	KEMENTRIAN PENDIDIKAN DAN KEBUDAYAAN UNIVERSITAS BRAWIJAYA FAKULTAS TEKNIK JURUSAN TEKNIK ELEKTRO Jalan MT Haryono 167 Telp & Fax. 0341 554166 Malang 65145	**KODE** **PJ-01**

PENGESAHAN
PUBLIKASI HASIL PENELITIAN SKRIPSI
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK UNIVERSITAS BRAWIJAYA

NAMA : ANGGA SEPTI SUYANTO

NIM : 1150603091110 - 08

PROGRAM STUDI : TEKNIK REKAYASA KOMPUTER

JUDUL SKRIPSI : KONVERSI CITRA LABIRIN KE DALAM EDGE DAN VERTEX

TELAH DI-REVIEW DAN DISETUJUI ISINYA OLEH:

Pembimbing 1	Pembimbing 2
Ir. Muhammad Aswin, MT.	**Adharul Muttaqin, ST., MT.**
NIP. 19640626 199002 1 001	**NIP. 19760121 200501 1 001**

KONVERSI CITRA LABIRIN KE DALAM EDGE DAN VERTEX

Publikasi Jurnal Skripsi

Disusun Oleh :

ANGGA SEPTI SUYANTO

NIM : 1150603091110 - 08

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2014

KONVERSI CITRA LABIRIN KE DALAM EDGE DAN VERTEX

ANGGA SEPTI SUYANTO.[1] , Muhammad Aswin, Ir., MT.[2], Adharul Muttaqin, ST., MT.[2]
[1]Mahasiswa Teknik Elektro Univ. Brawijaya, [2]Dosen Teknik Elektro Univ. Brawijaya
Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya
Jalan MT. Haryono 167, Malang 65145, Indonesia
E-mail: goairhead@ymail.com

Abstract— **The main aim of this research is to develop a system that can be used to process a two-dimensional labyrinth image into a edge and vertex. On an labyrinth image consisting of walls and roads, the system will be able to represent it in the form of edge and vertex that will be used by the search algorithm best path.**

The research was conducted by taking an labyrinth image from files. From that image, then the next process is changing the image into a gray image (grayscale) and then convert it into the form of a binary image . The next operation is to find the midpoint of each wall and the street that would be selected to get the vertex. The next process is checking the two vertices are adjacent to each other to determine that the vertices are interconnected or not and then subtracting the coordinates of the vertices to obtain the weight of the edge. The final result will be a graph that is stored in plaintext form. Testing system by comparing the results of edge and vertex of manual calculations with the results of the system are made. And then, test the final output by load the output data into applications that have been made.

Index Terms — Digital Image Processing, Grayscalling, Threshold, Graph.

Abstrak–- **Penelitian ini bertujuan untuk mengembangkan suatu sistem yang dapat digunakan untuk mengolah suatu bentuk citra labirin dua dimensi kedalam *edge* dan *vertex*. Pada suatu citra labirin yang terdiri dari dinding dan jalan, sistem ini akan dapat merepresentasikannya kedalam bentuk *edge* dan *vertex* yang nantinya digunakan oleh algoritma pencari jalur terbaik.**

Penelitian ini dilakukan dengan cara mengambil suatu citra labirin dari *file*. Dari citra tersebut, kemudian dilakukan perbaikan citra dengan mengubah citra ke dalam citra abu – abu (*grayscale*) dan selanjutnya mengubahnya ke dalam bentuk citra biner atau hitam – putih. Operasi selanjutnya adalah dengan mencari titik tengah dari masing – masing dinding dan jalan yang nantinya akan diseleksi untuk mendapatkan *vertex*nya. Pencarian edge dilakukan melalui pengecekan dua *vertex* yang saling berdekatan untuk mengetahui *vertex* tersebut saling berhubungan atau tidak dan dilanjutkan dengan mengurangkan koordinat *vertex* tersebut untuk memperoleh *weight* dari *edge* tersebut. Hasil akhir berupa *graph* yang nantinya disimpan dalam bentuk *plaintext*. Pengujian sistem dilakukan dengan membandingkan hasil *edge* dan *vertex* dari perhitungan manual dengan hasil dari sistem yang dibuat. Serta melakukan pengujian hasil *output* dengan melakukan *load data output* ke dalam aplikasi yang telah dibuat.

