논문번호 13-01-05

Journal of IKEEE.Vol.17,No.1,029~035,March 2013
http://dx.doi.org/10.7471/ikeee.2013.17.1.029

평면 점집합에서 정렬을 고려한 개선된 컨벡스 헐 알고리즘

An Improved Convex Hull Algorithm Considering Sort in Plane Point Set

박 병 주*, 이 재 흥*★

Byeong-Ju Park*, Jae-Heung Lee*★

Abstract

In this paper, we suggest an improved Convex Hull algorithm considering sort in plane point set. This algorithm has low computational complexity since processing data are reduced by characteristic of extreme points. Also it obtains a complete convex set with just one processing using an convex vertex discrimination criterion. Initially it requires sorting of point set. However we can't quickly sort because of its heavy operations. This problem was solved by replacing value and index. We measure the execution time of algorithms by generating a random set of points. The results of the experiment show that it is about 2 times faster than the existing algorithm.

요 약

본 연구에서는 임의의 정렬되지 않은 점집합에서 정렬을 고려한 개선된 Convex Hull 알고리즘을 제안한다. 이 알고리즘은 Convex Hull의 극점 특성을 이용하여 처리 데이터를 한정하기 때문에 계산복잡도가 낮다. 각 단계마다 볼록 정점을 판별하는 조건을 이용하여 한 번의 스캔으로 완전한 Convex Set을 구한다. 알고리즘 초기에 점집합의 정렬이 필요한데, 이때 걸리는 시간이 알고리즘 전체 동작시간의 대부분을 차지하기 때문에 값과 인덱스를 대치하여 빠르게 정렬하였다. 일반적인 상황을 가정하여 랜덤한 점집합으로 알고리즘의 동작시간을 측정하였으며 기존의 알고리즘에 비해 약 두 배의 속도 향상이 있음을 확인하였다.

Key words : Convex Hull, Convex Polygon, Extreme Point, Image Processing, Parallel Processing

* Dept. of Computer Engineering, Hanbat University
dinobei89@gmail.com 042-821-1420

★ Corresponding author

※ Acknowledgment
This research was financially supported by the Ministry of Education, Science Technology (MEST) and National Research Foundation of Korea(NRF) through the Human Resource Training Project for Regional Innovation

Manuscript received Feb. 26, 2013; revised Mar. 15, 2013; accepted Mar 15. 2013

Ⅰ. 서론

Convex Hull 문제는 점집합 혹은 도형이 주어졌을 때 그것을 포함하는 최소 면적의 Convex Polygon을 구하는 문제이다[1]. 연산 결과인 Polygon은 기존 점집합의 부분집합이며 Convex Set이라고도 한다. 이 알고리즘은 계산 기하학(Computational Geometry)에서 기본적인 문제로 다양한 분야에서 사용된다. 가령 컴퓨터그래픽스, GIS (Geographic Information Syst-

em), HCI (Human Computer Interaction), 로보틱스와 같은 분야에 응용될 수 있다[1].

Convex Hull을 SVM (Support Vector Machine)과 같은 인식기에서 사용할 불변 특징 추출[2]에 관한 연구나 빠른 인식기 학습을 위해 응용하는 연구[3] 등 패턴인식 분야에서의 연구가 활발하다. 또, 모핑 알고리즘[4], 볼륨 렌더링 알고리즘[5]과 같은 영상처리 분야에서의 연구도 활발하며 이외 다양한 분야에서의 실제 연구사례를 찾아볼 수 있다.

현재 많이 사용하는 Convex Hull 알고리즘은 주어진 모든 정점을 탐색한다. 이들 정점의 대부분은 실제 Convex Set의 내부에 포함되는 정점이기 때문에 탐색이 불필요한 점들이 대부분이다. 이렇게 모든 정점을 탐색하는 것은 곧 계산복잡도가 커지는 문제를 초래하게 된다. 따라서 많은 정점에 대한 실시간 응용에는 불리하다.

