PENDETEKSIAN LUBANG DI JALAN SECARA SEMI-OTOMATIS

SEMI-AUTOMATIC POTHOLE DETECTION

Priyanto Hidayatullah[1], Ferry Ferizal[2], Rheo Habibie Ramadhan[3], Bening Qadarsih[4], dan Febrianto Mulyawan[5]

([1,2]Staf Pengajar Jurusan Teknik Komputer Politeknik Negeri Bandung; [3,4,5]Pendiri Els Soft)

ABSTRAK

Pendeteksian lubang di jalan, oleh Dinas Pekerjaan Umum (PU) Jawa Barat, saat ini dilakukan dengan tenaga manusia sepenuhnya. Pekerja menyusuri ruas jalan untuk menemukan kerusakan jalan serta mengukur kedalaman, membuat, dan menghitung luas *patch* (penanda lubang ketika akan diperbaiki). Cara seperti ini membutuhkan waktu yang relatif lama. Dengan menggunakan teknologi pengolahan citra, pendeteksian dan perhitungan luas lubang dapat dipercepat dan tetap akurat. Hasil yang telah dicapai adalah terbangunnya sebuah aplikasi yang mampu mendeteksi dan menghitung luas *patch* lubang jalan dengan metode semiotomatis dengan bantuan operator. Hasil penghitungan luas *patch* pada aplikasi memiliki persentase *error* rata-rata sebesar 7,58%. Dengan adanya aplikasi ini, pencatatan kondisi jalan yang mencakup pendeteksian dan penghitungan luas *patch* lubang menjadi lebih cepat dengan tingkat kesalahan relatif rendah. Aplikasi ini dapat dikembangkan sehingga mampu digunakan untuk mendeteksi semua jenis kerusakan jalan, mampu melakukan pendeteksian secara otomatis tanpa bantuan operator, dan mampu melakukan validasi pendeteksian yang lebih baik sehingga hasil pendeteksian lebih akurat.

Kata kunci**:** pendeteksian lubang jalan, pengolahan citra, fitur citra, derajat keabuan, kontur, histogram

ABSTRACT

Pothole detection conducted by the Public Works Services of West Java is currently completely *done by human power.W orkers travel along the roads to find and measure the depth of the pothole, create and calculate the area of patch (pothole marker). However, this way requires a relatively long time. By using image processing technology, the detection and calculation of the pothole can be accelerated and remain accurate. The results achieved are an application that is able to detect and calculate the area of pothole patch with semi_automatic methods with the help of the operator. The result of extensive patches on the application has a percentage of average error of 7.58%. With thisapplication, recording the condition of roads that includes the detection and calculating potholes area are faster with relatively low error rate. This application can be improved so that it can be used to detect all types of roads damage, capable of performing detection automatically without operator assistance and is able to do*

better validation of the detection and as a result more accurate detection result can be achieved.

Keywords: pothole detection, image processing, image features, gray level, contours of image, histogram.

PENDAHULUAN

Perbaikan jalan yang dilakukan di Jawa Barat melalui tiga tahap. Pertama, dilakukan pencatatan kondisi suatu ruas jalan untuk mendapatkan data kerusakan yang mencakup lokasi ruas jalan (provinsi, kilometer, dan nama ruas jalan), ukuran ruas jalan (panjang, lebar, lebar bahu, dan lebar saluran ruas jalan), jenis kerusakan, luas kerusakan, kedalaman lubang untuk kerusakan berjenis lubang, kedalaman legokan untuk kerusakan berjenis legokan dan informasi kondisi keseluruhan ruas jalan. Pada tahap ini, sebelum dilakukan penghitungan luas tiap-tiap kerusakan yang ditemukan, dilakukan pembuatan *patch*. *Patch* (Gambar 1) adalah *outline* berbentuk poligon segi-n yang mengelilingi kerusakan. Luas kerusakan yang dihitung di sini yaitu luas *patch* dari kerusakan.