Kata Kunci— Digital Image Processing, Grayscalling, Threshold, Graph.

I. PENDAHULUAN

Labirin merupakan sebuah sistem jalur yang rumit, berliku-liku, serta memiliki banyak jalan buntu. Labirin bisa menjadi permainan di atas kertas, namun dapat juga dibuat dengan skala besar dengan menggunakan tanaman yang cukup besar untuk dilewati. Konsep dari sebuah labirin adalah bagaimana menemukan jalur terbaik antara titik *start* sampai ke titik *finish*.

Secara umum labirin tidak hanya di representasikan dalam suatu bentuk *riil* lorong dan dinding tiga dimensi, tapi juga dalam bentuk citra dua dimensi. Dalam bentuk labirin citra dua dimensi, untuk menemukan jalur terbaik memerlukan suatu metode yang efektif. Beberapa unsur terpenting dalam mengolah suatu labirin dua dimensi seperti : *node* atau titik persimpangan dari pertemuan lorong, jarak antara masing – masing *node* tersebut, serta bentuk koneksi dari *node* – *node* yang ada sehingga tidak masuk kejalan yang salah atau jalan buntu. Untuk dapat mencari unsur – unsur tersebut dalam mencari jalur terbaik dari suatu citra labirin dua dimensi, suatu perangkat komputer memerlukan sebuah metode atau algoritma yang dapat mengolah citra labirin tersebut untuk mendapatkan suatu *edge* dan *vertex*

Digital image processing adalah salah satu metode yang dapat digunakan untuk dapat mengkonversi suatu citra labirin menjadi suatu *edge* dan *vertex* yang nantinya menjadi masukan algoritma pencari jalur terbaik.

Suatu bentuk labirin dua dimensi memerlukan beberapa tahapan proses untuk dapat menemukan *edge* dan *vertex* sebagai input pencarian jalur terbaik. *Grayscalling* dan *Thresholding* adalah tahap perbaikan suatu citra labirin dua dimensi yang nantinya akan membuat citra labirin lebih baik dan dapat memudahkan proses pemecahan citra labirin menjadi *edge* dan *vertex.* Setelah citra diperbaiki barulah algoritma pencarian *edge* dan *vertex* melakukan

prosesnya sehingga terbentuk suatu citra labirin dua dimensi dengan *property* adalah *list node*, pola hubungan antar *node,* serta jarak masing – masing *node* yang terhubung.

II. TINJAUAN PUSTAKA

A. Digital Image Processing

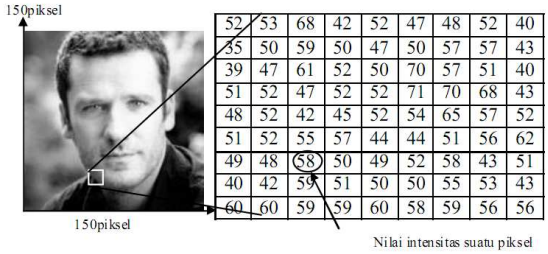
Digital Image Processing adalah proses pengolahan gambar dua dimensi oleh perangkat komputer *digital* . Ada pun menurut Gonzalez dan Woods (*2001,p2-3*), *digital image processing* merupakan proses pengambilan atribut – atribut pada gambar dengan *input* dan *output* yang berupa gambar. *Digital image processing* mempunyai banyak macam aplikasi pada berbagai bidang, seperti : penajaman gambar, pendeteksian objek pada gambar, pengurangan *noise*, konversi gambar berwarna ke *grayscale* dan sebaliknya, kompresi *data* pada gambar, dan sebagainya.

Koversi Citra *True Color* ke *Grayscale*

Citra berwarna diubah menjadi citra *grayscale* dengan mengubah format citra yang awalnya adalah *RGB* menjadi *YUV* lalu diambil *Y*-nya saja. Secara matematis, perhitungan citra *grayscale* akan menjadi :
Keterangan : R = Merah, G = Hijau, B = Biru,

Grayscale = 0.299R + 0.587G + 0.114B

Dengan cara ini, maka citra berwarna akan berubah menjadi *grayscale* tanpa mengubah keaslian dari *RGB*.

