PERANGKAT LUNAK EDGE DETECTION DENGAN ATURAN CELLULAR AUTOMATA

Swingly Purba, Rikardo H. Siahaan dan Galuh Ramadhani

Prodi Teknik Informatika, Fakultas Teknologi Industri, Institut Sains Dan Teknologi T.D. Pardede
Jalan DR.T.D. PardedeNo. 8, Medan 20153, Sumatera Utara

Email : swinglypurba@istp.ac.id
rikardosiahaan@istp.ac.id
ramadhanigaluh10@gmail.com,

ABSTRAK

Edge detection adalah suatu proses untuk menemukan dan mendeteksi batas-batas dalam suatu citra baik citra berwarna atau citra *grayscale* dengan menggunakan suatu operator tertentu sehingga menghasilkan suatu citra dalam warna hitam dan putih (*black and white*). Warna hitam dan putih tersebut menunjukkan batas antara latar dan objek dalam citra. Salah satu operator yang dapat digunakan untuk menghasilkan tepi adalah operator Prewitt, dimana dalam tulisan ini, operator ini digunakan sebagai pembanding dengan aturan *cellular automata*.Cara kerja aturan *cellular automata* dalam mendeteksi tepi adalah dengan mengubah nilai aturan yang diinput ke bentuk biner 10 bit, lalu dilakukan proses perhitungan jumlah tetangga yang berwarna hitam pada setiap pixel. Dari jumlah tetangga itulah, maka diperoleh apakah titik tersebut merupakan tepi atau bukan.Hasil dari tulisan ini adalah sebuah program yang mampu melakukan deteksi tepi pada citra dengan menggunakan aturan *cellular automata* yang hasilnya diperoleh berdasarkan input nilai yang ditentukan, kemudian diperoleh hasil perbandingan waktu eksekusi antara operator Prewitt dengan aturan *cellular automa.*

Kata Kunci : Aturan Cellular Automata

ABSTRACT

Edge detection is a process for finding and detecting boundaries in an image, either a color image or a grayscale image using a certain operator so as to produce an image in black and white (black and white). The black and white colors indicate the boundary between the background and the object in the image. One operator that can be used to generate edges is the Prewitt operator, which in this paper is used as a comparison with cellular automata rules. The way cellular automata rules work in detecting edges is by changing the value of the input rule to 10-bit binary form, then the process of calculating the number of neighbors that are black in each pixel. From that number of neighbors, it is obtained whether the point is an edge or not. The result of this paper is a program that is capable of performing edge detection in images using cellular automata rules whose results are obtained based on the specified input value, then the results of the comparison of execution time between Prewitt operator with cellular automa rules.

Keywords: *Cellular Automata Rules*

1. PENDAHULUAN

Edge Detection yang termasuk dalam terminologi pemrosesan citra dan *computer vision*, khususnya pada bidang *feature detection* dan *feature extraction*, bertujuan untuk memperoleh algoritma yang mengidentifikasi titik-titik pada citra digital dimana kecerahan gambar mengalami perubahan tajam atau *discontinuities*. Dengan mendeteksi perubahan kecerahan gambar, maka akan diperoleh tepi dari citra yang dapat dimanfaatkan untuk menandai bagian yang menjadi detail citra atau untuk memperbaiki detail dari citra yang kabur yang terjadi akibat efek akuisisi citra.

Manfaat dari penulisan tugas akhir ini adalah memahami proses *edge detection* dengan aturan *cellular automata*, menghasilkan perangkat lunak untuk melakukan *edge detection* dengan aturan *cellular automata* dan memperoleh batas-batas objek pada citra input serta memperoleh perbandingan ketajaman tepi dan waktu eksekusi antara aturan *cellular automata* dengan operator prewitt.

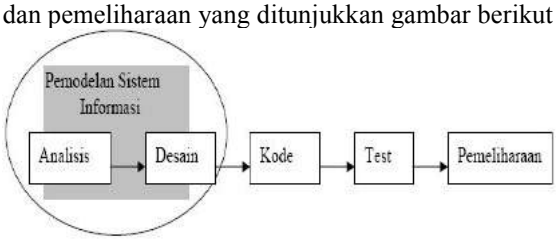
2. METODE PENELITIAN

Perangkat lunak yang dapat melakukan *edge detection* terhadap citra input dengan menggunakan metode aturan *cellular automata* dan membandingkannya dengan salah satu operator *edge detection* yaitu operator Prewitt.