Dalam melakukan proses pendeteksian lubang di jalan, Dinas PU Jawa Barat belum memanfaatkan teknologi komputer. Selama ini, proses tersebut masih dilakukan secara manual. Proses pendeteksian dan pencatatan tersebut bisa memakan waktu dua pekan untuk jalan sepanjang 1 km.

Dalam artikel ini, akan dibahas sebuah alternatif untuk membantu proses pencatatan kondisi jalan menggunakan teknologi Pengolahan Citra Digital. Pada aplikasinya, dimasukkan video hasil rekaman kondisi jalan yang rusak.

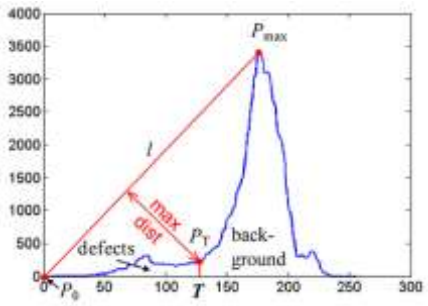
Gambar 1. *Patch* pada lubang jalan dari Muhammad Idris (2010)

TINJAUAN PUSTAKA

Beberapa literatur yang berkembang mengenai pendeteksian lubang jalan menggunakan metode yang beragam. Adi (2005) menggunakan pegas sebagai alat pendeteksi getaran dan Matthies (2003) menggunakan sensor suhu. Adapun Hesami (2009) menggunakan laser dalam pendeteksian lubang beserta kedalamannya. Bersamaan perkembangan teknologi, para peneliti mengembangkan metodenya dengan memanfaatkan teknologi yang terdapat pada perangkat bergerak. Sebagai contoh, De Zoysa (2007) dan Mednis (2011) menggunakan *accelerometer* pada *smartphone*.

Sebagian literatur mendeteksi lubang di jalan menggunakan pengolahan dan analisis citra. Menurut Koch dkk (2011), lubang memiliki ciri visual lebih gelap daripada sekitarnya, bentuknya cenderung elips yang diakibatkan distorsi perspektif,

teksturnya lebih kasar dibandingkan jalan di sekelilingnya. Dalam penelitiannya, Koch dkk membagi pendeteksian lubang di jalan menjadi tiga tahap, yaitu: segmentasi citra, ekstraksi bentuk, dan pembandingan tekstur. Tahap segementasi citra dilakukan dengan *thresholding segmentation* yang penentuan *threshold*-nya menggunakan histogram (Gambar 2). Ekstraksi bentuk dilakukan dengan *morphology thinning* dan ekstraksi tekstur menggunakan pendekatan statistik yaitu standar deviasi dari intensitas keabuan.



Gambar 2. Penentuan *Threshold* Berdasarkan Histogram oleh Koch (2011)

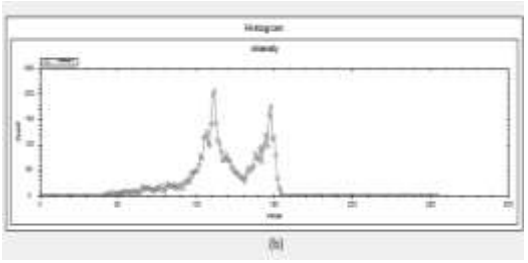
Vijay (2007) mengatakan pendeteksian lubang bisa bermasalah ketika terjadi efek perpektif linier. Efek ini menimbulkan besarnya sebuah objek yang dekat dengan kamera akan tampak berbeda dengan objek yang jauh dari kamera. Oleh karena itu, citra hasil perekaman kondisi jalan harus dikoreksi terlebih dahulu sebelum diproses lebih lanjut.

Joubert dkk (2011) menggunakan Kinect dan GPS untuk mendeteksi lubang jalan dan posisinya. Untuk mengetahui sisi dari sebuah lubang jalan, Joubert dkk mengestimasi permukaan jalan menggunakan algoritma *random sample consensus* (RANSAC). Titik-titik yang bukan merupakan bagian dari permukaan tersebut adalah titik-titik dari lubang jalan. Algortima deteksi kontur kemudian diterapkan untuk mengidentifikasi sisi-sisi dari lubang jalan.

