Convex Hull Algorithms

Thiago Pereira - tpereira@impa.br
Computational Geometry Report
IMPA - Instituto de Matemtica Pura e Aplicada

September 20th, 2008

1 Introduction

The objective of this work is comparing algorithms for the planar convex hull problem. We implemented three convex hull algorithms, as well as a clever pre-processing technique. We were interested in evaluating the work required to implement these algorithms and validate the efficiency results predicted by complexity theory.

2 Implementation

We implemented the Jarvis March [1, 2] algorithm with complexity $O(nh)$. Here n is number of input points and h is the size of the resulting convex hull. Since h is not know a priori, Jarvis March is known as an output-sensitive algorithm. This is not asymptotically optimal since $h = O(n)$ in the general case leading to an overall complexity of $O(n^2)$. In practice this algorithm is still useful since it is simpler to implement and can be very efficient if the hull size is known to be small.

We also implemented two versions of the Graham Scan algorithm. The first version will be simply called Graham Scan in this report. This algorithm finds an extreme point and sorts the points angularly with respect to it as can be seen in [1, 2]. It meets the optimal complexity of $O(n \log(n))$. This Graham Scan implementation is largely based on [3].

The second one is known as Upper Lower hull [4, 2] requires only sorting the points along the x direction. It's complexity is also $O(n \log(n))$. This sorting requires simpler operations. Since sorting is the step that determines the complexity of these algorithms, it is expected that a simpler sorting algorithm will result in a faster program, even though both have the same complexity.

Jarvis March was the easiest of the three to code. Upper Lower hull is easier than Graham Scan to code since it does not require the angular sorting step. Yet the time taken to code the three algorithms was similar.

We also made experiments with an $O(n)$ pre-processing step that eliminates points that are trivially in the convex hull. All of the above algorithms were

tested with and without pre-processing the point set. This linear step is asymptotically irrelevant when compared to either $O(n \log(n))$ for both Graham Scan or $O(nh)$ for Jarvis March. As such these harder steps will be run on fewer points and should lead to an improvement in the overall running time. We can eliminate any point found to be a convex combination of other three points. This means if we can find a triangle that is likely to contain many points, this pre-processing step will be very effective. We find the first u and last v points in lexicographical order ordering by x first. Next we search in $O(n)$ for the point w that maximizes the area of triangle uvw. Experiments showed that using only this triangle for point elimination did more harm than good. The number of points left is reduced but not enough to compensate for the time spent with pre-processing. The solution was simple: find two triangles, one that maximizes and one that minimizes the signed area. Minimizing a signed area gives a triangle which has large absolute area. All of this is still $O(n)$ and point elimination is now effective at reducing overall processing time as will be shown in the next section.

The three algorithms cited above were tested on random sets of points. Four scenarios were tested using different point probability distribution: uniform points in a square, uniform points in a triangle, uniform points in a circle and uniform points along a circunference. Each of these scenarios was tested with 100, 1000, 10000, 100000, 1000000 and 10000000 sampled points. The same set of point samples was used for the algorithms.

The algorithms were implemented in C++ using the Standard Template Library (STL) for sorting and data structures.

3 Results and Discussion

We first tested the algorithms on the test set available in the course's website obtaining Figure 6. Tests were also made on degenerate cases with input of one, two and three points, and also collinear cases.

We begin by presenting the testing scenarios used for performance evaluation. For the different point set sizes and distributions, Table 1 shows the size of the convex hull obtained. The same data is shown in Figure 1 in a plot of $\log(y) \times \log(n)$. In this chart, the slope of each line a can be interpreted as a polynomial growth $y = n^a$. It can be easily seen that the size of the convex hull for points in a circunference is n, for points in a circle is approximately $O(n^{1/3})$ and for points in a square or triangle growth is sub-polynomial approximately $O(\log(n))$. All these observations are coherent with the estimates provided by stochastic geometry [5]. Some of the resulting convex hulls can be seen in Figures 7, 8, 9 and 10.

We next present the tables and charts for the experiments with points in a square, triangle, circle and circunference. All times are shown in seconds.

