# IV. PENGUJIAN

Pengujian dilakukan dengan membandingkan kedua algoritma pembuatan convex hull yang meliputi berbagai bentuk simple polygon baik yang sederhana maupun yang kompleks, dan kecepatan proses pembuatan convex hull. Pada pengujian kecepatan akan dibuat tabel-tabel hasil pengujian dengan melakukan percobaan beberapa kali untuk mendapatkan hasil yang akurat.

Perlu diketahui juga spesifikasi komputer yang dipakai dalam pengujian adalah :

- Prosesor komputer Intel Pentium MMX **233** Mhz
- Memory RAM **32** Megabyte
- **VGA** Card Matrox Millenium 2 Megabyte

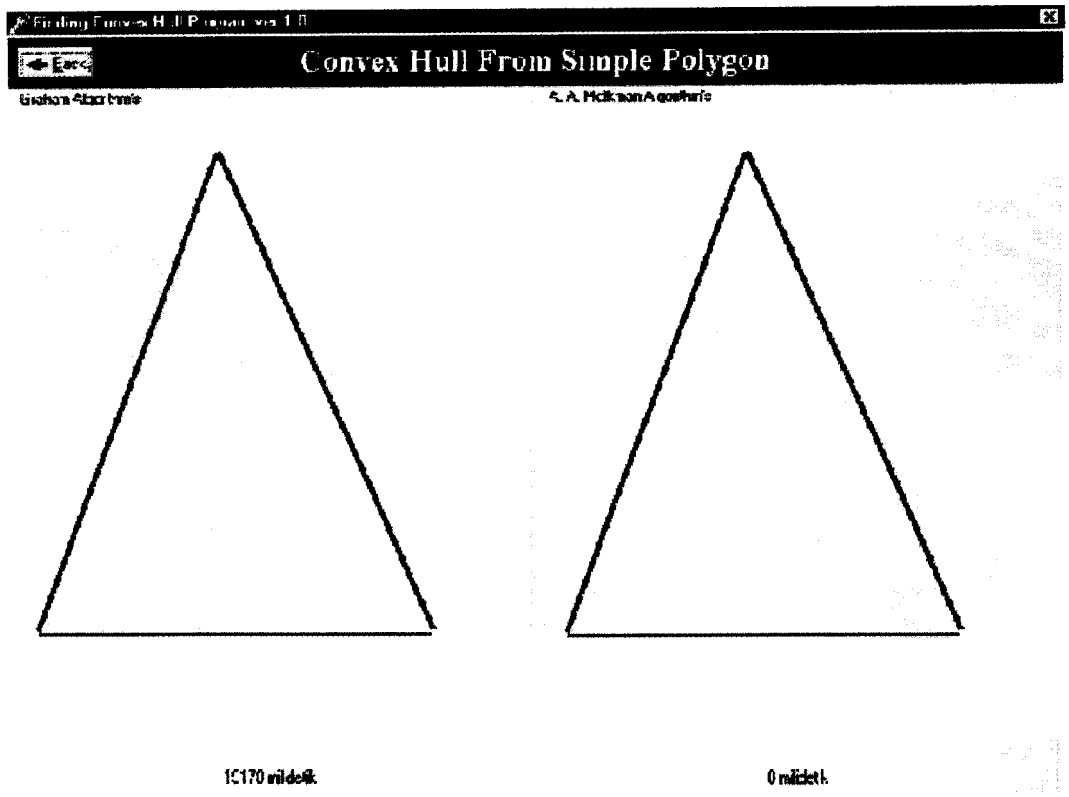
## 1. TAMPILAN PROGRAM

Dalam aplikasi program ini menggunakan button click untuk menjalankan procedure dimana ada empat macam button click yaitu, input, proses, detail dan keluar. Proses penggambaran convex hull dan perhitungan waktu eksekusi dimulai pada saat tombol button click proses yang terdapat pada aplikasi program ditekan. Adapun tampilan program untuk mencari convex hull dapat dilihat pada gambar 4.1. Dapat dilihat bahwa layar monitor dibagi menjadi dua bagian, yaitu gambar asli dan convex hull. Pada daerah gambar asli , bisa diinputkan titik-titik yang membentuk simple polygon. Button click input berguna untuk membersihnya layar dari gambar sebelumnya. Button click detail digunakan untuk menampilkan hasil penggambaran convex hull menurut algoritma Three Coins (Graham) dan algoritma Melkman. Button click exit digunakan untuk keluar dari program ini


Gambar 4.1.

Tampilan Utama

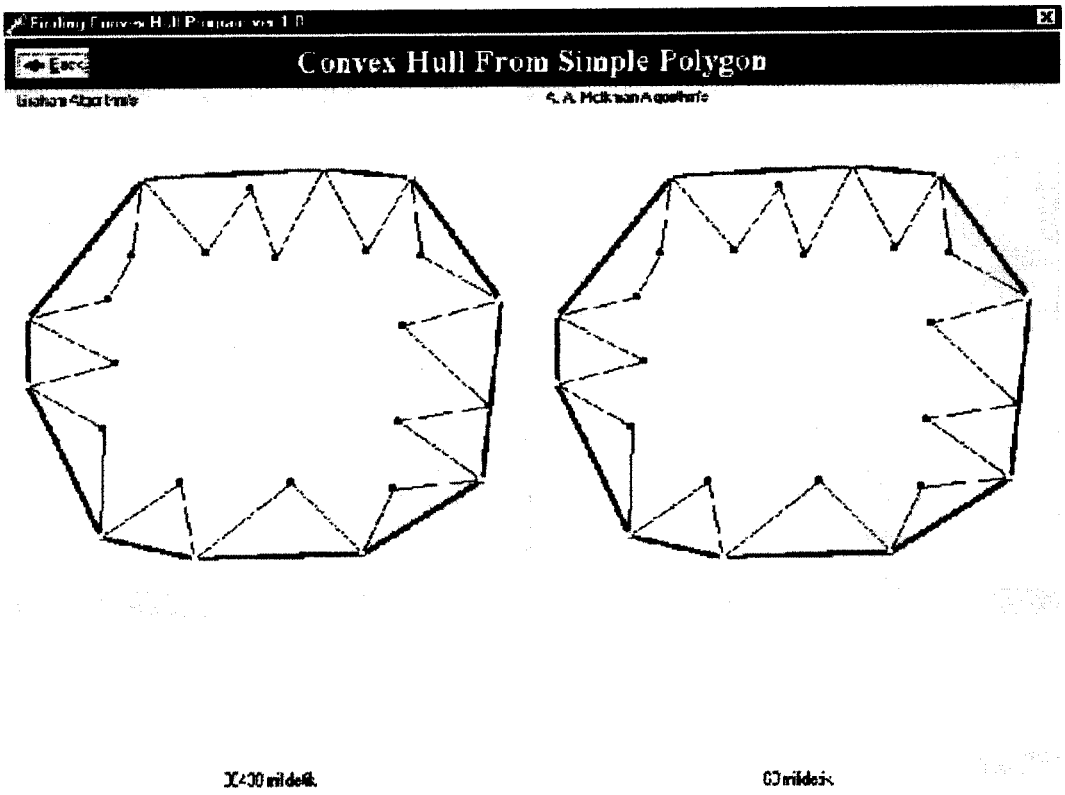
Pada bab pengujian ini akan dibahas analisa program berdasarkan waktu eksekusi dan akurasi convex hull yang dibentuk untuk berbagai macam bentuk simple polygon dari setiap algoritma yang digunakan.




Gambar 4.2

Convex Polygon dengan 3 vertices (segitiga)

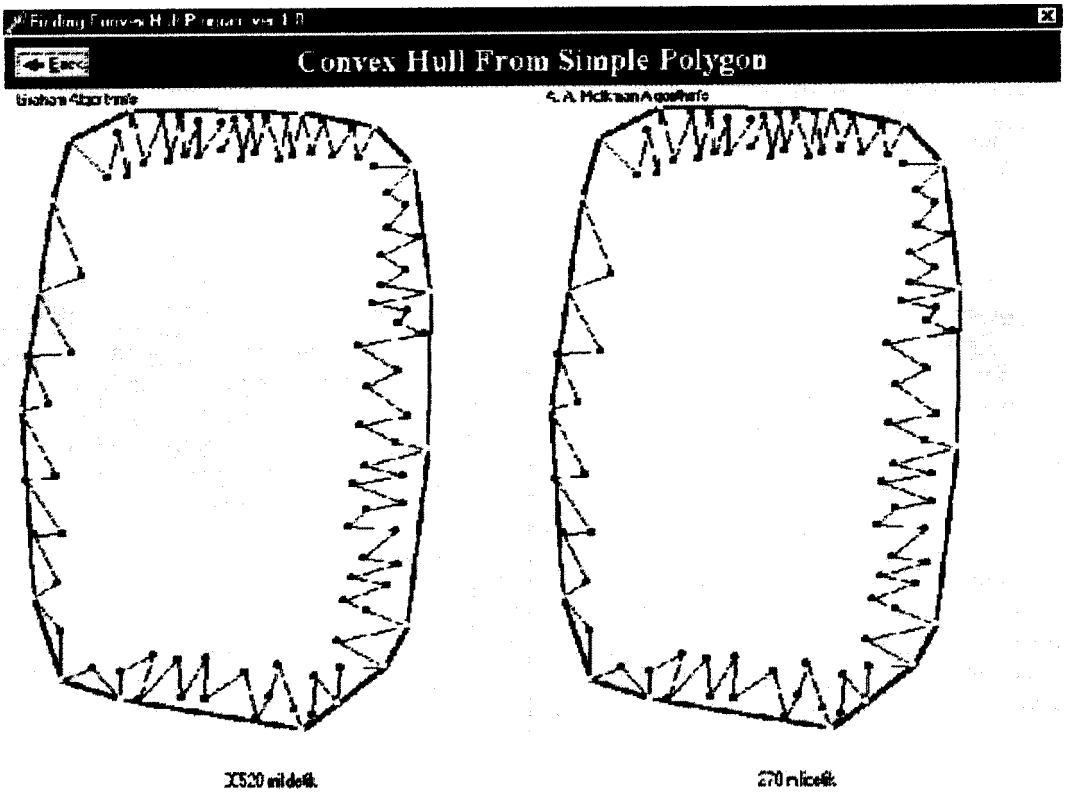
Keterangan pada gambar 4.2 dapat dilihat bahwa untuk menggambarkan convex hull untuk segitiga merupakan bentuk simple convex polygon yang paling sederhana terdiri dari 3 vertices. Dengan algoritma Graham waktu yang diperlukan 19,170 milidetik sedangkan kalau dengan Algoritma Melkman 0 milidetik karena relatif sangat cepat. Convex hull dari segitiga adalah segitiga itu sendiri, sehingga Melkman dengan sangat cepat langsung menggambarkannya.




Gambar 4.3

Simple Polygon dengan 25 vertices

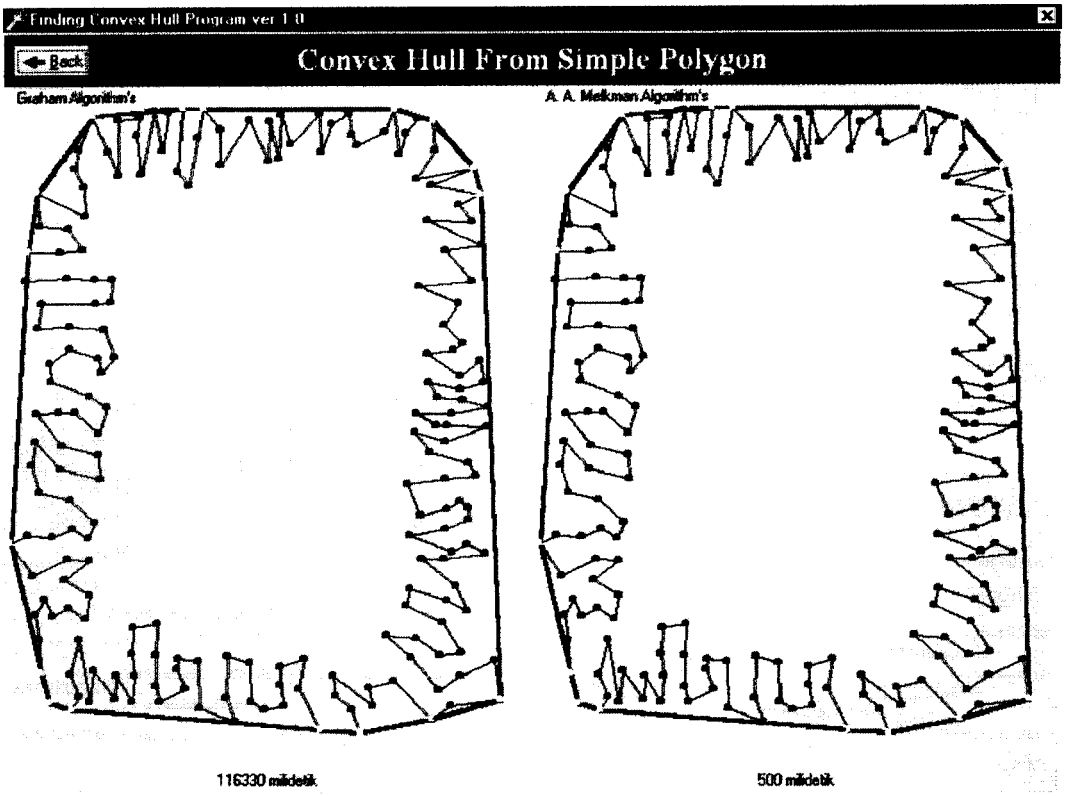
Keterangan : pada gambar 4.3 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 25 vertices. algoritma Graham memerlukan waktu 30430/1000 = 30,430 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 60/1000 = 0.06 milidetik




