Pengolahan Citra Digital Deteksi Tepi Untuk Membandingkan Metode *Sobel, Robert* dan *Canny*

Putu Teguh Krisna Putra, Ni Kadek Ayu Wirdiani
Jurusan Teknologi Informasi, Fakultas Teknik, Universitas Udayana
Bukit Jimbaran, Bali, Indonesia, telp. +62361703315
e-mail: teguhkrisna91@yahoo.com, ayu_wirdi@yahoo.com

Abstrak

Pengolahan citra digital merupakan teknologi yang dapat dimanfaatkan dalam meningkatkan kualitas citra dan proses identifikasi citra untuk memperkaya informasi yang bersumber pada citra tersebut. Penulisan ini bertujuan untuk membandingkan metode deteksi tepi pada pengolahan citra digital. Deteksi tepi merupakan sebuah proses di mana proses tersebut berfungsi untukmendeteksi garis tepi yang membatasi dua wilayah citra. Deteksi tepiitu sendiri bertujuan untuk menandai bagian yang menjadi detail citra, dan memperbaikiserta mengubah citra. Penulisan ini membahas tentang perbandingan hasil deteksi tepi dengan metode *Canny, Sobel* dan *Robert* dengan menggunakan aplikasi Delphi 7. Perangkat lunak yang digunakan adalah Borland Delphi 7Enterprise sebagai bahasa pemrogramannya. Hasil dari perbandingan ketiga metode tersebut adalah metode *Canny* lebih baik dalam melakukan pengolahan kualitas citra untuk penghitungan deteksi tepi karena output dari metode *Canny* memiliki batas dan tepi yang lebih jelas.

Kata Kunci: Pengolahan Citra, Deteksi Tepi, *Canny*, *Sobel*, *Robert*, Delphi 7

Abstract

Digital image processing is a technology that can be utilized in improving image quality and image for an identification process which nformation is based on image. This research aims to compare the methods of edge detection in digital image processing. Edge detection is a process in which the process is used to detect the outline of the border of two image regions. Edge detection is intended to mark the part that becomes the image detail, and improve and transform the image. Writing is about the comparison of the results with the method ofedge detection onCanny, and RobertSobel application using Delphi 7. Software used is Borland Delphi 7 Enterprise as the programming language. The results of the comparison of three methods is the method of Canny perform better in image quality processing for edge detection calculation because the output of the Canny method has clear boundaries and edges.

Keywords: *Image Processing, Edge Detection, Canny, Sobel, Robert, Delphi 7*

1. Pendahuluan

Citra sebagai salah satu komponen multimedia memegang peranan sangat penting sebagai bentuk informasi visual. Citra mempunyai karakteristik yang tidak dimiliki oleh data teks, yaitu citra kaya akan informasi. Maksud dari “citra kaya akan informasi” adalah citra dapat memberikan informasi yang lebih banyak dibandingkan dengan informasi yang disajikan dalam bentuk teks [1].

Secara sederhana, citra adalah image pada bidang dwimatra (dua dimensi). Sumber cahaya menerangi objek, objek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh alat-alat optik, misalnya mata pada manusia, kamera, pemindai (scanner), dan sebagainya, sehingga bayangan objek yang disebut citra tersebut terekam [1]. Tetapi kemudian konsep citra dan pengolahannyadihubungkan dengan pengubahan dan perbaikan citra (gambar) yang bertujuan antaralain:

a. Memperbaiki kesalahan data sinyal gambar akibat transmisi dan selama akuisisi sinyal.
b. Meningkatkan penampakan gambar sehingga dapat 'diterima' oleh sistem penglihatan manusia.

Dengan perkembangan komputer dan alat pengambilan gambar secara digital yang semakin berkembang saat ini, sehingga menghasilkan banyak fasilitas untuk melakukanproses

pengolahan gambar agar lebih sesuai dengan kebutuhan. Salah satunya adalahdeteksi tepi pada gambar, karena dengan menggunakan proses deteksi tepi gambar makaproses pengolahan manipulasi pada gambar akan lebih mudah dilakukan. Deteksi tepigambar merupakan sebuah proses dimana suatu proses yang menghasilkan tepi-tepi dariobyek-obyek citra yang bertujuan untuk menandai bagian yang menjadi detail citra sertamemperbaiki detail citra yang kabur.Tujuan dari penulisan ini tidak lain adalah untuk membuat sebuah perbandingan antaramasing-masing operator deteksi tepi, manakah yang lebih baik yang dapat digunakansesuai dengan kebutuhan [1] [2].