For the square and triangle tests, $h = O(\log n)$. This means $O(nh) = O(n \log n)$, that is Jarvis March is optimal. The experiments agree with this theoretical result. It can be seen that Jarvis March and Graham Scan have

Samples	Rectangle	Triangle	Circle	Circunference
100	9	12	15	100
1000	15	14	35	1000
10000	23	18	73	10000
100000	33	29	146	100000
1000000	40	32	349	1000000
10000000	45	36	730	10000000

Table 1: Convex Hull Size

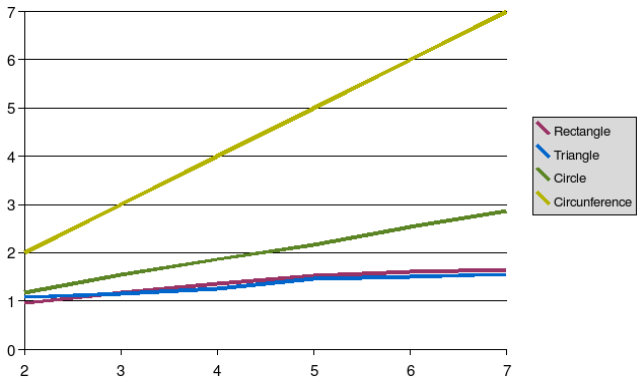

Figure 1: Convex Hull Size Growth

Algorithms	100	1000	10000	100000	1000000	10000000
Jarvis	0	0	0.04	0.54	6.49	73.05
Jarvis Pre	0	0	0.02	0.27	3.45	38.65
Graham	0	0.01	0.03	0.36	4.24	48.72
Graham Pre	0	0	0.01	0.2	2.22	26.04
Upper Lower	0	0	0.01	0.11	1.43	15.98
Upper Lower Pre	0	0	0.01	0.1	1.04	11.2
Points Left	21	213	2900	43210	450803	4553277
HullSize	9	15	23	33	40	45

Table 2: Square Distribution

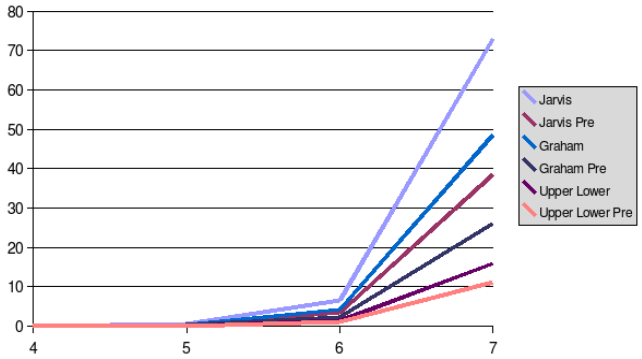

Figure 2: Square Point Distribution

similar running times.

For points in a circunference, the size of the convex hull is the same as the size of the input n. This is the worst possible scenario for the Jarvis March algorithm $O(n^2)$ as $h = O(n)$. For 1000000 and 10000000 points Jarvis March was not executed since it is very slow.

For larger sampling density (1000000 or 10000000 points), the algorithms were not behaving properly at first. This was due to lack of robustness in the code. This problem was solved by using floating point comparisons with smaller tolerances ($1e-10$) and by avoiding floating point comparisons whenever possible. Even after these improvements some points are still being lost in the circunference algorithm with large number of points. It still clear from this experiment that much care needs to be taken when implementing geometry algorithms, specially concerning rounding errors.

In all the experiments above the Upper Lower Hull algorithm with lexicographical sorting outperforms the Graham Scan with angular sorting by a factor that ranges between 2 and 4.