Gambar 4.4

Simple Polygon dengan 100 vertices

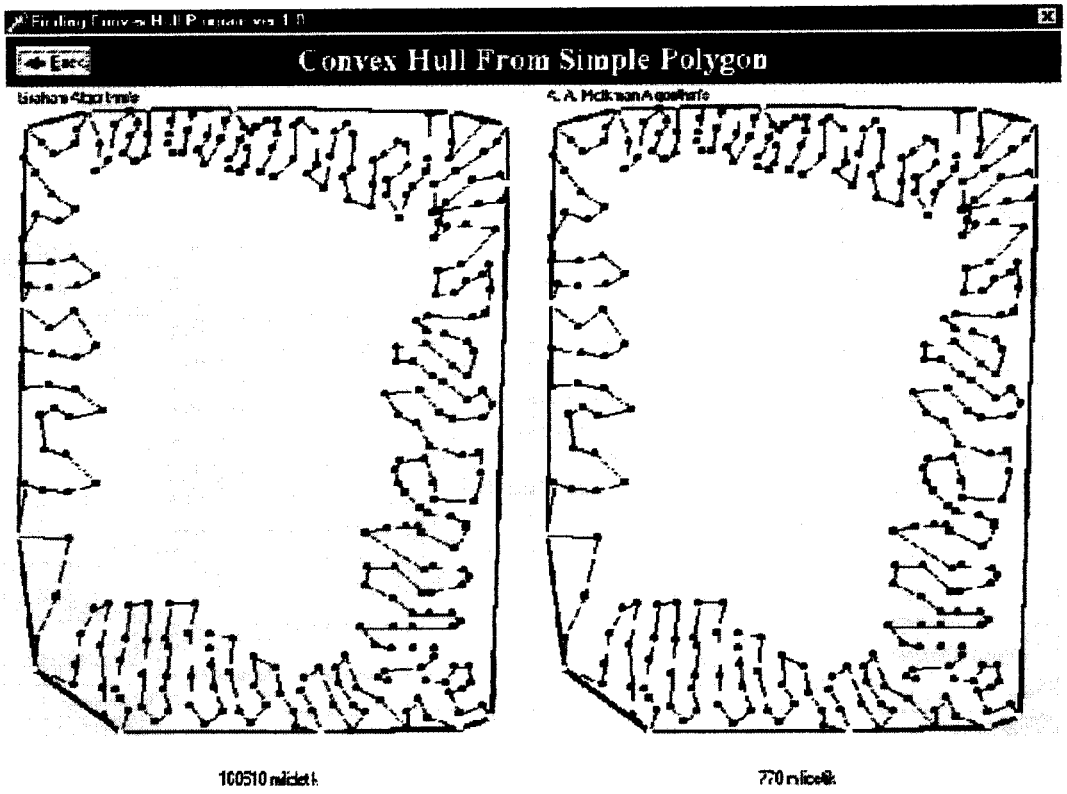
Keterangan : pada gambar 4.4. dapat dilihat bahwa untuk menggarnbarkan convex hull pada simple polygon dengan 100 vertices, algoritma Graham memerlukan waktu *66520/*1000 = 66,520 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 270/ 1000 = 0,27 milidetik. Makin banyak vertices maka waktu yang diperlukan untuk membuat convex hull menjadi lebih banyak juga.




Gambar 4.5.

Simple Polygon dengan 200 vertices

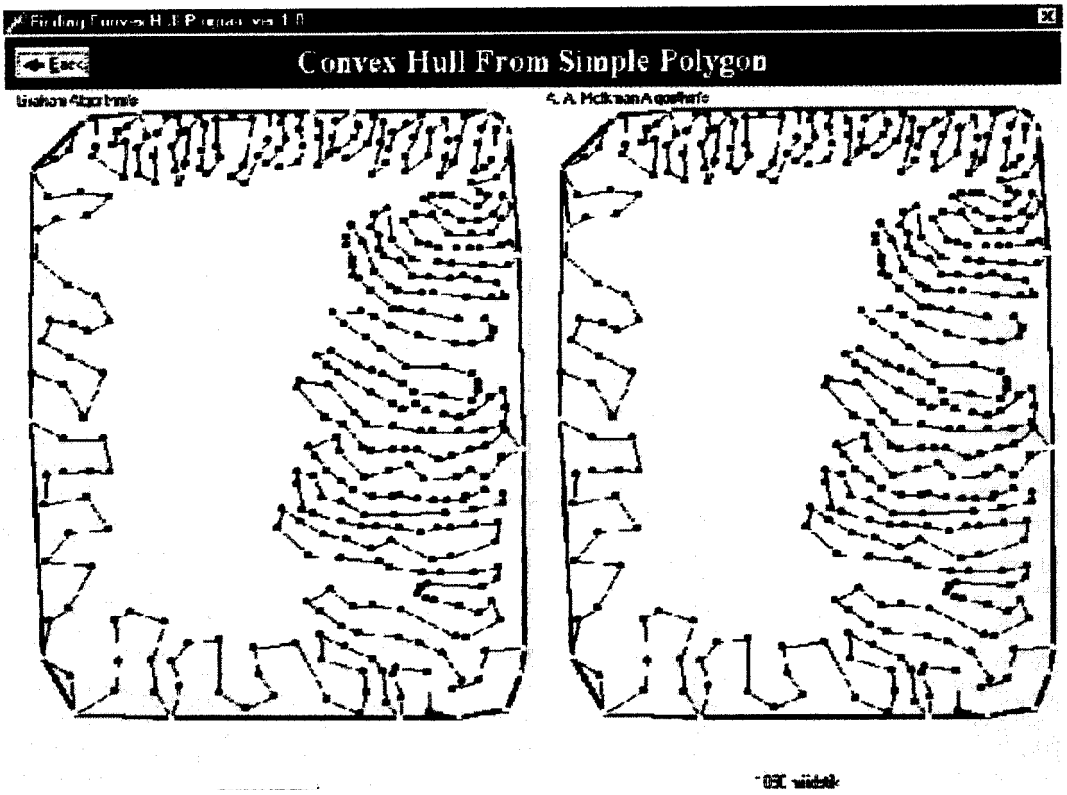
Keterangan : pada gambar 4.5 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 200 vertices, algoritma Graham memerlukan waktu 116330/1000 = 116,330 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 500/1000 = 0,5 milidetik. Makin banyak vertices maka waktu yang diperlukan untuk membuat convex hull menjadi lebih banyak juga




Gambar 4.6

Simple Polygon dengan 300 vertices

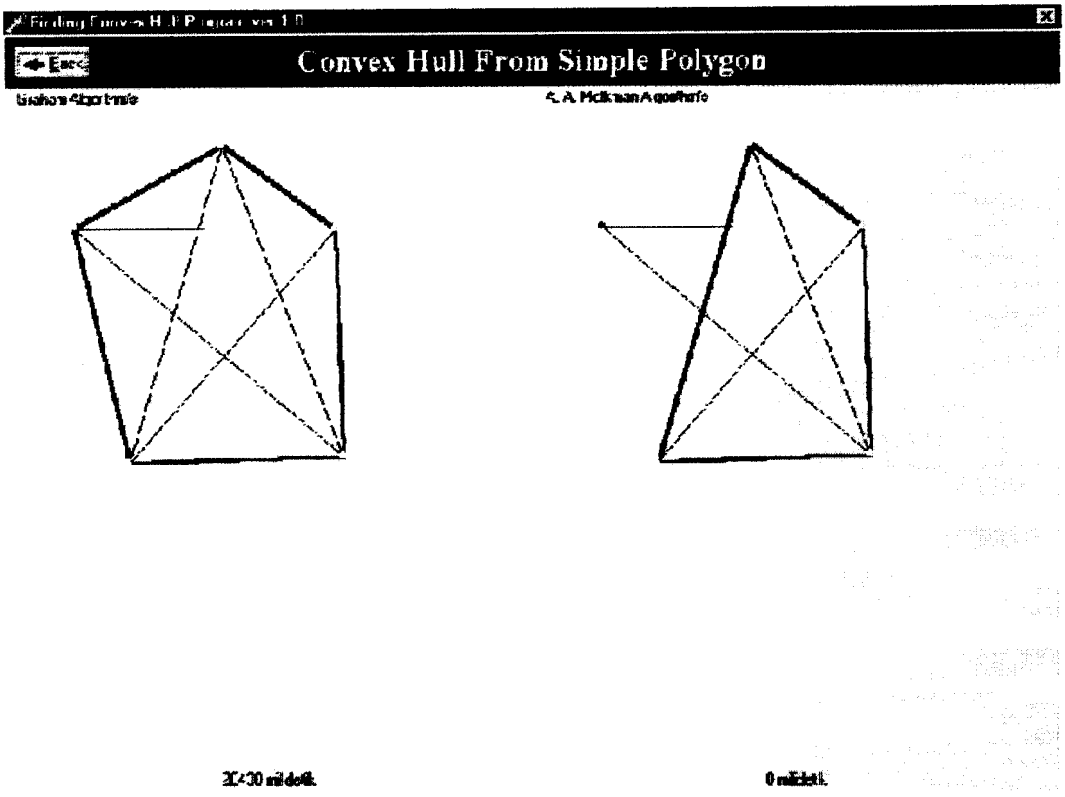
Keterangan : pada gambar 4.6 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 300 vertices, algoritma Graham memerlukan waktu I685 10/1000 = 168,51 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 770/1000 = 0,77 milidetik. Makin banyak vertices maka waktu yang diperlukan untuk membuat convex hull menjadi lebih banyak juga.




Gambar 4.7

Simple Polygon dengan 400 vertices

Keterangan : pada gambar 4.7 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 400 vertices, algoritma Graham memerlukan waktu 239360/1000 = 239,36 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 1050/1000 = 1,05 milidetik. Makin banyak vertices maka waktu yang diperlukan untuk membuat convex hull menjadi lebih banyak juga.

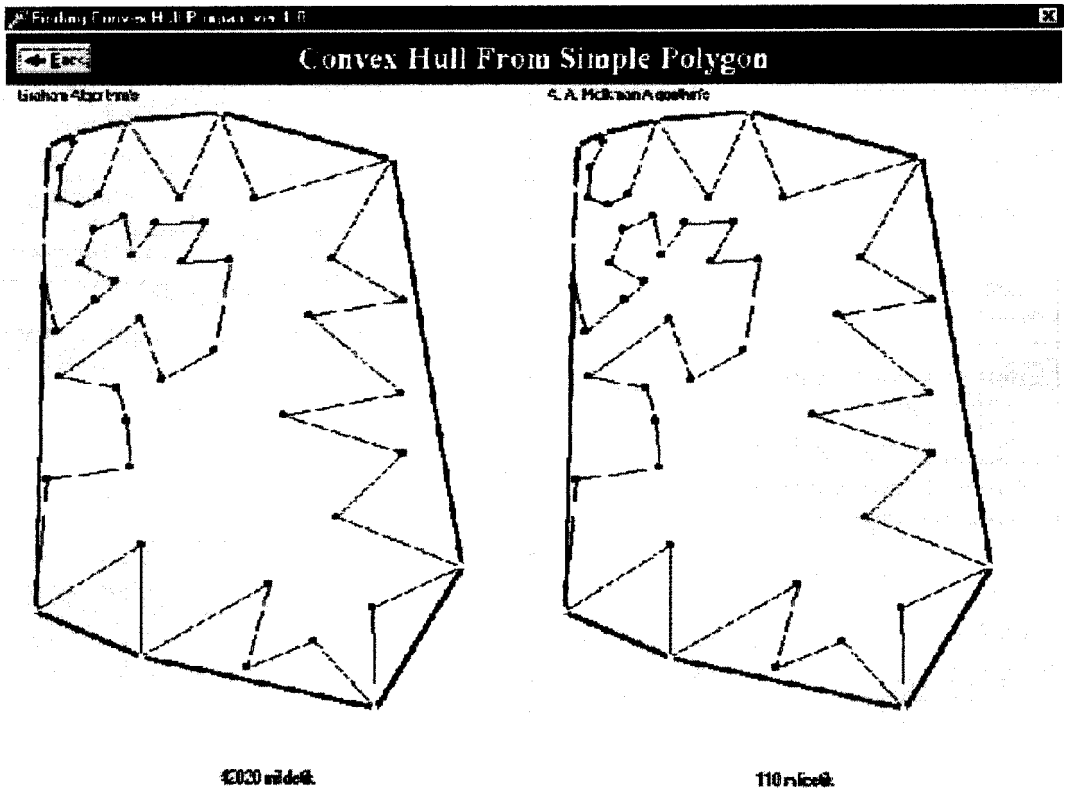



Gambar 4.8.

Bukan Simple Polygon

Keterangan : Selama input yang dimasukkan adalah simple polygon biarpun bentuknya kompleks dan banyak titik vertexnya, algoritma Melkman masih bisa mengatasinya, tapi apabila sudah bukan simple polygon, maka hanya algoritma Graham yang bisa membuat convex hullnya.