2. Metode Penelitian

Dalam pembangunan aplikasi ini dibutuhkan beberapa script program yang digunakanuntuk melengkapi kesempurnaan dari program deteksi tepi gambar ini. Adapun scriptyang digunakan dalam pembuatan program ini. Berikut pembagian utntuk masing-masing fungsi pada aplikasi pengolahan citra digital deteksi tepi.

2.1 Fungsi Konversi Input Gambar

Konversi input gambar adalah untuk menginputkan gambar/objek yang akan dilakukan proses deteksi tepi. Format yang digunakan adalah berekstensi bitmap (**bmp*). Format ini merupakan format gambar yang memiliki struktur yang masih sempurna sehingga informasi citra yang terkandung pada gambar input lebih lengkap. Berikut adalah fungsi untuk input gambar:

```
procedure TFormMenu.OpenImage1Click(Sender: TObject);
begin
  OpenPictureDialog.Execute;
  FileNameImageInput:=OpenPictureDialog.FileName;
  if(FileNameImageInput<>'')then
    begin
FormImageInput.ImageInput.Picture.Bitmap.LoadFromFile(FileNameImageInput);
    FormImageInput.Visible:=true;
    BitmapImageInput:=FormImageInput.ImageInput.Picture.Bitmap;

    case BitmapImageInput.PixelFormat of
    pf1bit : begin
              MatriksImageInput:=ImageBinerToMatriks(BitmapImageInput);
             end;
    pf8bit : begin
              MatriksImageInput:=ImageGreyToMatriks(BitmapImageInput);
             end;
    pf24bit : begin
              MatriksRImageInput:=ImageRGBToMatriksR(BitmapImageInput);
              MatriksGImageInput:=ImageRGBToMatriksG(BitmapImageInput);
              MatriksBImageInput:=ImageRGBToMatriksB(BitmapImageInput);
              end;
end;
```

Kode Program 1. Fungsi konversi Input Gambar

Citra bitmap diinputkan dari komputer dan pc user. Jadi aplikasi ini tidak perlu menggunakan database sebagai tempat penyimpanan gambar. Setelah itu dilakukan penghitungan pixel untuk mencari perhitungan matrik yang akan digunakan sebagai acuan pada metode masing-masing deteksi tepi.

2.2 Fungsi Konversi Deteksi *Robert*

Konversi deteksi *Robert* merupakan metode yang digunakan sebagai pembanding pertama dalam penelitian ini. Berikut adalah fungsi *Robert Edge Detection:*

```
function Image*Robert*(MInput:Matriks):Matriks;
var
```

```
  i,j: integer;
  MOutput : Matriks;
  kernel : integer;
begin
  SetLength(MOutput,Length(MInput)-1,Length(MInput[0])-1);
  for i:=0 to (Length(MOutput)-2) do
    begin
      for j:=0 to (Length(MOutput[0])-2) do
        begin
         Kernel:=Abs(MInput[i,j]-MInput[i+1,j+1])+Abs(Minput[i,j+1]-
MInput[i+1,j]);
         If kernel<0 then kernel:=0;
         If kernel>255 then kernel:=255;

        MOutput[i,j]:=kernel;
        end;
    end;
  Image Robert:=MOutput;
end;
```

Kode Program 2. Fungsi Konversi Deteksi *Robert*

Kode Program 2diawali dengan fungsi *ImageRobert* untuk pemanggialan dari matriks yang akan dilakukan sebagai pengitungan metode ini. Dan melakukan perintah *If kernel*sebagai batas akhir *lupping* maksimal yakni 255. Pemanggilan pada procedure *Robert* ini terlihat bagaimana cara pembuatan sebuah matrik yangterdapat pada algoritma *Robert* dengan menggunakan fungsi array yang terdapatpada Delphi.

