REDUKSI WARNA CITRA DOKUMEN DIGITAL MENGGUNAKAN TEKNIK EPSF

SKRIPSI

Oleh :

FARID SUKMANA
(0610960025-96)

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2011

REDUKSI WARNA CITRA DOKUMEN DIGITAL MENGGUNAKAN TEKNIK EPSF

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Oeh :

FARID SUKMANA
(0610960025-96)

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2011

LEMBAR PENGESAHAN SKRIPSI

Reduksi Warna Citra Dokumen Digital Menggunakan Teknik EPSF

Oleh:
FARID SUKMANA
0610960025-96

Setelah dipertahankan di depan Majelis Penguji
Pada tanggal 04 Januari 2011
dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana
dalam bidang Ilmu Komputer

Dosen Pembimbing I	Dosen Pembimbing II
Drs. Marji, MT	**Nurul Hidayat, Spd.,M.Sc**
NIP. 19660423199111 1001	**NIP. 19680430 2002121 001**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA UniversitasBrawijaya

Dr. Agus Suryanto, M.Sc
NIP. 19690807 199412 1 001

UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama : Farid Sukmana
NIM : 0610960025-96
Jurusan : Matematika
Penulis tugas Akhir berjudul : Reduksi Warna Citra Dokumen Digital Menggunakan Teknik EPSF

Dengan ini menyatakan bahwa :

1. **Isi dari tugas Akhir yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam Tugas Akhir ini.**
2. **Apabila dikemudian hari ternyata Tugas Akhir yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 04 April 2011
Yang menyatakan,

(Farid Sukmana)
NIM. 0610960025

UNIVERSITAS BRAWIJAYA

TEKNIK EPSF PADA REDUKSI WARNA DOKUMEN DIGITAL MENGGUNAKAN *MEAN SHIFT PROCEDURE*

ABSTRAK

Segmentasi merupakan salah satu bagian pengolahan citra digital yang digunakan untuk memisahkan pola-pola gambar. Sehingga dari suatu citra didapatkan informasi dari objek utama yang dicari. Permasalahan dalam penelitian ini yaitu jika citra yang digunakan adalah citra dokumen digital yang memiliki tekstur garis dan warna yang rumit dan komplek. Untuk itu sebelum dilakukan segmentasi citra objek utama perlu dilakukan penyederhanaan tekstur-tekstur tersebut sehingga memungkinkan citra dokumen tersebut memiliki keseragaman *background*. Dan untuk dapat dihasilkan penyederhaan *background* perlu dilakukan reduksi warna pada dokumen baik objek maupun background. Inilah yang mendasari dilakukan penelitian ini. Dan untuk mereduksi warna dari *pixel* yang diambil dalam suatu citra terlebih dahulu warna *pixel* tersebut dikelompokkan sebagai kandidat pusat *cluster* yang ditampilkan dalam sebaran warna. Dan dari kandidat pusat cluster yang ada diolah untuk mendapatkan pusat *cluster* akhir sebagai nilai pixel untuk citra hasil. Untuk mendapatkan pusat cluster akhir, pixel-pixel pada citra dilewatkan melalui beberapa tahapan antara lain *Edge Preserving Smoothing Filter(EPSF), Sobel Edge Detection, Initial Color, Mean Shift Procedure* dan *Manhattan Distance*. Teknik EPSF pada tahapan tersebut digunakan untuk mengurangi noise dalam citra sedangkan untuk melakukan pengelompokkan atau *clustering* pixel menggunakan *mean shift procedure*. Dan citra yang dihasilkan dari *output* program digunakan 2 citra hasil untuk membandingkan kedua citra.Dimana keduanya memiliki input parameter yang sama dan yang berbeda citra hasil pertama dilewatkan melalui EPSF dan citra kedua tanpa melalui EPSF. Untuk memperjelas hasil kedua citra dilakukan konversi ke citra biner dengan *threshold* yang ditentukan oleh system. Konversi ke citra biner dimaksudkan untuk melihat efisiensi hasil reduksi warna objek maupun background. Berdasarkan hasil uji coba diperoleh citra tanpa memalui EPSF memiliki kualitas yang lebih baik dibandingkan dengan citra yang melalui EPSF ditunjukkan dengan *error* yang lebih kecil. Namun dari segi citra biner kualitas citra yang melalui EPSF lebih baik digunakan untuk segmentasi dibandingkan citra tanpa

melalui EPSF. Dikarenakan pada citra tersebut lebih sedikit noise jika dibandingkan dengan citra tanpa melalui EPSF.

EPSF IN REDUCTION COLOR FOR DIGITAL DOCUMENT WITH MEAN SHIFT PROCEDURE

ABSTRACT

Image segmentation is one of part image processing which be used to segment pattern in image. So could got it information from aim object. The problem in this research if image which used is digital document image which has complex pattern textur and color. Before that aim object need texturs simplification so could has background is homogenous. And to get result background is more simply needed color reduction for document to object or background. This is the basic of the research. And to reduct color from pixel whict take from original image, first all pixel will be clustered and will be final cluster. To look distribution pixel of image will be appeared in 2D color distribution. And cluster center candidate will be processed to ge final center cluster as pixelvalue for final image. To get final center cluster .pixels in image will pass some step like Edge Preserving Smoothing Filter (EPSF), Sobel Edge Detection, Initial Color, Mean Shift Procedure dan Manhattan Distance. EPSF is step will be used to reduct noise from image , to clustering pixel the step which used is mean shift procedure. And image which resulted from the output of program will be used two image to compare both of them, first image use EPSF but twice image not pass EPSF step. And to get good observation the result of them will be converted to binary image with threshold which given by sytem. Convertion to binary image be intended to look efficiency of color reduction for object or background. On the strength of the experiment be get ,image without passed EPSF has more qulity than image which passed EPSF. That be pointed by a few error in image which passed EPSF. But from quality of binary image , image which passed EPSF has more quality than image without passed EPSF, because that image has a few noise than image without passed EPSF.

UNIVERSITAS BRAWIJAYA

KATA PENGANTAR

Puji syukur kami panjatkan kehadirat Allah SWT atas limpahan rahmat dan hidayah-Nya, sehingga saya dapat menyelesaikan skripsi dengan judul "Teknik *EPSF* pada Reduksi Warna Citra Dokumen Digital Menggunakan *Mean Shift Procedure* " dengan baik dan lancar. Skripsi ini merupakan salah satu syarat untuk menyelesaikan program sarjana Strata-1 pada Program Studi Ilmu Komputer Jurusan Matematika Universitas Brawijaya.

Peneliti berharap dengan adanya penelitian ini dapat diteruskan dalam pengenalan tulisan citra dokumen digital.

Dalam menyelesaikan skripsi ini saya mendapatkan banyak bantuan, bimbingan dan petunjuk dari berbagai pihak. Pada kesempatan ini saya ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Drs. Marji, MT selaku pembimbing I dan Ketua Program Studi Ilmu Komputer Fakultas MIPA Universitas Brawijaya.
2. Nurul Hidayat, SPd. MSc pembimbing II Program Studi Ilmu Komputer Fakultas MIPA Universitas Brawijaya.
3. Dr. Drs. Agus Suryanto, MSc selaku Ketua Jurusan Matematika Fakultas MIPA Universitas Brawijaya.
4. Orang Tua kami, yang selalu memberikan doa agar kami berhasil.
5. Teman-teman dari Program Studi Ilmu Komputer Universitas Brawijaya yang telah memberikan bantuan dan bimbingan dalam penyelesaian tugas dan penyusunan laporan Skripsi ini.