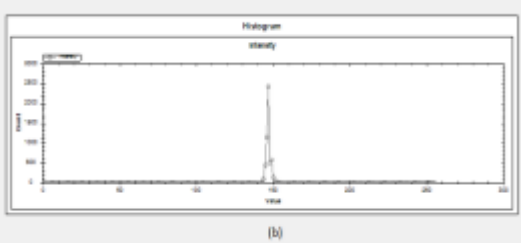
METODE YANG DIUSULKAN

Berdasarkan hasil wawancara dengan nara sumber dari Dinas PU Jawa Barat, didapat definisi dari lubang jalan yaitu suatu jenis kerusakan jalan yang sebagian atau seluruh bagian lapisan jalan rusak atau hilang. Ciri umum lubang berdasarkan karakteristik visualnya yaitu lubang memiliki kedalaman, warna lubang berbeda dengan warna jalan di sekelilingnya (cenderung lebih gelap), tekstur lubang biasanya lebih kasar, dan lubang berbentuk kontur tertutup. Gambar 3 sampai dengan Gambar 5 memberikan ilustrasi akan karakter visual lubang jalan.

Gambar 3. Jalan dengan Lubang



Gambar 4. Histogram Bagian Lubang



Gambar 5. Histogram Bagian Jalan Yang Tidak Berlubang

Proses pendeteksian lubang dan bukan lubang dimulai dengan perekaman video kondisi jalan dengan menggunakan kamera yang dipasang pada kendaraan dengan sudut kemiringan 45^o, 60^o, dan 90^o.

Gambar 6. Posisi kamera

Menurut Koch (2011), citra dalam mengukur tingkat kecerahan/kegelapan informasi warna pada citra RGB tidak diperlukan sehingga citra RGB perlu diubah ke dalam citra *grayscale*.

Selanjutnya, dilakukan *noise removal* pada video kondisi jalan. Filter yang digunakan adalah *low pass filter* seperti *mean filter*, *median filter*, dan *gaussian filter*. Ukuran kernel filter yang digunakan adalah 3x3 dan 5x5.

Langkah selanjutnya adalah segementasi lubang dengan jalan bukan lubang. Metode *thresholding segmentation* yang digunakan oleh Koch dkk (2011) kurang efektif untuk sampel yang digunakan di dalam artikel ini. Oleh karena itu, dalam artikel ini, diusulkan metode lain, yaitu segmentasi dengan *threshold* dengan penentuan *threshold* menggunakan Rata-rata Intensitas Keabuan Citra dengan rumus

$$U(x) = \begin{Bmatrix} 1, x >= T \\ 0, x < T \end{Bmatrix} \quad \text{.....(1)}$$

$$T = \text{———} \quad \text{.....(2)}$$

dengan

U = citra hasil segmentasi

x = nilai pixel citra sebelum disegmentasi

T = *threshold*

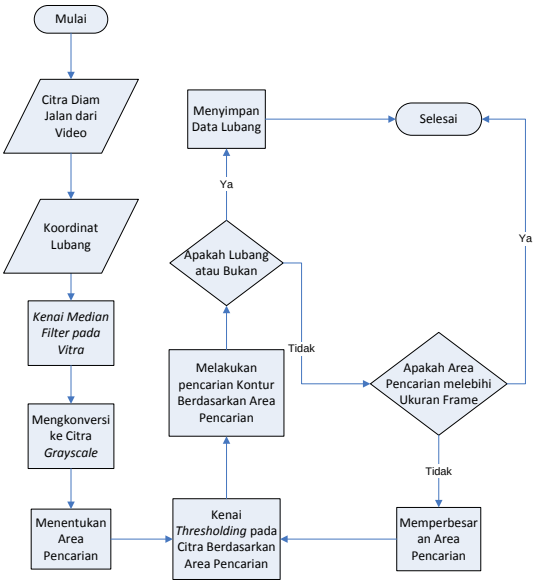
Hasil dari proses segementasi tersebut adalah citra biner.