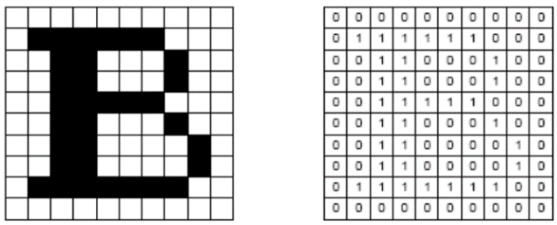


52	53	68	42	52	47	48	52	40
35	50	59	50	47	50	57	57	43
39	47	61	52	50	70	57	51	40
51	52	47	52	52	71	70	68	43
48	52	42	45	52	54	65	57	52
51	52	55	57	44	44	51	56	62
49	48	58	50	49	52	58	43	51
40	42	59	51	50	50	55	53	43
60	60	59	59	60	58	59	56	56

Gambar 1. Citra *Grayscale* 150 x 150 *pixel*

Thresholding

Untuk mengubah citra *grayscale* menjadi citra hitam putih, digunakan metode *thresholding* (*Peters II, 1995*). Caranya adalah dengan menentukan nilai *threshold* yang menjadi batas. Nilai *pixel* yang berada dibawah batas yang ditentukan akan diubah menjadi 0 (hitam) dan nilai *pixel* yang nilainya diatas nilai *threshold* akan diubah menjadi 255 (putih).



0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0	0	0
0	0	1	1	0	0	0	1	0	0
0	0	1	1	0	0	0	1	0	0
0	0	1	1	1	1	1	0	0	0
0	0	1	1	0	0	0	1	0	0
0	0	1	1	0	0	0	0	1	0
0	0	1	1	0	0	0	0	1	0
0	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0

Gambar 2. Citra Biner Hasil *Threshold*

B. *Graph*

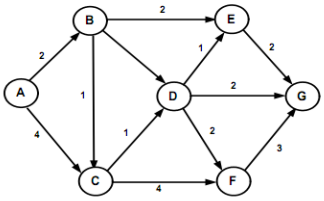
Graph adalah kumpulan *node* (*vertex*) di dalam bidang dua dimensi yang dihubungkan dengan sekumpulan garis (*edge*). *Graph* dapat digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Representasi visual dari *graph* adalah dengan menyatakan objek sebagai bulatan atau titik (*Vertex*), sedangkan hubungan antara objek dinyatakan dengan garis (*Edge*).

G = (V, E)

Dimana
G = *Graph*
V = Simpul atau *Vertex*, atau *Node*, atau Titik
E = Busur atau *Edge*

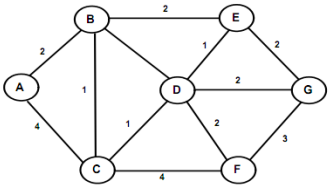
Macam–Macam *Graph* Menurut Arah dan Bobotnya

1. *Graph* berarah dan berbobot



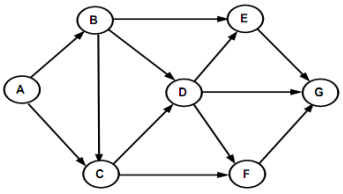
Gambar 3. *Graph* berarah dan berbobot

2. *Graph* tidak berarah dan berbobot



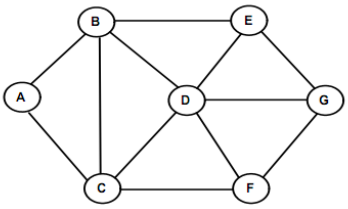
Gambar 4. *Graph* tidak berarah dan berbobot

3. *Graph* berarah tidak berbobot



Gambar 5. *Graph* berarah tidak berbobot

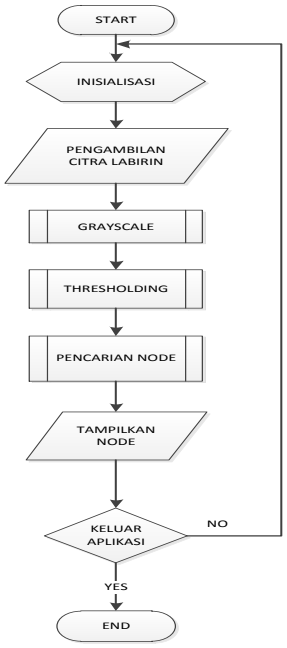
4. *Graph* tidak berarah dan tidak berbobot



Gambar 6. *Graph* tidak berarah tidak berbobot

III. PERANCANGAN DAN IMPLEMENTASI PERANGKAT LUNAK

Sistem konversi labirin ke dalam *edge* dan *vertex* terbagi dalam beberapa tahap meliputi perbaikan citra, pencarian tebal dinding dan lorong labirin, serta melakukan pencarian *node* dan disimpan ke dalam bentuk *graph* .