2.3 Fungsi Konversi Deteksi *Sobel*

Konversi deteksi *Sobel* merupakan metode yang digunakan sebagai pembanding kedua dalam penelitian ini. Berikut adalah fungsi*Sobel Edge Detection*:

```
function ImageSobel(MInput:Matriks):Matriks;
var
  i,j: integer;
  gx,gy:integer;
  MOutput : Matriks;
  kernel : integer;
begin
  SetLength(MOutput,Length(MInput)-1,Length(MInput[0])-1);
  for i:=0 to (Length(MOutput)-2) do
    begin
      for j:=0 to (Length(MOutput[0])-2) do
        begin

         gx:=(MInput[i,j]+2*MInput[i,j+1]+MInput[i,j+2])-
(MInput[i+2,j]+2*MInput[i+2,j]+MInput[i+2,j+2]);
         gy:=(MInput[i,j]+2*MInput[i+1,j]+MInput[i+2,j])-
(MInput[i,j+2]+2*MInput[i+1,j+2]+MInput[i+2,j+2]);
         kernel:=round(sqrt(power(gx,2)+power(gy,2)));
 If kernel<0 then kernel:=0;
         If kernel>255 then kernel:=255;

        MOutput[i,j]:=kernel;
        end;
    end;
  ImageSobel:=MOutput;
```

```
end;
```

Kode Program 3. Fungsi Konversi Deteksi *Sobel*

Kode Program 3 diawali dengan fungsi *ImageSobel* untuk pemanggialan dari matriks yang akan dilakukan sebagai pengitungan metode ini. Dan melakukan perintah *If kernel*sebagai batas akhir *lupping* maksimal yakni 255. Operator *Sobel* ini adalah kemampuan untuk mengurangi noise sebelum melakukan perhitungan deteksi tepi. Sehingga besar gradient dapat di hitung. Biasanya operator *Sobel* menempatkan penekanan atau pembobotan pada piksel-piksel yang lebih dekat dengan titik pusat jendela, sehingga pengaruh piksel-piksel tetangga akan berbeda sesuai dengan letaknya terhadap titik di mana gradien dihitung. Dari susunan nilai-nilai pembobotan pada jendela juga terlihat bahwa perhitungan terhadap gradien juga merupakan gabungan dari posisi mendatar dan posisi vertikal.

2.4 Fungsi Konversi Deteksi *Canny*

Konversi deteksi *Canny* merupakan metode yang digunakan sebagai pembanding ketiga dalam penelitian ini. Berikut adalah fungsi*Canny Edge Detection*:

```
function CannyGaussSobel(ImageGauss : Matriks; upperThreshold : Integer;
lowerThreshold : Integer):Matriks;
var
  FilterGx, FilterGy,ImageGaussSobel,AngleMatriks,NewMatriks : Matriks;
  rowFilter,colFilter,summGx,summGy,angle,row,col : integer;
  ValueSobel,temp,thisAngle : real;
  edgeEnd : Boolean;

begin
  SetLength(FilterGx,3,3);
  SetLength(FilterGy,3,3);

  // Gx
  FilterGx[0,0] := -1; FilterGx[1,0] := 0; FilterGx[2,0] := 1;
  FilterGx[0,1] := -2; FilterGx[1,1] := 0; FilterGx[2,1] := 2;
  FilterGx[0,2] := -1; FilterGx[1,2] := 0; FilterGx[2,2] := 1;
  // Gy
  FilterGy[0,0] := 1; FilterGy[1,0] := 2; FilterGy[2,0] := 1;
  FilterGy[0,1] := 0; FilterGy[1,1] := 0; FilterGy[2,1] := 0;
  FilterGy[0,2] := -1; FilterGy[1,2] := -2; FilterGy[2,2] := -1;

  SetLength(ImageGaussSobel,Length(ImageGauss)-2,Length(ImageGauss[0])-2);
  SetLength(AngleMatriks,Length(ImageGauss)-2,Length(ImageGauss[0])-2);

  for row:= 0 to (Length(ImageGaussSobel)-1) do
  begin
    for col:= 0 to (Length(ImageGaussSobel[0])-1) do
    begin
        summGx := 0;
        summGy := 0;
        for rowFilter:= 0 to 2 do
        begin
          for colFilter:= 0 to 2 do
          begin
            summGx :=
summGx+((ImageGauss[row+rowFilter,col+colFilter])*(FilterGx[rowFilter,colF
ilter]));
            summGy :=
summGy+((ImageGauss[row+rowFilter,col+colFilter])*(FilterGy[rowFilter,colF
ilter]));
```

```
          end
        end;
        //ShowMessage(IntToStr(summ));
        ValueSobel := round(sqrt(POWER(summGx,2) + POWER(summGy,2)));
        if ValueSobel>255 then temp:=255
            else temp:=ValueSobel;

        // calculate angle
        thisAngle := round((ArcTan2(summGx,summGy)/3.14159) * 180.0);
        if (((thisAngle < 22.5) and (thisAngle > -22.5)) or ((thisAngle >
157.5) and  (thisAngle < -157.5)) ) then
          angle := 0;
        if (((thisAngle > 22.5) and (thisAngle < 67.5)) or ((thisAngle < -
112.5) and  (thisAngle > -157.5)) ) then
          angle := 45;
        if (((thisAngle > 67.5) and (thisAngle < 112.5)) or ((thisAngle <
-67.5) and  (thisAngle > -112.5)) ) then
          angle := 90;
        if (((thisAngle > 112.5) and (thisAngle < 157.5)) or ((thisAngle <
-22.5) and  (thisAngle > -67.5)) ) then
          angle := 135;
        // tampung ke matrik angle
        AngleMatriks[row,col] := angle;
        // set value to matrik output from gauss with Sobel
        ImageGaussSobel[row,col] := round(temp);
    end;
  end;
CannyGaussSobel:= ImageGaussSobel;
end;
```