Setelah didapatkan citra tersegmentasi, langkah selanjutnya adalah mencari kontur lubang menggunakan *Moore-Neighbor tracing*. Algoritma ini dibuat berdasarkan konsep *Moore Neighborhood*. Ide dari algoritma ini adalah menelusuri kedelapan tetangga *Moore Neighborhood* dengan nilai satu (dapat searah atau berlawanan jarum jam, asalkan konsisten) dari suatu pixel yang bernilai satu. Pertama, sebuah pixel dengan nilai satu dicari dan dinyatakan sebagai pixel "*start*" lalu dari pixel "*start*" ini, mulai dilakukan penelusuran kedelapan tetangga *Moore Neighborhood.* Jika pada saat penelusuran ini ditemukan pixel hitam, mundur (*back track*) ke posisi pixel sebelum pixel hitam ditemukan. Dari sini, kembali telusuri kedelapan tetangga Moore Neigborhood dari pixel hitam yang baru ditemukan. Algoritma ini terus berulang hingga kembali ke pixel "start" (posisi kembali ke pixel awal).

Berdasarkan ciri umum lubang jalan yaitu memiliki kontur tertutup, pencarian kontur lubang *Moore-Neighbor tracing* dimodifikasi sehingga hanya kontur tertutup yang dihasilkan. Caranya adalah dengan mengubah kriteria penghentian algoritma *Moore-Neighbor tracing.* Pada algoritma aslinya, proses pendeteksian kontur akan berhenti ketika penelusuran kontur kembali ke pixel start dan ketika sebuah pixel hitam dua kali ditelusuri. Hal ini memungkinkan hasilnya berupa kontur terbuka. Pada algoritma yang dimodifikasi, ditentukan kondisi berhenti hanya bila penelusuran kontur kembali ke pixel start yang bisa dipastikan hasilnya adalah kontur tertutup.

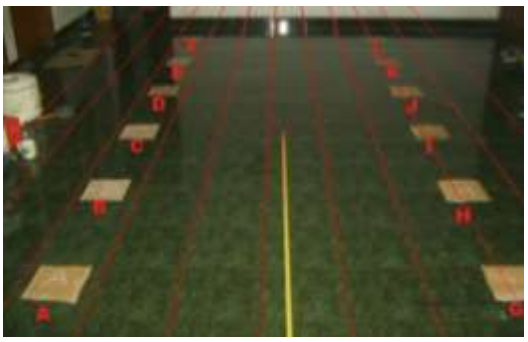
Selain itu, algoritma tersebut juga dimodifikasi sehingga setelah ditemukan sebuah kontur, proses tidak akan berhenti, tetapi akan dicari lagi kontur lain sampai tidak ada lagi kontur tersisa. Karena kontur yang berhasil dideteksi memungkin lebih dari satu, kontur disimpan di dalam *array of pixel* multidimensi.

Flowchart dari metode yang diusulkan di dalam artikel ini bisa dilihat pada Gambar 7.



Gambar 7. *Flowchart* Dari Metode Yang Diusulkan

Setelah proses pendeteksian lubang, selanjutnya adalah penghitungan luas *patch* lubang.



Gambar 8. Percobaan Pertama Penanganan Distorsi Perspektif

Karena pada citra digital sebuah lubang dan *patch*-nya direpresentasikan oleh sekumpulan pixel, untuk menghitung luas *patch* lubang, perlu diketahui satu pixel dalam posisi tertentu mewakili berapa centimeter persegi (luas) di dunia sebenarnya (nilai rasio centimeter persegi/pixel). Untuk mengetahui nilai rasio ini, dilakukan serangkaian percobaan. Teori yang digunakan untuk

penghitungan luas *patch* lubang yaitu regresi non-linear polinomial dan eliminasi Gauss Jordan.