본 논문에서는 이러한 문제를 해결하기 위해 2차원 영상에서 Convex Set을 구하기 위하여 Convex Hull의 특성인 극점(Extreme Point)을 이용하였다. 이에 따라 알고리즘의 계산 복잡도가 낮고 구현도 간단한 개선된 Convex Hull 알고리즘을 제안한다. 제안하는 알고리즘의 성능을 확인하기 위해 기존 알고리즘과 제안하는 알고리즘의 수행시간을 측정하고 비교하였다.

II. 기존의 Convex Hull 알고리즘

기존의 Convex Hull 알고리즘은 크게 두 가지 방법으로 나눌 수 있다. 이들의 주된 차이는 주어진 점집합에 대해 정렬을 하는가에 있다.

우선, 널리 알려진 기존의 알고리즘을 시간 복잡도와 함께 살펴보면 Graham's Scan[6], Quickhull[7]과 같은 알고리즘은 $O(nlogn)$, Gift-wrapping 알고리즘은 $O(nh)$의 시간복잡도를 가진다. 여기서 h는 극점의 개수를 뜻한다. Quickhull 알고리즘은 Divide and Conquer 방식을 이용하여 Convex Set을 계산하기 때문에 점집합을 정렬하지 않고 알고리즘이 수행된다. 이들 알고리즘 자체는 최적이지만 비싼 계산 때문에 다른 최적 알고리즘들이 제안되었다.

Dong Wei[8]의 알고리즘과 Xianquan Zhang[9]의 알고리즘은 극점을 통해 계산복잡도를 줄인 알고리즘이다. 극점을 사용하는 알고리즘은 특정 범위 내에 속하는 좌표만을 신속히 탐색해야 하기 때문에 초기에 점집합을 정렬해야 한다. 따라서 알고리즘의 시간복잡도는 정렬의 시간복잡도에 지배되어 정렬 알고리즘의 이론상 최소 시간복잡도인 $O(nlogn)$이 되게 된다.

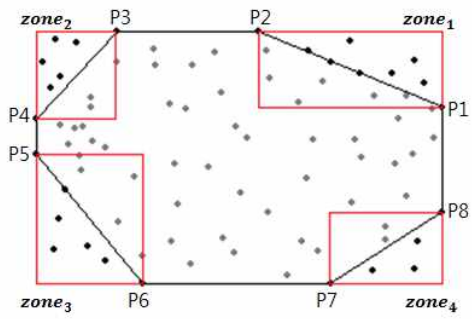

Fig 1. Extreme point of point set & area division
그림 1. 점집합의 극점과 영역 분할

극점을 사용하는 알고리즘[8]을 조금 더 자세히 살펴보면, 극점에 의해 점집합은 다섯 개의 영역으로 나뉨을 알 수 있다. 그림 1은 극점과 영역 분할을 표현하고 있는데, 극점을 모두 이어 Polygon을 구성하면 이 Polygon 내부에 포함된 정점들은 모두 볼록(Convex Vertex) 정점이 됨을 알 수 있다. 4개의 영역 각각에 대해 볼록 정점 후보를 추려낸 뒤, 반시계 방향으로 Graham's Scan의 Circulation과 Back Tracking 방법을 사용하여 오목 정점(Concave Vertex)을 제거하게 되고 남은 정점들이 Convex Set이 된다.

이들 알고리즘은 초기에 점집합의 정렬이 필수적으로 요구되며 x, y에 대해 두 번의 정렬을 하도록 되어 있다.

III. 개선된 Convex Hull 알고리즘

1. 알고리즘 구성

극점을 이용하는 기존의 알고리즘에서는 정렬 문제에 대한 별도의 언급 없이 기존의 정렬 알고리즘을 사용하도록 하고 있다. 하지만 실제 Convex Hull 알고리즘에서 정렬에 걸리는 시간의 비중이 매우 높기 때문에 정렬 방법에 대한 고려가 있어야 한다.

