Jurnal Informatika, Vol. 13, No. 1, Mei 2015, 1-8 DOI: 10.9744/informatika.13.1.1-8
ISSN 1411-0105

SEGMENTASI OBYEK PADA CITRA DIGITAL MENGGUNAKAN METODE OTSU *THRESHOLDING*

Slamet Imam Syafi'i[1*], Rima Tri Wahyuningrum[2], dan Arif Muntasa[3]
[1,2,3] Program Studi Teknik Informatika, Fakultas Teknik, Universitas Trunojoyo Madura
Jl. Raya Telang, PO BOX 2, Kamal, Bangkalan – 69162, Indonesia
*Penulis korespondensi; Email: slametux@gmail.com

Abstrak: Citra digital memiliki ukuran dan obyek berupa *foreground* dan *background*. Untuk memisahkannya, perlu dilakukan sebuah proses segmentasi citra. Salah satu metode dalam proses segmentasi citra adalah menggunakan Metode Otsu *thresholding*. Penelitian ini dibagi menjadi lima proses, yaitu *input* data citra, *pre-processing*, segmentasi, *cleaning*, dan perhitungan akurasi. Tahap pertama adalah *input* data citra digital RGB yang di dalamnya terdiri dari beberapa obyek. Tahap kedua adalah konversi dari citra RGB ke citra *grayscale*. Tahap ketiga adalah mencari nilai ambang secara otomatis menggunakan Metode Otsu *thresholding*, kemudian dikonversi ke citra biner. Tahap keempat adalah proses *invert image*, *noise removal* dengan nilai ambang 150, dan *morphology*. Tahap terakhir adalah proses perhitungan akurasi dilakukan untuk mengukur kinerja dari metode segmentasi yang selanjutnya hasil dari proses tersebut dibandingkan dengan citra *Ground Truth* hasil pengamatan *user* secara langsung untuk menghitung tingkat akurasi. Pengujian dilakukan pada *Weizmann Segmentation Database* sebanyak 30 citra digital RGB. Akurasi yang didapat dari pengujian tersebut sebesar 93,33%.

Kata kunci: Metode Otsu thresholding; Region properties; Segmentasi; Weizmann database.

Abstract: *Digital image has size and object in the form of foreground and background. To separate it, it is necessary to be conducted the image segmentation process. Otsu thresholding method is one of image segmentation method. In this research is divided into five processes, which are input image, pre-processing, segmentation, cleaning, and accuracy calculation. First process was input color images which consists of multiple objects. Second process was conversion from color image to grayscale image. Third process was automatically calculated threshold value using Otsu thresholding method, followed by binary image transformation. The fourth process, the result of third process is changed into negative image as the segmentation results, noise removal with a threshold value of 150, and morphology. The last accuracy calculation is conducted to measure proposed segmentation method performance. The experimental result have been compared to the image of Ground Truth as the direct user observation to calculate accuracy. To examine the proposed method, Weizmann Segmentation Database is used as data set. It conconsist of 30 color images. The experimental results show that 93.33% accuracy were achieved.*

Keywords*: Otsu thresholding method; Region properties; Segmentation; Weizmann database.*

PENDAHULUAN

Segmentasi citra digital dilakukan dengan membedakan antara obyek dan latar belakang, antara lain dengan memanfaatkan operasi pengambangan secara otomatis. Nilai *thresholding* (ambang) pada citra *grayscale* didapat dengan menggunakan Metode Otsu [1]. Dengan demikian, Metode Otsu *thresholding* cocok untuk mencari nilai ambang dari sebuah citra *grayscale*. Sehingga menghasilkan citra segmentasi yang bagus. Metode Otsu *thresholding* merupakan metode segmentasi yang cukup akurat dalam mendapatkan daerah yang merupakan obyek tersegmentasi dengan menggunakan histogram *grayscale*.

Proses sebelum dilakukan perhitungan tingkat akurasi terhadap citra hasil segmentasi, terlebih dahulu harus melalui proses *invert image* dan *noise removal*, agar area yang bukan termasuk obyek dapat dihilangkan [2].

Proses selanjutnya adalah perbaikan citra, *morphology* merupakan sebuah proses perbaikan pada citra biner. Prosesnya adalah menggabungkan titik-titik latar yang ada di dalam obyek menjadi bagian dari obyek. Agar piksel yang bukan termasuk obyek dan berada di dalam obyek pada citra digital diganti dengan piksel berwarna putih [3]. Lubang atau *holes* pada citra didefinisikan sebagai wilayah yang memiliki latar belakang dengan dikelilingi oleh perbatasan piksel yang terhubung dengan *foreground* atau obyek. *Filling holes* ini digunakan untuk mengisi bagian tengah obyek yang berlubang.

Setelah didapat hasil perbaikan citra, proses selanjutnya adalah menghitung tingkat akurasi dari setiap citra hasil segmentasi. Prosesnya untuk men-

dapatkan tingkat akurasi, diantaranya adalah dengan melakukan pelabelan menggunakan *region properties* atau *regionprops* [4]. Proses yang terakhir adalah perhitungan tingkat akurasi dengan menggunakan *Ground Truth* hasil dari pengamatan user secara langsung.