Pertama, diambil citra 16 buah objek karton dengan ukuran 20x20 cm (400 cm^2) dengan tinggi kamera 135 cm dan sudut kamera 60^o. Penempatan objek-objek ini dapat dilihat pada gambar 8.

Jumlah pixel setiap karton dan lokasinya dalam citra (koordinat titik tengah karton) kemudian dihitung. Dari penghitungan ini, dihasilkan data pada tabel 1.

Karena adanya distorsi perspektif, perlu diketahui keterkaitan antara jarak (dari sumbu x dan sumbu y) dan jumlah pixel yang merepresentasikan karton berukuran 20x20cm. Dapat dilihat pada tabel 11 bahwa perubahan rasio cm^2/pixel bersamaan dengan bertambahnya nilai pada sumbu x relatif jauh lebih kecil daripada perubahan rasio cm^2/pixel bersamaan bertambahnya nilai pada sumbu y. Atas dasar inilah, diasumsikan bahwa nilai yang memengaruhi rasio cm^2/pixel hanya jarak objek dari sumbu x (jarak objek dari bagian atas citra).

Tabel 1. Hasil Percobaan Pertama Hitung Luas

y / x		89		134		210		363
225	P	179						
273	L	215						
382	H	180						
429	D	185						
198			O	395				
257			K	342				
394			G	320				
454			C	324				
150					N	816		
232					J	819		
418					F	802		
496					B	746		
63							M	2321
179							I	2479

y / x		89		134		210		363
461							E	2711
574							A	2556

Keterangan :

	Jumlah *pixel* yang mewakili karton
	Kode karton
	Nilai koordinat x
	Nilai koordinat y

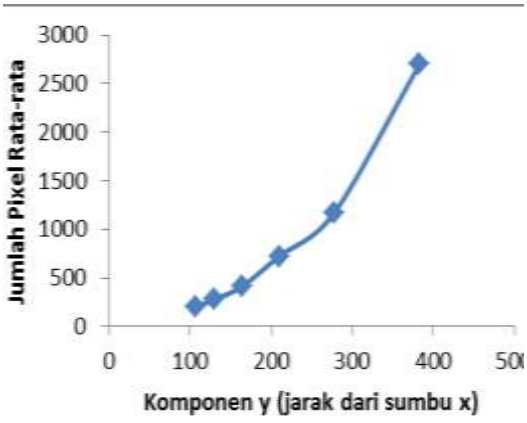
Setelah dibuat asumsi ini, dilakukan kembali percobaan kedua dengan setting yang sama, yang berfokus pada pencarian keterkaitan antara jarak objek dari sumbu x dan rasio cm^2/pixel-nya.

Tabel 2. Hasil Percobaan Kedua

Y	**Objek Pada Karton A-F**		**Objek Pada Karton G-L**		**Rata-Rata**	
	#Pixel	Cp*	*#Pixel*	Cp*	*#Pixel*	Cp*
106	188	2.12	213	1.87	200.5	2.00
130	260	1.53	300	1.33	280	1.43
164	408	0.98	415	0.96	411.5	0.97
210	712	0.56	738	0.54	725	0.55
277	1151	0.34	1173	0.34	1162	0.34
383	2686	0.14	2709	0.14	2697.5	0.14

*Cp = cm^2/*pixel*

Jika kolom jumlah pixel rata-rata dan y di atas direpresentasikan dalam diagram kartesian, terlihat sebuah kurva (Gambar 9).



Gambar 9. Representasi kurva dari data

Untuk dapat mengetahui nilai jumlah pixel yang mewakili 400 cm^2 di mana pun pada citra, diperlukan sebuah fungsi yang memuat atau mendekati seluruh data pada tabel (mewakili kurva yang terbentuk). Fungsi kurva ini dapat dicari dengan menggunakan metode regresi polinomial.

Pada kasus ini, komponen y (jarak dari sumbu x) menjadi peubah bebas (*independent variable*, disimbolkan dengan X) yang menentukan nilai jumlah pixel yang mewakili 400 cm^2 sebagai peubah tak bebas (*depedent variable*, disimbolkan dengan Y).