그림 2와 같이 $O(nlogn)$으로 동작하는 정렬 알고리즘을 이용하면 기존의 Convex Hull 알고리즘의 전체 수행 시간보다 더 오래 걸린다. 이러한 문제를 알고 있지만 정렬 알고리즘의 동작속도 향상에 대한 근본적인 개선은 이론상 불가능하다. 따라서 입력으로 주어질 점집합의 좌표를 특정한 크기의 영상에 속한 것으로 한정하여 빠르게 정렬을 수행하고 Convex Set을 계산하도록 하였다. 일단 점집합이 정렬된 이후에는 극점을 탐색하여 Convex Set의 후보 영역을 한정

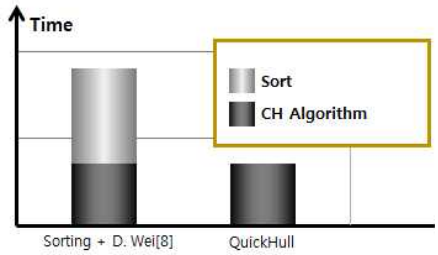

Fig 2. Rate of the execution time including sorting
그림 2. 정렬을 포함한 동작시간 비율

하게 된다. 이후에는 Dong Wei[8] 알고리즘과는 다르게 한 번의 탐색으로 온전한 Convex Set을 구하게 된다.

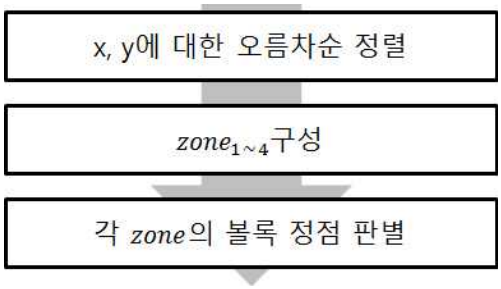

Fig 3. Algorithm configuration
그림 3. 알고리즘 구성도

본 논문에서 제안하는 알고리즘은 그림 3과 같이 크게 세 단계로 이뤄지며 전체 동작은 그림 4와 같다.

2. 점집합의 정렬

$W\times H$ 크기의 영상에서 임의의 점집합이 입력으로 주어질 때 이를 먼저 정렬해야 한다.

이때 정점 좌표 값의 범위가 특정한 크기의 영상으로 한정된 상황에서 빠르게 정렬하는 방법을 이용한다. 정렬할 원소의 개수가 N이고 값의 범위가 $0\sim(M-1)$인 점집합 배열을 G라고 하고, 길이가 M

인 포인터 배열을 pG라 하자. 그림 5와 같이 포인터 배열 pG의 각 인덱스에는 스택이 들어간다.

$G[0]=g_0, G[1]=g_1, ..., G[N-1]=g_{N-1}$일 때, 모든 i에 대해 $pG[g_i]=i$를 할당한다. pG에는 G의 인덱스