Penelitian ini mengimplementasikan segmentasi Metode Otsu *threholding* untuk mendapatkan nilai ambang secara otomatis dari sebuah citra *grayscale*. Tujuannya adalah untuk menerapkan Metode Otsu *thresholding* sebagai metode segmentasi dan mengetahui tingkat akurasi dengan *Ground Truth*. Dalam penelitian ini terdiri dari lima proses, yaitu *input* data citra, *pre-processing*, segmentasi, *cleaning*, dan perhitungan akurasi. Segmentasi dilakukan pada *Weizmann Segmentation Database*, data yang digunakan sebanyak 30 citra dengan format PNG. Hasil dari penelitian ini berupa citra segmentasi dan tingkat akurasi.

METODE PENELITIAN

Input Citra RGB

Pembacaan citra digital harus dilakukan untuk dapat memproses citra tersebut. Ada beberapa jenis file gambar yang dapat dibaca oleh Matlab, seperti JPG, GIF, TIF, BMP, dan salah satunya adalah jenis file yang digunakan dalam penelitian ini yaitu berformat .PNG [5]. Citra yang digunakan sebanyak 30 citra RGB yang didapat dari *Weizmann Segmentation Database*.

Grayscale

Citra *grayscale* adalah citra yang memiliki nilai dari putih dengan intensitas paling besar (255) sampai hitam yang memiliki intensitas paling rendah (0) [6]. Proses konversi dari citra RGB ke grayscale, dapat dilihat pada persamaan berikut:

$$Gray = ((R * 0.2989) + (G * 0.5870) + (B * 0.1140)) \quad (1)$$

Histogram Citra

Histogram citra dapat diartikan sebagai ukuran penyebaran piksel dari suatu citra. Histogram diperoleh dengan menghitung jumlah kemunculan dari setiap nilai piksel, yang kemudian dipetakan terhadap nilai intensitas dari citra *grayscale* [7].

Segmentasi Citra

Segmentasi citra merupakan bagian dari proses pengolahan citra. Segmentasi citra (*image segmentation*) mempunyai arti membagi suatu citra menjadi wilayah-wilayah yang homogen berdasarkan kriteria keserupaan tertentu antara tingkat keabuan suatu piksel dengan tingkat keabuan piksel-piksel tetangganya, kemudian hasil dari proses segmentasi ini akan digunakan untuk proses lebih lanjut [8].

Metode Otsu *thresholding*

Metode Otsu merupakan salah satu metode untuk segmentasi citra digital dengan menggunakan nilai ambang secara otomatis, yakni mengubah citra digital warna abu-abu menjadi hitam putih berdasarkan perbandingan nilai ambang dengan nilai warna piksel citra digital. Metode Otsu *thresholding* diperkenalkan pertama kali oleh Nobuyuki Otsu, dalam jurnal ilmiahnya yang berjudul "*A Threshold Selection Method from Grayscale Histogram*" pada tahun 1979 [9].

Untuk mendapatkan nilai *threshold* ada perhitungan yang harus dilakukan. Langkah awal yang harus dilakukan adalah membuat histogram. Dari histogram dapat diketahui jumlah piksel untuk setiap tingkat keabuan. Tingkat keabuan citra dinyatakan dengan i sampai dengan *L*. Level ke *i* dimulai dari *1*, yaitu piksel 0. Untuk *L*, maksimal level adalah 256 dengan piksel bernilai 255.

Nilai ambang yang akan dicari dari suatu citra *grayscale* dinyatakan dengan *k*. Nilai *k* berkisar antara 0 sampai dengan *L*-1, dengan nilai *L*=256 (simbol histogram adalah *Pi*) [9]. Jadi probabilitas setip piksel pada level ke *i* dinyatakan dengan persamaan (1):

$$P_i = \frac{n_i}{N} \quad (2)$$

Keterangan:
Pi = Probabilitas piksel ke-*i*
n_i = Jumlah piksel dengan tingkat keabuan *i*
N = Total jumlah piksel pada citra

Langkah selanjutnya mencari nilai jumlah kumulatif, rerata kumulatif dan intensitas global. mencari nilai tersebut dapat melihat persamaan (3), persamaan (4), dan persamaan (5).

