Edge Detection dengan Algoritma Canny

Christopher Danil
STMIK IBBI
Jl. Sei Deli No. 18 Medan, Telp. 061-4567111 Fax. 061-4527548
e-mail: christopherdanil@gmail.com

Abstrak

Metode *edge detection* akan mendeteksi semua *edge* atau garis-garis yang membentuk objek gambar dan akan memperjelas kembali pada bagian-bagian tersebut. Tujuan pendeteksian ini adalah bagaimana agar objek di dalam gambar dapat dikenali dan disederhanakan bentuknya dari bentuk sebelumnya. Metode *Canny edge detection* merupakan pengembangan dari metode dasar *edge detection*. Perancangan sebuah prosedur dengan menerapkan langkah-langkah metode *Canny edge detection* akan menghasilkan sebuah tampilan gambar yang berbeda dengan menampilkan efek relief didalamnya. Efek relief adalah seperti sebuah tampilan batu kasar yang diukir, yaitu garis-garis kasar yang membentuk sebuah penggambaran objek di dalamnya. Efek relief terbentuk dari bayangan terang dan gelap. Kedua bayangan ini terjadi akibat adanya sorotan sinar mengenai gambar dari arah tertentu. Kelebihan dari metode Canny ini adalah kemampuan untuk mengurangi noise sebelum melakukan perhitungan deteksi tepi sehingga tepi-tepi yang dihasilkan lebih banyak.

Kata kunci: *Deteksi Tepi, Operator Canny*

Abstract

Edge detection methods will detect all edges or lines that form an image object and would clarify these parts. The purpose of this detection is how to keep the object in the image can be recognized and simplified form of the previous shape. Canny edge detection method is an extension of the basic method of edge detection. Designing a procedure to implement measures Canny edge detection method will produce an image that is different from showing the effects of relief in it. Relief effect is like a rough stone-carved look, the rough lines that form an object representation in it. Formed from the relief effect of light and dark shadows. Both of these shadows are the result of the beam on the image of a particular direction. The advantages of the Canny method is the ability to reduce noise prior to the calculation of edge detection so that the edges produced more.

Keywords*: Edge Detection, Canny Operator*

1. Pendahuluan

Tepi (*edge*) adalah perubahan nilai intensitas derajat keabuan yang cepat atau tiba-tiba (besar) dalam jarak yang singkat. Tujuan mendeteksi tepi sendiri adalah untuk mengelompokkan objek-objek dalam citra, dan juga digunakan untuk menganalisis citra lebih lanjut. Ada banyak algoritma yang digunakan untuk mendeteksi tepi, salah satu diantaranya adalah deteksi tepi Canny (*Canny Edge detection*).

Canny edge detector dikembangkan oleh John F. Canny pada tahun 1986 dan menggunakan algoritma multi-tahap untuk mendeteksi berbagai tepi dalam gambar. Walaupun metode tersebut telah berumur cukup lama, namun metode tersebut telah menjadi metode deteksi tepi standar dan masih dipakai dalam penelitan.

Adapun kategori algoritma yang dikembangkan oleh John F. Canny adalah sebagai berikut :

1. Deteksi : Kemungkinan mendeteksi titik tepi yang benar harus dimaksimalkan sementara kemungkinan salah mendeteksi titik tepi harus diminimalkan. Hal ini diamksudkan untuk memaksimalkan rasio *signal-to-noise*.
2. Lokalisasi : Tepi terdeteksi harus sedekat mungkin dengan tepi yang nyata.
3. Jumlah tanggapan : Satu tepi nyata tidak harus menghasilkan lebih dari satu ujung yang terdeteksi.

Dengan rumusan John F. Canny tentang kriteria ini, maka *Canny edge detector* optimal untuk

Kelas tepian tertentu (dikenal sebagai *step edge*). Gambar pada Gambar 1 berikut ini untuk menunjukkan bagaimana cara kerja algoritma Canny.