Gambar 7. Flowchart progam keseluruhan

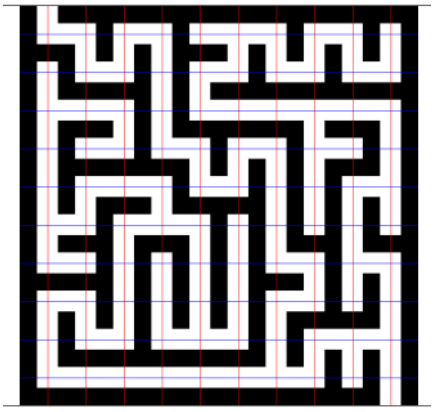
A. Perbaikan Citra

Perbaikan citra dilakukan secara berurutan yaitu mengkonversi citra *true color* ke dalam bentuk *grayscale* dan kemudian mengkonversi ke dalam bentuk citra biner. Sehingga citra labirin biner mudah untuk di proses oleh sistem.

B. Pencarian Lebar Dinding dan Lorong

Untuk mencari ketebalan dinding dan jalan pada labirin, dilakukan dalam dua tahap. Pertama adalah pengecekan dari sisi sebelah kiri citra labirin dan yang kedua adlah pengecekan dari sisi sebelah atas. Tahap awal dalam pengolahan citra labirin ini yaitu menentukan jumlah titik sampel yang nantinya digunakan sebagai titik – titik pencarian untuk mencari nilai lebar *border* pada citra labirin dan mencari nilai lebar dinding dan lorong citra labirin. Titik sampel yang digunakan pada program ini adalah sebanyak 100 titik sampel, hal tersebut dimaksudkan untuk mendapatkan keakuratan data yang diperoleh baik nilai lebar *border* maupun nilai lebar dinding dan lorong.

Proses pertama adalah mencari lebar dinding dan lorong horizontal dilakukan secara *vertical* dimulai dari (*xPosition, y*). Proses pencarian dimulai dengan mendapatkan nilai *pixel* hitam dari sisi atas bergerak kebawah hingga bertemu *pixel* putih, selanjutnya mendapatkan nilai *pixel* putih dan akan berhenti saat pencarian bertemu *pixel* hitam. Setelah didapatkan nilainya, kemudian dicari nilai yang paling banyak keluar untuk dijadikan sebagai lebar dari dinding dan lorong.



Gambar 8**.** Hasil *drawing* awal proses pencarian tebal dinding dan lorong

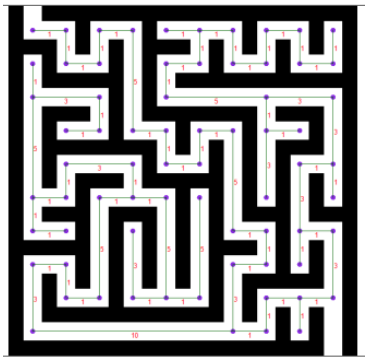
C. Pencarian Node

Saat labirin telah di proses sehingga terbentuk *node-node* awal, hal yang dilakukan selanjutnya adalah menyeleksi *node-node* yang sudah ada. *Node* awal diseleksi sesuai dengan parameter yang ditunjukkan oleh tabel 1.