Saya berharap skripsi ini bermanfaat bagi saya selaku mahasiswa yang melakukan penelitian, serta kepada para pembaca laporan skripsi ini. Dan saya menyadari masih banyak kekurangan dalam laporan skripsi ini karena keterbatasan kemampuan dan ilmu. Oleh karena itu, saya sangat mengharapakan saran dan kritik yang membangun.

Malang, 25 Nopember 2010

Penyusun

UNIVERSITAS BRAWIJAYA

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

UNIVERSITAS BRAWIJAYA

DAFTAR *SOURCE CODE*

UNIVERSITAS BRAWIJAYA

BAB I
PENDAHULUAN

1.1 Latar Belakang

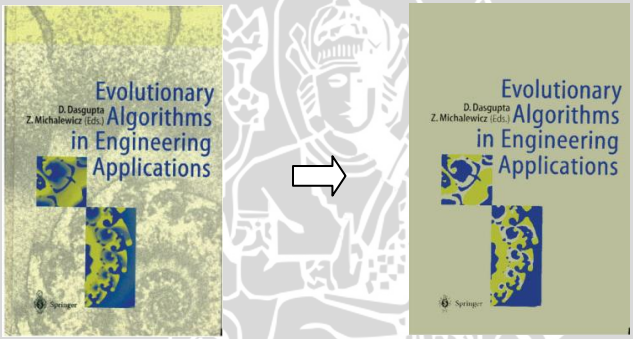
Segmentasi merupakan salah satu bagian dari pengolahan citra yang digunakan untuk memisahkan pola-pola yang terdapat pada gambar baik warna maupun tekstur dari gambar. Biasanya suatu pola dalam suatu gambar dipisahkan berdasarkan tekstur warnanya. Dan menjadikan area satu dengan yang lainnya berbeda. Pada dokumen digital bergambar dimisalkan suatu sampul buku pada umumnya memiliki corak warna yang beragam. Sehingga untuk mendapatkan objek utama perlu adanya penyederhanaan terhadap tekstur dari *background*. Untuk menyederhanakan tekstur tersebut maka perlu adanya keseragaman dari warna *backcground* dengan melakukan pengurangan warna asli dari suatu dokumen digital bergambar baik warna objek maupun *background* atau lebih dikenal dengan reduksi warna.

Reduksi warna pada dokumen digital berguna untuk memisahkan area *background* dan objek utama. Dengan keanekaragaman pola warna yang tinggi maka perlu adanya suatu keseragaman warna yang berdekatan. Karena tujuan ketika mencetak suatu dokumen yaitu untuk mendapatkan informasi yang sedetail mungkin. Karena dokumen akan berbeda penyajian informasi jika secara visual berbeda. Informasi yang dimaksud yaitu bagaimana seseorang mengambil isi dari suatu dokumen misal adalah kata-kata yang terdapat pada dokumen tersebut. Untuk mengurangi ketidakseragaman warna *background*, garis grafik dan tekstur warna maka perlu dilakukan pengurangan jumlah warna dan mengatur warna asli menuju warna hasil yang lebih sedikit dari sebelumnya.

Dokumen digital selain memiliki ketidakseragaman pola warna juga memiliki tingkat *noise* yang tingggi dan biasanya *noise* ini tidak diinginkan adanya dalam suatu dokumen. Sehingga perlu adanya pengurangan terhadap *noise* dengan melakukan rahap *preprocessing* (*smoothing*)

Dari beberapa permasalahan di atas untuk memperoleh hasil reduksi warna dokumen digital maka diperlukan beberapa metode untuk mengerjakannya. Untuk megurangi *noise* dari citra yaitu dengan menggunakan teknik *edge preserving smoothing filter*

(*EPSF*). Menurut Nikolou, Papmarkos (2008:1) teknik tersebut dibandingkan dengan teknik *smoothing* yang lain dapat menjaga detail objek yang baik. Detail objek yang dimaksud disini adalah area transisi antara objek dengan *background*. Sedangkan untuk mengurangi jumlah warna citra asli dilakukan tahap *sobel edge detection* untuk mengambil sampel warna dari citra dan *mean shift procedure* untuk memperoleh warna yang lebih sedikit dari sebelumnya dengan mencari warna yang paling dekat dengan menggunakan rata-rata nilai *cluster* (kumpulan nilai-nilai pixel dalam area yang dipilih). Penggunaan teknik tersebut telah diteliti oleh Nikolou Papmarkos dan hasil penelitian terhadap citra dokumen digital pada gambar 1.1 sebagai gambar asli dan gambar 1.2 sebagai gambar hasil reduksi.



Gambar 1.1 Sebelum Direduksi Gambar 1.2 Setelah Direduksi

Berdasarkan permasalahan yang ada tersebut dan metode yang digunakan maka judul yang diambil dalam penelitian ini yaitu "Reduksi Warna Citra Dokumen Digital Menggunakan Teknik *EPSF*".

1.2 Rumusan Masalah

Adapun permasalah yang akan dibahas dalam penelitian ini antara lain :

a. Bagaimana menerapkan teknik *EPS*F untuk melakukan reduksi warna pada citra dokumen digital menggunakan deteksi tepi *Sobel* dan *Mean Shift Procedure*.

b. Bagaimana pengaruh *h* (tinggi area) dan jumlah iterasi pada *mean shift procedure* terhadap hasil penelitian.

1.3 Batasan Masalah

a. Penggunaan citra terbatas pada citra bergambar yang memiliki *background* dan objek, bukan warna *grayscale*.
b. Tidak dilakukan penyimpanan terhadap warna putih karena akan terjadi permasalahan dalam penyimpanan ke area distribusi 2D pada program.
c. Pengujian citra visual diuji terbatas untuk 20 orang.

1.4 Tujuan Pembahasan

a. Menerapkan teknik *EPSF* untuk melakukan reduksi warna pada citra dokumen digital menggunakan deteksi tepi *Sobel* dan *Mean Shift Procedure*.
b. Mengetahui pengaruh *h* (tinggi area) dan jumlah iterasi pada *mean shift procedure* terhadap hasil penelitian.

1.5 Manfaat

Dengan adanya penelitian ini diharapkan mampu memecahkan permasalahan dalam segmentasi citra dokumen digital yang memiliki tingkat kerumitan yang tinggi disebabkan *background* dan pola warna yang tidak seragam.

1.6 Metodolgi Pemecahan Masalah

Metodologi yang digunakan untuk memecahkan masalah dalam penyusunan tugas akhir ini meliputi :

1. Studi literatur
   Mempelajari teori-teori yang berhubungan dengan metode-metode yang digunakan dalam *image processing.*
2. Pendefinisian dan analisis masalah
   Mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat.
3. Perancangan dan implementasi sistem
   Membuat perancangan perangkat lunak dengan analisis terstruktur dan mengimplementasikan hasil rancangan tersebut yaitu Teknik *EPSF* pada reduksi warna citra dokumen digital menggunakan deteksi tepi *sobel* dan *mean shift procedure*

4. Uji coba dan analisa hasil implementasi
   Menguji perangkat lunak, dan menganalisa hasil dari implementasi tersebut apakah sudah sesuai dengan tujuan yang dipersamaankan sebelumnya, untuk kemudian menarik kesimpulannya.