Berikut adalah formulasi untuk menghitung jumlah kumulatif (*cumulative sum*) dari $\omega(k)$, untuk *L* = 0, 1, 2, ..., *L*-1:

$$\omega(k) = \sum_{i=0}^{k} p_i \quad (3)$$

Berikut adalah formulasi untuk menghitung rerata kumulatif (*cumulative mean*) dari $\mu(k)$, untuk *L* = 0, 1, 2, ..., *L*-1:

$$\mu(k) = \sum_{i=0}^{k} i.p_i \quad (4)$$

Berikut adalah formulasi untuk menghitung rerata intensitas global $\mu_T(k)$:

$$\mu_T(k) = \sum_{i=0}^{L-1} i.p_i \quad (5)$$

Pada persamaan (3), persamaan (4), maupun persamaan (5), nilai *k* menyatakan tingkat level keabuan dimana setiap rentang piksel akan dihitung. Langkah selanjutnya adalah menentukan varian antar kelas (*between class variance*). Persamaan untuk *between class variance* (6):

$$\sigma_B^2(k) = \frac{[\mu_T\omega(k) - \mu(k)]^2}{\omega(k)[1-\omega(k)]} \quad (6)$$

Hasil dari perhitungan *between class variance* dicari nilai maksimal. Nilai yang paling besar digunakan sebagai *threshold* atau nilai ambang (*k*), dengan persamaan (7):

$$\sigma_B^2(k^*) = max_{1\leq x\leq L}\ \sigma_B^2(k) \quad (7)$$

Keterangan:
$\omega(k)$ = Jumlah Kumulatif
$\mu(k)$ = Rerata Kumulatif
$\mu_T(k)$= Rerata Intensitas Global
σ_B^2 = Nilai Ambang

Between class variance bertujuan untuk mencari nilai ambang dari sebuah citra *grayscale*, nilai ambang atau *threshold* digunakan sebagai nilai acuan untuk mengubah citra *grayscale* ke citra biner. Setiap citra memiliki nilai ambang yang berbeda-beda.

Binerisasi

Dalam pengolahan citra digital, proses binerisasi adalah mengubah citra *grayscale* menjadi citra biner, artinya mengubah warna tiap-tiap piksel pada citra bernilai 0 dan 255 ke dalam piksel bernilai 0 dan 1. Sehingga citra hanya berwarna hitam dan putih. Pada proses binerisasi, menggunakan nilai ambang untuk menentukan nilai *grayscale* tertentu yang diubah menjadi piksel bernilai 0 atau 1 [2].

Negasi Citra

Negasi Citra (*invert images*) merupakan proses penggantian warna, untuk piksel berwarna putih diganti dengan piksel berwarna hitam. Sedangkan piksel berwarna hitam diganti dengan pisel berwarna putih [2]. Proses ini bertujuan agar pada saat proses penghapusan *noise* dapat lebih akurat.

Penghapusan *Noise*

Noise merupakan gangguan pada citra berupa bintik-bintik yang memiliki ukuran piksel lebih kecil dari obyek. *Noise* harus diatasi sebelum citra dianalisis, karena adanya *noise* dapat mengurangi tingkat akurasi dari proses segmentasi. Penghapusan *noise* (*noise removal*) dapat berupa proses *filtering* atau dengan menghapus piksel dengan ukuran tertentu [2]. Salah satunya adalah dengan menghapus area pada citra yang memiliki ukuran tertentu, dengan nilai ambang yang telah ditentukan secara manual.

Perbaikan Citra

Proses dari perbaikan citra (*morphology*) adalah menggabungkan titik-titik latar yang ada di dalam obyek menjadi bagian dari obyek [4]. Proses perbaikan citra dengan menggunakan *morphology*, di antaranya adalah *dilasi*, *erosi*, dan *filling holes* atau pengisian lubang. Lubang atau *holes* pada citra didefinisikan sebagai wilayah yang memiliki latar belakang dan dikelilingi oleh perbatasan piksel yang terhubung dengan *foreground* atau obyek.

Filling holes digunakan untuk mengisi bagian tengah dari obyek yang berlubang. Agar dapat mengisi lubang, titik di setiap lubang (*holes*), f_m, diberi nilai 1 (untuk citra biner) di semua titik sampai mencapai tepi *border*, 1-*f* [3]. Dapat dilihat pada persamaan (8).

$$f_m(x,y) = \begin{cases} 1-f(x,y) & berada\ pada\ tepi\ f \\ 0 & selain\ itu \end{cases} \quad (8)$$

Pelabelan

Pelabelan (*labelling*) berfungsi untuk memisalkan beberapa luas wilayah piksel yang tidak saling berhubungan [4]. Piksel-piksel yang berupa latar citra diberi nilai label "0", piksel-piksel lain yang berupa obyek dari citra diberi label "1" dan seterusnya sesuai dengan letak urutan perkolom.

Region Properties

Region properties (*regionprops*) berfungsi untuk mengukur sekumpulan properti-properti dari setiap wilayah yang telah dilabeli dalam matrik label. Untuk mengetahui jumlah obyek yang terdapat pada gambar yang terdeteksi adalah dengan cara memakai *labelling objects*, salah satunya adalah *regionprops* [10]. *Regionprops* hanya dapat mendeteksi obyek yang berwarna putih sebagai *foreground*, sedangkan untuk yang berwarna hitam akan dianggap sebagai *background*.