Gambar 1. Gambar yang digunakan sebagai contoh *Canny Edge Detection*

Algoritma Canny berjalan dalam 5 langkah yang terpisah yaitu:

1. *Smoothing* : Mengaburkan gambar untuk menghilangkan *noise*
2. *Finding gradien* : Tepian harus ditandai pada gambar memiliki gradien yang besar.
3. *Non-maksimum-suppresion* : Hanya maxima lokal yang harus ditandai sebagai *egde*.
4. *Double thresholding* : Tepian yang berpotensi ditentukan oleh *thresholding*.
5. *Edge Tracking by hysteresis* : Tepian final ditentukan dengan menekan semua sisi yang tidak terhubung dengan tepian yang sangat kuat.

Tidak dapat dipungkiri bahwa semua gambar yang diambil dari kamera akan berisi sejumlah *noise*. Untuk mencegah *noise* salah dideteksi sebagai tepian, maka *noise* harus dikurangi. Oleh karena itu pada langkah pertema gambar harus diperhalus dengan mengggunakan *Gaussian filter*. Inti dari *Gaussian filter* adalah standar deviasi dengan $\sigma = 1.4$ ditunjukkan pada persamaan (1) di bawah ini. Sedangkan hasil dari implementasi Gaussian filter terhadap gambar ditunjukkan pada Gambar 2.

$$B = \frac{1}{159} \cdot \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \quad \text{.....(1)}$$

(a) Original (b) Diperhalus

Gambar 2. Gambar *Grayscale* yang Diperhalus dengan *Gaussian Filter* untuk Menekan *Noise*

Algoritma Canny pada dasarnya menemukan titik tepi pada gambar *grayscale* dengan perubahan nilai intensitas yang paling besar, daerah ini ditemukan dengan menentukan gradien gambar. Gradien pada setiap piksel gambar yang telah diperhalus ditentukan dengan menerapkan operator Sobel. Langkah Kedua adalah memperkirakan gradien pada arah x dan y. Hal tersebut ditunjukkan dalam Persamaan (2).

$$K_{GX} = \begin{bmatrix} 1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{.....(2)}$$

$$K_{GY} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Magnitudo gradien (juga dikenal sebagai kekuatan tepi) dapat ditentukan sebagai jarak Euclidean yang diukur mengukur dengan menerapkan hukum Pythagoras seperti yang ditunjukkan dalam

Persamaan (3). Yang terkadang disederhanakan dengan menerapkan ukuran jarak Manhattan seperti yang ditunjukkan dalam Persamaan (4) untuk mengurangi kompleksitas komputasi.

$$|G = \sqrt{G_x^2 + G_y^2} \quad \text{.....(3)}$$

$$G| - G_x| + |G_y| \quad \text{.....(4)}$$

dimana:
Gx dan Gy adalah gradien pada masing-masing arah x dan y.

Hal ini tampak jelas dari Gambar 3, bahwa gambar dengan gradien yang besar sering menunjukkan tepian yang cukup jelas. Namun, tepian biasanya luas dan dengan demikian tidak dapat menunjukkan persis di mana tepian yang sebenarnya. Untuk menentukan tepian yang sebenarnya ini, arah tepian harus ditentukan dan disimpan seperti ditunjukkan dalam Persamaan (5).

$$\theta = \arctan\left(\frac{|G_y|}{|G_x|}\right) \quad \text{.....(5)}$$

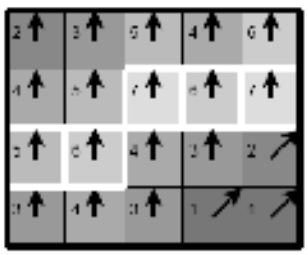
(a) Gambar yang diperhalus (b) Magnitudo Gradien

Gambar 3. Magnitude Gradien dari penerapan Operator Sobel terhadap Gambar yang Telah Diperhalus

Pada langkah ketiga bertujuan untuk mengkorversikan tepian yang masih *blurred* pada gambar hasil magnitude gradien hingga menhasilkan tepian yang tajam. Pada dasarnya hal ini dilakukan dengan mempertahankan semua maxima lokal dalam gambar gradien dan menghapus segala sesuatu yang lain. Algoritma adalah untuk setiap piksel pada gambar gradien adalah sebagai berikut:

1. Putar arah gradien θ ke arah 45˚ terdekat, kemudian hubungkan dengan 8 titik tetangga yang terhubung dengannya.
2. Bandingkan nilai piksel tepian saat ini dengan nilai piksel tepian dalam arah positif dan negatif gradien. Jika arah gradien adalah utara (θ =90 ◦), bandingkan dengan piksel ke utara dan selatan.
3. Jika nilai piksel tepian saat ini adalah yang terbesar, maka simpan nilai tepian tersebut, namun jika bukan, hapus nilai tersebut.