Kode Program 4. Fungsi konversi Deteksi *Canny*

Kode Program 4 menggunakan perintah *function CannyGaussSobel*untuk mengkonversi citra *grayscale* menjadi citra biner. Melakukan perintah *Filter.Gx* untuk menghilangkan noise yang ada pada gambar lalu dilakukan perhitungan matriks.

2.5 Fungsi *Output* Gambar

Konversi *output*gambar adalah untuk *form* untuk mengeluarkan otput dari masing-masing hasil dari ketiga metode yang telah dilakukan proses deteksi tepi. Format yang dihasilkan adalah berekstensi bitmap (*bmp). Berikut *script* dari fungsi *output* gambar.

```
unit UnitImageOutput;

interface

uses
  Windows,  Messages,  SysUtils,  Variants,  Classes,  Graphics,  Controls,
Forms,
  Dialogs, ExtCtrls;

type
  TFormImageOutput = class(TForm)
    ImageOutput: TImage;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  FormImageOutput: TFormImageOutput;

implementation

{$R *.dfm}

end.
```

Kode Program 5.Fungsi Output Gambar

Kode Program 5 menggunakan variabel dari *FormImageOutput: TFormImageOutput* yang digunakan sebagi pemanggilan dari hasil deteksi tepi perbandingan metode.

3. Kajian Pustaka

3.1 Pengolahan Citra Digital

Citra sebagai salah satu komponen multimedia memegang peranan sangat penting sebagai bentuk informasi visual. Citra mempunyai karakteristik yang tidak dimiliki oleh data teks, yaitu citra kaya akan informasi. Maksud dari "citra kaya akan informasi" adalah citra dapat memberikan informasi yang lebih banyak dibandingkan dengan informasi yang disajikan dalam bentuk teks.

Secara sederhana, citra adalah image pada bidang dwimatra (dua dimensi). Sumber cahaya menerangi objek, objek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh alat-alat optik, misalnya mata pada manusia, kamera, pemindai (scanner), dan sebagainya, sehingga bayangan objek yang disebut citra tersebut terekam [1].

.

3.2 *Pixel*

Pixel merupakan elemen terkecil dari suatu citra digital yang dihitung secara detail. *Pixel* sendiri berasal dari akronim bahasa Inggris yaitu*Picture Element* yang disingkat menjadi *Pixel. Pixel* tersusun secara berdekatan dengan teratur sehingga terbentuk suatu citra. Semakin banyak dan rapat susunan pixel dalam suatu citra maka semakin tajam citra yang diperoleh. Suatu *pixel* memiliki rentang nilai tertentu. Rentang nilai setiap *pixel* berbeda-beda, hal ini disebabkan dari ruang warna yang digunakan. Secara umum citra digital menggunakan format ruang warna RGB (*red, green, blue*) dimana ruang warna RGB memiliki rentang nilai 0 sampai dengan 255 [1][3].