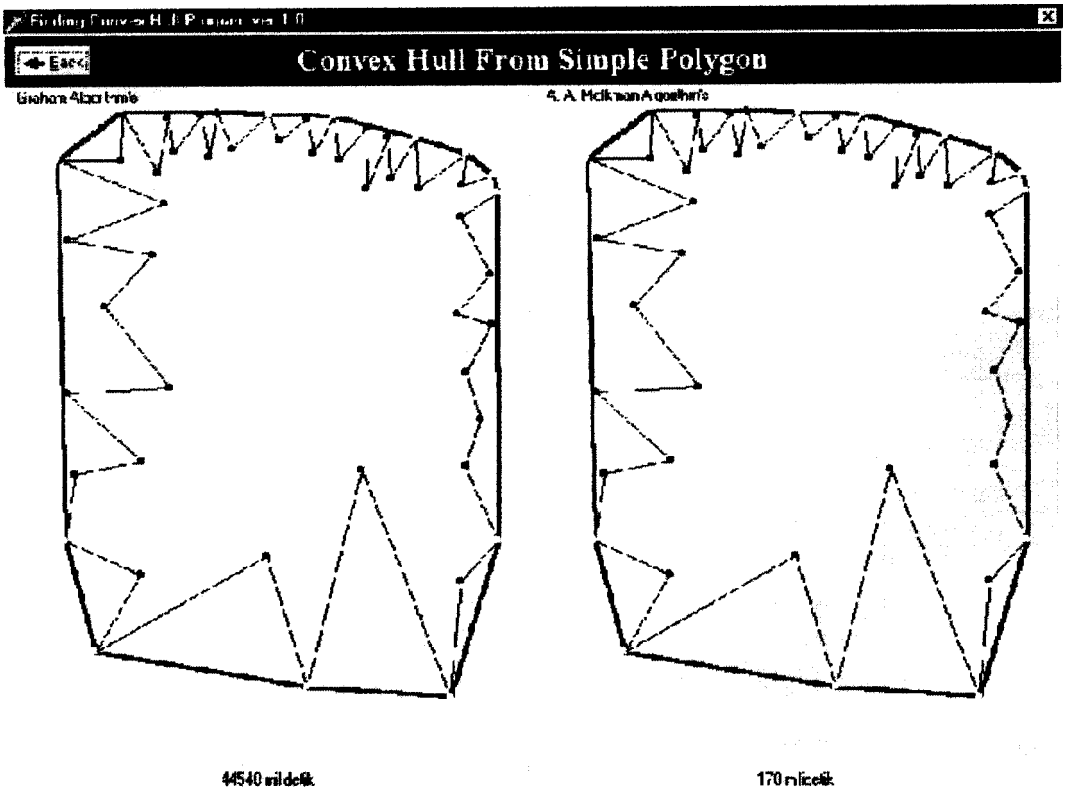
Selanjutnya diadakan percobaan dengan memasukkan bentuk simple polygon yang mempunyai 50 vertices dengan berbagai macam bentuk yang memungkinkan, kemudian dicari convex hullnya dengan kedua macam algoritma dan diulang sebanyak 25 kali, dapat dilihat pada gambar 4.9 sampai dengan gambar 4.33.




Gambar 4.9

Simple Polygon1 dengan 50 vertices

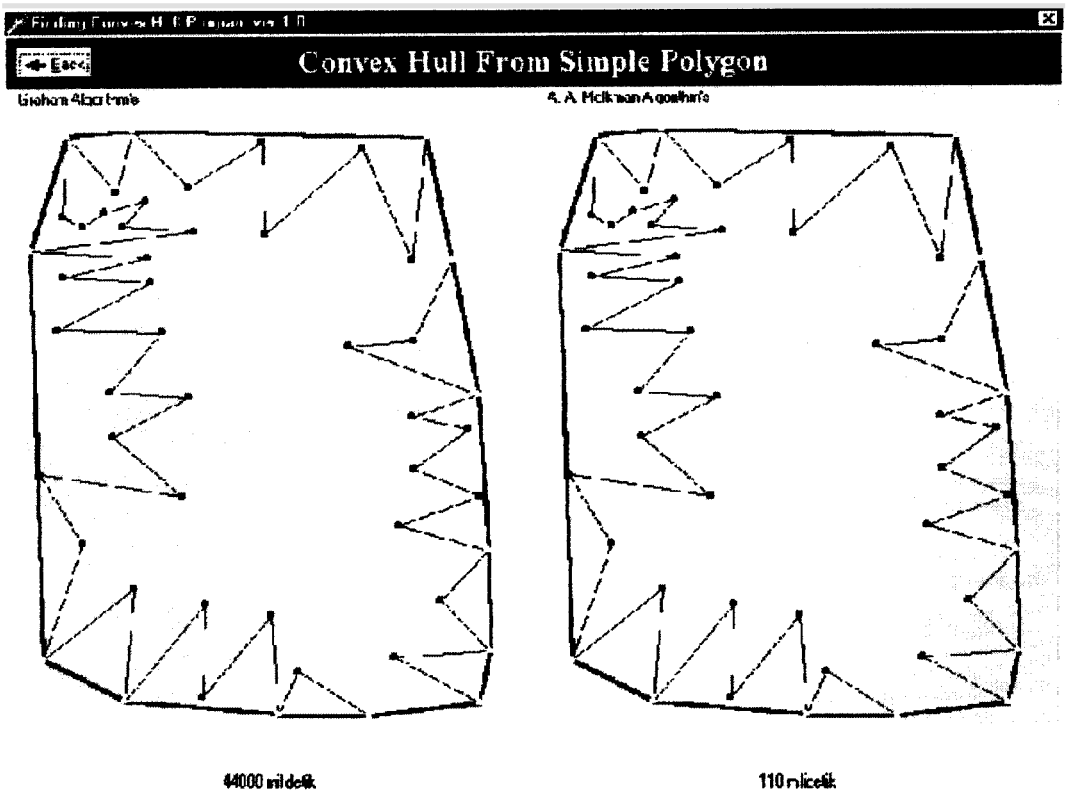
Keterangan : pada gambar 4.9 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 42020/1000 = 44,.020 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 110/1000 = 0,11 milidetik.




Gambar 4.10

Simple Polygon2 dengan 50 vertices

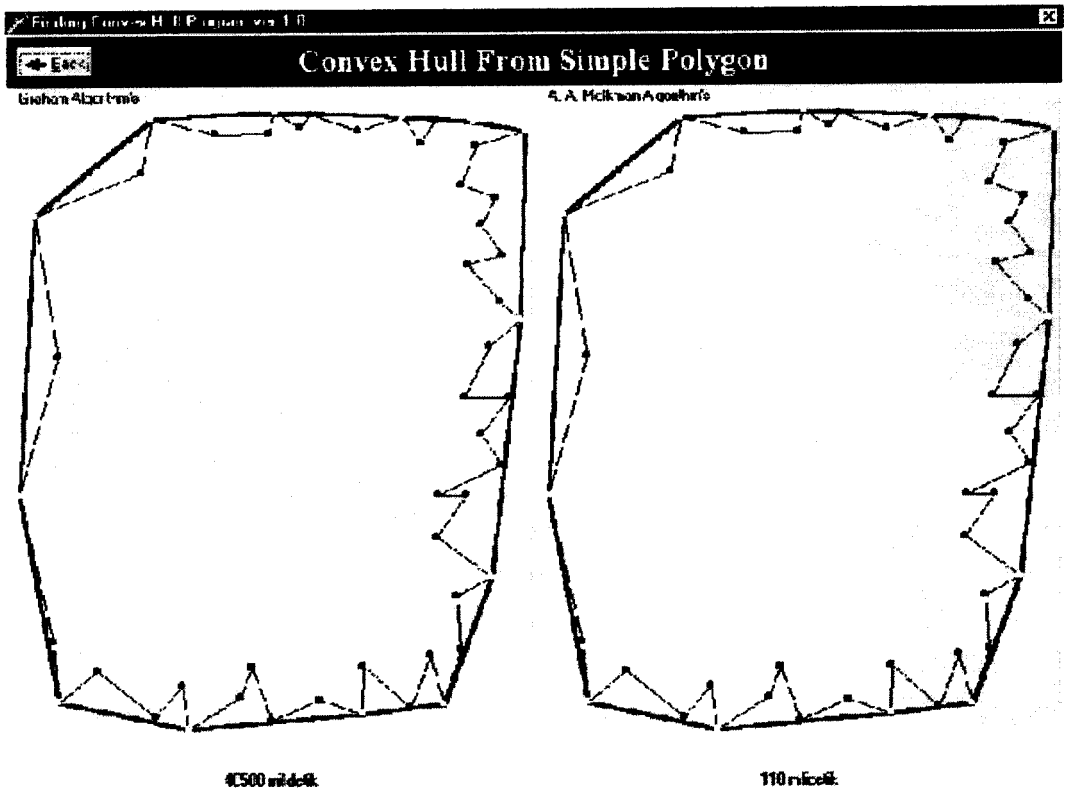
Keterangan : pada gambar 4.10 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu $44540/1000 = 4434$ milidetik sedangkan algoritma Melkman hanya memerlukan waktu $170/1000 = 0{,}17$ milidetik.




Gambar 4.11

Simple Polygon3 dengan 50 vertices

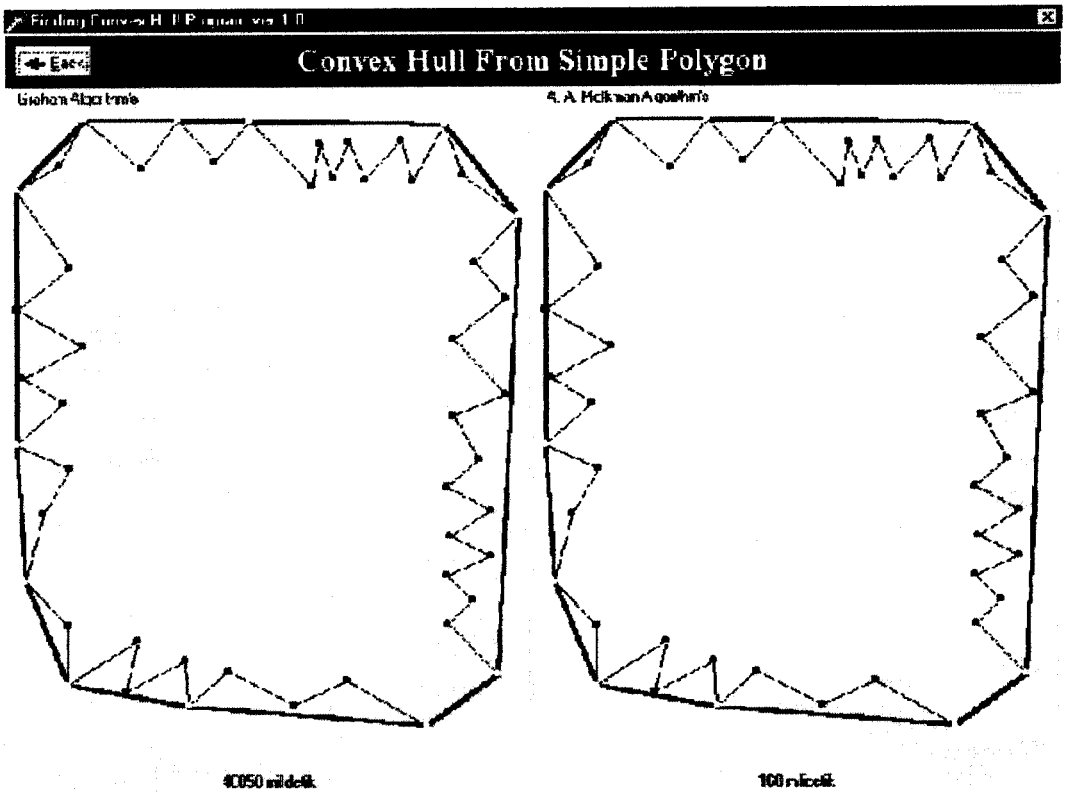
Keterangan : pada gambar 4.11 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 44880/1000 = 44,88 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 110/1000 = 0,11 milidetik




Gambar 4.12

Simple Polygon4 dengan 50 vertices

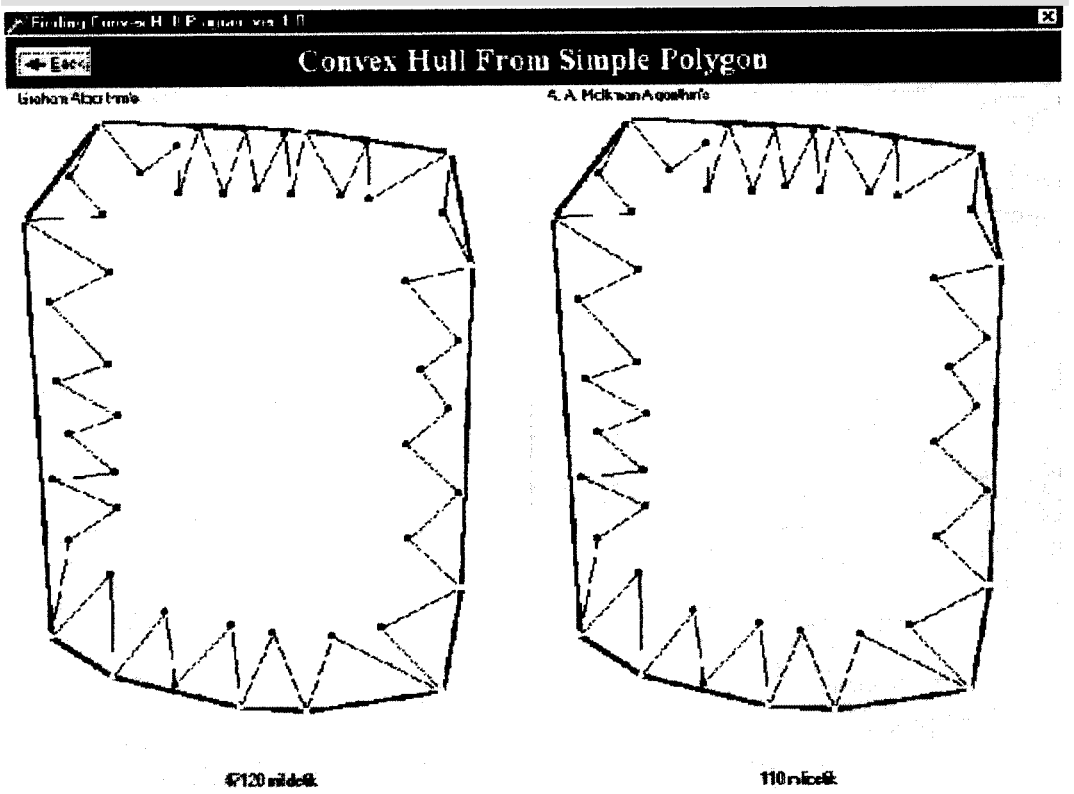
Keterangan : pada gambar 4.12 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 48500/1000 = 48,5 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 110/1000 = 0,11 milidetik.