Selanjutnya menentukan titik tengah pada setiap obyek, titik tengah didapat dengan menentukan terlebih dahulu matrik setiap obyek. Setiap baris dari matrik berisi titik koordinat *x* dan *y* dari salah satu poin. Matrik pada setiap obyek memiliki batasan-batasan, yakni batas atas-kiri, atas-kanan, kiri-atas, kanan-atas, kiri-bawah, kanan-bawah, bawah-kiri, dan

bawah-kanan. Dari batasan-batasan tersebut, akan didapat titik tengah dari setiap obyek yang ada pada citra.

Menghitung Obyek

Menghitung obyek (*objects count*) merupakan proses menampilkan nilai dari citra yang telah dilabeli, seberapa banyak obyek yang ada pada citra dan berupa angka yang ditampilkan di *command window*. Proses ini juga berjalan bersama proses pelabelan, sehingga untuk hasilnya pasti akan menghasilkan jumlah obyek yang sesuai dengan proses *region properties*. *Objects count* juga berfungsi untuk mengetahui tingkat akurasi dalam menghitung obyek yang ada dalam sebuah citra.

Pemisahan Obyek

Pemisahan obyek (*crop objects*) merupakan proses pemotongan dari obyek antara satu dengan yang lainnya, tergantung banyaknya citra yang ada pada sebuah citra. Obyek yang sudah berhasil terdeteksi, kemudian akan dipisahkan atau dipotong sesuai dengan banyaknya obyek yang ada pada citra *region properties*. Semakin banyak obyek yang terdeteksi, maka akan semakin banyak pula citra baru yang akan ditampilkan.

Ground Truth

Ground Truth merupakan citra pembanding dengan citra hasil segmentasi, pada *Weizmann Segmentation Database* tidak terdapat citra *Ground Truth*. Jadi citra *Ground Truth* untuk Tugas Akhir ini didapat berdasarkan hasil pengamatan *user* secara langsung.

Citra hasil segmentasi dibandingkan dengan citra *Ground Truth* dari hasil pengamatan *user* secara langsung untuk mengetahui tingkat akurasi dari proses segmentasi obyek pada citra digital menggunakan Metode Otsu *thresholding*. Formula untuk menghitung tingkat akurasi dapat dilihat pada persamaan (9):

$$Akurasi = \frac{Jumlah\ Citra\ Benar}{Jumlah\ Total\ Citra} \times 100 \qquad (9)$$

ALUR KERJA SISTEM

Alur kerja sistem pada penelitian ini dibagi menjadi lima tahapan. Tahap pertama adalah *input* data citra digital RGB dengan menggunakan 30 citra RGB yang didapat dari *Weizmann Segmentation Database*. Tahap kedua adalah konversi dari citra RGB ke citra *grayscale*. Tahap ketiga adalah segmentasi menggunakan Metode Otsu *thresholding*, kemudian dikonversi ke dalam citra biner. Tahap keempat adalah proses *invert image*, *noise removal* dengan nilai ambang 150, dan *morphology* menggunakan *filling holes*. Tahap terakhir adalah proses perhitungan tingkat akurasi diantaranya adalah proses *labelling*, *regionprops*, *objects count*, dan *crop objects*, dan akurasi dengan *Ground Truth*.

Gambar 1. Diagram alur kerja sistem

HASIL DAN PEMBAHASAN

Uji coba yang dilakukan adalah untuk mengetahui nilai jumlah total piksel (*L*) dan intensitas piksel (*i*) yang sesuai dengan rumus Metode Otsu *thresholding*. Uji coba yang lain adalah penggunaan kondisi pada *invert image* yang tepat, agar hasil *noise removal* lebih akurat. Uji coba yang selanjutnya adalah menentukan nilai ambang dari *noise removal*, mulai dari 10, 50, 100, 150, dan 200. Kemudian tingkat akurasi dari beberapa rangkaian proses yang dianggap berpengaruh. Ada beberapa rangkaian uji coba yang dilakukan. Pada skenario uji coba, yang diuji coba adalah nilai total piksel, nilai intensitas piksel pada Metode Otsu *thresholding*, penggunaan kondisi dan tanpa kondisi pada *invert image* serta parameter pada *noise removal* yaitu nilai ambang.

Metode Otsu *thresholding*

Pada skenario ini dilakukan dua pengujian, yakni menentukan total jumlah intensitas piksel (*L*) dan intensitas piksel (*i*) yang sesuai dengan rumus Metode Otsu *thresholding*. Dapat dilihat pada Tabel 1.

Tabel 1. Skenario uji coba nilai *L* dan *i* pada Metode Otsu

Skenario Uji Coba Metode Otsu *thresholding*	Uji Coba Total Jumlah Intensitas Piksel (*L*) dan Intensitas Piksel (*i*)
Skenario uji coba 1	*L* = 255 dan *i* = 0,1,2, ..., *L*
Skenario uji coba 2	*L* = 256 dan *i* = 0,1,2, ..., *L*-1

Hasil nilai ambang dari proses segmentasi menggunakan Metode Otsu *thresholding* dapat dilihat pada Tabel 2.