3.3 Citra Grayscale

Proses konversi citra bewarna ke citra *grayscale* dapat dilakukan dengan persamaan berikut:

$$I(x,y) = \frac{R+G+B}{3} \quad \text{.....(1)}$$

Variabel *I* adalah nilai (tingkat) warna *grayscale* pada posisi *(x,y)* sedangkan *R*, *G*, dan *B* berturut-turut menyatakan nilai komponen ruang warna *R* (merah), *G* (hijau), dan *B* (biru) dari nilai setiap piksel citra bewarna pada posisi (x, y). Jadi dapat dikatakan bahwa citra *grayscale* merupakan nilai rata-rata dari RGB setiap piksel [3].

.

3.4 Citra Bitmap

Citra bitmap sering juga disebut sebagai citra raster atau DIBs (Device Independen Bitmap) merupakan format citra setandar pada sistem operasi Microsoft Windows dan IBM OS/2 (Key,2002). Pada citra dengan format bitmap, data-data citra disimpan pada setiap pixelnya. Citra bitmap dipresentasikan dalam bentuk matriks.

Ukuran sebenarnya untuk n-bit (2n warna) bitmap dalam byte dapat dihitung:

$$\approx 54 + 4 \cdot 2^n + \frac{lebar \cdot tinggi \cdot n}{8} \quad \text{.....(2)}$$

Citra bitmap mampu mendukung warna dari monochrome sampai dengan true color (4,3 milyar warna). Dengan kemampuan pendukung warna yang begitu besar maka tipe bitmap

sangat mendukung dalam pengolah warna, namun tipe bitmap sangat lemah dalam pengolahan objek karena bitmap tidak mendukung pengolahan objek berbasis vektor.

Bitmap atau yang sering dikenal dengan BMP adalah representasi dari citra grafis yang terdiri dari susunan titik yang tersimpan di memori komputer. Untuk menampilkan citra BMP pada monitor atau mencetaknya pada printer, komputer menerjemahkan BMP ini menjadi pixel (pada layar) atau titik tinta (pada printer).

Struktur dari file BMP dibagi menjadi 4 bagian yaitu bagian pertama adalah header, kemudian diikuti oleh bagian informasi, dilanjutkan oleh bagian-bagian warna dan yang terakhir adalah bagian pixel [3].

3.5 *Image Processing*

Image processing adalah suatu metode yang digunakan untuk memproses ataumemanipulasi gambar dalam bentuk 2 dimensi. *Image processing* adalah suatu bentuk pengolahan atau pemrosesan sinyal dengan input berupa gambar *(image)* dan ditransformasikan menjadi gambar lain sebagai keluarannya dengan teknik tertentu. *Image processing* dilakukan untuk memperbaiki kesalahan data sinyal gambaryang terjadi akibat transmisi dan selama akuisisi sinyal, serta untuk meningkatkan kualitas penampakan gambar agar lebih mudah diinterpretasi oleh sistem penglihatan manusia baik denganmelakukan manipulasi dan juga penganalisisan terhadap gambar. *Image processing* dapat juga dikatakan sebagai segala operasi untuk memperbaiki, menganalisa, atau mengubah suatu gambar. Konsep dasar pemrosesan suatu objek pada gambarmenggunakan *image processing* diambil dari kemampuan indera penglihatan manusia yang selanjutnya dihubungkan dengan kemampuan otak manusia [3].

3.6 Citra Biner

Citra biner adalah citra yang memiliki nilai warna 0 dan 255.Citra biner akan terlihat hanya dengan warna hitam dan putih saja [1][3]. Untuk mengubah citra warna menjadi citra biner, maka citra warna RGB dirubah terlebih dahulu menjadi citra *grayscale*. Nilai warna pada citra *grayscale* kemudian ditentukan dengan nilai acuan (*threshold*). Jika nilai *grayscale* citra lebih kecil dari*threshold,* maka nilainya diubah menjadi 0 (warna hitam). Sedangkan jika nilai *grayscale* citra lebih besar atau sama dengan *threshold* maka nilainya diubah menjadi 255 (warna putih).