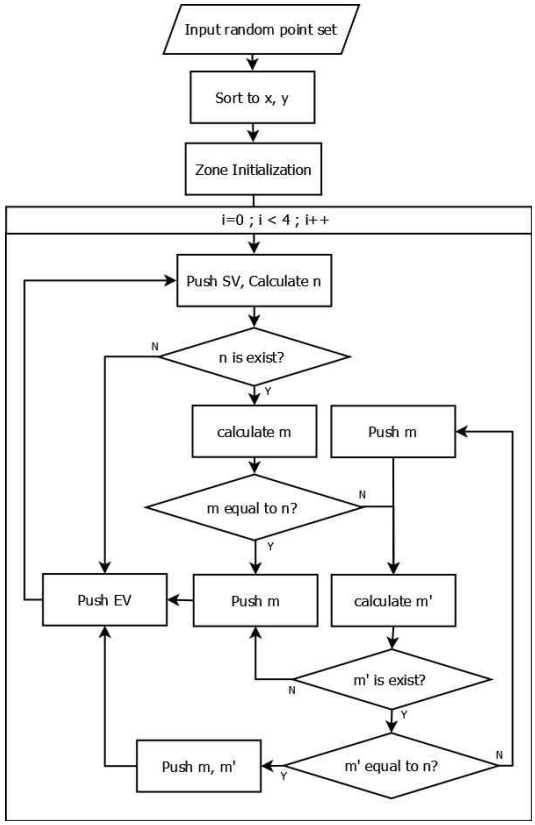

Fig 4. The proposed algorithm
그림 4. 제안하는 알고리즘

값이 들어있다. 즉 값과 인덱스를 대치한 상태가 된다. 이 상태에서 $pG[0]$부터 $pG[M-1]$까지 순서대로 스택에 있는 모든 데이터를 Pop하면 점집합 G의 인덱스 i가 나온다. 각각의 i에 대해서 값으로 대치($G[i]$)하면 정렬된 데이터를 얻을 수 있다. 이렇게 x, y 좌표에 대해서 각각 정렬한 뒤 극점 계산을 통해 Convex Set을 구한다.

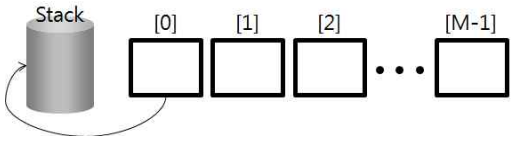

Fig 5. Stack pointer array pG
그림 5. 스택 포인터 배열 pG

3. $zone_{1\sim4}$ 구성

연산 결과 얻어지는 Convex Set 점집합을 편의상 *CS*라고 하고 스택(Stack) 구조로 가정한다. 여기에서 말하는 극점의 의미는 인덱스 x, y 순서대로 MAXIMUM 혹은 MINIMUM을 갖는 점을 말한다.

$zone_1$에서는 메인 인덱스가 x이고 서브 인덱스가 y이다. $zone_2$부터 $zone_4$까지 메인 인덱스와 서브 인덱스가 대치된다. 즉, $zone_2$에서는 메인 인덱스가 y가 되고 서브 인덱스가 x가 된다. $zone_3$에서는 $zone_1$과 같으며, $zone_4$에서는 $zone_2$와 같다. 또, 각 Zone에서는 메인 인덱스와 서브 인덱스 순서로 최댓값을 구할 것인지 최솟값을 구할 것인지 조건이 필요한데, 각각 $cond_1, cond_2$ 라고 언급한다. $zone_1$에서의 $cond_1$은 MAXIMUM이고 $cond_2$는 MINIMUM이다. $zone_2$에서의 $cond_1$은 MINIMUM이고 $cond_2$는 MINIMUM이다. $zone_3$에서의 $cond_1$은 MINIMUM이고 $cond_2$는 MAXIMUM이다. $zone_4$에서의 $cond_1$은 MAXIMUM이고 $cond_2$는 MAXIMUM이다. 표 1은 $zone_1$부터 $zone_4$까지 메인/서브 인덱스와 $cond_1$, $cond_2$를 정리한 것이다.

Table 1. Parameters for calculation of the Extreme Point for each zone
표 1. 각 Zone마다 극점 계산을 위한 파라미터

	main/sub idx		$cond_1$	$cond_2$
$zone_1$	x	y	MAX	MIN
$zone_2$	y	x	MIN	MIN
$zone_3$	x	y	MIN	MAX
$zone_4$	y	x	MAX	MAX

표 1에 정리한 조건대로 구하면 최소 2개부터 최대 8개의 극점이 구해지게 된다. 위에서 언급한 극점의 정의를 확장하면 내부 극점(Inner Extreme Point)은 특정 영역(left, top, right, bottom) 내의 점들만 고려하면 된다. 내부 극점과의 용어 차이를 위해 위에서 언급한 극점은 전역 극점(Global Extreme Point)이라고 정의한다.