1.7 Sistematika Penulisan

Sistematika penulisan laporan tugas akhir ini adalah sebagai berikut :

BAB I Pendahuluan

Berisi uraian tentang latar belakang , persamaanan masalah, batasan masalah , tujuan pembahasan , metodologi pemecahan masalah, dan sistematika penulisan laporan.

BAB II Tinjauan Pustaka

Berisi dasar teori tentang metode dalam menentukan tiap langkah dalam penelitian.

BAB III Metodologi Penelitian

Berisi uraian tentang konsep-konsep dan tahap-tahap dalam melakukan reduksi warna pada komplek dokumen bergambar.

BAB IV Implemetnasi dan Pembahasan

Dalam bab ini dijelaskan penerapan algoritma ke dalam program yang sebenarnya dan melakukan analaisa hasil program dari reduksi warna.

BAB VI Penutup

Bab ini berisi kesimpulan akhir serta saran-saran yang berguna dalam perbaikan dan pengembangan lebih lanjut.

BAB II
TINJAUAN PUSTAKA

2.1 *Edge Preserving Smoothing Filter* (EPSF)

Edge preserving dalam dunia pengolahan citra digital digunakan untuk melakukan segmentasi gambar. Walaupun banyak algoritma *filtering* yang bisa digunakan tetapi hanya beberapa yang memberikan hasil yang memuaskan. Dan dalam beberapa kasus terhadap deteksi objek, algoritma yang diterapkan tidak efisien sehingga menyebabkan proses deteksi bekerja dibawah standard. Oleh Karena itu dibutuhkan algoritma yang dapat diandalkan. (Garnica, 2000 : 2)

Tujuan dari *edge preserving* yaitu untuk melakukan *smoothing* sehingga *noise* dalam citra dapat dihilangkan. Seperti yang diungkapkan Garnica (2000:2) bahwa dalam setiap *image* digital pasti terdapat sejumlah *noise* putih. Dan dalam segmentasi *image* ada berbagai *feature* yang dapat ditemukan seperti elemen tepi gambar yang kecil sebagai penanda adanya *noise* dari *image*. sehingga dalam melakukan segmentasi pertama kali harus dilakukan *filtering* dengan menggunakan citra yang sebenarnya.

Smoothing dengan menggunakan *Gaussian filter* bertujuan untuk melakukan penghalusan dan menjaga area tepi dari suatu objek agar lebih baik daripada sebelumnya. Karena Gaussian digunakan dalam meningkatkan tingkat kehalusan dari suatu citra. Dan dalam Gaussian tingkat kehalusan ditentukan oleh standar deviasi. (Fisher, Perkins, Walker dan Wolfart, 2003)

Teknik *smoothing* yang digunakan dalam penelitian ini yaitu *EPSF* dengan mengabaikan penggunaan teknik *gaussian*. Hal ini diungkapkan langsung Nikolaou dan Paparmakos (2008:16) yang menyatakan bahwa metode yang digunakan seperti *gaussian filter* memang mampu menghilangkan *noise* dari suatu *image* tapi menghilangkan informasi dan detail dari batasan (tepi) objek yang dibutuhkan pada citra. Maka dari itu untuk mengatasi masalah itu digunakan metode *EPSF*.

Dalam *EPSF* digunakan koefisien *convolution mask* (*ci*) untuk tiap *pixel*. Koefisien ini digunakan untuk mendapatkan efek *blurring* pada citra. Untuk mendapatkan koefisien ini didasarkan pada *Manhattan color distance* yang didasarkan pada pusat *pixel* dan *pixel* tetangga. Nilai *di* berada pada range $0 \leq di \leq 1$, yang dinyatakan dalam persamaan 2.1

$$di = \frac{|Rac - Rai| + |Gac - Gai| + |Bac - Bai|}{3 \times 255}, \; 0 \leq di \leq 1 \qquad (2.1)$$

Definisi

di : jarak antara pusat *pixel* dengan *pixel* tetangga
Rac, Gac, Bac : Nilai R,G dan B pada pusat warna
Rai, Gai, Bai : Nilai R,G dan B pada *pixel* tetangga

Dimana *Rac*, *Gac*, dan *Bac* adalah pusat *cluster*. Setelah didapatkan *di* maka dapat dihitung *ci* yang dinyatakan dalam persamaan 2.2:

$$ci = (1 - di)^p, \text{ dimana } p \geq 1 \qquad (2.2)$$

Definisi

ci : nilai koefisien konvolusi
di : jarak antara pusat *pixel* dengan *pixel* tetangga
p : nilai pangkat berdasarkan input yang diinginkan

Sehingga *convolution mask* didapatkan :

$$f(x) = \frac{1}{\sum_{i=1}^{8} ci} \begin{matrix} c1 & c2 & c3 \\ c4 & 0 & c5 \\ c6 & c7 & c8 \end{matrix} \qquad (2.3)$$

Definisi

f(x) : *convolution mask*

Untuk mendapatkan nilai pixel citra hasil digunakan persamaan 2.4

$$\boldsymbol{G(x) = H(x) * f(x)} \qquad (2.4)$$

Definisi :

f(x) : *convolution mask*

H(x) : nilai *pixel* citra yang sebenarnya
G(x) : nilai *pixel* citra hasil

2.2 *Convolution Mask*

Teknik konvolusi ini perlu dijelaskan pertama kali karena akan terkait dalam penggunaan beberapa teknik yang dipakai dalam tiap tahap reduksi warna ini .

Definisi singkat dari tehnik konvolusi :

> Konvolusi merupakan operasi matematika yang digunakan dalam image untuk mendapatkan hasil dari perkalian dua array yang berbeda ukuran. Dan dengan dimensi yang sama maka hasil yang didapatakan juga sama. Sedangkan kedua array tersebut dikenal dengan kernel. Dalam konteks yang digunakan pada pengolahan citra untuk mendapatkan hasil konvolusi maka diperlukan array input pertama sebagai *graylevel* gambar (Gambar 2.1) dan kemudian array kedua dalam bentuk array yamg lebih kecil yang disebut kernel (Gambar 2.2). Gambar tersebut menunjukkan gambar asli dan kernel yang digunakan. (Fisher, Perkins, Walker dan Wolfart, 2003)

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}	I_{28}	I_{29}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}	I_{38}	I_{39}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}	I_{48}	I_{49}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}	I_{58}	I_{59}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}	I_{68}	I_{69}

Gambar 2.1 Citra Asli

K_{11}	K_{12}	K_{13}
K_{21}	K_{22}	K_{23}

Gambar 2.2 Kernel

Dan secara matematis persamaan dari konvolusi menurut Fisher, Perkins, Walker dan Wolfart (2003) dinyatakan dalam persamaan 2.5

$$O(i,j) = \sum_{k=1}^{m} \sum_{l=1}^{n} I(i+k-1, j+l-1) K(k,l) \quad (2.5)$$

Definisi :

I	: matriks citra asli
K	: matriks kernel
m, n	: baris dan kolom dari citra yang difilter
i, j	: lokasi dari pixel yang akan dilakukan filtering
k, l	: lokasi pixel pada matriks citra asli

2.3 *Sobel Edge Detection*

Penelitian ini menggunakan tehnik *edge detection* untuk melakukan tahap *subsampling.* Dimana *edge detection* merupakan metode yang digunakan untuk menyaring informasi yang terdapat dalam citra dan menjaga bagian-bagian penting citra dalam hal ini berupa tepi-tepi citra. Dimana hasilnya berupa tepi citra yang memiliki nilai *contrast* lebih tinggi dibandingkan dengan yang lain. Ada beberapa metode yang dgunakan dalam melakukan pendeteksian tepi.