The point elimination pre-processing step was very effective. Overall running

Algorithms	100	1000	10000	100000	1000000	10000000
Jarvis	0	0	0.03	0.48	4.43	58.65
Jarvis Pre	0	0	0.01	0.05	0.43	4.09
Graham	0	0	0.03	0.35	4.16	49.3
Graham Pre	0	0.01	0	0.04	0.42	4.06
Upper Lower	0	0	0.01	0.12	1.4	16.12
Upper Lower Pre	0	0	0.01	0.05	0.41	4.05
Points Left	18	38	296	683	1367	6888
HullSize	12	14	18	29	27	36

Table 3: Triangle Distribution

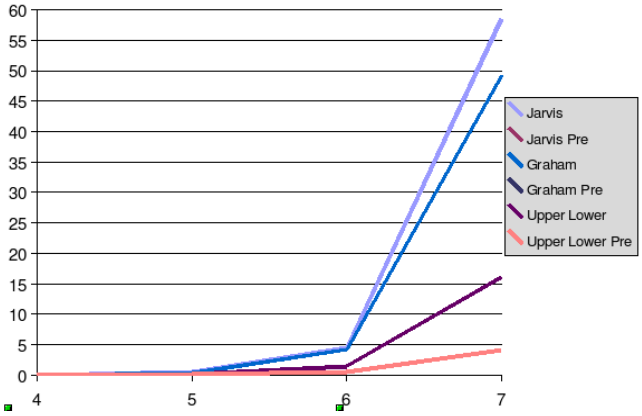

Figure 3: Triangle Point Distribution

Algorithms	100	1000	10000	100000	1000000	10000000
Jarvis	0	0.01	0.12	2.37	56.1	1174
Jarvis Pre	0	0	0.05	0.89	21.5	429.92
Graham	0	0.01	0.03	0.36	4.17	49.26
Graham Pre	0	0	0.01	0.16	1.87	20.7
Upper Lower	0	0	0.01	0.13	1.44	16.09
Upper Lower Pre	0	0	0	0.09	0.91	9.64
Points Left	37	372	3676	36226	363751	3633142
HullSize	15	35	73	146	349	730

Table 4: Circle Distribution

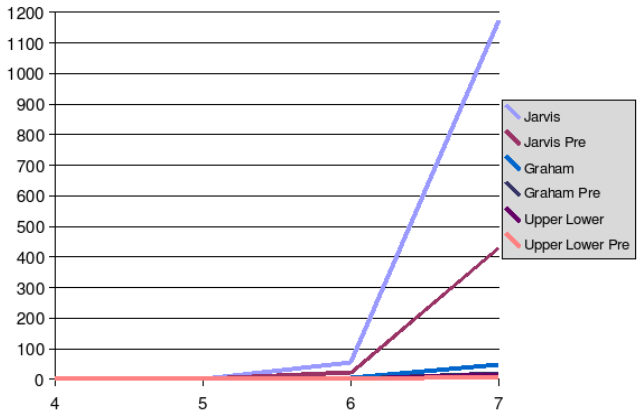

Figure 4: Circle Point Distribution

Algorithms	100	1000	10000	100000	1000000	10000000
Jarvis	0	0.16	16.12	1604	-	-
Jarvis Pre	0	0.17	16.21	1601	-	-
Graham	0	0	0.03	0.35	4.17	49.45
Graham Pre	0	0	0.05	0.4	4.62	53.97
Upper Lower	0	0	0.01	0.12	1.41	16.03
Upper Lower Pre	0	0.01	0.01	0.18	1.86	20.59
Points Left	100	1000	10000	100000	1000000	10000000
HullSize	100	1000	10000	99997	999742	9960644

Table 5: Circunference Distribution

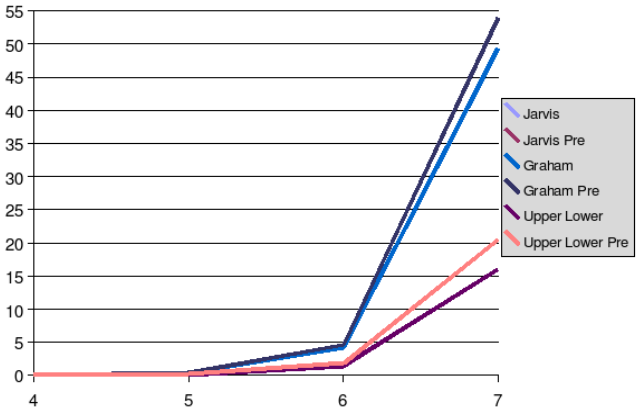

Figure 5: Circunference Point Distribution

time (including the elimination and the convex hull algorithm itself) was usually improved by a factor of 2. This was observed, except for the circunference test. In this case no points are eliminated so that spending time with pre-processing only makes it worse. For points in a square or circle about 60% of the points are eliminated. For the triangle test only a very small fraction of the points are left after elimination, resulting in a much faster algorithm.