Gambar 4.13

Simple Polygon5 dengan 50 vertices

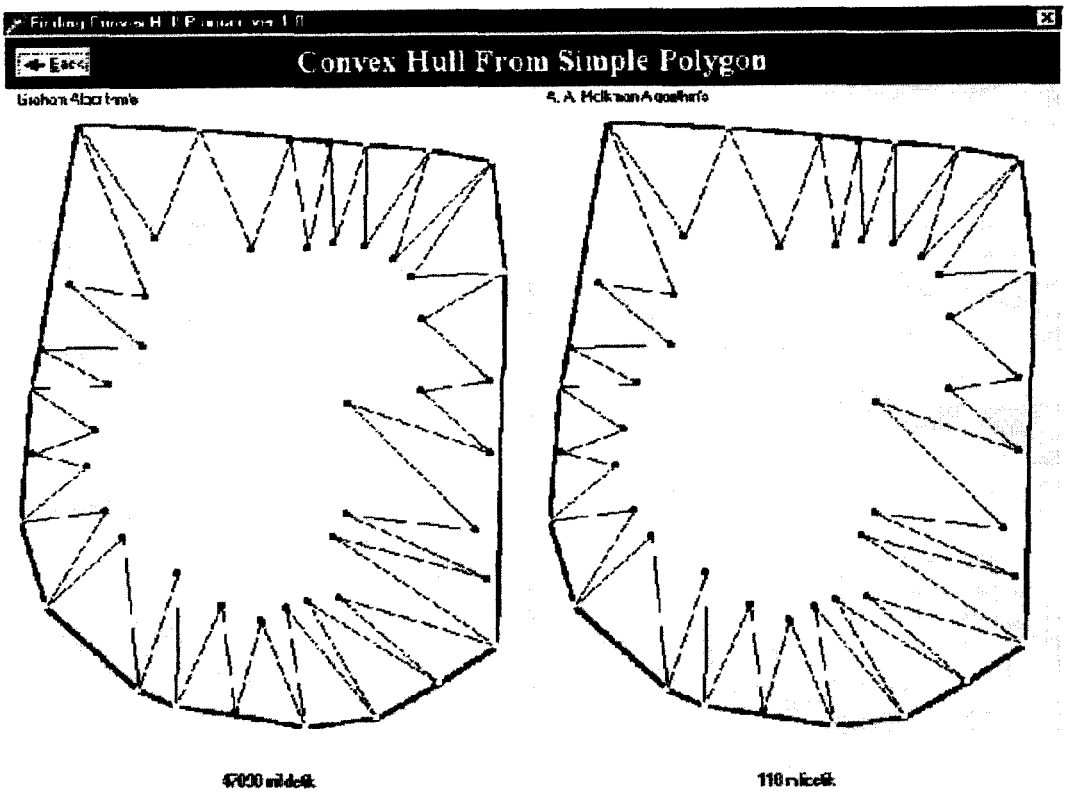
Keterangan : pada gambar 4.13 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu $46850/1000 = 46{,}85$ milidetik sedangkan algoritma Melkman hanya memerlukan waktu $160/I000 = 0{,}16$ milidetik.




Gambar 4.I4

Simple Polygon6 dengan 50 vertices

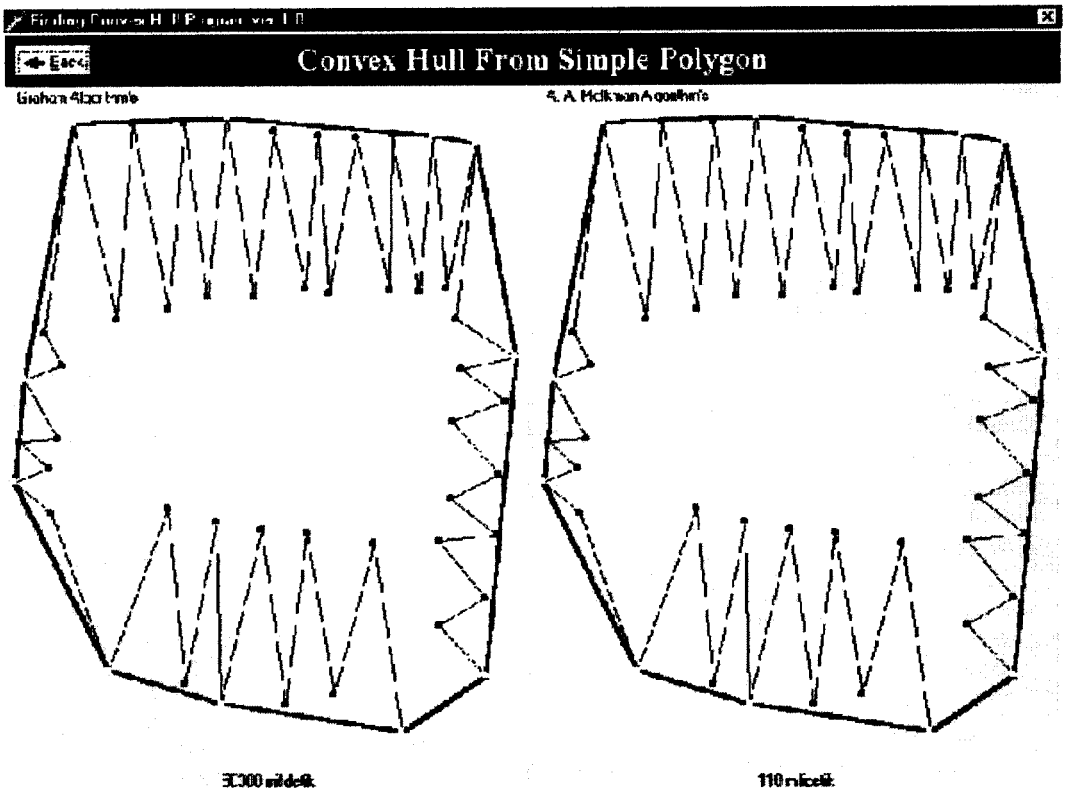
Keterangan : pada gambar 4.14 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 47120/1000 = 47,12 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 110/1000 = 0,11 milidetik.




Gambar 4. 15

Simple Polygon7 dengan 50 vertices

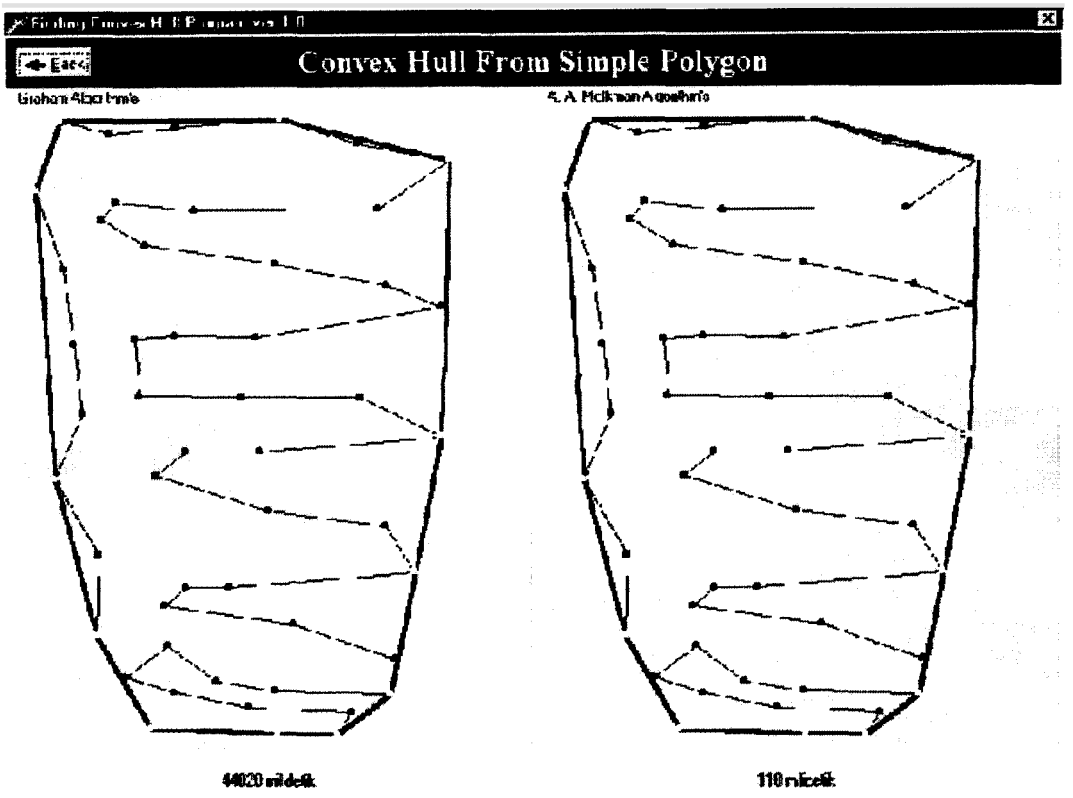
Keterangan : pada gambar 4.15 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 47890/1000 = 47,89 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 110/ 1000= 0,11 milidetik.




Gambar 4. I6

Simple Polygon8 dengan 50 vertices

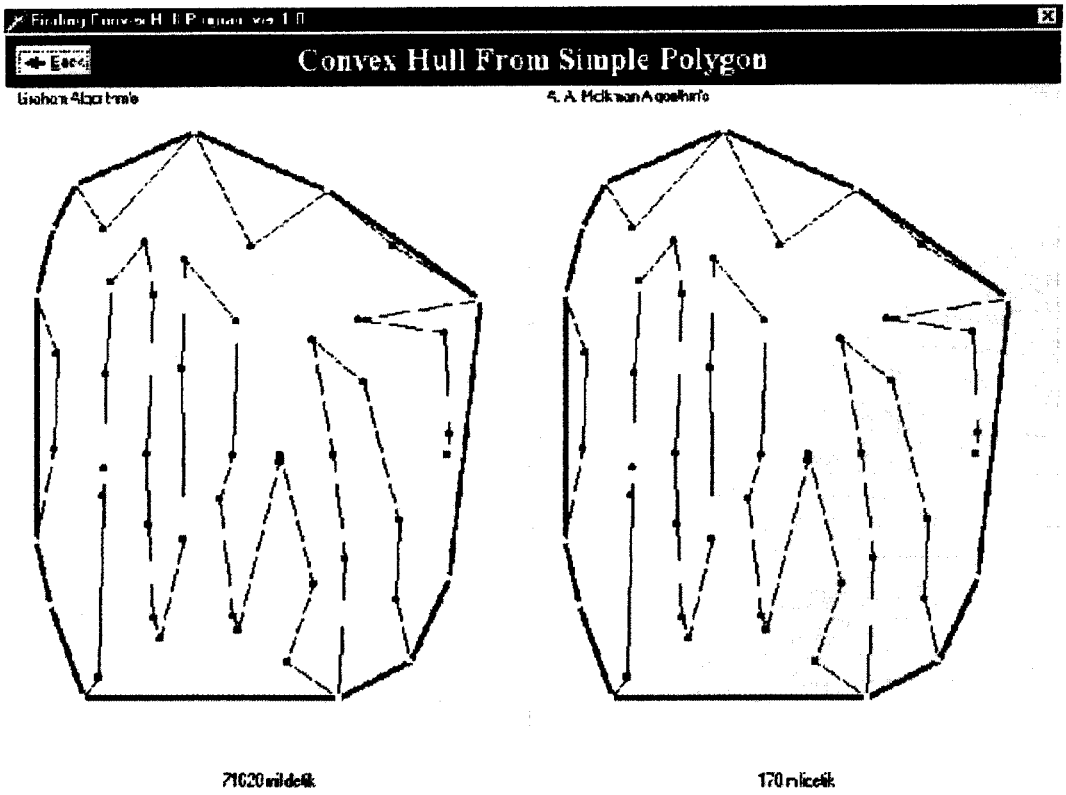
Keterangan : pada gambar 4.16 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 56300/1000 = *56,3* milidetik sedangkan algoritma Melkman hanya memerlukan waktu 110/1000 = 0,l 1 milidetik




Gambar 4 17

Simple Polygon9 dengan 50 vertices

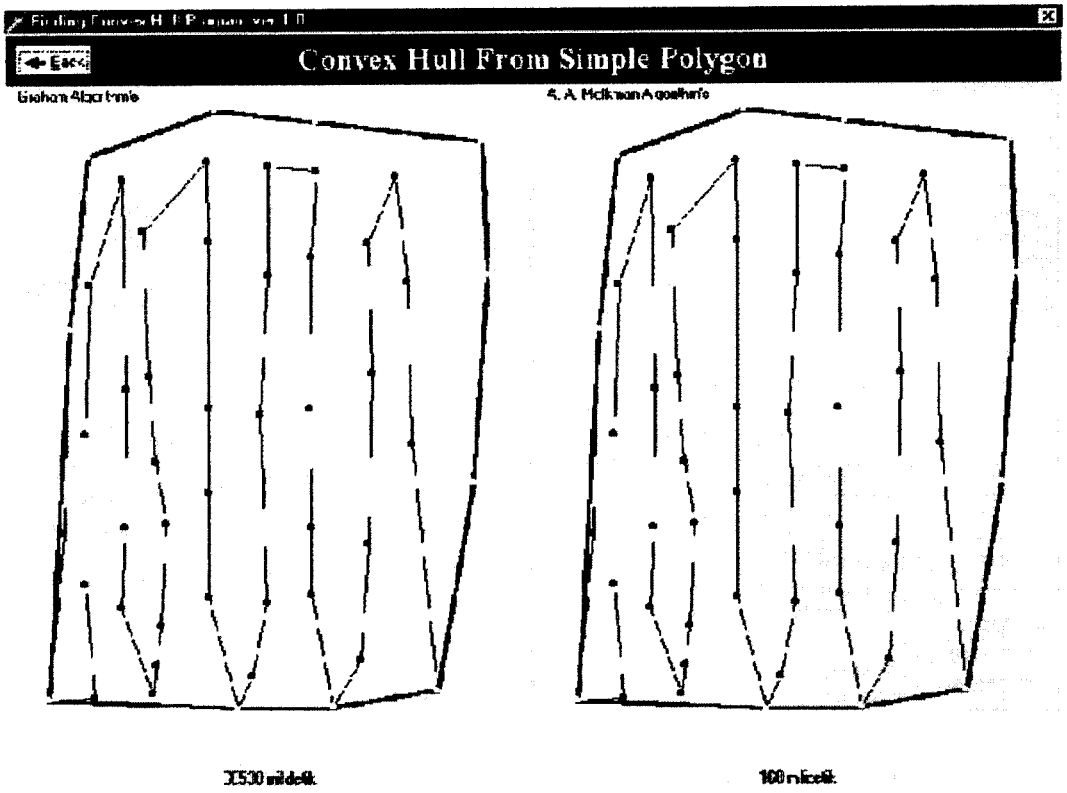
Keterangan pada gambar 4.17 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 44820/ 1000 = 44,82 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 110/1000 = 0,11 milidetik.