Rumus umum fungsi polinomial yang dicari adalah

$$Y = a_0 + a_1X + a_2X^2 \ldots + a_kX^k \quad \text{.....(3)}$$

dengan k adalah orde dari polinomial nilai konstanta a_0, a_1 .. a_k dapat diketahui dengan rumus umum pada persamaan (4).

$$\begin{bmatrix} n & \sum x_i & .. & \sum x_i^k \\ \sum x_i & ..... & & \sum x_i^{k+1} \\ \vdots & & & \vdots \\ \sum x_i^k & ..... & & \sum x_i^{k+2} \end{bmatrix} \begin{Bmatrix} a_0 \\ \vdots \\ a_k \end{Bmatrix} = \begin{Bmatrix} \sum x_i^0 y_i \\ \vdots \\ \sum x_i^k y_i \end{Bmatrix}$$

(4)

dengan n adalah jumlah data yang ada dan k adalah orde dari polinomial.

Pada kasus ini, dicobakan polinomal dengan orde dua dengan rumus fungsinya adalah

$$Y = a_0 + a_1X + a_2X^2 \quad \text{.....(5)}$$

$$Y = 405 - 4.555X + 0.027X^2 \quad \text{.....(6)}$$

Dengan diketahuinya fungsi di atas, jumlah pixel yang mewakili 400 cm2 pada dunia nyata yang berlaku untuk pixel dengan jarak dari sumbu x tertentu dapat diperkirakan dengan memasukkan nilai jarak dari sumbu x ke dalam fungsi. Dengan diketahuinya jumlah pixel yang mewakili 400 cm^2, nilai rasio cm^2/pixel dan *error*-nya bisa didapat dengan dengan rumus

$$rasio cm^2 / pixel = \frac{400 cm^2}{jumlah pixel} \quad \text{.....(7)}$$

$$\%error = \frac{nilai perkiraan - nilai asli}{nilai asli} x100$$

(8)

HASIL DAN PEMBAHASAN

Spesifikasi perangkat keras untuk pembangaunan dan pengujian aplikasi yang dibuat di dalam artikel ini adalah Notebook Asus E82JQ-A1dengan spesifikasi *harddisk* SATAII 500GB 5400RPM, intel core i7 720QM 1.6Ghz, 4GB RAM DDR2, Window 7 ultimate 32bit. *Software* yang

digunakan dalam membangun aplikasi ini sebagai berikut: Microsoft Visual Studio 2010, OpenCV 2.2, EmguCV 2.2.1.1150 (sebagai jembatan antara OpenCV dan .NET *framework*).

Tahap persiapan dari metode yang diusulkan adalah perekaman kondisi jalan. Telah dilakukan tiga percobaan dengan sudut yang berbeda untuk mengetahui sudut mana yang paling optimal. Hasilnya terlihat pada tabel 3 yang dapat disimpulkan bahwa sudut penempatan kamera 60^o paling baik dibandingkan sudut yang lain karena *field of view* (FOV) yang dihasilkan lebih besar daripada sudut yang lain sehingga sisi kiri dan sisi kanan jalan dapat terlihat kamera secara bersamaan. Selain itu, hasil segmentasi video yang diambil menggunakan sudut ini jauh lebih baik. Adapun distorsi perspektif yang menjadi kelemahan dari sudut penempatan kamera ini ditangani dengan regresi non-polinomial dan eliminasi Gauss-Jordan.

Tabel 3. Kelebihan dan Kekurangan dari Posisi Kamera

Sudut	Kelebihan	Kekurangan
45^0	- *Field of view* (FOV) > FOV 90^0 - Proses segmentasi mudah dilakukan	Timbul distorsi perspektif yang mengganggu dalam penghitungan luas *patch* lubang.
60^0	- FOV 60^0 > FOV 45^0 - Proses segmentasi mudah dilakukan	Timbul distorsi perspektif yang mengganggu dalam penghitungan luas *patch* lubang.
90^0	Tidak ada distorsi perspektif yang menggangu dalam penghitungan luas *patch* lubang.	- FOV 90^0 < FOV 45^0 - Proses segmentasi sulit dilakukan karena dari sudut pandang ini lubang dan jalan terlihat mirip.