그림 6은 본 논문에서 제안하는 알고리즘에서 사용하는 용어를 도식화한 것이다. 이 알고리즘은 반복적으로 내부 극점을 구하여 볼록 정점만을 계산하게 된다. 알고리즘 동작 중에 매번 *CS*에 볼록 정점이 추가되게 된다. 한 번의 정점 스캔으로 온전한 *CS*를 얻기 위해 기울기를 사용한다. 가장 최근에 2개의 점이 이

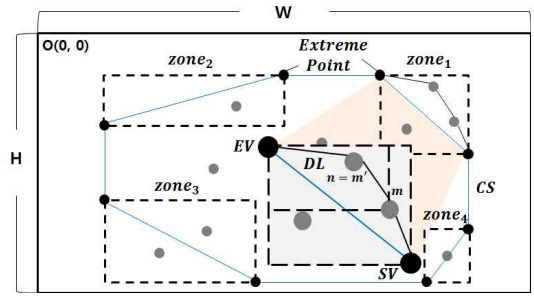

Fig 6. Each term schematization
그림 6. 각 용어 도식화

루는 유향 직선을 *DL* (Directed Line)이라고 정의하고 m과 n은 특정 영역 내의 내부 극점을 뜻한다. m은 *SV*와 *EV* 사이에서 $cond_1, cond_2$ 조건 순서대로 구해진다. 이와 반대로 n은 $cond_2, cond_1$ 조건 순서대로 구해진다. m'은 m과 *EV* 사이에서 구한 m을 뜻한다.

4. 각 *zone*의 볼록 정점 판별

알고리즘은 좌표축 상단이 원점(0, 0)일 때를 기술하였다. 표 1과 같은 조건으로 전역 극점을 구해서 얻어지는 점을 각각 $P_1 \sim P_8$이라고 하자. 특성상 중복되는 정점도 그대로 인정하기로 한다. 얻은 8개의 극점에서 2개씩 묶어서 만들어지는 영역부터 반 시계방향으로 4개의 Zone을 할당한다.

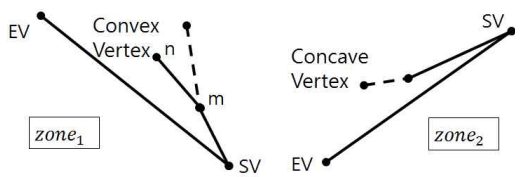

Fig 7. Convex or Concave judgement
그림 7. Convex/Concave 판단

그림 7과 같이 각 Zone에서의 Convex Set을 보면, 반시계방향으로 *DL*의 기울기는 점점 작아짐을 알 수 있다. 즉 $Slope(\overline{SVm}) > Slope(\overline{mn})$ 이다.

이와 같은 특성을 이용하여 *CS*에 정점을 넣을 때 *DL*의 기울기를 계산하여 현재 정점을 넣었을 때의 *DL*의 기울기와 비교하여 더 작을 때에만 *CS*에 Push

연산을 수행한다. 이러한 연산을 편의상 CS에 Push 하였다고 기술하였다. 이 때, 현재 탐색 영역 내에 점이 하나도 없을 때와 하나만 있을 경우 그리고 둘 이상일 경우를 나누어 처리한다. 정점이 한 개 이하로 있을 때는 예외로 보고 특정한 처리로 해당 Zone의 연산을 마친다. 해당 영역 내에 두 개 이상이 있을 때는 한 개 이하가 남을 때까지 같은 동작을 반복하여 처리한다. 먼저 m과 EV 영역 내의 극점 m인 m'이 존재하지 않는다면, 이는 특수한 경우로 m과 n이 일직선상에 있는 것을 뜻한다. 일반적으로 아무 정점도 존재하지 않을 때는 m'으로 n이 선택되어야 하기 때문이다. 따라서 CS에 m과 EV를 순서대로 Push하고 해당 Zone의 연산을 마친다. m'을 성공적으로 구했다면, m'과 n이 동일할 경우와 다를 경우로 나누어 처리한다. 전자일 때는 더 이상의 m'이 없다는 것을 알 수 있으며, 그림 7의 $zone_2$에서 이러한 경우를 보였다. 따라서 CS에 m, m', EV 순서로 Push한다. 후자일 때는 아직 탐색해야 할 정점이 1개 이상 남아있다는 뜻이므로 m을 Push하고 다시 반복 연산의 초기로 돌아간다.

각 Zone에서의 연산은 방향만 다르고 동작 순서는 동일하므로 위에서 언급한 내용이 $zone_1$부터 $zone_4$까지 모든 연산을 설명하고 있다.