Tabel 2. Metode Otsu *thresholding*

Citra	*L* = 255 *i* = 0, 1, 2, ..., *L*	Verifikasi	*L* = 256 = 0, 1, 2, ..., *L*-	Verifikasi
01.png	0.21176	Tidak	0.20784	Sama
02.png	0.62353	Tidak	0.61961	Sama
03.png	0.51373	Tidak	0.5098	Sama
04.png	0.35294	Sama	0.35294	Sama
05.png	0.49804	Tidak	0.49412	Sama
06.png	0.41176	Tidak	0.40784	Sama
07.png	0.52549	Sama	0.52549	Sama
08.png	0.55686	Tidak	0.55294	Sama
09.png	0.44706	Sama	0.44706	Sama
10.png	0.70588	Sama	0.70588	Sama
11.png	0.72157	Sama	0.72157	Sama
12.png	0.43137	Tidak	0.42745	Sama
13.png	0.43137	Sama	0.43137	Sama
14.png	0.39608	Sama	0.39608	Sama
15.png	0.47059	Sama	0.47059	Sama
16.png	0.37647	Tidak	0.37255	Sama
17.png	0.36863	Tidak	0.36471	Sama
18.png	0.62745	Sama	0.62745	Sama
19.png	0.35686	Tidak	0.35294	Sama
20.png	0.39216	Sama	0.39216	Sama
21.png	0.58431	Tidak	0.58039	Sama
22.png	0.52157	Tidak	0.51765	Sama
23.png	0.33725	Tidak	0.33333	Sama
24.png	0.44314	Sama	0.44314	Sama
25.png	0.58824	Sama	0.58824	Sama
26.png	0.21961	Tidak	0.21569	Sama
27.png	0.43137	Tidak	0.42745	Sama
28.png	0.54118	Sama	0.53725	Sama
29.png	0.36863	Sama	0.36863	Sama
30.png	0.56863	Sama	0.56863	Sama

Hasil skenario uji coba 1 dengan menggunakan nilai *L* = 255 dan *i* = 0, 1, 2, ..., *L*. Memiliki nilai *threshold* (ambang) yang berbeda setelah dibandingkan dengan nilai ambang dari *thresholding* menggunakan Otsu *toolbox* Matlab. Nilai ambang yang sama untuk lima digit setelah koma hanya pada citra 04, 07, 09, 10, 11, 13, 14, 15, 18, 20, 24, 25, 29, dan 30. Sedangkan untuk sisanya, nilainya berbeda. Sedangkan untuk hasil skenario uji coba 2 dengan menggunakan nilai *L* = 256 dan *i* = 0, 1, 2, ..., *L*-1. Memiliki nilai *threshold* (ambang) yang sama dengan nilai ambang dari *thresholding* menggunakan Otsu *toolbox* Matlab.

Invert Image

Pada skenario ini, dilakukan proses mengubah warna piksel hitam menjadi warna piksel putih. Begitu sebaliknya, untuk warna piksel warna putih menjadi warna piksel hitam. Ada dua skenario uji coba yang dilakukan, yang pertama adalah *invert image* tanpa kondisi dan yang kedua dengan menggunakan kondisi. Hasil akhir dari proses invert image adalah citra dengan warna obyek adalah putih, atau nilai piksel obyek adalah 0. Skenario uji coba dapat dilihat pada Tabel 3.

Tabel 3. Skenario uji coba *invert image* tanpa dan dengan menggunakan kondisi

Skenario Uji Coba *Invert Image*	Uji Coba
Skenario uji coba 1	Tanpa Kondisi
Skenario uji coba 2	Menggunakan Kondisi

Hasil verifikasi dari proses *invert images* dapat dilihat pada Tabel 4.

Tabel 4. *Invert image*

Tanpa Kondisi			Menggunakan Kondisi		
Citra	Berhasil		Citra	Berhasil	
	Iya	Tidak		Iya	Tidak
01.png	Iya		01.png	Iya	
02.png	Iya		02.png	Iya	
03.png	Iya		03.png	Iya	
04.png	Iya		04.png	Iya	
05.png	Iya		05.png	Iya	
06.png	Iya		06.png	Iya	
07.png		Tidak	07.png	Iya	
08.png	Iya		08.png	Iya	
09.png	Iya		09.png	Iya	
10.png	Iya		10.png	Iya	
11.png	Iya		11.png	Iya	
12.png	Iya		12.png	Iya	
13.png	Iya		13.png	Iya	
14.png	Iya		14.png	Iya	
15.png	Iya		15.png	Iya	
16.png		Tidak	16.png	Iya	
17.png	Iya		17.png	Iya	
18.png		Tidak	18.png	Iya	
19.png	Iya		19.png	Iya	
20.png	Iya		20.png	Iya	
21.png		Tidak	21.png	Iya	
22.png	Iya		22.png	Iya	
23.png	Iya		23.png	Iya	
24.png	Iya		24.png	Iya	
25.png		Tidak	25.png	Iya	
26.png	Iya		26.png	Iya	
27.png	Iya		27.png	Iya	
28.png		Tidak	28.png	Iya	
29.png	Iya		29.png	Iya	
30.png	Iya		30.png	Iya	

Hasil skenario uji coba 1 *invert image* tanpa menggunakan kondisi, terdapat 6 citra yang tidak berhasil. Diantaranya adalah citra 07, 16, 18, 21, 25,

dan 28, sisanya berhasil semua. Sedangkan untuk sisanya berhasil semua. Sedangkan untuk hasil skenario uji coba 2 *invert image* dengan menggunakan kondisi, semua citra berhasil dilakukan proses *invert image*.