Pada saat video menunjukkan lubang, video diberhentikan kemudian diekstrak menjadi citra. Selanjutnya, citra tersebut dikonversi ke dalam bentuk citra *grayscale* (Gambar 11) kemudian tahap *noise removal* pada citra *grayscale* hasil dari tahap sebelumnya. Telah dicobakan tiga jenis *low pass filter* yaitu *mean filter, median filter,* dan *gaussian filter* dan ukuran kernel 3x3 dan 5x5. Hasilnya menunjukkan bahwa secara visual, median filter memberikan hasil lebih baik daripada filter lain walaupun waktu yang diperlukan lebih lama. Namun, perbedaan waktunya tidak signifikan. Selain itu, ukuran kernel 5x5 lebih baik dibandingkan 3x3. Tabel 4 menunjukkan waktu yang diperlukan masing-masing filter untuk menghilangkan *noise* dari 2804 citra dengan ukuran 640x480 pixel dengan kernel 5x5.

Tabel 4. Perbandingan Penggunakan Waktu Proses *Noise Removal*

Filter	Waktu
Mean	57.050 milidetik
Median	60.651 milidetik
Gaussian	47.720 milidetik

Setelah itu, proses *thresholding segmentation*. Sebelumnya, sudah dicobakan segmentasi 2.804 citra dengan ukuran 630x480 pixel menggunakan metode yang diusulkan oleh Koch (2011). Namun, diperlukan waktu 11,7 menit untuk menentukan *threshold*-nya yang tentunya cukup lama. Jika menggunakan metode nilai rata-rata derajat keabuan, waktu yang dibutuhkan hanya 1,7 menit dengan hasil yang baik. Hasilnya terlihat pada Gambar 12. Dari gambar tersebut, terlihat dengan cukup jelas bahwa lubang cukup tersegmentasi dari jalan yang akan memudahkan proses selanjutnya.

Gambar 10. Citra awal dari jalan yang memiliki lubang

Gambar 11. Citra *grayscale*

Gambar12. Citra yang telah dikenai *thresholding* (citra biner)

Gambar 13. Citra lubang yang telah dideteksi konturnya dan diberi *patch*

Tahap berikutnya adalah pendeteksian kontur yang hasilnya ada pada Gambar 13. Dari gambar tersebut, terlihat pula bahwa kontur dapat terdeteksi dengan baik. Setelah kontur terdeteksi, lubang diberi *patch*.

Setelah dibuat *patch*, selanjutnya proses perhitungan luas *patch*. Namun sebelumnya, perlu dihitung *error* yang mungkin terjadi. Jika dihitung *error* pada percobaan pertama menggunakan persamaan (8) di atas, didapatlah rata-rata persentase *error* 6,288%.

Setelah itu, dilakukan perhitungan *error* dengan kasus nyata yaitu percobaan penghitungan luas *patch* lubang di ruas jalan Sarijadi Bandung. Dipilih tiga lubang jalan di ruas jalan Sarijadi Bandung. Setiap lubang diukur panjang, lebar, dan luasnya. Data

panjang, lebar, dan luas disajikan dalan tabel 5.

Tabel 5. Luas lubang sebenarnya

No	Panjang	Lebar	Luas
1	190 cm	139 cm	26410 cm^2
2	97 cm	136 cm	13192 cm^2
3	79 cm	73 cm	5767 cm^2

Hasil percobaan menunjukkan persentase *error* rata-rata juga turun cukup signifikan jika jarak titik terdekat dengan sumbu x lebih besar daripada 59 pixel dengan persentase *error* rata-rata adalah 7,58%. Oleh karena itu, dibuat batasan bahwa aplikasi yang dibangun hanya menghitung lubang yang terdeteksi jika jarak titik terdekat dengan sumbu x lebih besar daripada 59 pixel dan persentase *error* 7,58% akan menjadi konstanta penambah hasil penghitungan luas *patch* sesungguhnya.