Sebuah contoh sederhana dari penghapusan non maksimum ditunjukkan pada Gambar 4. Hampir semua piksel yang memiliki arah gradien yang menunjukkan arah utara, oleh karena itu mereka dibandingkan dengan piksel atas dan bawah. Piksel yang berubah menjadi maksimal dalam perbandingan ini ditandai dengan warna putih pada perbatasan, sisanya dihapus. Gambar 5 menunjukkan efek pada gambar tes.



Gambar 4. Ilustrasi Penghapusan Non Maksimum

(a) Hasil Gradien (b) Tepian setelah Penghapusan Non Maksimum

Gambar 5. Penghapusan Non Maksimum

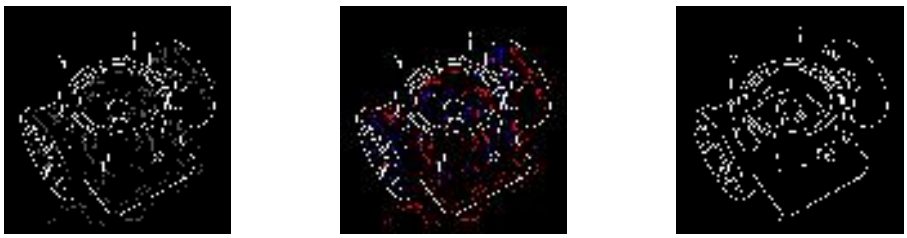
Piksel tepian yang tersisa setelah dilakukan penghapusan non maksimum ditandai dengan nilai piksel per piksel yang kuat. Kebanyakan dari titik ini adalah tapian yang nyata pada gambar, akan tetapi beberapa kemungkinan disebabkan oleh *noise* atau variasi wana karena permukaan yang kasar. Cara paling sederhana untuk membedakannya adalaha menggunakan nilai *threshold* (ambang batas) sehingga hanya tepian dengan nilai yang kuar yang akan dipertahankan. Disini pada algoritma Canny menggunakan sistem *thresholding* ganda dimana tepian dengan nilai yang lebih besar dari *threshold* atas ditandai sebagai titik kuat, tepian dengan nilai yang lebih kecil dari *threshold* bawah akan dihapus, dan tepian dengan nilai piksel antara *threshold* atas dengan *threshold* bawah akan ditandai sebagai tepian yang lemah. Hasil pada contoh gambar menggunakan *threshold* 20 dan 80 ditunjukkan pada Gambar 6.

(a) Tepian setelah Penghapusan Non Maksimum (b). Thresholding Ganda

Gambar 6. *Edge Thresholding*

Tepian yang kuat diintepretasikan sebagai " tepian yang pasti " dan dapat segera dimasukkan sebagai tepian pada gambar akhir. Tepi lemak termasuk jika dan hanya jika mereka terhubung ke tepi yang kuat, dengan logika bahwa noise dan variasi warna tidak mungkin untuk menghasilkan tepi yang kuat (dengan penyesuaian yang tepat dari thresholding). Dengan demikian tepian yang kuatlah yang akan menghasilkan tepian yang asli pada gambar. Tepian yang lemah dapat terjadi karena memang merupakan tepian yang nyata atau noise / variasi warna.

Edge tracking dapat dilakukan dengan analisis BLOB (*Binary Large Object*). Piksel tepian dibagi dalam BLOB yang terkoneksi menggunakan 8 hubungan tetangga. BLOB yang mengandung setidaknya 1 piksel tepian yang kuat akan disimpan, sedangkan lainnya dihapus. Efek *edge tracking* pada gambar contoh ditunjukkan pada Gambar 7.