Tabel 1. Parameter Penentuan *Node*

Nomor Vertex	Batas Kiri	Batas Atas	Batas Kanan	Batas Bawah	Keterangan
1	Ada dinding	Ada dinding	Tidak	Tidak	Tikungan
2	Ada dinding	Ada dinding	Ada dinding	Tidak	Jalan buntu
3	Ada dinding	Ada dinding	Tidak	Ada dinding	Jalan buntu
4	Ada dinding	Tidak	Ada dinding	Ada dinding	Jalan buntu
5	Ada dinding	Tidak	Tidak	Ada dinding	Tikungan
6	Ada dinding	Tidak	Tidak	Tidak	Simpang Tiga
7	Tidak	Ada dinding	Ada dinding	Ada dinding	Jalan buntu
8	Tidak	Ada dinding	Ada dinding	Tidak	Tikungan
9	Tidak	Ada dinding	Tidak	Tidak	Simpang Tiga
10	Tidak	Tidak	Ada dinding	Ada dinding	Tikungan
11	Tidak	Tidak	Ada dinding	Tidak	Simpang Tiga
12	Tidak	Tidak	Tidak	Ada dinding	Simpang Tiga
13	Tidak	Tidak	Tidak	Tidak	Perempatan

Setelah *node* diseleksi langkah selanjutnya adalah mencari nilai *edge* dari masing-masing *vertex* yang terhubung dan melakukan *overlay node* serta edge yang menghubungkan antar *node*.

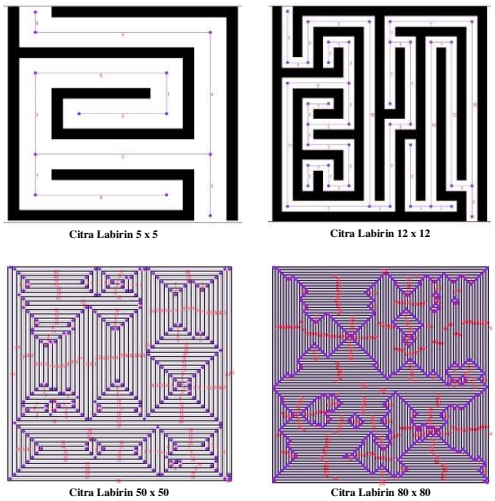


Gambar 9. Hasil akhir citra labirin.

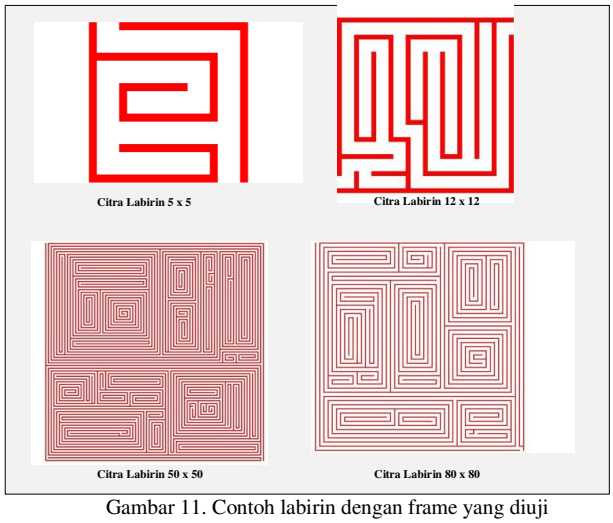
IV. PENGUJIAN DAN PEMBAHASAN

A. Pengujian citra labirin biasa

Pengujian pertama dilakukan pada 20 citra labirin biasa tanpa *frame* dan *noise*, untuk melihat apakah aplikasi yang dibuat dapat berjalan pada model citra labirin dasar dan membandingkan jumlah *node* dengan hasil perhitungan manual. Serta waktu eksekusi program pada citra labirin yang berbeda ukuran dan resolusinya.



Gambar 10. Contoh labirin biasa yang diuji



Gambar 11. Contoh labirin dengan frame yang diuji

Tabel 2. Hasil pengujian citra labirin biasa

No	Labirin	Jumlah *node* program	Hasil perhitung-an manual
1	5 x 5	12	12
2	7 x 7	16	16
3	8 x 8	18	18
4	11 x 11	72	72
5	12 x 12	40	40
6	21 x 21	150	150
7	22 x 22	186	186
8	25 x 25	55	55
9	27 x 27	61	61
10	33 x 33	101	101
11	38 x 38	105	105
12	41 x 41	130	130
13	47 x 47	154	154
14	50 x 50	250	250
15	60 x 60	289	289
16	70 x 70	372	372
17	80 x 80	478	478
18	90 x 90 (3)	6251	577
19	90 x 90 (2)	8100	577
20	100 x 100	6732	535

Dari hasil pengujian pada citra labirin biasa, program yang dibuat dapat mendeteksi *node* dengan baik dan sesuai dengan perhitungan manual. Tetapi saat menangani citra labirin besar dengan nilai dinding dan lorong yang sangat kecil, algoritma yang dibuat membutuhkan waktu yang cukup lama karena terlalu banyak *node* yang diseleksi serta pengecekkan nilai *edge* masing-masing *node* .