Terdapat beberapa pendekatan yang dapat diambil dalam mengembangkan suatu perangkat lunak, diantaranya metode sekuensial linier, model prototipe, model RAD, model inkremental, model spiral, dan metode lainnya.

A. Model Sekuensial Linier

Model sekuensial linier untuk *software engineering,* sering disebut juga dengan siklus kehidupan klasik atau model air terjun (*Waterfall*). Model ini mengusulkan sebuah pendekatan kepada perkembangan *software* yang sistematik dan sekuensial yang dimulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan yang ditunjukkan gambar berikut :



Gambar 1 : Model Sekuensial Linier

Dimodelkan setelah siklus rekayasa konvensional, model sekuensial linier melingkupi aktivitas – aktivitas sebagai berikut :

1. Rekayasa dan pemodelan sistem/informasi
   Karena sistem merupakan bagian dari sebuah sistem yang lebih besar, kerja dimulai dengan membangun syarat dari semua elemen sistem dan mengalokasikan beberapa *subset* dari kebutuhan ke *software* tersebut. Pandangan sistem ini penting ketika *software* harus berhubungan dengan elemen-elemen yang lain seperti *software*, manusia, dan database. Rekayasa dan analisis sistem menyangkut pengumpulan kebutuhan pada tingkat sistem dengan sejumlah kecil analisis serta desain tingkat puncak. Rekayasa informasi mencakup juga pengumpulan kebutuhan pada tingkat bisnis strategis dan tingkat area bisnis.
2. Analisis Kebutuhan *Software*
   Proses pengumpulan kebutuhan diintensifkan dan difokuskan, khususnya pada *software*. Untuk memahami sifat program yang dibangun, analis harus memahami domain informasi, tingkah laku, unjuk kerja, dan *interface* yang diperlukan. Kebutuhan baik untuk sistem maupun *software* didokumentasikan dan dilihat lagi dengan pelanggan.
3. Desain
   Desain *software* sebenarnya adalah proses multi langkah yang berfokus pada empat atribut sebuah program yang berbeda, yaitu struktur data, arsitektur *software*, representasi *interface*, dan detail (algoritma) prosedural. Proses desain menerjemahkan syarat/kebutuhan ke dalam sebuah representasi *software* yang dapat diperkirakan demi kualitas sebelum dimulai pemunculan kode. Sebagaimana persyaratan, desain didokumentasikan dan menjadi bagian dari konfigurasi *software*.
4. Generasi Kode
   Dalam tahap pembuatan kode, tugas yang dilakukan adalah mengubah desain yang telah dirancang sebelumnya ke dalam bahasa mesin yang bisa dibaca. Jika desain dilakukan dengan cara yang lengkap, pembuatan kode dapat diselesaikan secara mekanis.
5. Pengujian
   Sekali program dibuat, pengujian program dimulai. Proses pengujian berfokus pada logika *internal software*, memastikan bahwa semua pernyataan sudah diuji, dan pada *eksternal* fungsional, yaitu mengarahkan pengujian untuk menemukan kesalahan–kesalahan dan memastikan bahwa input yang dibatasi akan memberikan hasil aktual yang sesuai dengan hasil yang dibutuhkan.
6. Pemeliharaan
   Software akan mengalami perubahan setelah disampaikan kepada pelanggan (perkecualian yang mungkin adalah *software* yang dilekatkan). Perubahan akan terjadi karena kesalahan yang ditemukan, karena *software* harus disesuaikan untuk mengakomodasi perubahan–perubahan di

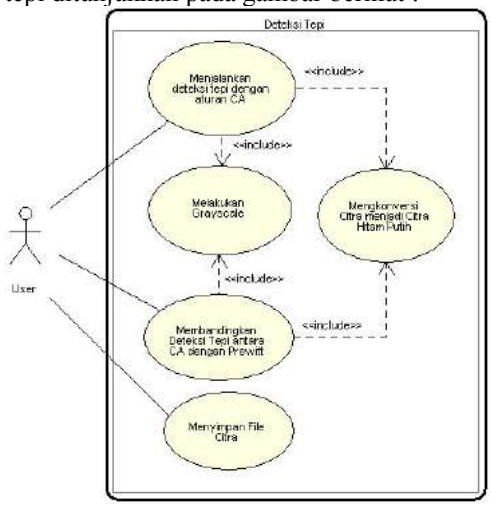
dalam lingkungan *eksternal*nya (contohnya perubahan yang dibutuhkan sebagai akibat dari perangkat *peripheral* atau sistem operasi yang baru), atau karena pelanggan membutuhkan perkembangan fungsional atau unjuk kerja. Pemeliharaan *software* mengaplikasikan lagi setiap fase program sebelumnya dan tidak membuat program yang baru lagi.