(a) Thresholding Ganda (b) *Edge Tracking by hysteresis* (c) Hasil Akhir

Gambar 7. *Edge Tracking* dan Hasil Akhir

2. Metode Penelitian

Tahapan analisis sistem merupakan tahapan yang sangat penting karena kesalahan di dalam tahapan ini akan menyebabkan kesalahan pada tahapan selanjutnya. Proses analisis sistem dalam pengembangan sistem merupakan suatu prosedur yang dilakukan untuk pemeriksaan masalah dan penyusunan pemecahan masalah yang timbul serta membuat spesifikasi sistem yang baru. Adapun tahapan dari analisis sistem dapat digambarkan dalam bentuk algoritma sebagai berikut:

Algoritma Form Utama

1. *Load Form* utama, *FormEdge Detection*, *Form* Memilih gambar.
2. Gambar yang dapat dipilih merupakan gambar dengan format BMP, JPG dan GIF
3. Gambar yang dipilih ditampilkan pada kotak gambat yang pertama.

Algoritma Form Edge Detection

1. Inisialisasi threshold low dan threshold high
2. Inpu gambar
3. Tekan tombol proses, akan melakukan langkah-langkah operator Canny yang akan ditampilkan pada masing-masing kotak gambar
4. Jika ditekan tombol "*Close*" tutup aplikasi yang sedang dimainkan.

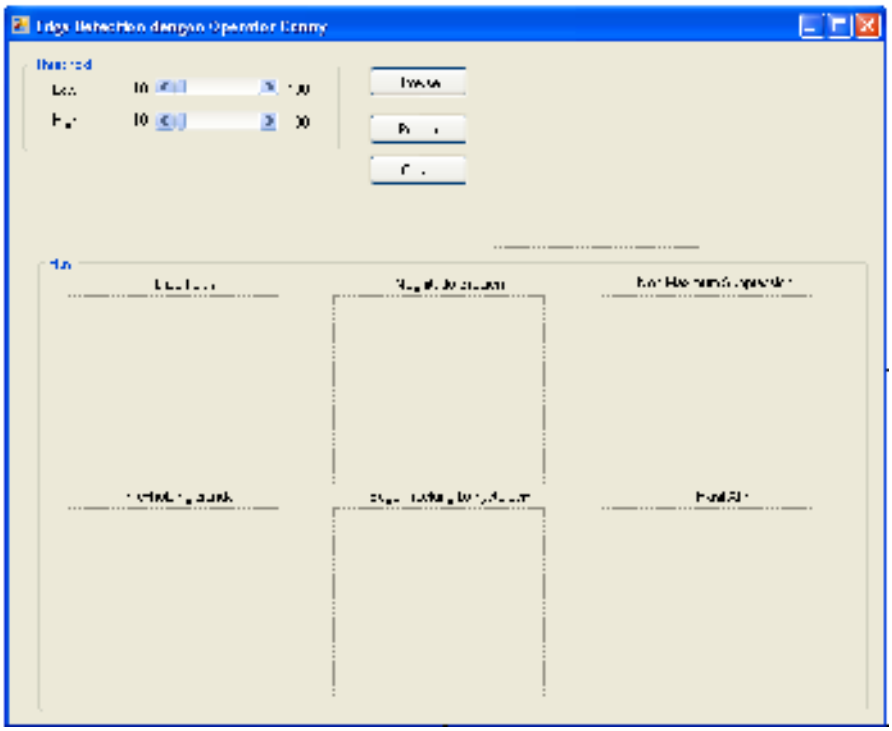
Algoritma Edge Detection

```
function ok( x, y: integer ): boolean;
   begin
     result := ( x >= 0 ) and ( y >= 0 ) and ( x < b.Width ) and ( y < b.Height );
   end;
 begin
   result := false;
   if not edge[y][x] then exit;

   edge[y][x] := false;
   result := mag[y][x] >= thrlow;
   for j := -1 to 1 do
     for i := -1 to 1 do
       if ok( x + i, y + j ) then
         result := trace( x + i, y + j ) or result;

   //jika hasil rekursi menghasilkan true
   //(terhubung dengan piksel yang nilainya lebih besar dari hi-threshold)
   if result then begin
     pr := br.ScanLine[y];
     pr[x] := rgb_hitam;//citra yang baru berwarna latar putih
   end;
 end;
begin
 bg := gaussian( b, 1 );
 result := citra_create( b.Width, b.Height );//buat citra(bitmap) baru

 setlength( p, b.Height );
 for j := 0 to high( p ) do
   p[j] := bg.ScanLine[j];

 setlength( gx, b.Height );
 for j := 0 to high( gx ) do
   setlength( gx[j], b.Width );

 setlength( gy, b.Height );
 for j := 0 to high( gx ) do
   setlength( gy[j], b.Width );

 setlength( mag, b.Height );
 for j := 0 to high( mag ) do
   setlength( mag[j], b.Width );
```

```
  setlength( edge, b.Height );
  for j := 0 to high( edge ) do
    setlength( edge[j], b.Width );

  //hitung gradien gx dan gy menggunakan operator sobel
  for j := 1 to b.Height - 2 do begin
    for i := 1 to b.Width - 2 do begin
      gy[j][i] := ( p[j + 1][i - 1].r - p[j - 1][i - 1].r ) + 2 * ( p[j + 1][i].r - p[j - 1][i].r ) + ( p[j + 1][i + 1].r - p[j
- 1][i + 1].r );
      gx[j][i] := ( p[j - 1][i + 1].r - p[j - 1][i - 1].r )
        + 2 * ( p[j][i + 1].r - p[j][i - 1].r )
        + ( p[j + 1][i + 1].r - p[j + 1][i - 1].r );
      mag[j][i] := sqrt( gx[j][i] * gx[j][i] + gy[j][i] * gy[j][i] );
    end;
  end;

  //non-maximal suppression
  for j := 1 to b.Height - 2 do begin
    for i := 1 to b.Width - 2 do begin
      dir := arctan2( gy[j][i], gx[j][i] );
      dx := round( cos( dir ) );
      dy := round( sin( dir ) );

      edge[j][i] := ( mag[j][i] >= mag[j + dy][i + dx] ) and ( mag[j][i] >= mag[j - dy][i - dx] );
      if edge[j][i] then begin
        edge[j + dy][i + dx] := false;
        edge[j - dy][i - dx] := false;
      end;
    end;
  end;

  //apply hysteresis
  thrlow := 10;
  thrhigh := 90;
  br := result;
  for j := 1 to b.Height - 2 do begin
    for i := 1 to b.Width - 2 do begin
      if edge[j][i] and (mag[y][x] >= thrhigh) then trace( i, j );//trace melakukan edge-linking
    end;
  end;
```