Nilai Ambang *Noise Removal*

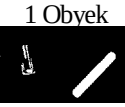
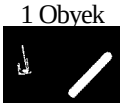
Pada skenario ini, untuk nilai ambang dari *noise removal* yang dicoba adalah 10, 50, 100, 150, dan 200. Nilai ambang ini yang nantinya akan mempengaruhi penghapusan *noise* yang ada pada citra dan bukan termasuk dalam obyek. Dapat dilihat pada Tabel 5.

Tabel 5. Skenario uji coba nilai ambang *noise removal*

Skenario Uji Coba *Noise Removal*	Uji Coba
Skenario uji coba 1	Nilai Ambang 10
Skenario uji coba 2	Nilai Ambang 50
Skenario uji coba 3	Nilai Ambang 100
Skenario uji coba 4	Nilai Ambang 150
Skenario uji coba 5	Nilai Ambang 200

Pada Tabel 6 merupakan contoh citra dari 30 citra yang telah diuji coba. Hasil akurasi dari proses penghapusan *noise* dengan menggunakan nilai ambang yang berbeda menghasilkan citra dengan akurasi yang semakin bagus.

Tabel 6. Hasil *noise removal*

Nilai Ambang 10	Nilai Ambang 50	Nilai Ambang 100	Nilai Ambang 150	Nilai Ambang 200
				
09.png 46 Obyek	09.png 3 Obyek	09.png 2 Obyek	09.png 2 Obyek	09.png 2 Obyek
				
19.png 2 Obyek	19.png 2 Obyek	19.png 2 Obyek	19.png 2 Obyek	19.png 1 Obyek
				
20.png 60 Obyek	20.png 12 Obyek	20.png 5 Obyek	20.png 2 Obyek	20.png 2 Obyek
				
24.png 7 Obyek	24.png 3 Obyek	24.png 2 Obyek	24.png 2 Obyek	24.png 2 Obyek
				
29.png 1 Obyek	29.png 1 Obyek	29.png 1 Obyek	29.png 1 Obyek	29.png 1 Obyek
				
30.png 5 Obyek	30.png 4 Obyek	30.png 4 Obyek	30.png 4 Obyek	30.png 3 Obyek

Dari 30 citra uji coba, dua citra mengalami *error* mulai dari proses segmentasi, yakni citra 20 dan citra 30. Sedangkan untuk sisanya, tidak mengalami masalah.

Hasil skenario uji coba dapat dilihat pada Tabel 6. Untuk uji coba 1 penghapusan *noise* dengan nilai ambang 10, citra yang berhasil menampilkan obyek secara tepat adalah citra nomor 01, 02, 05, 10, 14, 19, 22, 27, 28, dan 29.

Hasil skenario uji coba 2 *noise removal* dengan nilai ambang 50, citra yang gagal menampilkan obyek secara tepat adalah citra nomor 06, 09, 11, 12, 20, 24, dan 30.

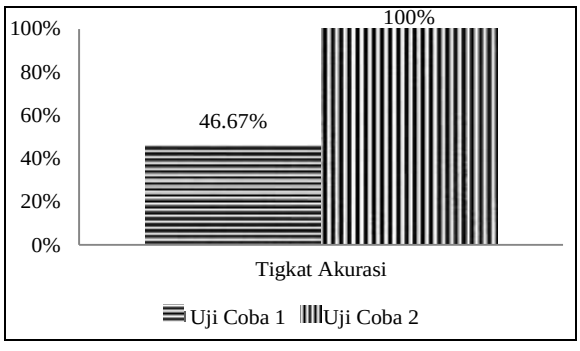
Hasil skenario uji coba 3 *noise removal* dengan nilai ambang 100, citra yang gagal menampilkan obyek secara tepat adalah citra nomor 20, dan 30.

Hasil skenario uji coba 4 *noise removal* dengan nilai ambang 150, citra yang gagal menampilkan obyek secara tepat adalah citra nomor 20 dan 30, *noise* sudah mulai banyak berkurang dari sebelumnya.

Hasil skenario uji coba yang terakhir *noise removal* dengan nilai ambang 200, citra yang gagal menampilkan obyek secara tepat adalah citra nomor 20 dan 30, ditambah dengan citra 19 yang bahkan seharusnya menjadi obyek dianggap sebagai *noise*, sehingga citra mengalami *error*. Salah satu obyek yang ada pada citra 19 terdeteksi sebagai *noise* ketika dilakukan penghapusan *noise* dengan nilai ambang 200, disebabkan oleh ukuran dari obyek terlalu kecil. Sehingga *noise removal* dengan nilai ambang 200, tidak bisa digunakan.