4 Conclusion

The experiments have shown that the Jarvis March algorithm was never faster than either of the Graham Scan algorithms, while sometimes it was much slower. We still cannot rule this algorithm out. Since it is output sensitive, it is expected to be faster if the number of points on the convex hull is small. It can even be considered a linear algorithm if the hull size is a constant.

It was also found that Upper Lower Hull is easier to implement and runs faster than Graham Scan, being recommended for most cases even though both algorithms are assymptotically the same.

All the experiments have shown that complexity analysis theory is very useful in predicting algorithm behaviour. Yet one should not forget that the $O()$ and $\Omega()$ notations may hide large constants and may hide some slower growth functions.

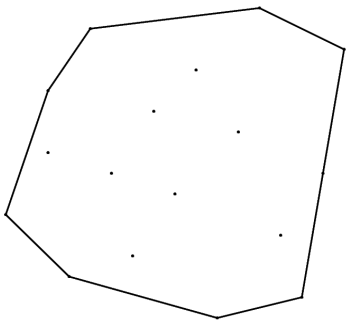

Figure 6: Standard Test Set

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition.* McGraw-Hill Science/Engineering/Math, July 2001.

[2] Luiz Henrique de Figueiredo. *Notas de Aula de Geometria Computacional.* 2005.

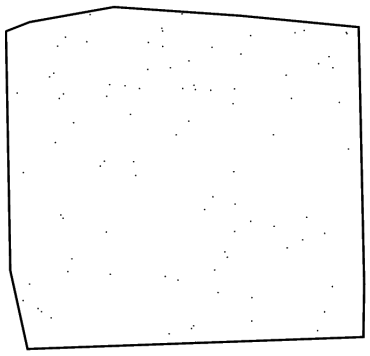

Figure 7: Convex Hull of a Square with 100 points

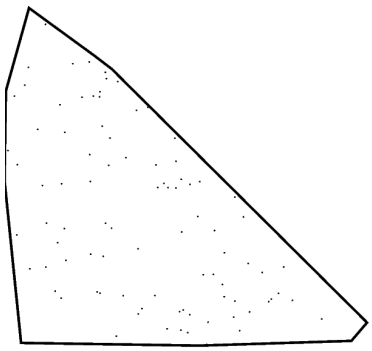

Figure 8: Convex Hull of a Triangle with 100 points

[3] Roberto Cavalcante Daniel Fleischman, Fabio Dias Moreira. Linkedisamba icpc notebook. 2006.

[4] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational geometry: algorithms and applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[5] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction (Monographs in Computer Science)*. Springer, August 1985.

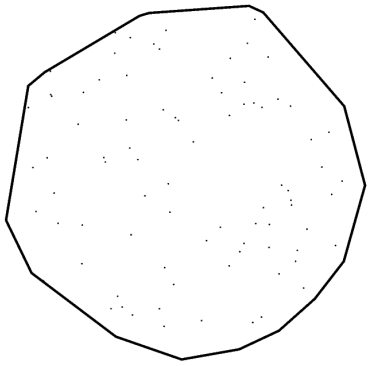

Figure 9: Convex Hull of a Circle with 100 points

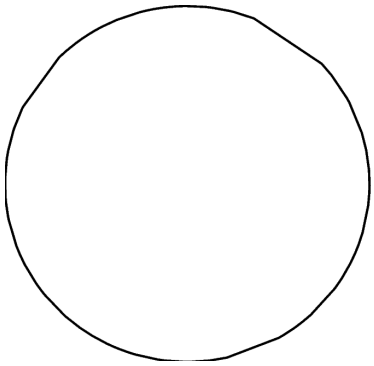

Figure 10: Convex Hull of a Circunference with 100 points