contains S. That is, there is no other polygon (or polyhedron) Λ with $S \subseteq \Lambda \subseteq \Omega$. Also, this convex hull has the smallest area and the smallest perimeter of all polygons containing the set S.

Convex hulls find use in a number of different applications including:

- Collision avoidance in robotics.
- Pattern recognition and digital image processing.

Fundamentals of Neural Networks

Neural Networks are relatively crude electronic models based on the neural structure of the brain. The brain basically learns from experience. It is natural proof that some problems that are beyond the scope of current computers are indeed solvable by small energy efficient packages. This sweeping success can be attributed to a few key factors:

- **Power.** Neural networks are very sophisticated modeling techniques capable of modeling extremely complex functions. In particular, neural networks are ***nonlinear***. Neural networks also keep in check the curse of dimensionality problem that bedevils attempts to model nonlinear functions with large numbers of variables.
- **Ease of Use.** Neural networks learn by example. The neural network user gathers representative data, and then invokes training algorithms to automatically learn the structure of the data.

Some Important properties of neural networks

- **Trainability:** Networks can be taught to form associations between any input and output patterns.
- **Generalization:** Networks don't just memorize the training data; rather, they learn the underlying patterns, so they can generalize from the training data to new examples.
- **Nonlinearity:** Networks can compute nonlinear, nonparametric functions of their input, enabling them to perform arbitrarily complex transformations of data.
- **Robustness:** Networks are tolerant of both physical damage and noisy data; in fact noisy data can help the networks to form better generalizations.
- **Uniformity:** Networks offer a uniform computational paradigm which can easily integrate constraints from different types of inputs.
- **Parallelism:** Networks are highly parallel in nature, so they are well-suited to implementations on massively parallel computers.

Literature Review

The convex hull problem has had a long history going back to the beginning of computational geometry and has been an intensively studied subject even up to the present day. The computation of the convex-hull of a finite set of points, particularly on the plane, has been studied extensively and has wide applications in pattern recognition, image processing, cluster analysis, statistics, robust estimation, operations research, computer graphics, robotics, shape analysis, and several other fields.

Since 1970s, the problem of convex-hull computation has been an interesting area of research. As a result, a wide variety of algorithms are available in the literature to solve this problem. Early papers dealt primarily with the planar case d = 2. These wide varieties of algorithms were classified as (a) computing exact convex hull and (b) computing approximate convex hull. They were also known as sequential and parallel. Sequential means using a single processor for the computation and parallel means using multiple processors for the purpose of computation. The birth of computational geometry is often credited to Ronald Graham's research and subsequent 1972 paper titled "An efficient algorithm for determining the convex hull of a finite planar set." His algorithm was a response to Bells Lab's request for a faster algorithm. They had to determine the convex hull of ten thousand points rapidly, a challenging number in the late 1960s with existing O(n2) algorithms. Graham was hired and developed what is now known as Graham's Scan, an O(nlogn) convex hull algorithm.

Some of the algorithms for solving convex hull problem

Table1: For Solving Convex Hull Problem

S.No	Algorithm	Speed	Discovered By
1	Brute Force	O(n3)	[Anon, the dark ages]
2	Gift Wrapping	O(nh)	[Chand & Kapur, 1970]
3	Graham's Scan	O(n log n)	[Graham, 1972]
4	Jarvis March	O(nh)	[Jarvis, 1973]
5	Quick Hull	O(nh)	[Eddy, 1977], [Bykat, 1978]
6	Divide and Conquer	O(n log n)	[Preparata & Hong, 1977]
7	Monotone Chain	O(n log n)	[Andrew, 1979]
8	Incremental	O(n log n)	[Kallay, 1984]
9	Marriage-Before-Conquest	O(n log n)	[Kirkpatrick & Seidel, 1986]

Proposed Algorithm

In this research paper, we have developed an algorithm for the construction of convex hulls. Conceptually the algorithm is a connectionist implementation of Gift Wrapping algorithm. We have implemented the algorithm by a self organizing neural network.

A Self Organizing Map is a special class of artificial neural networks based on unsupervised competitive learning. It refers to the ability to learn from the input without having any prior supervising information. A Self Organizing Map inspects the input vectors for some hidden patterns, and by cooperatively adjusting the weights of output layer of neurons to make localized response to the input data.