Ⅳ 실험 및 고찰

본 연구에서는 극점 특성을 이용하여 기존의 Convex Hull 알고리즘을 개선하였고 기존의 알고리즘들과 비교하기 위해 연산 속도를 측정하였다. 정점 8000개를 예로 들면 퀵 정렬(Quick Sorting) 하였을 때 0.69ms의 시간이 걸린다. 만약 기존의 정렬 알고리즘을 이용한다면 총 2번 정렬하기 때문에 1.38ms 동안 정렬하게 된다. 이러한 결과는 각 단계별 평균 소요 시간을 비교하여 표 2에 나타내었다. 같은 입력에서 정렬을 사용하지 않는 Convex Hull 알고리즘[7]에서는 0.420ms로 3배가량 빨랐다. 표 3과 그림 8에서는 정렬을 사용하지 않는 기존의 알고리즘과 비교한 실험 결과를 보인다.

표 2를 보면 우선적으로 단계 '1. 정렬'에서 걸리는 시간 차이가 매우 크다. '2. 극점 계산'과 '3. $zone_{1\sim4}$ 탐색'에서의 수행시간은 Dong Wei[8]의 방법에서 Convex Hull 후보 정점 때문에 약간의 성능 향상을 확인할 수 있다. 본 논문의 방법에서는 Convex Hull 후보 정점이 없기 때문에 '4. Concave 정점 제거' 단계가 없다. 나아가 Convex Hull 후보 정점의 수가 많을수록 '4. Concave 제거' 단계의 수행시간이 더 커질 것을 예상할 수 있다.

Table 2. The comparison of execution time for each step.

표 2. 각 단계별 수행 시간 비교 (단위:ms)

알고리즘 단계 (Points Number : 8000)	Dong Wei[8]	본 논문의 방법
1. 정렬	1.380	0.193
2. 극점 계산	0.033	0.028
3. $zone_{1\sim4}$ 탐색		
4. Concave 제거	0.008	-
총 수행시간	1.421	0.221

Table 3. The comparison of execution time for algorithm.

표 3. 알고리즘 수행시간 비교 (단위:ms)

Points Number	Quickhull [7]	Graham's Scan[6]	본 논문의 방법
500	0.032	0.411	0.023
1000	0.054	2.054	0.037
2000	0.114	5.76	0.064
4000	0.224	26.02	0.115
8000	0.420	108.313	0.221
16000	0.831	419.16	0.426
32000	1.653	1783.25	0.861
64000	3.356	7481.09	1.721

점집합 P는 $P=\{(x,y)|0\le x<1024,\ 0\le y<768\}$의 범위에 대해 랜덤하게 입력받았고 테스트는 2.66GHz Dual Core CPU의 PC에서 측정하였다. 측정된 시간은 1,000회 수행의 평균값을 사용하였다.

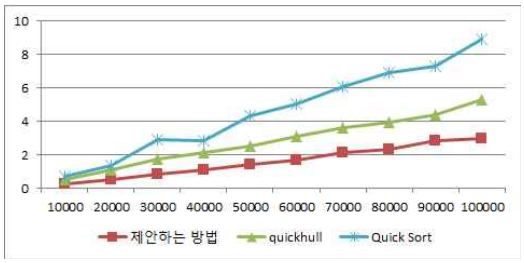

Fig 8. The results of the experiment

그림 8. 실험 결과 그래프

만약 많은 정점에 대한 Convex Hull을 계산해야하는 응용이라면 본 논문에서 제안하는 알고리즘을 쉽게 병렬 처리하여 속도를 높일 수 있다. 초기 x, y의 정렬을 병렬로 처리하고, 각 Zone의 연산은 독립적이기 때문에 병렬로 처리해도 알고리즘은 유효하다. 연산 완료 후에 병합을 통해 온전한 Convex Set을 구할 수 있다. 그림 9에서 보인 구조와 같이 병렬처리하면 처리하지 않은 것과 비교해 2~2.5배 정도 빠르게 동작하게 된다.