3. ANALISIS DAN PERANCANGAN SISTEM

Analisis persyaratan terhadap perangkat lunak yang akan dirancang mencakup analisis fungsional yang menjabarkan persyaratan fungsional yang harus dipenuhi oleh perangkat lunak dan analisis non fungsional yang menjabarkan persyaratan non fungsional dari perangkat lunak yang akan dirancang.

A. Analisis Persyaratan Fungsional

Untuk mengetahui persyaratan fungsional dari sistem yang dirancang, penulis menggunakan use case diagram. Use case diagram untuk aplikasi deteksi tepi ditunjukkan pada gambar berikut :



Gambar 2 : Diagram Use Case

B. Analisis Persyaratan Non Fungsional

Persyaratan Non Fungsional menjelaskan mengenai fungsi-fungsi pendukung perangkat lunak. Persyaratan non fungsional dapat di analisis dengan menggunakan kerangka PIECES yaitu:

1. *Performance*
   Aplikasi dapat mendeteksi tepi dari citra yang diinput oleh user baik citra warna maupun citra hitam putih.
2. *Information*
   Aplikasi dapat menunjukkan hasil dari pengolahan gambar berupa posisi tepi dari citra yang diinput dengan menggunakan aturan *cellular automata* dan Operator Prewitt.
3. *Economy*
   Tidak memerlukan aplikasi tambahan untuk menjalankan aplikasi ini.
4. *Control*
   Proses deteksi tepi akan diproses jika telah diinput citra yang akan diproses dan nomor aturan yang digunakan antara 0 sampai 1023 dan jika data tidak lengkap, maka akan muncul pesan *error*.
5. *Efficiency*
   Aplikasi menyediakan menu untuk membandingkan hasil eksekusi deteksi tepi antara aturan *cellular automata* dengan operator Prewitt secara visual dan waktu eksekusi yang diperlukan.
6. *Service*
   Aplikasi dirancang dengan menggunakan Microsoft Visual Basic .NET 2008 yang memiliki *interface* berbasis windows sehingga aplikasi mudah untuk digunakan.

C. Analisis Proses

Proses Utama pada aplikasi yang dirancang ada 2 yaitu proses *run* dan proses *compare*. Untuk mengetahui alur kerja atau logika proses dari kedua proses tersebut, penulis menggunakan *Activity Diagram*. *Activity Diagram* yang menunjukkan proses *run* dapat dilihat pada gambar 3 berikut :

Gambar 3 : Diagram untuk proses Run

Activity Diagram yang menunjukkan proses *compare* dapat dilihat pada gambar 4 berikut.

Gambar 4 : Diagram untuk proses Compare

4. IMPLEMENTASI DAN HASIL

a) Tampilan Program

Ketika program dijalankan, *form* utama akan menampilkan menu seperti gambar 10 berikut.

Pada *form* utama terdapat pilihan menu *run*, *compare*, *information,* dan *exit*. Jika dipilih menu *run* maka akan muncul *form run* yang tampilannya dapat dilihat pada gambar berikut.

Gambar 11 : Form Run

Setelah *form run* muncul, *user* diminta untuk memilih gambar yang akan diolah kemudian menginput nilai aturan yang akan digunakan pada tempat yang sudah disediakan. Tampilan *form* setelah dipilih gambar dan diinput nilai aturan dapat dilihat pada gambar berikut.

Gambar 12 : Form Run setelah diinput gambar dan diinput nilai atuuran

Setelah *user* telah menginput gambar dan nilai aturan, dengan memilih tombol *Go* maka proses *run* akan dijalankan. Jika proses *run* selesai, maka hasil deteksi tepi akan ditampilkan pada *picturebox*, hasil

dari deteksi tepi dapat dilihat pada gambar berikut :

Gambar 13 : Hasil deteksi tepi dengan aturan 990

Setelah proses *run* selesai, dengan memilih salah satu langkah pada *listbox* yang berada di posisi tengah, maka keterangan dari langkah yang dipilih akan ditampilkan pada *listbox* yang disebelah kanan seperti gambar berikut.