Menurut Nugroho (2005:4) suatu tepi adalah batas antara dua *region* yang memiliki *graylevel* yang relatif berbeda. Pada dasarnya ide yang ada dibalik sebagian besar teknik *edge-detection* adalah menggunakan perhitungan local *derivative operator*.

Pada penelitian ini digunakan teknik deteksi tepi *sobel*. Karena teknik ini menggunakan konvolusi yang relatif kecil sehingga waktu komputasi tidak membutuhkan waktu yang besar.

Namun operator sobel ini juga memilik kelemahan seperti yang diutarakan oleh Jebara (2006) operator sobel sangat peka terhadap tingkat kebisingan (*noise*) yang tinggi dan pada operator *sobel* hanya menghasilkan data tepi.

Persamaan 2.6 dan persamaan 2.7 merupakan persamaan yang digunakan secara berturut-turut dalam arah gradien x dan gradien y dalam teknik *sobel*.

$$\mathbf{Gx} = \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix} \qquad (2.6) \qquad \mathbf{Gy} = \begin{matrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{matrix} \qquad (2.7)$$

Definisi

Gx : nilai dari gradien untuk kernel arah x
Gy : nilai dari gradien untuk kernel arah y

Nilai gradien citra pada operasi *sobel* dapat diperoleh dengan persamaan 2.8

$$\mathbf{G} = \sqrt{\boldsymbol{G_x}^2 + \boldsymbol{G_y}^2} \quad \textbf{(}2.8\textbf{)}$$

Definisi

G : nilai gradien citra

Adapun beberapa persamaan yang bisa digunakan untuk menghitung *gradient magnitude* dalam deteksi tepi yang dinyatakan pada persamaan 2.9, 2.10 dan 2.11

Persamaan 1 : $||\nabla f|| = \sqrt{\Delta x^2 + \Delta y^2}$ **(**2.9**)**

Persamaan 2 : $||\nabla f|| = \max(abs(\Delta x), abs(\Delta y))$ **(**2.10**)**

Persamaan 3 : $||\nabla f|| = abs(\Delta x) + abs(\Delta y))$ **(**2.11**)**

Definisi

∇f : nilai dari *gradient magnitude* citra

$\Delta x, \Delta y$: nilai dari *gradient magnitude* arah x dan y

Dan syarat yang harus dipenuhi untuk setiap nilai G didapatkan dengan menerapkan persamaan 2.12

$G(xi, yi) \leq G(xi + n, yi + m)$ dimana $n = |-1{,}1|, m| - 1{,}1|$ **(**2.12**)**

Definisi

xi, yi : nilai *pixel* pusat warna
m, n : nilai antar -1 dan 1 sebagai perulangan *pixel* untuk menuju tetangga

maka akan didapatkan sample warna yang berada dalam interior objek dan menghindari sampel warna yang berada di sekitar objek. (Nikolaou, Nikos, 2008:17)

Untuk menghitung *gradient magnitude* berdasarkan warna RGB dinyatakan dengan persamaan 2.13, 2.14 dan 2.15

$$G\_red[x, y] = \sqrt{(Gx\_red[x, yj])^2 + (Gy\_red[x, y])^2} \quad (2.13)$$

$$G\_green[x, y] = \sqrt{(Gx\_green[x, y])^2 + (Gy\_green[x, y])^2} \quad (2.14)$$

$$G\_blue[x, y] = \sqrt{(Gx\_blue[x, y])^2 + (Gy\_blue[x, y])^2} \quad (2.15)$$

Definisi

G_red[x,y] : *Gradient magnitude* (*Red*) *pixel* x, y
G_green[x,y] : *Gradient magnitude* (*Green*) *pixel* x, y
G_blue[x,y] : *Gradient magnitude* (*Blue*) *pixel* x, y
Gx_red[x,y] : *Gradient magnitude* sumbu x (*Red*) *pixel* x, y
Gx_green[x,y] : *Gradient magnitude* sumbu x (*Green*) *pixel* x, y
Gx_blue[x,y] : *Gradient magnitude* sumbu x (*Blue*) *pixel* x, y
Gy_red[x,y] : *Gradient magnitude* sumbu y (*Red*) *pixel* x, y
Gy_green[x,y] : *Gradient magnitude* sumbu y (*Green*) *pixel* x, y
Gy_blue[x,y] : *Gradient magnitude* sumbu y (*Blue*) *pixel* x, y

Setelah semua nilai G pada RGB didapatkan maka kemudian dicari nilai maksimal dari nilai G dengan menggunakan persamaan 2.16

$$G[x, y] = max\ (G\_red[x, y], G\_green[x, y], G\_blue[x, y]) \quad (2.16)$$

Persamaan 2.17 digunakan untuk memberikan nilai pixel dengan warna putih.

$$Pixel[xi, yi].Merah = Pixel[xi, yi].Hijau = Pixel[xi, yi].Biru = 255 \quad (2.17)$$

Sedangkan persamaan 2.18, 2.19 dan 2.20 digunakan untuk memberikan nilai pixel dengan warna RGB.

$$Pixel[xi, yi].Merah = Pixel[xi, yi].Merah \quad (2.18)$$

Pixel[xi, yi].Hijau = Pixel[xi, yi].Hijau (2.19)
Pixel[xi, yi].Biru = Pixel[xi, yi].Biru (2.20)

2.4 *Histogram*

Histogram dalam penelitian ini berfungsi untuk mendapatkan peluang dari nilai rgb yang diterapkan dalam tahap *initial color*. Dimana *histogram* merupakan teknik untuk skala keabu-abuan pada gambar yang bertujuan untuk mendistribusikan *pixel* dari suatu gambar (I) untuk semua *graylevel* yang tersedia (L). Dimana diasumsikan pixel (i,j) membaca *pixel* dari gambar. Persamaan 2.21 digunakan untuk menghitung *histogram*. Sebagai contoh histogram citra kapal pada Gambar 2.3 dapat dilihat pada gambar 2.4.

$$f(k) = P\{I(i,j) = k\} = \frac{N_k}{N} \quad (2.21)$$

Definisi

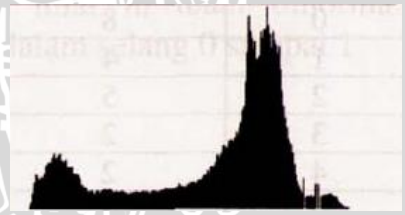
N_k : jumlah kejadian dari k dimana $k \epsilon [0, L-1]$ yang mewakili satu *graylevel*

N : jumlah *pixel* dari suatu gambar

f(k) : peluang dari greylevel k.