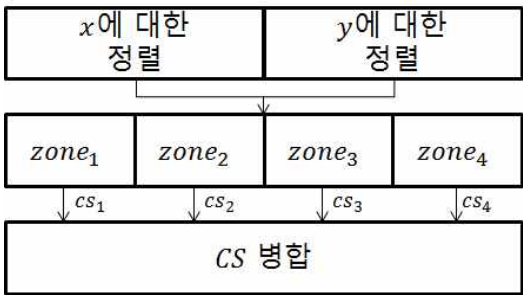

Fig 9. An example of parallel processing
그림 9. 병렬처리의 예

V 결론

본 연구에서는 주어진 점집합에 대한 정렬을 고려한 개선된 Convex Hull 계산 알고리즘을 제안하였다. 크게 점집합에 대한 정렬과 볼록 정점을 탐색하는 연산이 있는데 같은 개수의 점집합에 대해 정렬 알고리즘보다도 기존의 Convex Hull 알고리즘의 동작속도가 더 빠르기 때문에 정렬 문제를 가장 중요한 요인으로 보았다. 이를 해결하기 위해 특정 크기의 2차원 평면에 존재하는 점집합을 입력으로 받는 제한된 환경에서 동작하는 알고리즘으로 한정하였고 여분의 메모리를 사용하여 빠르게 정렬하였다.

또한, 극점을 이용하는 기존의 알고리즘을 개선하여 한 번의 탐색으로 온전한 Convex Set을 구했다. 랜덤하게 구성한 점집합을 이용하여 실험하였으며 정점의 개수와 무관하게 기존의 알고리즘에 비해 평균 2배의 속도로 동작하는 것을 확인하였다.

한편, 알고리즘의 구조상 병렬처리하기에 적합하기 때문에 이 또한 적용할 경우 2~2.5배 정도 빠르게 동작하는 것을 확인하였다.

References

[1] H. K. Cha, S. Y. Shin, "Efficient Convex Hull Algorithm and its Application", KIISE Vol.5, No.4, pp.59-70, 1987

[2] Minhas. R, Wu, J, "Invariant Feature Set in Convex Hull for Fast Image Registration" Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on, vol, no, pp.1557-1561, 7-10 Oct. 2007

[3] Chongming Wu, Xiaodan Wang, Dongying Bai, Hongda Zhang, "A Fast Training Algorithm for SVM Based on the Convex Hulls Algorithm" Signal Processing, 2008. ICSP 2008. 9th International Conference on, vol, no, pp.1578-1581, 26-29 Oct. 2008

[4] Chun-Qiong Gong, "A New Method for the Morph of Planar Polygons Based on Shape Feature" Information Technology, Computer Engineering and Management Sciences (ICM), 2011 International Conference on, vol.3, no, pp.7-10, 24-25 Sept. 2011

[5] He Fei, Li Xia, "A Improved Volume Rendering Algorithm Based on Convex Hull" Information Science and Engineering (ICISE), 2010 2nd International Conference on, vol, no, pp.1-3, 4-6. Dec. 2010

[6] R.L. Graham, "An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set", Information processing letters pp.132-133, 1972

[7] W Eddy, "A New Convex Hull Algorithm for Planar Sets", ACM Transactions on Mathematical Software, vol. 3, no. 4, pp.398-403, Dec 1977

[8] Dong Wei, XingHua Liu, "Improved Algorithm for Computing Convex Hull of Plane Point Set", Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on , vol., no., pp.1-4, 11-13. Dec. 2009

[9] X Zhang, Z Tang, J Yu, M Guo, "A Fast Convex Hull Algorithm for Binary Image", Informatica: An International Journal of Computing and Informatics, 2010

BIOGRAPHY

Park Byeong-Ju (Member)

2012 : BS degree in Computer Engineering, Hanbat University.
2012 ~ Present : MS Course in Computer Engineering, Hanbat University.

Lee Jae-Heung (Member)

1983 : BS degree in Electrical Engineering, Hanyang University.
1985 : MS degree in Electrical Engineering, Hanyang University.
1994 : PhD degree in Electrical Engineering, Hanyang University.
1989 ~ Present : Professor in Dept. of Computer Engineering, Hanbat University