B. Pengujian Citra Labirin dengan *frame*

Citra labirin dengan *frame* adalah salah satu bentuk citra labirin yang memerlukan metode khusus, karena dalam penanganannya harus terlebih dahulu mendeteksi *frame* diluar citra labirin baru kemudian mulai pemecahan dinding dan jalan labirin. Pengujian dilakukan kepada 20 sampel citra labirin yang mempunyai bentuk labirin dan *frame* yang berbeda – beda, baik dari file maupun citra hasil dari *scanner*.

Tabel 3. Hasil pengujian citra labirin dengan *frame*

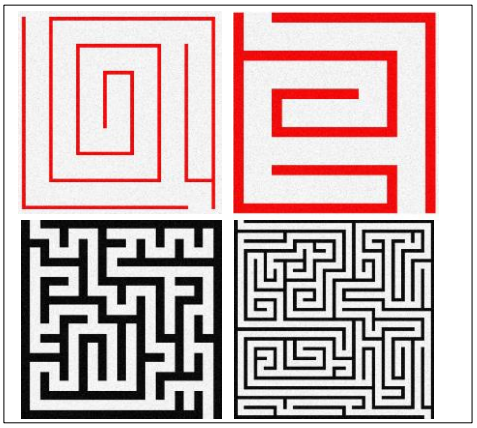
No	Labirin	Jumlah *node* program	Hasil perhitung-an manual
1	5 x 5	12	12
2	7 x 7 (*scanner*)	16	16
3	8 x 8	48	48
4	11 x 11 (*scanner*)	98	96
5	12 x 12	31	31
6	21 x 21 (*scanner*)	69	69
7	22 x 22	119	119
8	25 x 25 (*scanner*)	55	55
9	27 x 27	61	61
10	33 x 33 (*scanner*)	109	101
11	38 x 38	105	105
12	41 x 41 (*scanner*)	130	130
13	47 x 47	154	154
14	50 x 50	250	250
15	60 x 60	289	289
16	70 x 70	372	372
17	80 x 80	478	478
18	90 x 90 (3)	6251	577
19	90 x 90 (2)	8100	577
20	100 x 100	6732	535

Dari hasil pengujian pada citra labirin dengan *frame*, program yang dibuat dapat mendeteksi *node* dengan baik dan sesuai dengan perhitungan manual. Tetapi saat menangani citra labirin besar dengan nilai dinding dan lorong yang sangat kecil, algoritma yang dibuat membutuhkan waktu yang cukup lama karena terlalu banyak *node* yang diseleksi serta pengecekkan nilai edge masing-masing *node*.

Pada pengujian terhadap citra labirin hasil dari *scanner* terdapat beberapa hasil pengolahan yang tidak sesuai dengan perhitungan manual dikarenakan pada citra labirin hasil *scanner* memiliki *noise* yang menimbulkan kesalahan dalam proses pencarian nilai dinding dan lorong, serta posisi citra hasil *scanner* yang tidak simetris dan tegak lurus.

C. Pengujian Citra Labirin Dengan *Noise*

Citra labirin dengan *noise* adalah salah satu bentuk citra labirin yang didalamnya terdapat gangguan – gangguan baik pada *frame*, dinding maupun lorong sehingga dalam pengolahannya sering menimbulkan kesalahan pada hasil dari pengolahan citra. Pengujian dilakukan kepada 20 citra labirin yang diberi *gaussian noise* secara acak.