(Bassiou, Kotropoulos : 2).

Gambar 2.3 Citra Kapal



Gambar 2.4 Histogram Citra Kapal

2.5 *Initial Color*

Pendapat yang diungkapkan oleh Nikolaou dan Paparmakos (2008:16) menyatakan bahwa teknik ini digunakan untuk mendapatkan sampel dari citra. Hal ini bertujuan untuk mengurangi jumlah warna yang digunakan sebagai sampel sehingga jumlah *cost*

dari algoritma selanjutnya dapat lebih efisiean. Sehingga tahap ini juga bisa disebut sebagai tahap *sub-sampling* dengan memanfaatkan tehnik *edge detection* menggunakan *sobel* operator. Dan hal yang diperlukan antara lain:

a. Sampel yang didapatkan dari tahap *sub-sampling.*
b. Menentukan peluang dari sampel dengan menggunakan *Histogram.*

Dalam melakukan reduksi sejumlah warna dari sampel yang didapatkan sebelumnya maka pada initial color untuk menentukan kumpalan sampel yang digunakan persamaan 2.22, 2.23 dan 2.24

$$r_{mi} = \frac{(\sum_{r=-h}^{h}\sum_{g=-h}^{h}\sum_{b=-h}^{h} r.p(r,g,b)}{(\sum_{r=-h}^{h}\sum_{g=-h}^{h}\sum_{b=-h}^{h} p(r,g,b)} \quad (2.22)$$

$$g_{mi} = \frac{\sum_{r=-h}^{h}\sum_{g=-h}^{h}\sum_{b=-h}^{h} g.p(r,g,b)}{(\sum_{r=-h}^{h}\sum_{g=-h}^{h}\sum_{b=-h}^{h} p(r,g,b)} \quad (2.23)$$

$$b_{mi} = \frac{\sum_{r=-h}^{h}\sum_{g=-h}^{h}\sum_{b=-h}^{h} b.p(r,g,b)}{(\sum_{r=-h}^{h}\sum_{g=-h}^{h}\sum_{b=-h}^{h} p(r,g,b)} \quad (2.24)$$

Definisi

r_{mi}, g_{mi}, b_{mi} : nilai *pixel* RGB baru pada sampel
h : tinggi area
p : frekuensi *histogram*

2.6 Distribusi Warna dalam Ruang 2D

Distribusi warna ini bertujuan untuk menampilkan sebaran dari warna yang terdapat pada citra yang akan direduksi sehingga dapat diketahui seberapa banyak warna yang terdapat dalam gambar. Seperti yang diungkapkan Gearaud, Strub, Darbon (2001:2) yang mengatakan tujuan dari proyeksi dari WRG yaitu manampilkan semua label warna (r,g,*) yang mewakili semua warna dalam gambar asli. Label dari , l_m(r,g) digambarkan dengan sebuah *pixel* berwarna rgb sesuai dengan warna yang terdapat dalam gambar. Sedangkan warna putih dianggap bukan *pixel*.

2.7 *Cluster*

Menurut Jain dan Dubes mendefinisikan sebuah *cluster* sebagai sejumlah objek yang dikelompokkan secara bersama. Namun definisi cluster sendiri dapat beragam tergantung sudut pandang yang digunakan. Antara lain :

a. *Well-separated-cluster*
   Cluster adalah sekelompok titik dimana sebuah kelompok titik pada kelompok tersebut lebih dekat atau mirip dengan semua titik lainnya yang ada pada kelompok tersebut daripada titik-titik lain yang tidak terdapat pada kelompok tersebut.
b. *Center-based cluster*
   Cluster adalah sekelompok titik dimana semua titik pada kelompok tersebut lebih dekat dengan pusat cluster kelompok tersebut dibandingkan dengan pusat kelompok lain.
c. *Density-based cluster*
   Cluster adalah sebuah wilayah pada titik yang terpisahkan dari wilayah pada titik lainnya oleh wilayah berkepadatan rendah.

Adapaun cluster yang lebih cocok dengan penelitian ini adalah *well-seperated cluster*.

2.8 *Mean Shift Procedure*

Mean shift merupakan algoritma nonparametik *clustering* dalam suatu ruang vektor . Menurut (Subbarao, Raghav, 2009:8) menyatakan bahwa dengan melakukan generalisasi titik data yang terletak di *manifold riemann* memungkinkan *mean* untuk memperluas *mean shift clustering* dan *filtering* yang sering terjadi dalam ruang non vektor.

Mean shift procedure selalu terkait dengan *density*. Dimana *density* merupakan suatu area yang dianggap paling mendekati lokal maksimal dari suatu kumpulan *cluster*. Menurut Wang (3) menyatakan bahwa ukuran *density* dalam *mean shift* di beberapa titik dalam ruang 2D atau 3D didefinisikan dengan jumlah beban dari *pixel* terdekat. Area dari *density* titik dari tiap pixel terdekat didefiniskan sebagai *kernel* yang memiliki jarak yang kecil. Dan

mean shift akan menemukan setiap *pixel* yang menjadi lokal maksimal dari *density* yang diperkirakan sebelumnya. Dan setiap *pixel* akan bergerak menuju titik dari *gradient* yang dicapai.

Menurut (Bradsky, Gary : 46) ada langkah langkah yang harus dilakukan dalam menerapkan *mean shift* yaitu :

a. Menentukan ukuran dari *window* yang akan digunakan.
b. Memilih *initial* lokasi dari dari *window* yang telah ditentukan.
c. Menghitung pusat data dengan menggunakan *mean shift* dalam *window* yang telah ditentukan.
d. Pusat dari *window* dalam *mean shift* dihitung dalam tahap c.
e. Ulangi tahap c dan d sampai konvergen yaitu jika gradien sama dengan nol.

Menurut Comaniciu (2002 : 2) menyatakan di dalam suatu citra memiliki dimensi ruang R^d dengan poin dimisalkan xi = 1,2,…,n. Dalam sebuah *kernel density* yang disebut dengan *multivariate kernel density estimator* dengan kernel K(x) dan dengan ukuran dari luasan matrik H dan simetri positif *d x d* , maka untuk menghitung nilai dari x digunakan persamaan 2.25

$$\hat{f}(x) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{xi-x}{h}\right) \qquad (2.25)$$

Definisi

$\hat{f}(x)$: fungsi x untuk hasil meanshift
n : jumlah poin x pada ruang R^d
h : ukuran luasan dari matriks
xi : nilai x ke i yang terdapat pada matriks
x : nilai yang dihitung

K(x) adalah sebuah *kernel radial symmetric* yang ditunjukkan dengan persamaan 2.26

$$K(x) = c_k\, k(||x||^2) \qquad (2.26)$$

Definisi

k(x) : *profile* dari *kernel* , untuk x ≥ 0
ck : normal konstanta
K(x) : Kernel

Sehingga dapat diasumsikan bahwa nilai *K(x)* adalah positif.

Sedangkan nilai k(x) memiliki dua tetapan nilai dengan kondisi

$$k(x) \begin{cases} 1\text{-r untuk } r \leq 1 \\ 0 \text{ untuk yang lain} \end{cases} \qquad (2.27)$$

Berdasarkan persamaan *2.14* menurut (Comaniciu, Dorin , 2002 : 2) menyatakan bahwa nilai dari konstanta ck merupakan nilai yang didapatkan dari *volume* dengan panjang h.