Gambar 4.18

Simple Polygon10 dengan 50 vertices

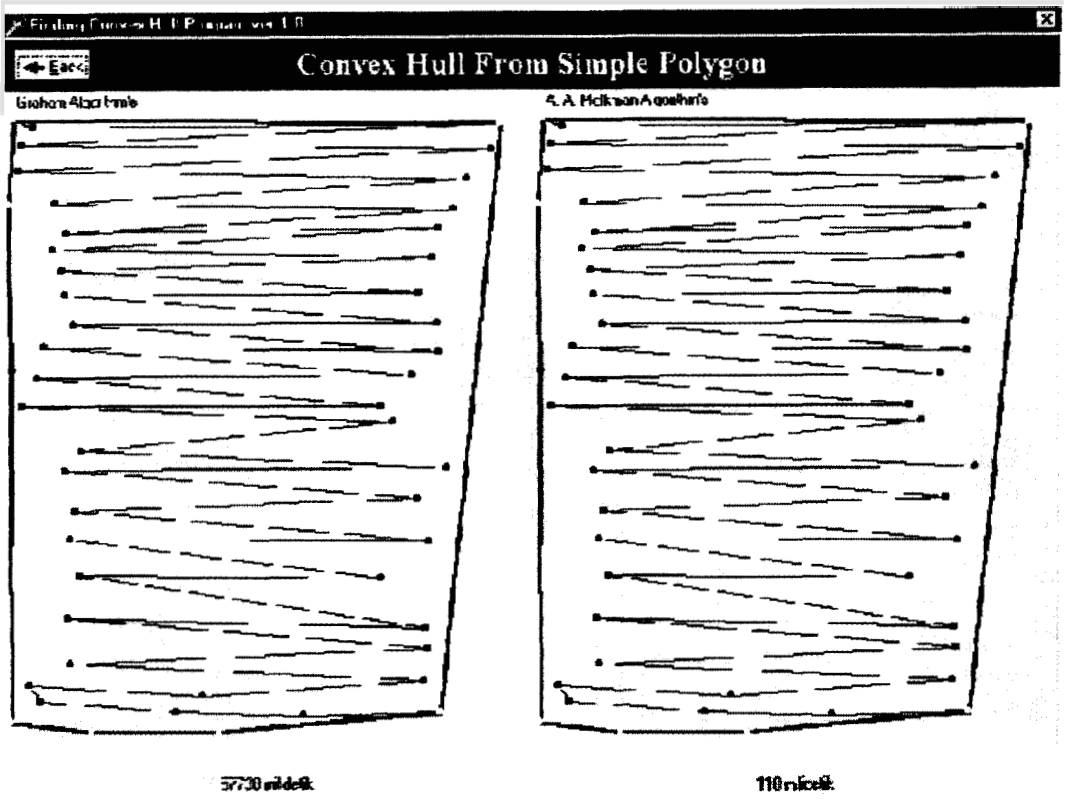
Keterangan : pada gambar 4.18 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 71620/1000 = 71,62 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 170/1000 = 0,17 milidetik.




Gambar 4. I9

Simple Polygon 11 dengan 50 vertices

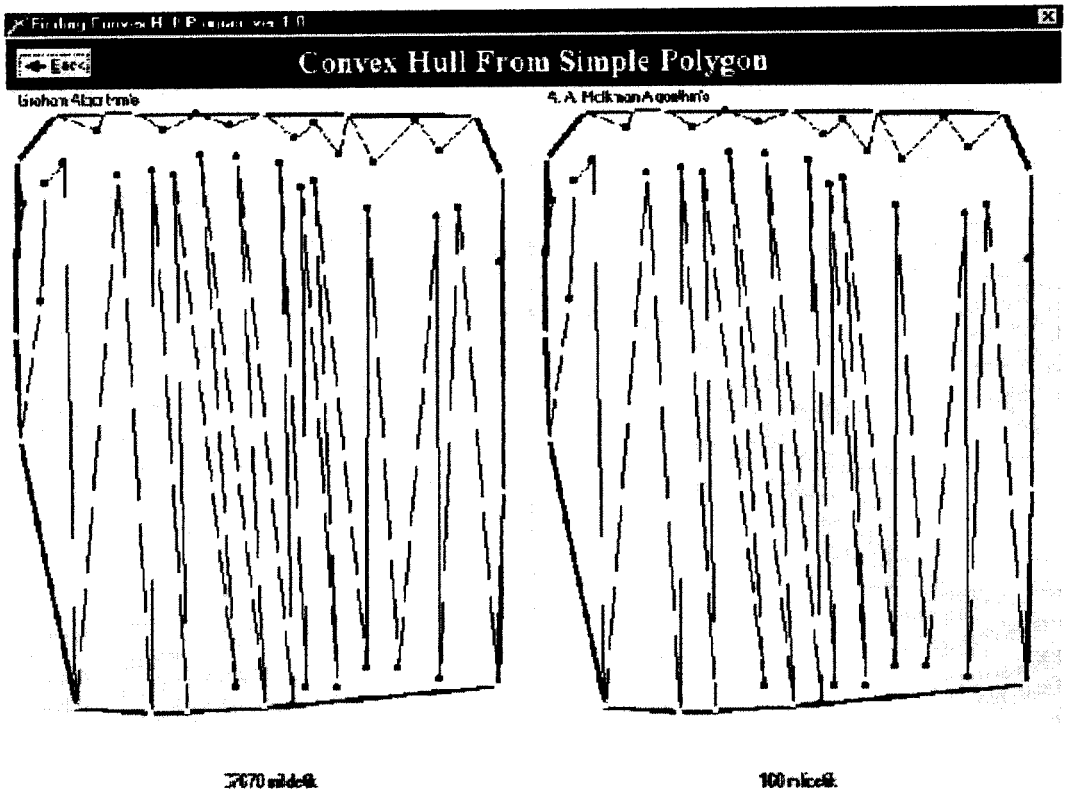
Keterangan : pada gambar 4.19 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu *60530/*1000 = *60,53* milidetik sedangkan algoritma Melkman hanya memerlukan waktu 160/1000 = 0,16 milidetik.




Gambar 4.20

Simple Polygon12 dengan 50 vertices

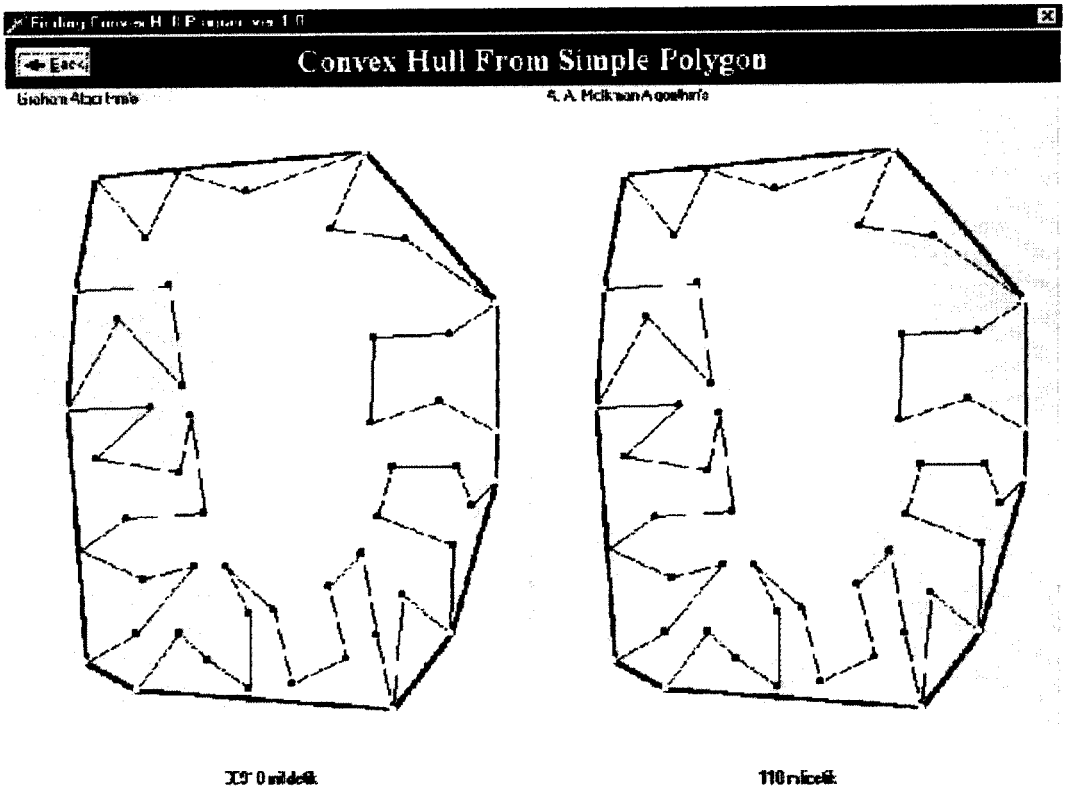
Keterangan : pada gambar 4.20 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 57730/1000 = 47,89 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 110/1000 = 0,11 milidetik.




Gambar 4.21

Simple Polygon13 dengan 50 vertices

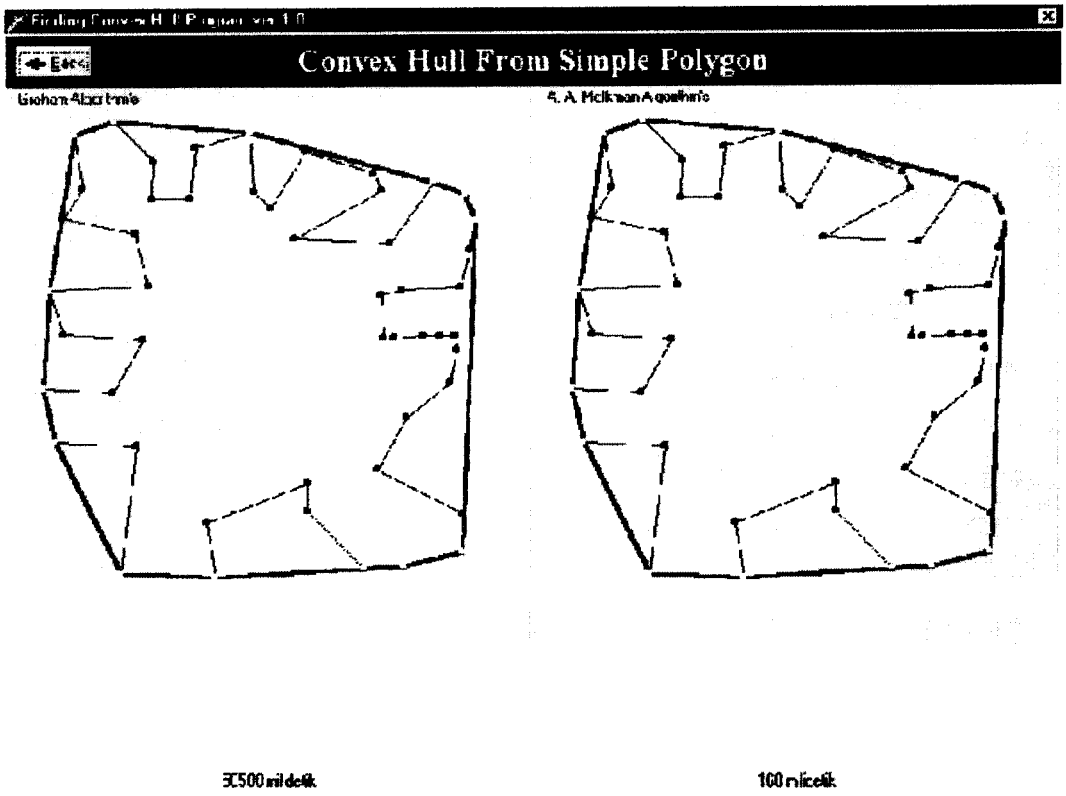
Keterangan : pada gambar 4.21 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 67670/1000 = 67,67 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 160/1000 = 0,16 milidetik




Gambar 4 22

Simple Polygon 14 dengan 50 vertices

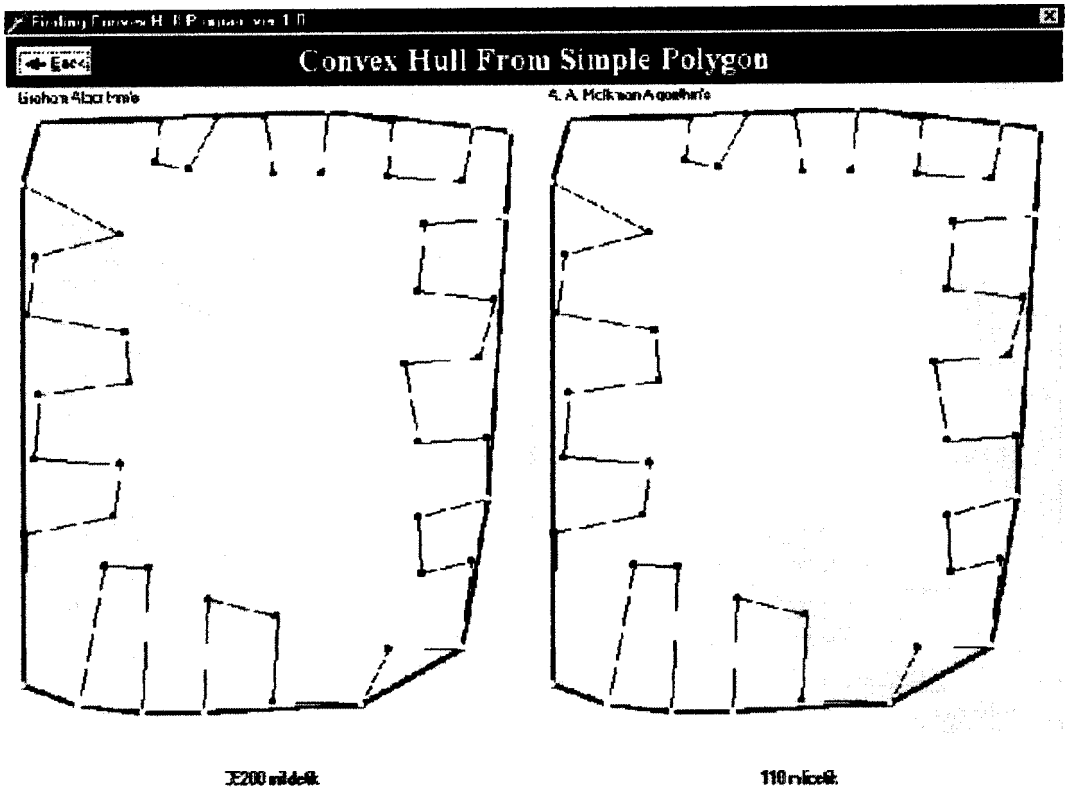
Keterangan : pada gambar 4.22 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 60910/1000 = 60,91 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 110/1000 = 0,11 milidetik.