SIMPULAN DAN SARAN

Aplikasi dapat memproses video hasil perekaman kondisi jalan dalam bentuk AVI. Penentuan *threshold* menggunakan rata-rata derajat keabuan lebih cepat dibandingkan dengan histogram dengan hasil yang tidak jauh berbeda. Namun, hasil pendeteksi lubang tidak sepenuhnya akurat karena ada objek yang bukan lubang terdeteksi sebagai lubang ataupun sebaliknya. Misalnya, beberapa bayangan terdeteksi sebagai lubang. Hal ini disebabkan keterbatasan fitur citra yang digunakan. Fitur derajat keabuan dan kontur ternyata belum cukup memvalidasi objek lubang. Oleh karena itu, diperlukan fitur-fitur lain yang dapat memperakurat pendeteksian.

Patch lubang yang dibuat oleh aplikasi sudah mengikuti distorsi perspektif. Untuk metode penghitungan luas *patch* lubang digunakan regresi polinomial orde 2 dengan persentase *error* rata-rata 7,58%. Jadi, untuk mendapatkan hasil penghitungan luas *patch* lubang yang mendekati luas *patch* lubang sebenarnya, luas *patch* pada aplikasi ditambahkan dengan 7,58% dari luas *patch* lubang pada aplikasi.

Ke depannya, disarankan agar aplikasi dapat mengolah video dalam format yang lebih beragam seperti mpg, mp4, mov, atau flv. Selain itu, ditambahkan fitur lain seperti fitur tekstur dan bentuk agar pendeteksian lubang lebih akurat. *Error* yang dihasilkan dari metode regresi polinomial mungkin dapat diminimalkan lagi. Tentunya diperlukan analisis lebih mendalam lagi mengenai metode perhitungan luas *patch* supaya didapat hasil yang benar-benar akurat.

DAFTAR PUSTAKA

Adi, Ruzbeh dan Minocher Homji. 2005. *"Intelligent Pothole Repair Vehicle"*, Texas A&M University.

De Zoysa, Kasun, Chamath Keppitiyagama, Gihan P. Seneviratne, W. W. A. T. Shihan. 2007 . *"A Public Transport System Based Sensor Network for Road Surface Condition Monitoring"*, NSDR'07.

Hesami, Reyhaneh dan Kerry J. McManus. 2009. *"Signal Processing Approach to Road Roughness Analysis and Measurement"*, Center for Sustainable Infrastructure,

Swinburne University of Technology, Hawthorn, Australia.

Idris, Muhammad., S.Si,.MT; Rustijan, ST; Rizki Adelwin, ST; Januri Sugeng, A.Md. 2010. *Inspeksi Keselamatan Jalan dan Pemanfaatan Hawkeye di dalam Pelaksanaan Inspeksi Keselamatan Jalan.*

Joubert, Deon, Ayanda Tyatyantsi, Jeffry Mphahlehle and Vivian Manchidi. 2011. *"Pothole Tagging System"*, Council for Scientific and Industrial Research, Pretoria, South Africa.

Koch, Chistian dan Ionnis Brilakis. 2011. *Pothole detection in asphalt pavement image*. s.l. : Elsevier Ltd.

Matthies, L dan A Rankin. 2003. *"Negative Obstacle Detection by Thermal Signature"*, Intelligent Robots and Systems.

Mednis, Artis, Girts Strazdins, Reinholds Zviedris, Georgijs Kanonirs, Leo Selavo. 2011. *"Real Time Pothole Detection using Android Smartphones with Accelerometers"*, University of Latvia.

Vijay, Shonil. 2007. *Design of an Infrastructure using WiFi for the pot-holedetection system*, Kanwal Rekhi School of Information Technology, Indian Institute of Technology, Bombay,.