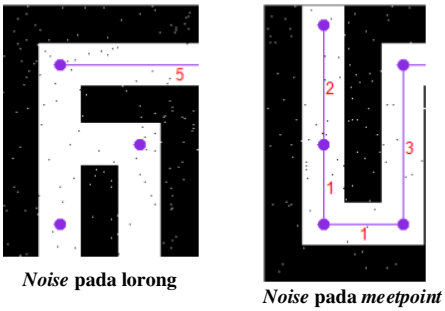


Gambar 12. Contoh labirin dengan frame yang diuji

Tabel 4. Hasil pengujian citra labirin dengan *gaussian noise*

No	Labirin	Jumlah *node* program	Hasil perhitung-an manual
1	676 x 694 px	12	12
2	700 x 700 px	16	16
3	669 x 669 px	72	72
4	651 x 651 px	150	150
5	700 x 700 px	55	55
6	750 x 750 px	103	102
7	898 x 857 px	130	130
8	878 x 705 px	250	250
9	756 x 705 px	289	289
10	863 x 897 px	574	478
11	679 x 704 px	12	12
12	709 x 758 px	16	16
13	669 x 669 px	72	72
14	669 x 655 px	150	150
15	704 x 710 px	55	55
16	745 x 750 px	103	102
17	898 x 857 px	130	130
18	1057 x 705 px	250	250
19	756 x 705 px	289	289
20	863 x 897 px	574	478

Dari hasil pengujian pada citra labirin dengan *Gaussian noise* secara acak pada citra labirin, program yang dibuat dapat menangani *noise* serta mendeteksi *node* yang *valid*. Tetapi terdapat beberapa kondisi *noise* pada lorong yang menyebabkan jumlah dan letak *node* tidak sesuai dengan yang diinginkan. Kondisi *noise* tersebut adalah saat letak *noise* berada tepat di *meetpoint* atau di titik *node* yang akan diseleksi serta saat ada *noise* yang terdapat di garis *edge* antar *vertex* pada lorong yang menyebabkan terbentuknya *node* baru dan membuat *edge* antar *node* menjadi berbeda. Kondisi *noise* pada lorong yang menyebabkan kesalahan *node* dan *edge* seperti ditunjukkan pada gambar 13.

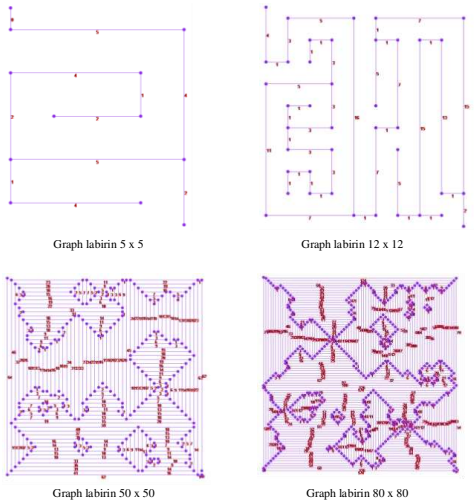


Noise* pada lorong** ***Noise* pada *meetpoint

Gambar 13. Kondisi *noise* pada lorong yang menyebabkan kesalahan

D. Hasil Visualisasi *Graph*

Citra labirin yang telah diproses akan di representasikan kedalam bentuk *graph* yang didalamnya terdapat *vertex* atau *node* serta *edge* masing-masing *vertex* yang terhubung beserta nilai *edge*. Bentuk representasi hasil pengolahan citra kedalam bentuk *graph* seperti ditunjukkan oleh gambar 14.



Gambar 14. Representasi *graph* hasil pengolahan citra

E. Pengujian Hasil Output Program

Dalam pengujian hasil *output* program adalah dengan melakukan *load* hasil pengolahan ke dalam aplikasi serta membandingkan dengan citra labirin awal. Pengujian hasil *output* program digunakan untuk mengetahui apakah hasil *output* program sesuai dengan yang diinginkan. Hasil pengujian seperti pada tabel 5.