Gambar 4.23

Simple Polygon15 dengan 50 vertices

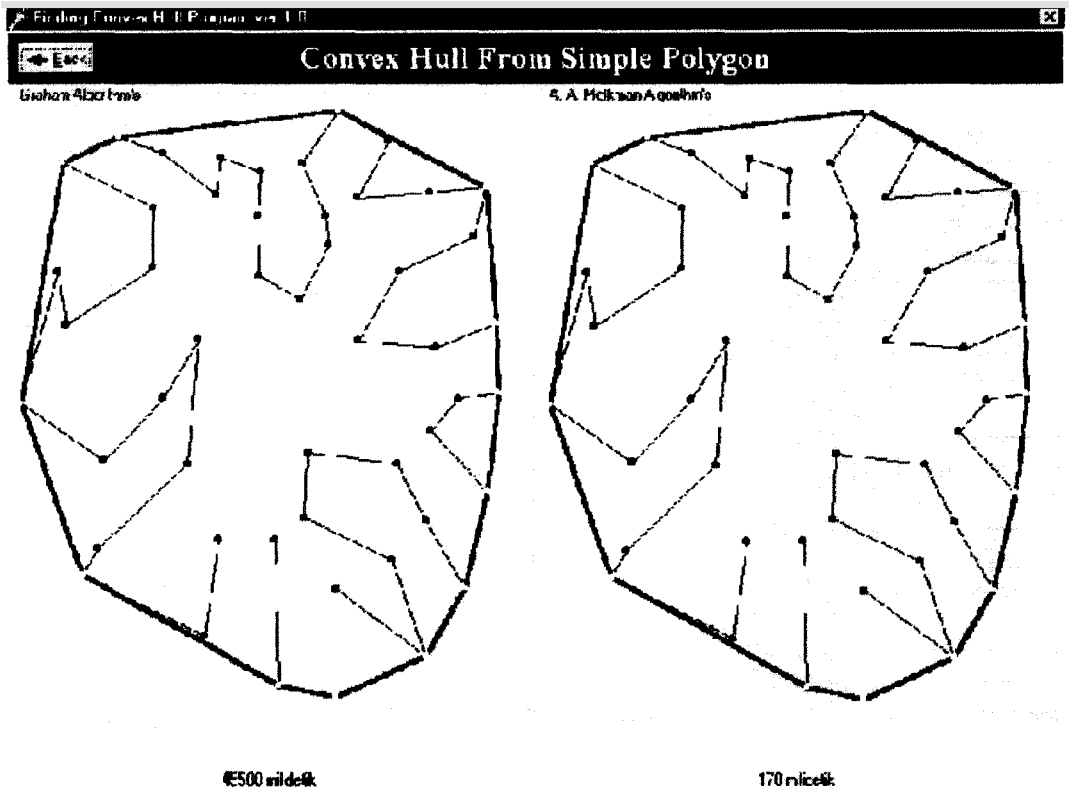
Keterangan pada gambar 4.23 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 58500/1000 = ***58,5*** milidetik sedangkan algoritma Melkman hanya memerlukan waktu 160/1000 = 0,16 milidetik.




Gambar 4.24

Simple Polygon 16dengan 50 vertices

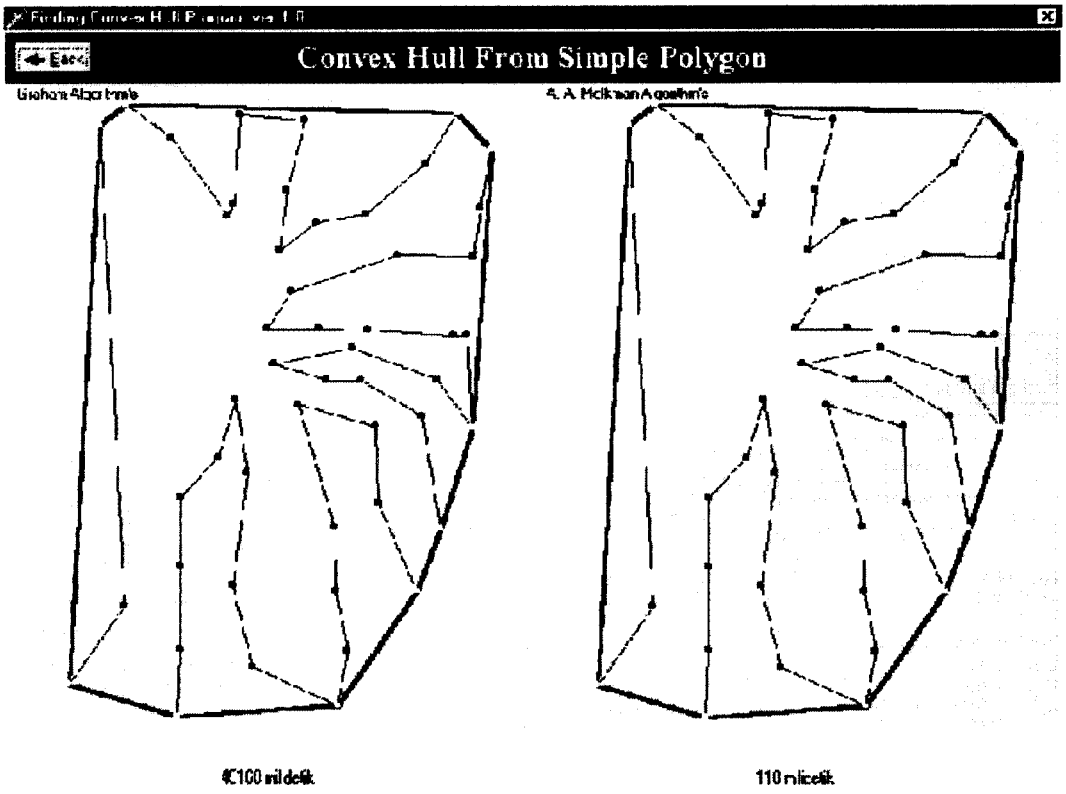
Keterangan : pada gambar 4.24 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 65200/1000 = 65,2 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 110/1000 = 0,11 milidetik.




Gambar 4.25

Simple Polygon 17 dengan 50 vertices

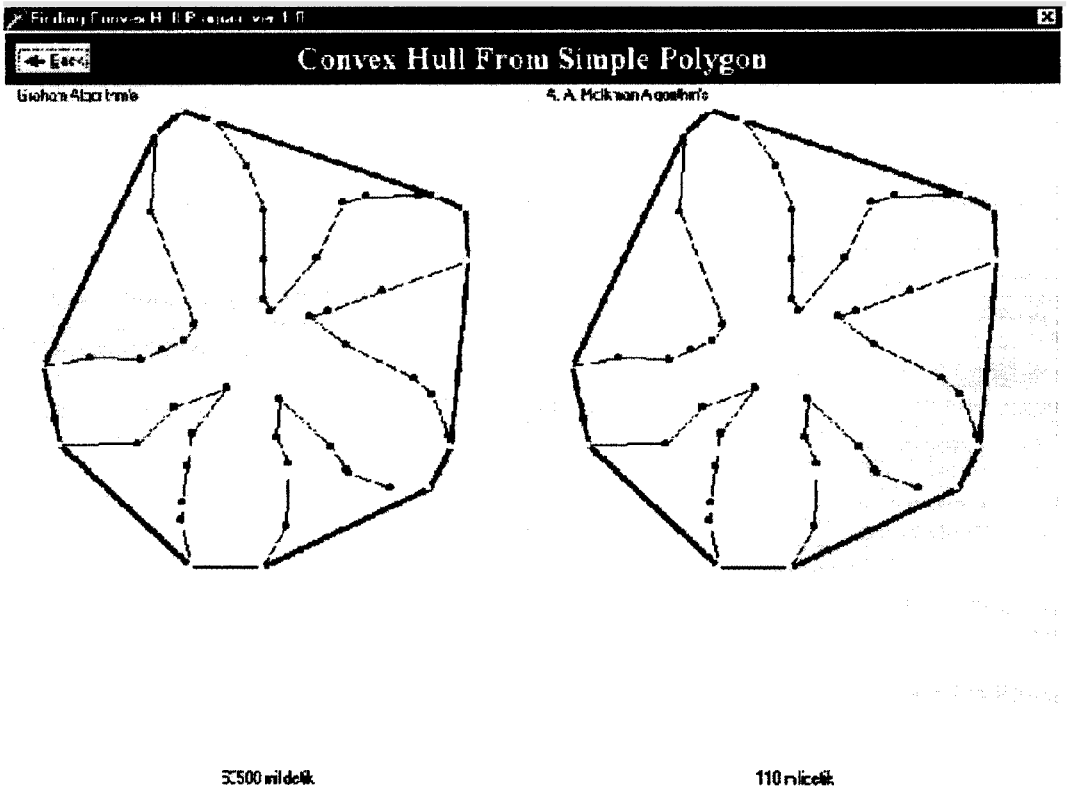
Keterangan : pada gambar 4.25 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 45580/1000 = 45,58 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 170/1000 = 0,17 milidetik




Gambar 4.26

Simple Polygon1**8** dengan 50 vertices

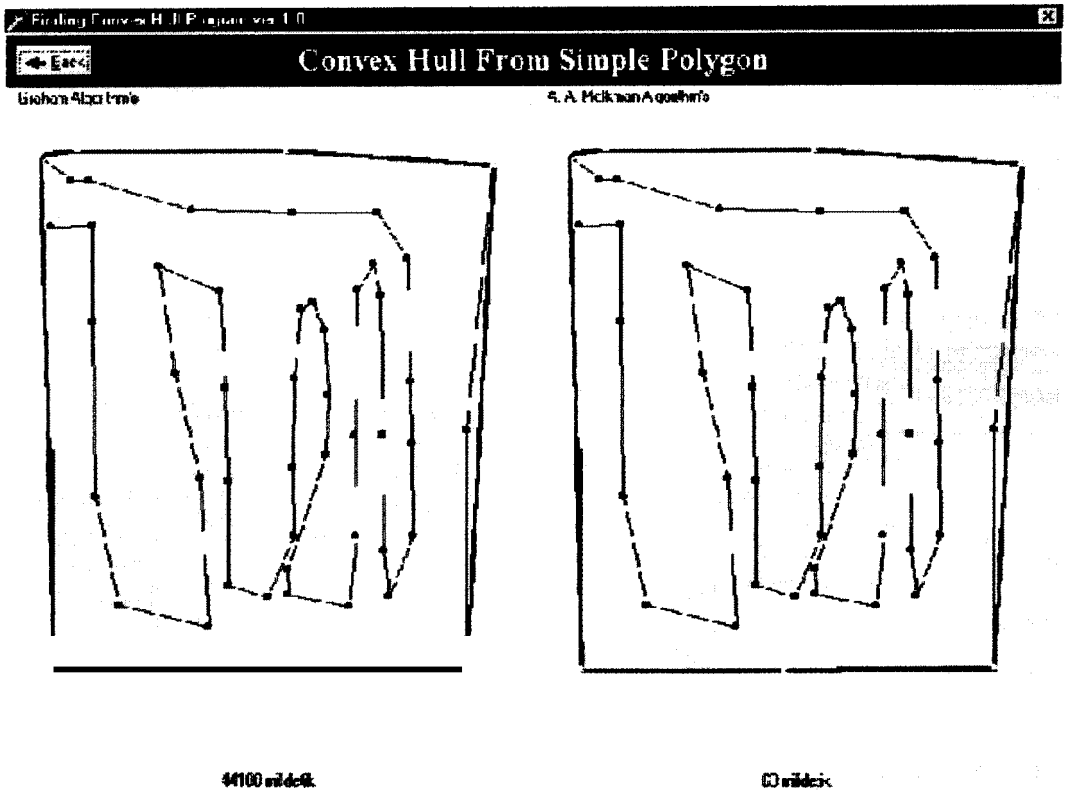
Keterangan . pada gambar 4.26 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 49 160/1000 = 49,16 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 1 10/1000 = 0,11 milidetik