3.7 Deteksi Tepi (*Edge*)

Tepi (*edge*) adalah perubahan nilai intensitas derajat keabuan yang mendadak besar dalam jarak yang dekat. Suatu titik (x,y) dikatakan sebagai tepi bila titik tersebut mempunyai perbedaan nilai piksel yang tinggi dengan nilai piksel tetangganya [3]. Perubahan mencapai maksimum pada saat nilai turunan pertamanya mencapai nilai maksimum atau nilai turunan kedua (2^{nd} *derivative*) bernilai 0. Deteksi tepi (*Edge detection*) adalah operasi yang dijalankan untuk mendeteksi garis tepi (*edges*) yang membatasi dua wilayah citra homogen yang memiliki tingkat kecerahan yang berbeda. Deteksi tepi pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra, tujuannya adalah :

a. Untuk menandai bagian yang menjadi detail citra
b. Untuk memperbaiki detail dari citra yang kabur, yang terjadi karena error atau adanya efek dari proses akuisisi citra.
c. Serta untuk mengubah citra 2D menjadi bentuk kurva

Suatu titik (x,y) dikatakan sebagai tepi (*edge*) dari suatu citra bila titik tersebutmempunyai perbedaan yang tinggi dengan tetangganya [4].

3.8 Dekteksi Tepi *Canny*

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.

Deteksi Tepi *Canny* dapat mendeteksi tepian yang sebenarnya dengan tingkat *error* yang minimum dengan kata lain, operator *Canny* didesain untuk menghasilkan citra tepian yang optimal [2].

3.9 Deteksi Tepi *Robert*

Operator *Robert* adalah nama lain dari teknik differensial yang sedang dikembangkan,yaitu differensial pada arah horisontal dan differensial pada arah vertikal, denganditambahkan proses konversi biner setelah dilakukan differensial. Teknik konversi bineryang disarankan adalah konversi biner dengan meratakan distribusi warna hitam danputih.Operator *Robert* ini juga disamakan dengan teknik DPCM *(Differential Pulse Code Modulation*). Operator *Robert* Cross merupakan salah satu operator yang menggunakan jendela matrik 2x2, operator ini melakukan perhitungan dengan mengambil arah diagonal untuk melakukan perhitungan nilai gradiennya.

Sebenarnya operator sedehana ini hanya memeriksa sebuah piksel tambahan pada satu arah gradient tetapi karena yang diperiksa adalah piksel dalam arah diagonal, maka secara keseluruhan piksel-piksel yang terlibat membentuk jendela matrik 2x2. Bentuk jendela yang demikian lebih menekankan pemeriksaan pada kedua arah diagonal, dari pada arah horizontal atau arah vertikal, sehingga perbedaan yang terletak pada sisi-sisi miring objek akan terdeteksi dengan lebih baik [2].

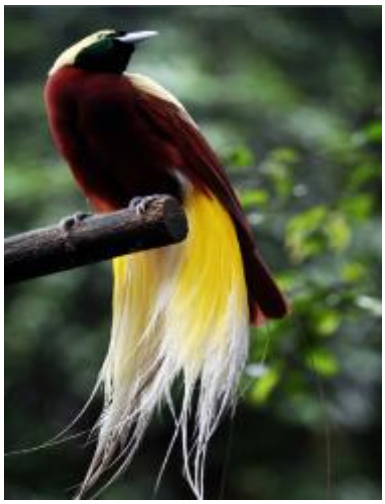
3.10 Deteksi Tepi *Sobel*

Metode *Sobel* merupakan pengembangan metode *Robert* dengan menggunakan *filter* HPF yang diberi satu angka nol penyangga. Metode ini mengambil prinsip dari fungsi laplacian dan gaussian yang dikenal sebagai fungsi untuk membangkitkan HPF. Kelebihan dari metode *Sobel* ini adalah kemampuan untuk mengurangi noise sebelum melakukan perhitungan deteksi tepi.