3. Hasil dan Analisis

Program *Edge Detection* ini dirancang untuk gambar hasil akhir berupa gambar yang menampilkan tepian dari gambar input. Program dirancang dalam bahasa *Visual Basic* dengan memanfaatkan komponen-komponen intrinsik yang terdapat pada *Visual Basic*. Rancangan *form* ini dapat dilihat seperti pada gambar 8.



Gambar 8. *Form Edge Detection*

Komponen utama pada *form* ini adalah *groupbox*, HscrollBar, *button*, dan *picture box*. Form ini digunakan untuk menghasilkan gambar dengan *edge* dengan menggunakan operator Canny. Bagian "HscrollBar" digunakan untuk menentukan nilai *threshold low* dan *high*. Tombol "*Browse*" untuk mencari *file* gambar ke dalam picture box dengan memunculkan sebuah kotak dialog. Tombol "Proses" digunakan untuk menghasilkan gambar *edge* dari gambar yang diinput sekaligus menghasilkan gambar menurut langkah-langkah dalam operator Canny. Tombol "Close" digunakan untuk menutup aplikasi yang sedangkan dijalankan.

4. Kesimpulan

Dari hasil dan analisis dapat diambil beberapa kesimpulan yaitu :

1. Dengan menggunakan operator Canny dapat menghasilkan gambar dengan tepian nyata yang baik.
2. Dengan menggunakan operator Canny, kemungkinan salah mendeteksi noise dan variasi warna sebagai edge semakin kecil
3. Algoritma operator Canny dilengkapi edge tracking yang memungkinkan penekanan semua sisi yang tidak terhubung dengan tepian yang sangat kuat

Referensi

Buku Teks:

[1.] Sergei Azernikov. Sweeping solids on manifolds. In Symposium on Solid and Physical Modeling. 2008.

[2.] John Canny. A computational approach to edge detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on, PAMI-8(6): 10-15

[3.] F. Mai, Y. Hung, H. Zhong, W. Sze. A hierarchical approach for fast and robust ellipse, extraction. Pattern Recognition. 2008.

[4.] Thomas B. Moeslund.. Image and Video Processing. 2008.