Gambar 4.27

Simple Polygon19 dengan 50 vertices

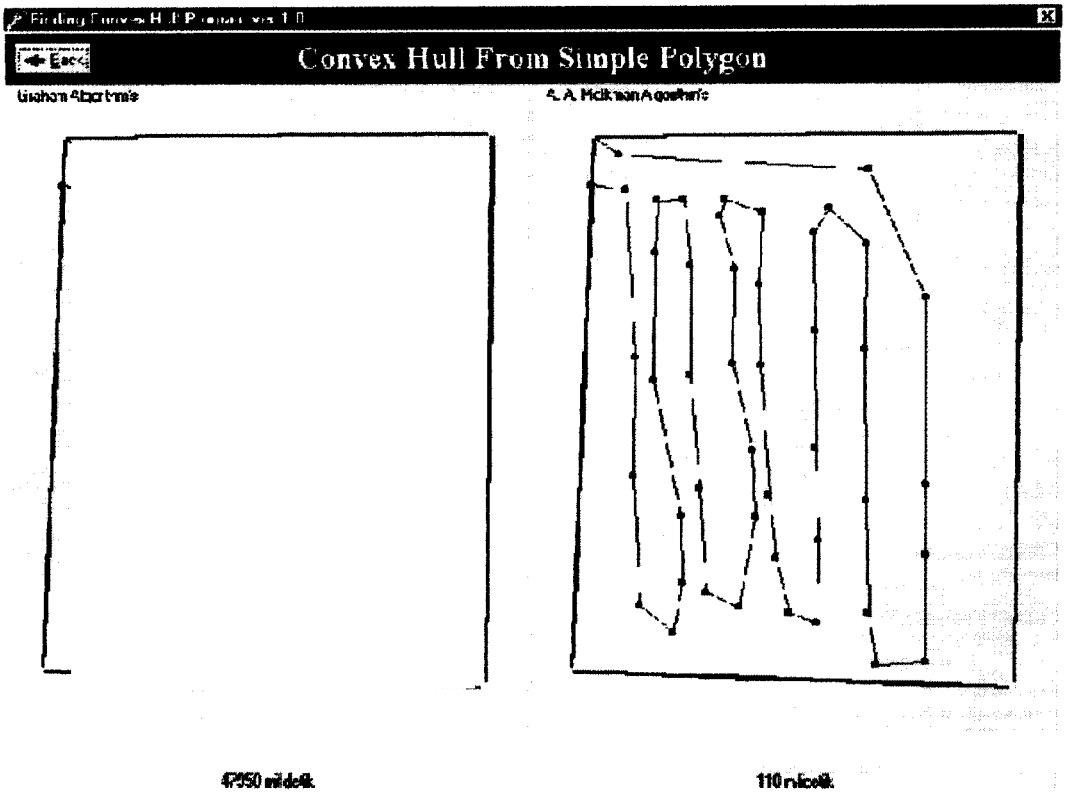
Keterangan : pada gambar 4.27 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu $53500/1000 = 53{,}5$ milidetik sedangkan algoritma Melkman hanya memerlukan waktu $110/1000 = 0{,}11$ milidetik.




Gambar 4.28

Simple Polygon20 dengan 50 vertices

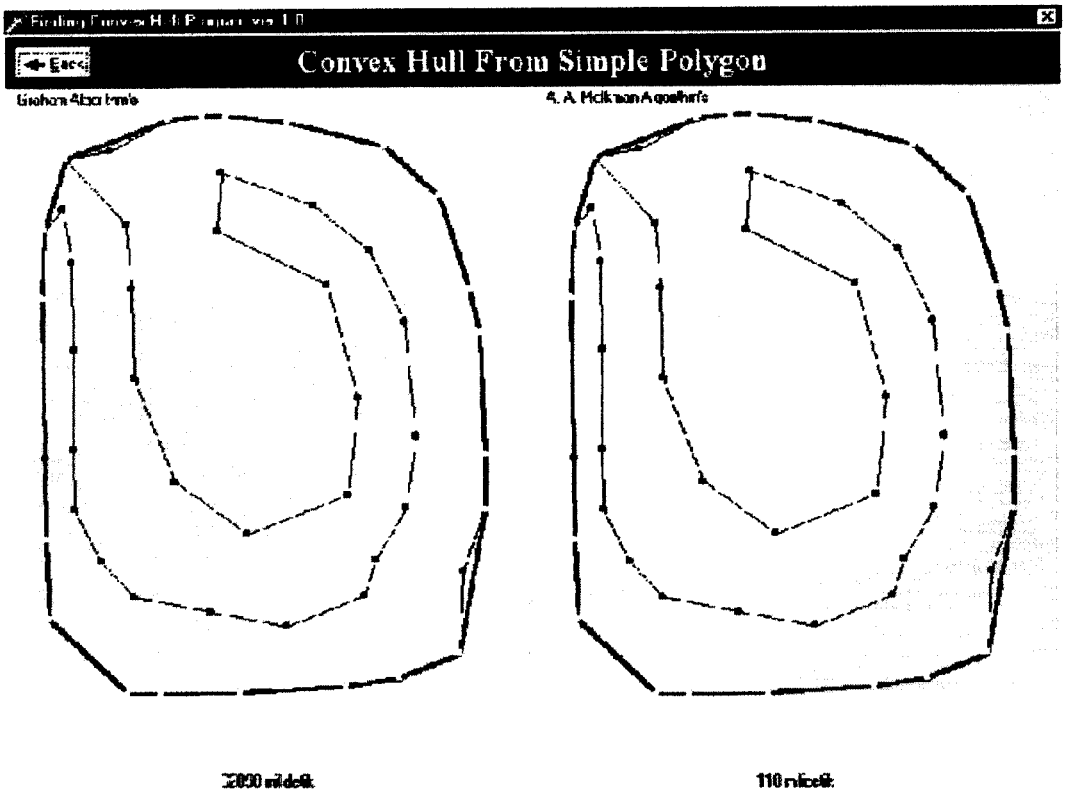
Keterangan : pada gambar **4.28** dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 44100/1000 = 44,1 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 60/1000 = *0,06* milidetik. Untuk bentuk yang ini terlihat jelas algoritma Melkman lebih cepat dibanding bentuk yang lain.




Gambar 4.29

Simple Polygon2 1 dengan 50 vertices

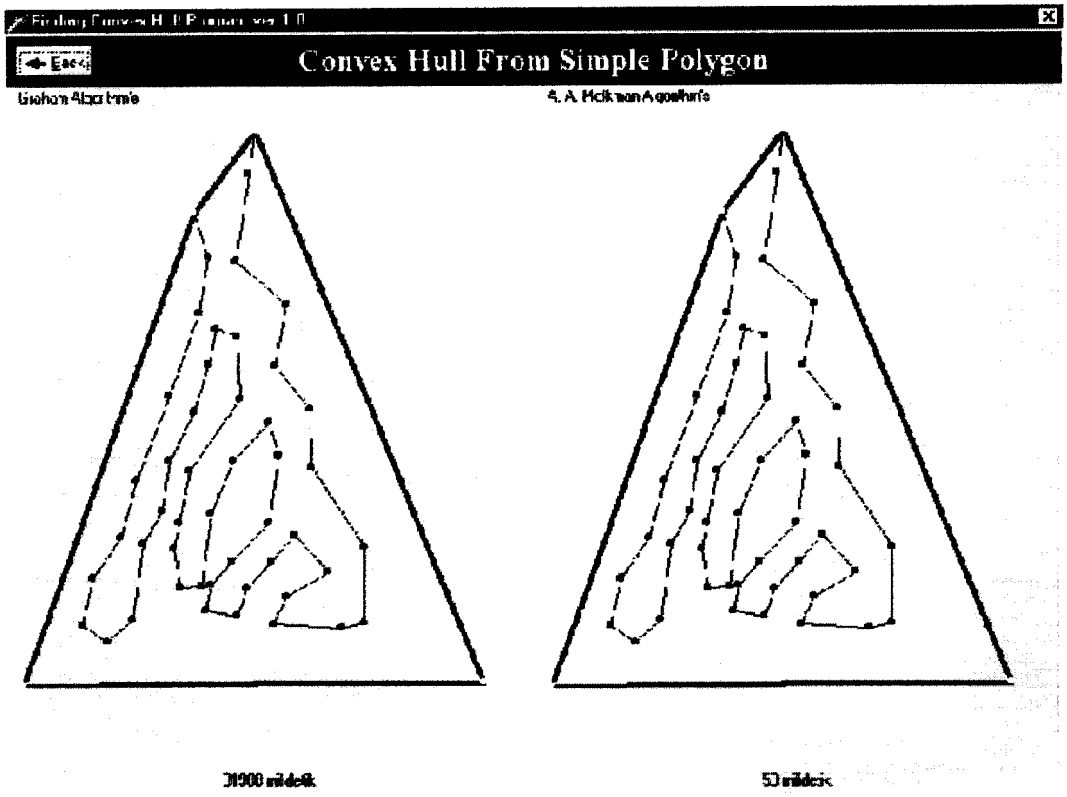
Keterangan pada gambar 4 29 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 47950/1000 = 47,95 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 110/1000 = 0,11 milidetik




Gambar 4.30

Simple Polygon22 dengan 50 vertices

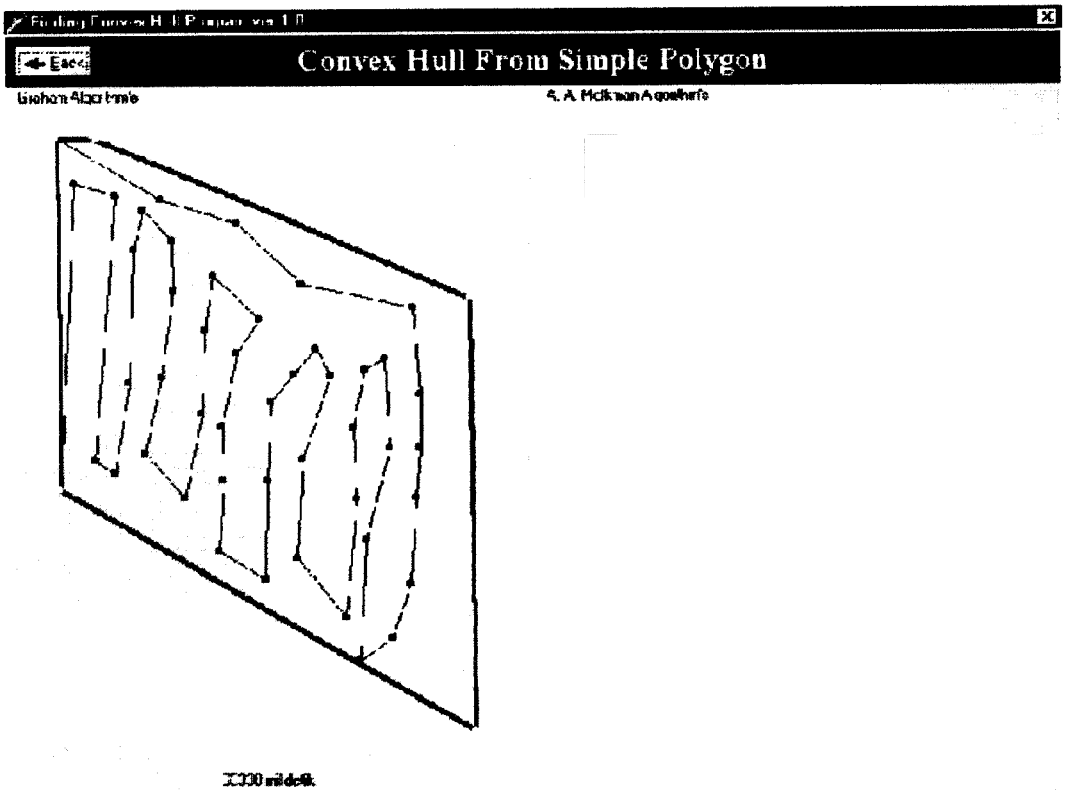
Keterangan : pada gambar 4.30 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 62890/1000 = 62,89 milidetik sedangkan algoritma Melkman hanya memerlukan waktu $110/1000 = 0{,}11$ milidetik.




Gambar 4.31

Simple Polygon23 dengan 50 vertices

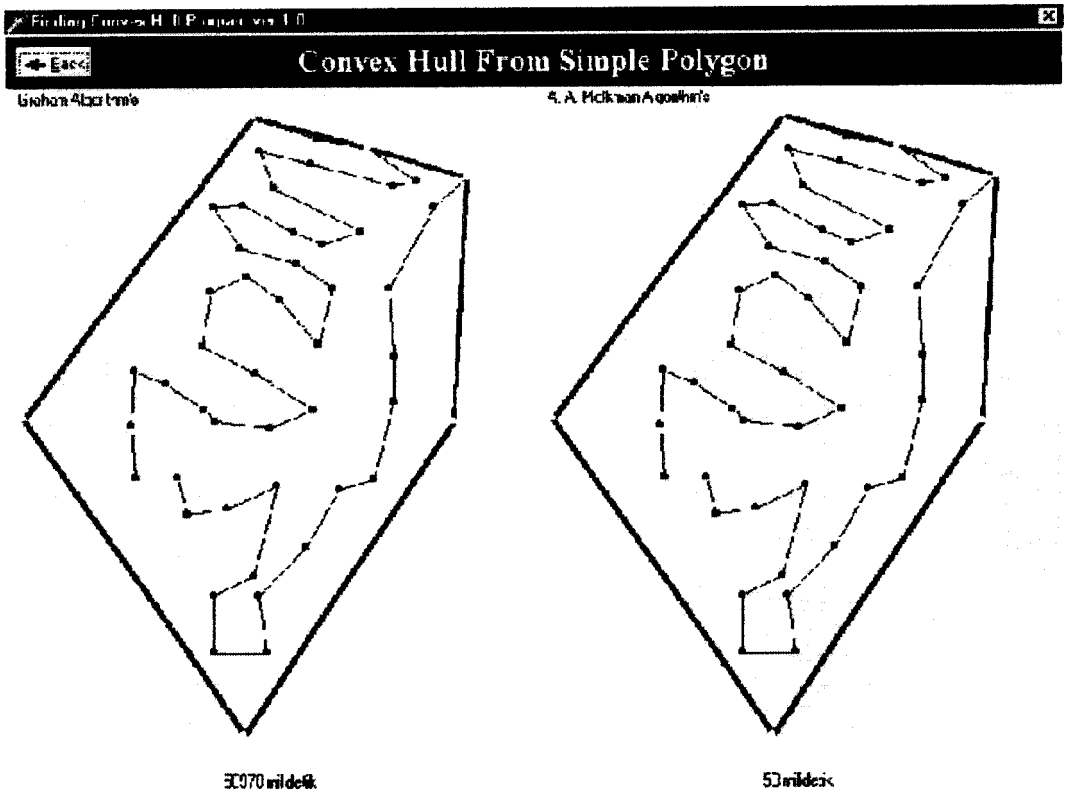
Keterangan : pada gambar 4.31 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu **8** 1900/1000 = **8** 1,9 milidetik sedangkan algoritma Melkman hanya memerlukan **waktu** 50/1000 = *0,05* milidetik. Untuk bentuk yang ini terlihat jelas algoritma Melkman lebih cepat dibanding untuk bentuk yang lain.