Algorithm

The algorithm for the construction of the convex hull is as follows:

Step 1 Generate random points in a plane and find the distance of each point from the left side of the plane and set the initial weights as Wi(0) = Xj – Xk

Step 2 Find the winning Neuron, say either nearest or farthest from the left side of the plane as it always lie on the hull edge.

Step 3 Choose the point as a successor processor P(s) and from Ps draw all possible edges.

Step 4 After finding the first hull vertex, now we need to find the other vertex so that on joining these two vertices we will get a hull edge.

(Selection of plane and the next hull vertex)

Step 4.1 Rotate the plane counter clockwise and again find the distance of each node from current side.

Step 4.2 Select the minimum distance point and draw an edge between the nodes and check whether the edge is an upward edge or downward edge. An edge is downward if its y1 is less than y2 and upward if y1 is greater than y2.

Step 4.3 Now check if there is any other node lying down to the line.

Step 4.3.1 Mark all those points whose value lies in between the top values of the selected nodes.

Step 4.3.2 Move along the edge drawn in step 4.2 and if the top value of the node under check is achieved but with lesser left value (X – Coordinate Value) on that edge, then that node lies above that line. (Applicable for Upward edges and for downward edges if the top value of the node is achieved with less left value then that node lies down to the line.) Mark all those points which lie down to the line.

Step 4.3.3 Out of the nodes which lie down to the line select the node with minimum distance from the previous side of the plane.

Step 4.3.4 Draw an edge between the newly selected points and repeat steps 4.3.1 and 4.3.2 for any other node lying down to the line keeping the side of the plane constant.

Step 4.3.5 If no other such point lies then repeat step 5 to generate other hull edges.

Step 5 The resultant will be a convex hull polygon.

Conclusion and Future Work

Every neuron is computing independently, it takes constant time. The next processor value is then assigned to the node Hence it takes O(1) time. The algorithm needs to selects the winner n number of times as n are the number of hull processors. Thus it takes O(n) time to compute (Best Case). (In worst case)If the algorithm calls itself recursively for the finding the points down to the drawn edge, This leads to O(n) recursions, therefore overall it is O(n2).

Future Scope: - In future scope, convex hulls can also be computed in higher dimensional space. Further, neural networks can be employed in convex hull determination in N-dimensional space. Another active area of research is in the development of efficient parallel algorithms for convex hull computation. It is beyond the scope of this thesis to extensively cover the literature.

References

1. Mulmuley K., Computational Geometry (1993) "An Introduction Through Randomized Algorithms". Prentice-Hall, Englewood Cliffs, N.J.
2. O'Rourke J. (1994) "Computational Geometry in C", Cambridge University Press.
3. Brionnimann.H, et al, Space-Efficient Planar Convex Hull Algorithms, Elsevier Science, This Research was Registered under Contract 9801749
4. Agarwal P.K. (1994): "Applications of Parametric Searching in Geometric Optimization," *J. Algorithms,* 17, 292–318.
5. Akl S.G. and Lyons K.A. (1993) "Parallel Computational Geometry." Englewood Cliffs, NJ: Prentice-Hall, 1993.
6. V. Capoyleas, G. Rote, and G. Woeginger(1991) "Geometric Clustering," *J. Algorithms,* 12, 341–356.
7. P.A. Devijver and J. Kittler(1982) "Pattern Recognition: A Statistical Approach", NJ: Prentice-Hall.
8. Earnshaw R.A. (1988) "Theoretical Foundations of Computer Graphics and CAD," in NATO ASI. Berlin, Germany: Springer-Verlag, 40.
9. H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel(1983) "On the Shape of a Set of Points in the Plane," *IEEE Trans. Inform. Theory,* IT-29, 551–559.
10. Y. K. Hwang and N. Ahuja(1993) "Cross Motion Planning – A Survey," *ACM Comput. Survey,* 24, 219–291.
11. F. P. Preparata and M. I. Shamos(1985) Computational Geometry: An Introduction. New York: Springer-Verlag.
12. J. T. Schwartz and C. K. Yap, Eds.(1987), Advances in Robotics I: Algorithmic and Geometric Aspects of Robotics. Hillsdale, NJ: Lawrence Erlbaum.