Tabel 5. Hasil pengujian *output* program

No	Pengujian	Jumlah Output	Hasil Node
1	Labirin biasa	20	*valid*
2	Labirin dengan *frame*	20	Terdapat 2 *error* dari *output* labirin pada citra *scanner* karena terdapat *noise* dan posisi citra labirin yang tidak simetris serta tegak lurus.
3	Labirin dengan *Gaussian noise*	20	Terdapat 4 *error* dari hasil *output* karena *noise* pada lorong terlalu besar
	Total	60	6 *error*

Dari hasil pengujian *output* program terdapat 6 *error*, yaitu perbedaan jumlah *node* yang dihasilkan. *Error* yang terjadi pada bentuk citra labirin yang mempunyai *noise* yang banyak atau besar pada lorong labirin yang mengakibatkan hasil *output* tidak sesuai, serta bentuk

citra dari hasil *scanner* yang tidak simetris dan tegak lurus sehingga menyebabkan nilai dinding serta lorong yang tidak *valid*, tetapi pada citra labirin biasa, *frame* serta dengan *noise* yang tidak terlalu besar, program yang telah dibuat masih dapat mendeteksi *edge* dan *vertex* dengan baik.

V. PENUTUP

A. Kesimpulan

Dari proses perancangan, implementasi dan pengujian program konversi citra labirin ke dalam *edge* dan *vertex* didapat kesimpulan antara lain.

1. Dalam program pengolahan citra labirin, untuk mendapatkan *edge* dan *vertex* yang *valid* dimulai dari proses perbaikan citra, mencari nilai *frame*, nilai dinding dan lorong labirin, *node* yang *valid*, *gateway*, menghubungkan antar *vertex* dan mendapatkan nilai *edge*, serta yang terakhir adalah menggambar kembali untuk memvisualisasikan kedalam bentuk *graph*.
2. Dari hasil pengujian, program yang telah dibuat dapat mendeteksi nilai frame dari citra labirin, serta dapat memperoleh *edge* dan *vertex* yang *valid* dan merepresentasikannya kedalam bentuk *graph.*
3. Dari hasil pengujian pada citra labirin dari *scanner* terdapat beberapa perbedaan nilai dinding dan lorong karena terdapat *noise* pada citra tersebut serta bentuk labirin yang tidak simetris dan tegak lurus.
4. Dari hasil pengujian terhadap citra labirin dengan *Gaussian noise* , program yang dibuat dapat menangani *noise* serta mendeteksi *node* yang *valid*. Tetapi terdapat beberapa kondisi *noise* pada lorong yang menyebabkan jumlah dan letak *node* tidak sesuai dengan yang diinginkan. Kondisi *noise* tersebut adalah saat letak *noise* berada tepat di *meetpoint* atau di titik *node* yang akan diseleksi serta saat ada *noise* yang terdapat di garis *edge* antar *vertex* pada lorong yang menyebabkan terbentuknya *node* baru dan membuat *edge* antar *node* menjadi berbeda.
5. Dari pengujian terhadap 20 citra labirin dengan *Gaussian noise* persentase *error* sebanyak 4 citra atau 20%, yaitu pada citra yg mempunyai *noise* pada lorong dengan kondisi *noise* berada pada *meetpoint* atau titik *node* serta *noise* yang terdapat di garis *edge* antar *vertex* pada lorong.
6. Dari total 60 bentuk citra yang terdiri dari, 20 citra biasa, 20 citra dengan frame dan 20 citra dengan *gaussian noise*. Persentase *error* sebanyak 8 citra atau 1,33%.

B. Saran

Saran yang dapat diberikan adalah :

1. Perlu ditambahkan metode khusus untuk menangani noise pada dinding dan lorong untuk memperkecil terjadinya *error* .
2. Perlu ditambahkan algoritma pencarian jalur terbaik untuk menguji *graph* yang telah dihasilkan.

DAFTAR REFERENSI

[1] Biggs, Norman. 1994. Algebraic Graph Theory Second Edition, Cambridge Mathematical Library.

[2] Budayasa, Ketut. 2007. Teori Graph dan Aplikasinya. Surabaya: Unesa.

[3] Gonzalez, R.C., Woods, Richard E.,2001. Digital Image Processing Second Edition, Prentice Hall, New Jersey.

[4] Munir, Rinaldi. 2005. Pengolahan Citra Digital Dengan Pendekatan Algoritmik. Jakarta: Elex Media Komputindo.

[5] Sutoyo, T., Mulyanto, Edy., Suhartono, Vincent., Nurhayati, Oky Dwi., Wijanarto. 2009. Teori Pengolahan Citra Digital. Yogyakarta: Andi.

[6] Tien, Tan Soei. 2001. Bahasa C# Untuk Pemrograman Berorientasi Objek. Jakarta: Elex Media Komputindo.