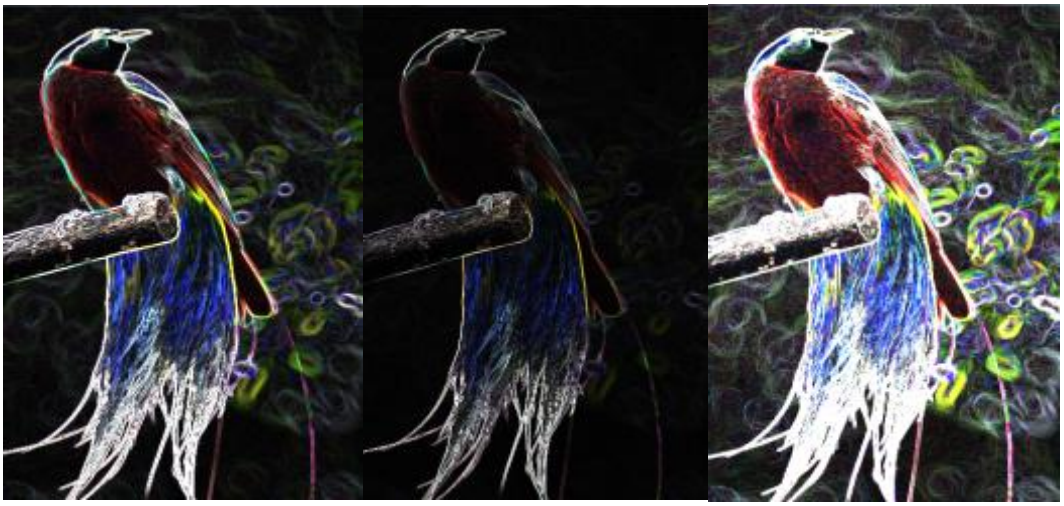
Biasanya operator *Sobel* menempatkan penekanan atau pembobotan pada piksel-piksel yang lebih dekat dengan titik pusat jendela, sehingga pengaruh piksel-piksel tetangga akan berbeda sesuai dengan letaknya terhadap titik di mana gradien dihitung. Dari susunan nilai-nilai pembobotan pada jendela juga terlihat bahwa perhitungan terhadap gradien juga merupakan gabungan dari posisi mendatar dan posisi vertikal [5].

4. Hasil dan Pembahasan

Berdasarkan pada fungsi yang telah dibahas pada metode penilitian, maka dihasilkan konversi citra yang diaplikasikansebagai berikut:



(a)



(b) (c) (d)

Gambar 1.Hasil pengolahan citra digital (a) Citra Input, (b) Output Deteksi Tepi *Robert*, (c) Output Deteksi Tepi *Sobel*, (d) Output Deteksi Tepi *Canny*

Gambar 1 menunjukkan aplikasi Delphi sederhana untuk melakukan proses pengolahan citra digital. Gambar 1 (a) merupakan Input dari gambar yang akan dilakukan proses penghitungan. Hasil pada Gambar 1 (a) telah menunjukkan citra grayscale yang benar. Gambar 1 (b) merupakan hasil dari konversimetode deteksi tepi *Robert*. Gambar 1 (c) merupakan hasil dari konversi metode deteksi tepi *Sobel*. Gambar 1 (d) merupakan hasil dari konversi metode deteksi tepi *Canny*.

5. Kesimpulan

Setelah dilakukan pengujian terhadapat aplikasi pengolahan citra digital deteksi tepi dan mendapatkan analisis dari beberapa percobaan yang dilakukan dapat disimpulkan bahwa metode *Canny* dapat melakukan penghitungan deteksi tepi lebih sempurna dibandingkan dengan metode *Robert* dan *Sobel*. Pada hasil dari koversi *Canny* terlihat tepi dan batas-batas pada citra output sangat jelas terlihat.

Daftar Pustaka

[1] Darma Putra. Pengolahan Citra Digital. Yogyakarta: CV. Andi Offset. 2010.
[2] Septian Dwi Cahyo. Analisis Perbandingan Tepi Pada Delphi. Teknik Informatika, Fakultas Teknologi Industri, Universitas Gunadarma.
[3] Dewi, Lilyana. Perencanaan Dan Pembuatan Aplikasi Untuk Transfer Warna Ke Gambar Greyscale Dengan Metode Global Image Matching. Teknik Informatika S1, Universitas Kristen Petra. 2003.
[4] Subchan Ajie Ari Bowo. Anlisis Deteksi Tepi Untuk Mengidentifikasi Pola Daun. Jurusan Teknik Elektro Fakultas Teknik Universitas Diponegoro.
[5] http://animasikomputer.blogspot.com/, diakses tanggal 28 Juni 2014.