Analisa Uji Coba Terhadap Metode Otsu *thresholding*

Dari uji coba terhadap total jumlah intensitas piksel (L) dan intensitas piksel (i) untuk Metode Otsu *thresholding*, didapat tingkat akurasi dari masing-masing percobaan dapat dilihat pada Gambar 2. Uji coba 1 yaitu menggunakan nilai $L = 255$ dan $i = 0, 1, 2, ..., L$. Uji coba 2 terhadap yaitu nilai $L = 256$ dan $i = 0, 1, 2, ..., L-1$ pada proses segmentasi citra menggunakan Metode Otsu. Hasilnya dapat dilihat pada Gambar 2.

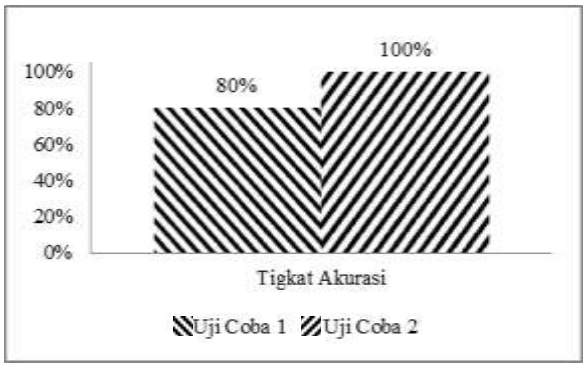


Gambar 2. Grafik akurasi Metode Otsu *thresholding*

Dari Gambar 2 didapat bahwa untuk Metode Otsu lebih bagus menggunakan $L = 256$ dan $i = 0, 1, 2, ..., L-1$ dibandingkan dengan $L = 255$ dan $i = 0, 1, 2, ..., L$. Hal ini dikarenakan pada rerata kumulatif kelas dan rerata intensitas global, nilai piksel dimulai dari 1 : L dengan nilai $L = 256$ apabila dimulai dari 0 : L dengan nilai $L = 255$ maka tidak akan diproses dan mengalami *error*. Serta nilai maksimal juga tidak dilakukan pengurangan.

Analisa Uji Coba Terhadap *Invert Image*

Dari uji coba terhadap *invert image* tanpa menggunakan kondisi dan dengan menggunakan kondisi, didapat tingkat akurasi dari masing-masing percobaan. Hasilnya dapat dilihat pada Gambar 3. Uji coba 1 tanpa menggunakan kondisi, dan untuk uji coba 2 dengan menggunakan kondisi pada proses *invert images*.



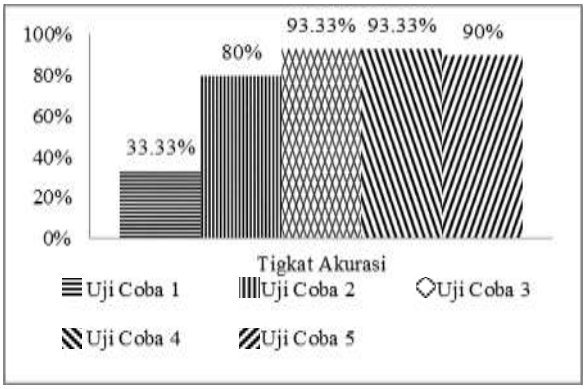
Gambar 3. Grafik akurasi *invert image*

Dari Gambar 3 didapat bahwa untuk proses *invert image* lebih bagus menggunakan kondisi dibandingkan dengan tanpa menggunakan kondisi. Hal ini dikarenakan pada proses tanpa menggunakan kondisi langsung dilakukan proses *invert* tanpa memperhatikan beberapa hal di dalamnya. Padahal obyek sudah berwarna putih atau memiliki nilai 1. Hasilnya citra yang sudah memiliki obyek piksel warna putih (1) tetap dikonversi ke hitam.

Berbeda pada saat menggunakan kondisi, citra yang hanya memiliki piksel warna putih (1) lebih banyak yang akan dilakukan proses *invert image*. Sedangkan untuk citra yang memiliki piksel warna putih (1) lebih sedikit, tidak akan mengalami proses *invert image*. Hasil dari proses *invert images* lebih bagus dengan menggunakan kondisi.

Analisa Uji Coba Terhadap Nilai Ambang *Noise Removal*

Dari uji coba terhadap nilai ambang pada *noise removal*, didapat tingkat akurasi dari masing-masing percobaan dan dapat dilihat pada Gambar 4. Uji coba 1 dengan menggunakan nilai ambang 10, uji coba 2 dengan menggunakan nilai ambang 50, uji coba 3 dengan menggunakan nilai ambang 100, uji coba 4 dengan menggunakan nilai ambang 150, dan untuk Uji coba 5 dengan menggunakan nilai ambang 200 pada proses *noise removal*. Hasilnya dapat dilihat pada Gambar 4.