Secara singkat dijelaskan sedikit tentang proses penurunan persamaan dari *mean shift*. Dimulai dari persamaan 2.26 dengan mensubtitusikan nilai 1-r atau 0 ke k(x), jika r adalah $||x||^2$ maka dari persamaan 2.26 dapat diketahui persamaan K(x) baru yang ditunjukkan pada persamaan 2.28

K(x) = c_k *(1-*$||x||^2$*) dimana* $||x|| \leq 1$ *dan* *0 untuk yang lain* (2.28)

Sedangkan syarat konvergen dari *menshift* yaitu jika telah memenuhi density maksimal yaitu :

$\nabla f(x) = 0$

$$\nabla f(x) = \frac{1}{nh^d}\sum_{i=1}^{n} ck\, k(||\frac{xi-x}{h}||^2)$$

Turunan dari $k(||\frac{xi-x}{h}||^2)$ adalah *(x-xi)* $k'(||\frac{xi-x}{h}||^2)$. Yang menghasilkan persamaan 2.29

$$= \frac{ck}{nh^d}\sum_{i=1}^{n} 2(\frac{xi-x}{h})\, k'(||\frac{xi-x}{h}||^2)$$

$$= \frac{2ck}{nh^{(d+2)}}\sum_{i=1}^{n}(xi-x)\, k'(||\frac{xi-x}{h}||^2) \qquad (2.29)$$

Beberapa persamaan yang dibutuhkan dalam mean shift ditunjukkan pada persamaan 2.30, 2.31 dan 2.32

g(r) = k´(r) (2.30)

$k'(r) = k'(||\frac{xi-x}{h}||^2)$ (2.31)

$gi = g(||\frac{xi-x}{h}||^2)$ (2.32)

Maka persamaan (2.32) dapat disubtitusikan ke persamaan (2.29) yang hasilnya adalah persamaan (2.34)

$$\nabla f(x) = \frac{2ck}{nh^{(d+2)}}\sum_{i=1}^{n}(xi-x)\,g(||\frac{xi-x}{h}||^2) \quad (2.33)$$

$$= \frac{2ck}{nh^{(d+2)}}\sum_{i=1}^{n}(xi-x)\,gi$$

$$= \frac{2ck}{nh^{(d+2)}}(\sum_{i=1}^{n} gi)(\frac{\sum_{i=1}^{n} xi\,gi}{\sum_{i=1}^{n} gi} - \boldsymbol{x}) \quad (2.34)$$

Definisi

g(x) : fungsi x untuk mendapatkan nilai baru hasil mean shift

Persamaan (2.34) merupakan persamaan dari *mean shift vector*.

2.9 *Color Similiarity*

Pada tahap ini berguna untuk melakukan *merging* terhadap warna yang hampir sama dengan memanfaatkan kesamaan berdasarkan *distance*. Dimana terdapat dua pengukuran yang dapat digunakan dalam melakukan kecocokan teradap dua objek yaitu dengan pengukuran *similiarity* dan *distance*. Perbedaan antara dua pengukuran ini yaitu untuk pengukuran *simialirity* semakin besar nilai yang dihasilkan semakin tinggi tingkat kecocokan dua objek sedangkan jika semakin kecil nilai maka semakin rendah tingkat kecocokan dua objek. Sedangkan pengukuran distance, semakin tinggi nilai *distance* yang dihasilkan maka semakin kecil tingkat kecocokan dua objek dan semakin kecil *distance* semakin tinggi tingkat kecocokan.

Terdapat beberapa teknik perhitungan dalam melakukan pengukuran *distance* antara lain :

a. Minkowski distance

b. Manhattan distance
c. Euclidean distance
d. Chebyshev distance
dan di dalam penulisan ini akan digunakan pengukuran *distance* dengan *manhattan distance*.

Persamaan umum dalam menghitung *manhattan distance* ditunjukkan pada persmaan 2.35

$$D = \sum_{n=1}^{m} |oin - ojn| \quad (2.35)$$

dimana :

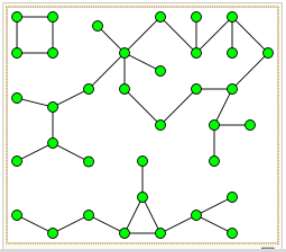
D merupakan nilai distance yang dicari
m merupakan banyaknya dimensi
oin merupakan objek pertama untuk dimensi ke n
ojn merupakan objek kedua untuk dimensi ke n

Persamaan 2.36 digunakan untuk menghitung nilai *manhattan distance* dengan objek berupa warna RGB.

$$dm = |ri - rj| + |gi - gj| + |bi - bj| \quad (2.36)$$

2.10 *Connected Component*

Connected component merupakan suatu graf yang menghubungkan dua atau lebih vertex dalam suatu path. Dalam sebuah grafik setidak-tidaknya memiliki *connected component* paling sedikit adalah satu. Seperti yang ditunjukkan dalam Gambar 2.5 dimana dalam satu grafik ini terdiri dari tiga *connected componenet*.



Gambar 2.5 Tiga Connected Component

Connected component dalam pelabelan terhadap citra dilakukan dengan melakukan scanning terhadap citra. Dan *connected component* pada pixel citra didasarkan pada konektifitas pixel yaitu

semua pixel yang terdapat pada *connected component* memiliki nilai-nilai intensitas pixel yang sama. *Scanning* dimulai dari pixel paling kiri ke kanan dan atas ke bawah dari suatu citra. Dan untuk mengidentifikasi bahwa pixel tersebut terhubung maka pixel yang berdekatan diatur nilai intensitas greylevelnya. *Connected component* bergerak sampai dicapai pada titik *p* (dimana *p* adalah pixel yang akan diberi label pada setiap tahap proses scanning) untuk *V=1*(dimana *V* memiliki rentang *graylavel* yang ditentukan sebelumnya missal pixel yang memiliki *graylavel* antara 1-100). Kemudian mengkaji empat tetangga p ((i) Disebelah kiri p, (ii) di atasnya dan (iii dan iv) kedua diagonal atas). Tahap pelabelan :

1. Jika semua tetangga memiliki nilai *V=0* maka tandai label baru untuk p.
2. Jika hanya terdapat satu tetangga yang memiliki nilai *V=1* maka tetapkan label untuk p.
3. Jika terdapat lebih dari satu tetangga yang memiliki nilai *V=1* maka tetapkan salah satu label p dan mencatat mengenai kesetaraan.

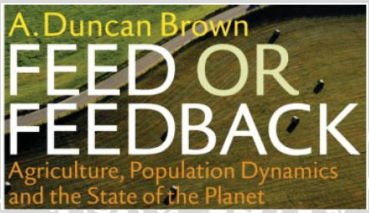
(Fisher R., Perkins S., Walker A. and Wolfart E., 2003)

2.11 Reduksi Warna pada Dokumen Digital

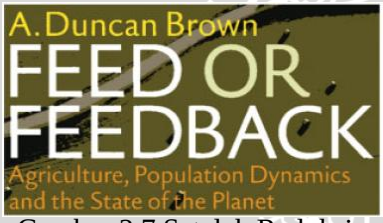
Penelitian yang dilakukan oleh Nikolou dan Papmarkos ini menunjukkan pengaruh penerapan *EPSF*. Dokumen digital yang digunakan untuk percobaan oleh mereka memiliki 78.623 warna dan ukurannya adalah 827 x 466 *pixel* (gambar 2.6). Hasil kuantitatif dari percobaan ini diberikan dalam Tabel 2.1. Gambar akhir yang diperoleh ketika EPSF digunakan sebagai tahap preprocessing ditampilkan pada Gambar 2.7. Ketika EPSF diabaikan, algoritma menghasilkan gambar 2.8.