Gambar 4.32

Simple Polygon24 dengan 50 vertices

Keterangan . pada gambar 4 32 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 63330/1000 = 63,33 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 50/1000 = *0,05* milidetik. Untuk bentuk yang ini terlihat jelas algoritma Melkman lebih cepat dibanding dengan bentuk yang lain.




Gambar 4.33

Simple Polygon25 dengan 50 vertices

Keterangan pada gambar 4.25 dapat dilihat bahwa untuk menggambarkan convex hull pada simple polygon dengan 50 vertices, algoritma Graham memerlukan waktu 50970/1000 = 50,97 milidetik sedangkan algoritma Melkman hanya memerlukan waktu 50/1000 = 0,05 milidetik. Untuk bentuk yang ini terlihat jelas algoritma Melkman lebih cepat dibanding dengan bentuk yang lain.

## 2. PENEMPATAN PERHITUNGAN WAKTU

Penempatan Perhitungan waktu pada setiap metode dilakukan sesaat sebelum proses penggambaran dimulai, dan perhitungan akan berhenti sesaat setelah proses perhitungan dilakukan sebanyak 1000 kali. Perhitungan sebanyak seribu kali bertujuan supaya didapatkan hasil perhitungan waktu yang lebih akurat terutama untuk eksekusi program yang relatif sangat cepat.

Flowchart penempatan perhitungan waktu pada algoritma Three Coins (Graham) dan algoritma Melkman dapat dilihat pada gambar **4.34.** Sedangkan implementasi program dapat dilihat sebagai berikut

Fungsi durasi untuk menghitung selisih waktu adapun variabel w1 untuk waktu awal dan w2 untuk waktu akhir.

Perhitungan hanya sampai menit, semua dalam satuan milidetik

```
function tform1.durasi(w1,w2  : tdatetime):integer;
var
   jam1,menit1,detik1,mili1 : word;
   jam2,menit2,detik2,mili2 : word;
   hasil,t1,t2,t3 : integer;
begin
  decodetime(w1,jam1, menit1,detik1,mili1);
  decodetime(w2,jam2,menit2,detik2,mili2);
  t1 := mili2-mili1;
  if t1 < 0 then
     begin
       t1 := mili2+1000-mili1;
       dec(detik2);
     end;
```

```
t2 := detik2-detik1;
if t2 < 0 then
     begin
       t2 := detik2+60-detik1;
       dec(menit2);
     end;
t3 := menit2-menit1;
if t3 < 0 then
     begin
       t3 := menit2+60-menit1;
       dec(jam2);
     end;
hasil := t3*60000+t2*1000+t1;
result := hasil;
end;


procedure TForm1.Proses1Click(Sender: TObject);
var j1,j2,j3 : tdatetime;
    i : integer;
begin
```

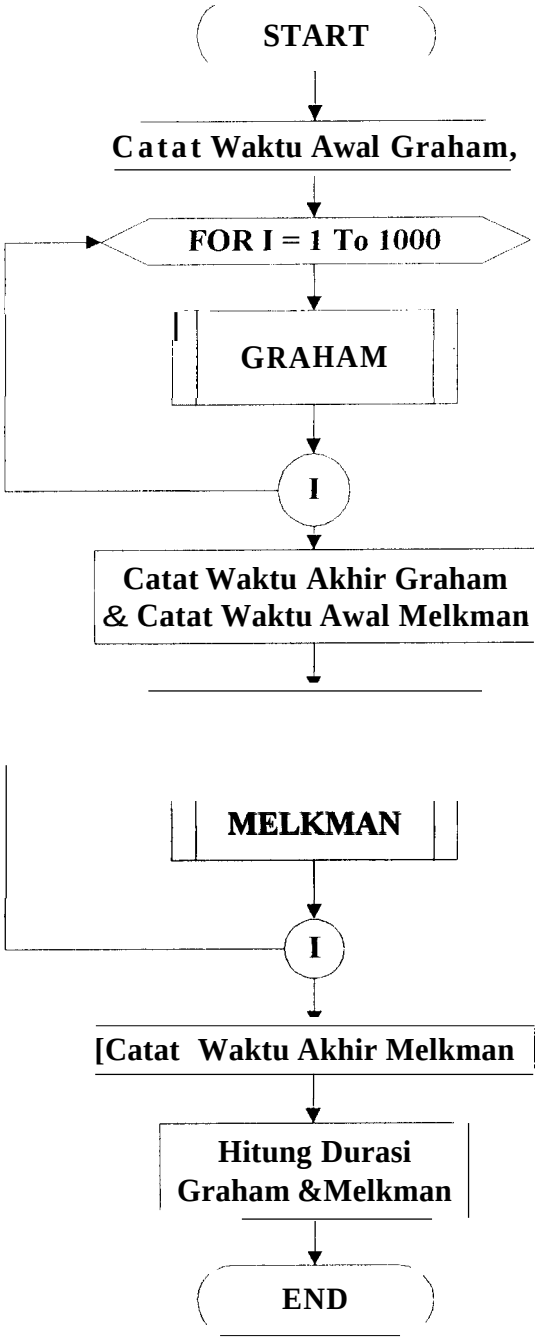
[jika jumlah titik lebih dari dua maka algoritma dapat dijalankan]

```
  if jum > 2 then
     begin
```

[gambar garis yang menghubungkan vertex awal dan akhir]

```
       image1.Canvas.moveto(t[0].x,t[0].y);
       image1.Canvas.LineTo(t[jum-1].x,t[jum-1].y);
       form1.Refresh;
```

```
    [Menyimpan waktu awal Graham]
        j1 := time;
    [Jalankan algoritma graham sebanyak seribu kali]
        for i := 1 to 1000 do graham;
    [Menyimpan waktu akhir Graham dan waktu awal Melkman ]
        j2 := time;
    [Jalankan algoritma Melkman sebanyak seribu kali]
        for i := 1 to 1000 do melkman;
    [Menyimpan waktu akhir Melkman]
        j3 := time;
    [Hitung waktu proses algoritma Graham]
        d1 := durasi(j1,j2);
    [Hitung waktu proses algoritma Melkman]
        d2 := durasi(j2,j3);
    end;
[Tombol proses di non aktifkan]
  proses1.Enabled := false;
[Tombol detail diaktifkan]
  detail1.Enabled := true;
  tekan := false;
end;
```




Gambar **4.34**

Flowchart Penempatan Perhitungan Waktu

Hasil dari percobaan penempatan perhitungan waktu dilakukan dengan menginputkan 50 vertex , dan percobaan dilakukan sebanyak 25 kali dengan

perhitungan algoritma diulang sebanyak 1000 kali untuk setiap percobaan dari masing-masing algoritma dapat dilihat pada tabel 4 1

Sedangkan untuk rata-rata dari satu kali perhitungan convex hull dari masing-masing data pada tabel 4 1 dibagi 1000 sehingga didapatkan hasil seperti terlihat pada tabel 4 2

Tabel 4.1.

Hasil Perhitungan Waktu

| NO. POLYGON | GRAHAM(ms) $O(n^2)$ | MELKMAN(ms) O(n) |
|---|---|---|
| 1 | 42020 | 110 |
| 2 | 44540 | 170 |
| 3 | 44880 | 110 |
| 4 | 48500 | 110 |
| 5 | 46850 | 160 |
| 6 | 47120 | 110 |
| 7 | 47890 | 110 |
| 8 | 56300 | 110 |
| 9 | 44820 | 110 |
| 10 | 71620 | 170 |
| 11 | 60530 | 160 |
| 12 | 57730 | 110 |
| 13 | 67670 | 160 |
| 14 | 60910 | 110 |
| 15 | 58500 | 160 |
| 16 | 65200 | 110 |
| 17 | 45580 | 170 |
| 18 | 49160 | 110 |
| 19 | 53500 | 110 |
| 20 | 44100 | 60 |
| 21 | 47950 | 110 |
| 22 | 62890 | 110 |
| 23 | 81900 | 50 |
| 24 | 63330 | 50 |
| 25 | 50970 | 50 |
| **JUMLAH** : | 1364460 | 2900 |

Tabel 4.2

Rata-Rata Hasil Perhitungan Waktu

| NO.POLYGON | GRAHAM(ms) $O(n^2)$ | MELKMAN(ms) O(n) |
|---|---|---|
| 1 | 42,020 | 0,110 |
| 2 | 44.540 | 0.170 |
| 3 | 44,880 | 0,110 |
| 4 | 48,500 | 0,110 |
| 5 | 46,850 | 0,160 |
| 6 | 47,120 | 0,110 |
| 7 | 47,890 | 0,110 |
| 8 | 56,300 | 0,110 |
| 9 | 44,820 | 0,110 |
| 10 | 71,620 | 0,170 |
| 11 | 60,530 | 0,160 |
| 12 | 57,730 | 0,110 |
| 13 | 67,670 | 0,160 |
| 14 | 60,910 | 0,110 |
| 15 | 58,500 | 0,160 |
| 16 | 65.200 | 0.110 |
| 17 | 45,580 | 0,170 |
| 18 | 49.160 | 0,110 |
| 19 | 53,500 | 0,110 |
| 20 | 44.100 | 0.060 |
| 21 | 47,950 | 0,110 |
| 22 | 62,890 | 0,110 |
| 23 | 81,900 | 0,050 |
| 24 | 63.330 | ***0,050*** |
| 25 | 50,970 | 0,050 |
| JUMLAH : | 1364,460 | 2,900 |

Dari hasil tabel 4.1 dan tabel 4.2 maka dapat dihitung

Kecepatan rata-rata untuk algoritma Graham : 1346,46/25 = 53,8584 **ms**

Kecepatan rata-rata untuk algoritma Melkman : 2,9 /25 = 0,1 I6 ms

Maka algoritma Melkman lebih cepat 464 kali dibandingkan dengan algoritma Three Coins (Graham)

## 3. KOMPLEKSITAS WAKTU

Ukuran umum yang dipakai untuk menentukan efektif tidaknya sebuah algoritma, sering dinyatakan dengan menggunakan fungsi kompleksitas. Fungsi ini mengindikasikan berapa lama waktu yang dibutuhkan oleh sebuah algoritma dengan input yang berbeda.

Adapun perhitungan kompleksitas waktu untuk program yang dibuat adalah :

```
procedure extermal;
var
   i,k : integer;
   j   : tpoint;
begin
  for i := 0 to jum-1 do sp[i] := t[i];                    (n)
  k := 0;
  for i := 1 to jum-1 do                                   (n)
    begin
            if t[i].y > t[k].y then
            k := i
      else
          if t[i].y = t[k].y then
             if t[i].x < t[k].x then
                k := i;
    end;
  j     := sp[k];
  sp[k] := sp[0];
  sp[O] := j;
end;
```

karena yang terjadi dalam prosedur extermal hanya proses sequence sehingga kompleksitas waktu procedure extermal adalah O(n).

```
procedure bubble;
var
   i,j : integer;
   k   : tpoint;
begin
  for i := 1 to jum-1 do                                    (n)
      for j := i+1 to jum-1 do                              (n-1)
           if which side(sp[0],sp[j],sp[i]) < 0 then
             begin
               k     := sp[i];
               sp[i] := sp[j];
               sp[j] := k;
             end;
end;
```

karena procedure bubble memiliki nested loop sehingga n(n-1) $= n^2 - n$, maka kompleksitas waktunya adalah $O(n^2)$

```
procedure graham;
var i : integer;
begin
  extermal;                                                 O(n)
  bubble;                                                   O(n²)
  push(sp[0]);
  push(sp[1]);
  push(sp[2]
  for i := 3 to jum-1 do                                    n-3
    begin
      while which side(q[atas-1],q[atas],sp[i]) > 0 do pop;
      push(sp[i]);
    end;
end;
```

maka yang dipilih adalah pangkat yang terbesar yaitu $O(n^2)$

```
procedure melkman;
var i,tanda : integer;
begin
```

```
// masukkan tiga titik pertama ke dalam decque
if hull(t[0],t[1],t[2]) > 0 then
   begin
     apush(t[0]);
     apush(t[1]);
   end
else
  begin
    apush(t[1]);
    apush(t[0]);
  end;
apush(t[2]);
bpush(t[2]); // insertkan titik ke-3 kedalam decque
i := 2;
tanda := 0;
while i < jum-1 do                                    (n)
  begin
    inc(i);
    while ((hull(t[i],d[0],d[1])>=0) and (hull(d[kep-1],
    d[kep],t[i])>=0) and (i<jum-1)) do inc(i);
    if ((hull(t[i],d[0],d[1])>=0) and (hull(d[kep-1],
    d[kep],t[i])>=0) and (i=jum-1)) then tanda:=1 else tanda:=0;
    while hull(d[kep-1],d[kep],t[i]) <= 0 do apop;
    apush(t[i]);
    while hull(t[i],d[0],d[1]) <= 0 do bpop;
    bpush(t[i]);
  end;
if ((tanda = 1) and (i=jum-1)) then
   begin
     apop;
     bpop;
   end;
end;
```

sehingga algoritma Melkman mempunyai kompleksitas waktu sebesar $O(n)$[1].

Dari hasil perbandingan kompleksitas waktu , maka algoritma Graham adalah $O(n^2)$ sedangkan algoritma Melkman adalah O(n).

---

[1] Lang. Pierre Computing the convex hull of a simple polygon in O(n)time with A A Melkman's algorithm, **[http://www.cs.mcgill.ca/~plang/copgeo/copgeo.html#Reference]**