Gambar 4. Grafik akurasi nilai ambang *noise removal*

Dari Gambar 4 didapat bahwa untuk nilai ambang dari *noise removal* yang menghasilkan perbaikan citra lebih bagus adalah nilai ambang 150. Hal ini dikarenakan pada penghapusan *noise* dengan nilai ambang 10, 50, dan 100 masih tidak bisa menghapus *noise* dengan ukuran lebih dari 100. Pada saat menggunakan nilai ambang 200, justru yang terdeteksi sebagai obyek dianggap sebuah *noise*, karena ukuran pikselnya yang kecil di bawah 200.

KESIMPULAN DAN SARAN

Dari tiga skenario uji, masing-masing diambil satu uji coba dengan hasil citra segmentasi yang bagus, yakni ketika menggunakan nilai $L = 256$ dan $i = 0, 1, 2, ..., L-1$. Pada proses *invert image* dengan menggunakan kondisi, agar proses *noise removal* lebih bagus. Terakhir adalah dengan menggunakan nilai ambang 150 pada proses *noise removal*.

1. Skenario uji pertama yang memenuhi syarat adalah pada uji coba 2 dengan nilai $L = 256$ dan $i = 0, 1, 2, ..., L-1$ akurasinya sebesar 100%. Skenario kedua yang memenuhi syarat adalah pada uji coba 2 dengan menggunakan kondisi saat dilakukan proses *invert image*, akurasinya sebesar 100%. Skenario ketiga yang memenuhi syarat adalah pada uji coba 4 dengan nilai ambang 150 pada *noise removal*, dengan tingkat akurasi sebesar 93,33%.
2. Penelitian tentang segmentasi obyek menggunakan metode otsu, dapat ditambah dengan deteksi tepi pada obyek agar obyek yang memiliki tingkat

kecerahan di atas nilai ambang tidak akan dianggap sebagai *background* dan akan tetap dianggap sebagai obyek. Proses deteksi tepi pada obyek dapat menggunakan Prewitt, Roberts, Canny, Sobel, atau menggunakan Laplacian Of Gaussian (LOG).

3. Penelitian tentang segmentasi obyek, dapat dikembangkan lagi untuk pengenalan obyek. Seperti pengenalan benda-benda, deteksi wajah, dan deteksi penyakit pada citra medis.

DAFTAR PUSTAKA

[1] Nabella, W. M., Sampurno, J., Nurhasanah. Analisis Citra Sinar-X Tulang Tangan Menggunakan Metode Thresholding Otsu untuk Identifikasi Osteoporosis. *POSITRON*, Vol. III, No. 1, hal. 12-15. 2013.

[2] Handoko, W.T., Ardhianto, E., Safriliyanto, E. Analisis dan Implementasi Image Denoising dengan Metode Normal Shrink sebagai Wavelet Thresholding Analysis. *DINAMIK Jurnal Teknologi Informasi*, Vol. 16, No. 1, hal. 56-63. 2011.

[3] Setiawan, A., Suryani, E., Wiharto. Segmentasi Citra Sel Darah Merah Berdasarkan Morfologi Sel untuk Mendeteksi Anemia Defisiensi Besi. *Jurnal Jurusan Informatika*, Universitas Sebelas Maret. 2013.

[4] Ardhianto, E., Hadikurniawati, W., Budiarso, Z. Implementasi Metode Image Subtracting dan Metode Regionprops untuk Mendeteksi Jumlah Obyek Berwarna RGB pada File Video. *DINAMIK Jurnal Teknologi Informasi*. Vol. 18, No.2, Hal. 91-100. 2013.

[5] Mandalasari, A.F., Uyun, S. *Segmentasi Citra Medis Menggunakan Metode Otsu dan Iterasi*. Tugas Akhir Teknik Informatika, Fakultas Sains dan Teknologi, UIN Sunan Kalijaga. 2013.

[6] Kumaseh, M. R., Latumakulita, L., Nainggolan, N. Segmentasi Citra Digital Ikan Menggunakan Metode Thresholding. *Jurnal Ilmiah Sains*, Vol. 13 No. 1. 2013.

[7] Fauzi, F., Arnia, F. Analisis Kinerja Metode Binerisasi pada Proses Pemisahan Text dari Background Menggunakan Perangkat Lunak OCR. *KITEKTRO Jurnal Online Teknik Elektro*. Vol.1, No.2 : 25-32, 2012.

[8] Cahyaningsih, S., Mulyono, A., Abidin, M. *Deteksi Osteoporosis dengan Thresholding Metode Otsu pada Citra X-Ray Tulang Rahang*. Tugas Akhir Jurusan Fisika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim. 2010.

[9] Otsu, N. A Threshold Selection Method from Gray-Level Histogram. *IEEE Transaction on Systems, Man, and Cybernetics*. Vol. SMC-9, 1. 1979.

[10] Subekti, I., Purnama, I. K. E., Purnomo, M. H. *Identifikasi Sel Darah Berbentuk Sabit Pada Citra Sel Darah Penderita Anemia*. Tugas Akhir Jurusan Teknik Elektro FTI, Institut Teknologi Sepuluh Nopember. 2013.