Gambar 14 : Tampilan form ketika dipilih langkah 1

Jika langkah 2 yang dipilih, maka tampilan *form* dapat dilihat pada gambar 15 :

Gambar 15 : Tampilan Form ketika dipilih langkah 2

Jika yang dipilih langkah 3, maka hasilnya dapat dilihat pada gambar 16

Gambar 16 : Tampilan form ketika dipilih langkah 3

Dengan input gambar yang sama dan nilai aturan yang berbeda, dalam hal ini invers dari nilai aturan 990 (1111011110) yaitu 33 (0000100001) maka diperoleh hasil deteksi tepi seperti gambar 4.8 berikut.

Gambar 17 : Hasil deteksi tepi dengan aturan 33

Jika menu *Compare* dipilih pada menu utama, maka akan muncul *form Compare* yang tampilannya dapat dilihat pada gambar 18 berikut.

Gambar 18 : Form Compare

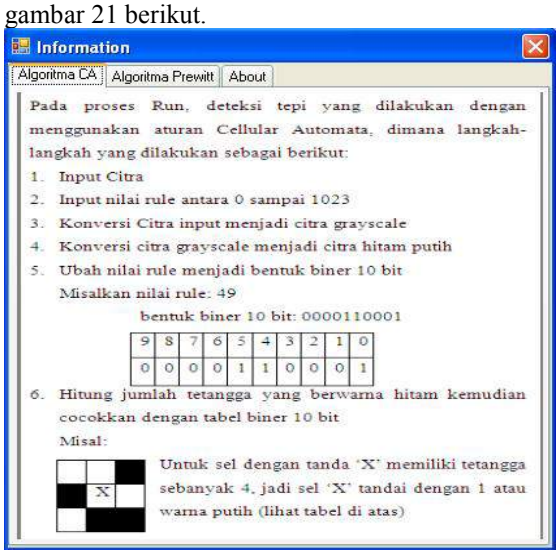
Setelah *form Compare* muncul, *user* diminta untuk memilih file gambar yang akan diproses dengan memilih tombol Open kemudian menentukan nilai aturan yang akan digunakan untuk proses deteksi tepi dengan aturan *cellular automata*. Tampilan *form* setelah diinput gambar dan nilai aturan dapat dilihat pada gambar 19 berikut.

Gambar 19 : *Form Compare* setelah dipilih gambar dan diinput nilai aturan

Setelah *user* telah menginput gambar dan nilai aturan, dengan memilih tombol *Go* maka proses *compare* akan dijalankan. Jika proses *compare* selesai, maka hasil deteksi tepi akan ditampilkan pada *picturebox*, hasil perbandingan dari deteksi tepi dapat dilihat pada gambar 20

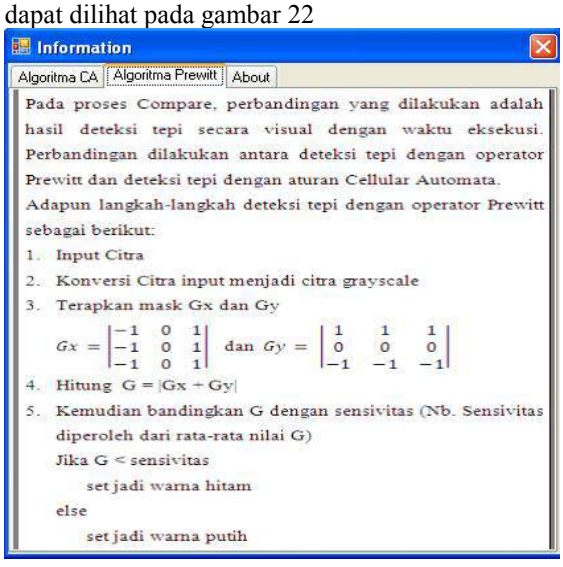
Gambar 20 : *Form Compare* setelah proses *Compare* selesai

Jika menu yang dipilih pada menu utama adalah menu *Information*, maka akan muncul *form Information* yang tampilannya dapat dilihat pada gambar 21 berikut.