Pertama kali yang harus diperhatikan adalah bahwa warna berkurang ketika tahap *preprocessing* diterapkan. Hal ini terjadi karena varians *EPSF* pada warna dan *cluster* lebih sedikit setelah dilakukan proses tersebut sehingga terjadilah proses awal reduksi warna.

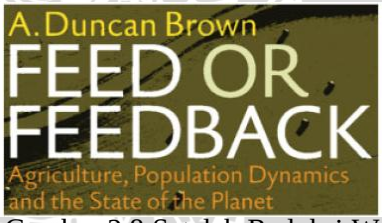
Pengaruh lain dari *EPSF* pada distribusi warna lebih tajam warna yang dihasilkan. Kesimpulan ini didasarkan pada fakta bahwa jumlah iterasi pada *mean shift* membutuhkan konvergensi, lebih kecil ketika EPSF diterapkan. Seperti yang terlihat dari hasil disajikan pada Tabel 2.1, dimana dalam permasalahan ini EPSF diterapkan, 20 kelas dicapai untuk mencapai pusat warna akhir setelah dilakukan 60 iterasi, jadi 60 / 20 = 3 rata-rata iterasi yang dilakukan. Di sisi lain ketika EPSF tidak diterapkan, rata-rata menjadi 113 / 23 = 4,91 iterasi.



Gambar 2.6 Dokumen Berwarna Asli Dengan 78.623 Warna



Gambar 2.7 Setelah Reduksi Dengan EPSF



Gambar 2.8 Setelah Reduksi Warna Tanpa EPSF

Gambar 2.9 Binary Edge Map Dengan EPSF

Gambar 2.10 Binary Edge Map Tanpa EPSF

Manfaat yang paling penting dari penggunaan *EPSF* ini menghilangkan dan mencegah terjadinya *noise* pada objek serta sejumlah connected component. Seperti terlihat pada Tabel 2.1, rasio antara jumlah *connected component* diperoleh tanpa dan dengan penerapan *EPSF* adalah 2631/518 = 5,079. Perbedaan ini dapat dilihat secara grafis dalam Gambar 2.9 dan 2.10 yang menunjukkan batas-batas objek. Efek dari *EPSF* lainnya adalah meningkatkan kualitas dari suatu objek. Pengurangan *connected component* secara signifikan dapat mempengaruhi waktu komputasi dan lebih diandalkan dalam melanjutkan segmentasi dokumen dan lokalisasi teks. (Nikolou ,N dan Paparmarkos N, 2008)

Tabel 2.1 Hasil Eksperimen

	Dengan EPSF	Tanpa EPSF
Processing time (second)	0.83	0.72
Initial classes	20	23
Mean-shift iteration	60	113
Jumlah warna akhir	7	6
Jumlah *connected component*	518	2631

Dan untuk mengukur kualitas citra digunakan *Mean Square Error* (MSE) dengan persamaan 2.37

$$MSE(\%) = \frac{1}{MxN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [g(x,y) - f(x,y)]^2 \; x\,100\% \qquad (2.37)$$

Definisi

M, N merupakan tinggi dan lebar area citra
g(x,y) merupakan citra asli
f(x,y) merupakan citra hasil

Dan untuk perhitungan persentase hasil berdasarkan visualisasi manusia menggunakan persamaan 2.38

$P = \frac{F}{N} x100$ (Bungin, 2005:26) (2.38)

Definisi

F merupakan frekuensi banyaknya responden yang menjawab

N merupakan jumlah keseluruhan

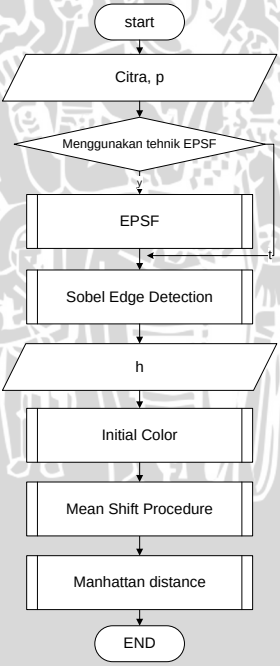
UNIVERSITAS BRAWIJAYA

BAB III
METODOLOGI PENELITIAN

3.1 Rancangan Sistem

Citra dalam sebuah dokumen berwarna biasanya memiliki tekstur dengan pola dan warna yang rumit. Sehingga memiliki tingkat kesulitan dalam segmentasi terhadap objek yang akan dipilih. Gambar (**3.1**) menunjukkan tahapan yang harus dikerjakan dalam melakukan reduksi warna.

Dalam penelitian ini input yang digunakan adalah citra dokumen berwarna (*full color)*. Yang akan menjadi pertimbangan dalam penelitian ini yaitu bagaimana hasil yang diperoleh jika mempertimbangkan 3 parameter yang digunakan yaitu p pada tahap EPSF dan h serta jumlah iterasi untuk tahap *Initial Color* dan *Mean Shift*.



Gambar 3.1 Flowchart Sistem

3.2 Rancangan Penelitian

Penelitian ini terdiri dari beberapa tahapan seperti yang terdapat pada Gambar 3.1. Tahap tersebut akan akan dijelaskan sebagai berikut :

3.2.1 Penerapan *EPSF* dalam Citra

200 100 40	100 125 100	200 220 200	200 100 125
140 160 130	100 170 180	160 200 220	210 140 120
230 240 250	110 120 110	150 160 170	110 210 230
110 120 100	120 140 150	130 100 50	135 125 115

Gambar 3.2 Citra Dengan Ukuran *Pixel* 4x4

Diketahui citra seperti Gambar 3.2 dengan nilai yang dicantumkan adalah nilai R, G, B pada masing-masing pixel dengan ukuran tiap sampel 3x3. Pixel berwarna kuning adalah pixel tetangga sedangkan pixel berwarna merah adalah pixel pusat.