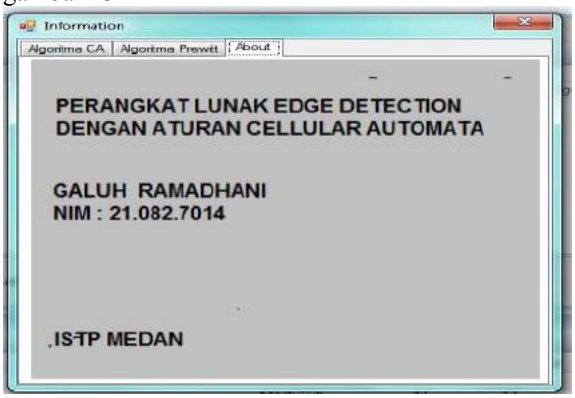
Gambar 21 Tampilan *Form Information* untuk tab Algoritma CA

jika dipilih tab Algoritma Prewitt, maka tampilannya dapat dilihat pada gambar 22



Gambar 22 Tampilan *Form Information* untuk tab Algoritma Prewitt

kemudian untuk tab About, dapat dilihat pada gambar 23



Gambar 23 Tampilan *Form Information* untuk tab About

5. KESIMPULAN DAN SARAN

1) Kesimpulan

Setelah menyelesaikan Tugas Akhir, penulis menarik beberapa kesimpulan sebagai berikut:

1. Perangkat Lunak dapat menghasilkan tepi dengan menginput nilai aturan antara 0 sampai 1023 dimana untuk tiap nilai aturan yang berbeda, dihasilkan keluaran yang berbeda juga.
2. Dari nilai aturan yang diinput, nilai aturan terbagi menjadi 2 kelompok yaitu kelompok I dari 0 sampai 511 dan kelompok II dari 512 sampai 1023. Pengelompokan itu dilakukan berdasarkan hasil deteksi tepinya, dimana hasil kelompok I merupakan invers dari hasil kelompok II.
3. Untuk hasil yang terbaik, nilai aturan yang diinput antara *range* 496 sampai 527 dengan ketentuan nilai aturan antara 0 sampai 511 diinput nilai aturan genap, sedangkan dengan nilai aturan antara 512 sampai 1024 diinput nilai aturan ganjil.
4. Beberapa nilai aturan lain yang menghasilkan *output* yang baik yaitu 30, 60, 126, 254, 480, 899, dan 993.
5. Dilihat dari perbandingan waktu proses deteksi tepi antara aturan *Cellular Automata* dengan Operator Prewitt menunjukkan bahwa waktu eksekusi yang diperlukan untuk aturan *Cellular Automata* lebih lama dibanding deteksi tepi dengan Operator Prewitt.
6. Dilihat dari ketajaman tepi yang dihasilkan, tepi pada *cellular automata* bisa lebih bagus dibanding operator Prewitt dan bisa lebih buruk dibanding operator Prewitt, hal ini dipengaruhi oleh nilai aturan yang digunakan.

2) Saran

Penulis ingin memberikan beberapa saran untuk pengembangan perangkat lunak lebih lanjut sebagai berikut:

1. Sebelum melakukan proses deteksi tepi, citra input dapat diproses dengan *image enhancement* sehingga diperoleh hasil yang lebih baik.
2. Untuk lebih memahami bagaimana cara kerja *cellular automata* terhadap *edge detection*, dapat dibuat simulasi dengan flash.

DAFTAR PUSTAKA

Achmad, B, Firdausy, K. 2005. "*Teknik Pengolahan Citra Digital menggunakan DELPHI*". Yogyakarta: Ardi Publishing.

Goi, H. T. 2003. "*An Original Method of Edge Detection Based on Cellular Automata*", Available: http://*cellular-automata*.um.ac.id/2009/12/an-original - method-of-edge-detection-based-on-*cellular-automata*/ diakses April 2010.

Kusumo, A. S., 2002. "*Visual Basic .NET versi 2002 dan 2003*". Jakarta: Gramedia.

Marr, D, E. Hildreth. 2008. "*Theory of Edge Detection*", Available: http://www.umiacs.umd.edu/~pturaga/ENEE731/papers/Edges/MarrHildreth80.pdf diakses April 2010.

Munir, Rinaldi. 2004. "*Pengolahan Citra Digital dengan Pendekatan Algoritmik*". Bandung: Informatika.

Pengolahan Citra. http://images.moedy9.multiply.multiplycontent.com/attachment /0/SMuuNwoKCBkAAHPHjZk1/Pengolahan%20Citra.pdf?nmid=115281461 diakses Mei 2010.

Pressman, Rogers. 1997. "*Rekayasa Perangkat Lunak Pendekatan Praktisi*". Jakarta: Andi.

Rennard, J.P. 2006. "*Introduction to Cellular Automata*", diambil dari http://www.rennard.org/alife/english/acgb.html diakses April 2010 diakses Maret 2010.