Langkah pertama mencari jarak (d) dengan menggunakan persamaan 2.2 :

$$d1 = \frac{|100-200|+|170-100|+180-100|}{3\ x\ 255} = \frac{250}{675} = 0.37$$

$$d2 = \frac{|100-100|+|170-125|+|180-100|}{3\ x\ 255} = \frac{125}{675} = 0.186$$

$$d3 = \frac{|100-200|+|170-220|+|180-200|}{3\ x\ 255} = \frac{170}{675} = 0.25$$

$$d4 = \frac{|100-140|+|170-160|+|180-130|}{3\ x\ 255} = \frac{100}{675} = 0.15$$

$$d5 = \frac{|100-160|+|170-200|+|180-220|}{3\ x\ 255} = \frac{136}{675} = 0.20$$

$$d6 = \frac{|100-230|+|170-240|+|180-250|}{3\ x\ 255} = \frac{270}{675} = 0.40$$

$$d7 = \frac{|100-110|+|170-120|+|180-110|}{3\ x\ 255} = \frac{170}{675} = 0.20$$

$$d4 = \frac{|100-150|+|170-160|+|180-170|}{3\ x\ 255} = \frac{70}{675} = 0.103$$

Menghitung *convolution mask* dengan p = 10 menggunakan persamaan 2.3. Sehingga :

$c1 = (1 - 0.37)^{10} = 0.0098$

$c2 = (1 - 0.19)^{10} = 0.1216$

$c3 = (1 - 0.25)^{10} = 0.0563$

$c4 = (1 - 0.15)^{10} = 0.1969$

$c5 = (1 - 0.20)^{10} = 0.1073$

$c6 = (1 - 0.40)^{10} = 0.0060$

$c7 = (1 - 0.20)^{10} = 0.1073$

$c8 = (1 - 0.103)^{10} = 0.3372$

Dan hasil yang didapatkan :

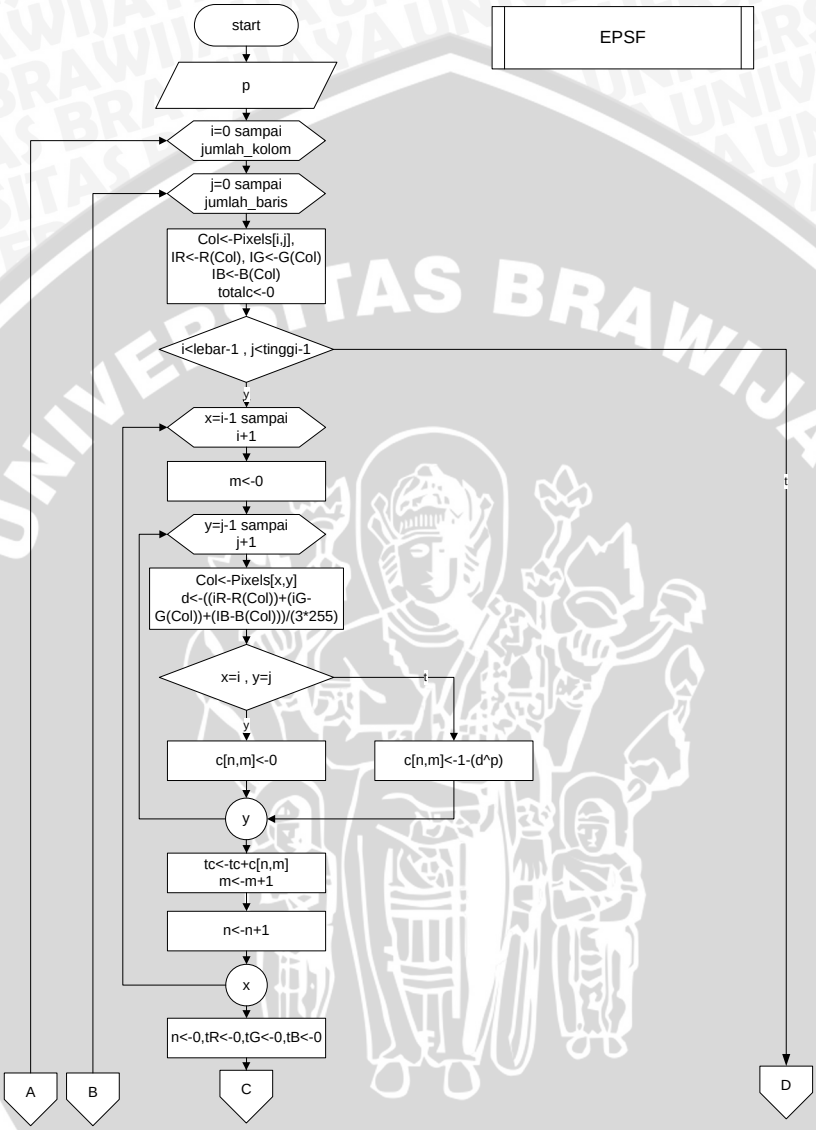
$$\frac{1}{\sum_{i=1}^{8} ci} = (0.0098 + 0.1216 + 0.0563 + 0.1969 + 0.1073 + 0.0060 + 0.1073 + 0.3372) = \frac{1}{0.9415}$$

Setelah mendapatkan nilai ci pada masing-masing *pixel* dan maka diterapkan persamaanan 2.4

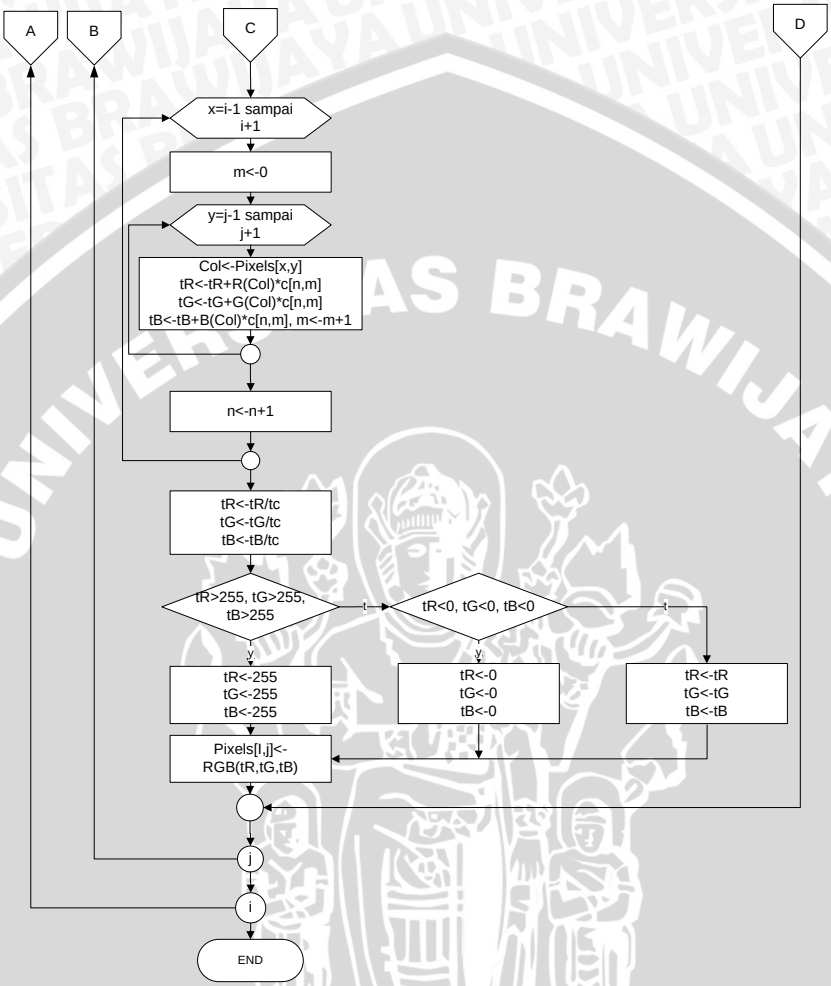
$$F(x) = \frac{1}{\sum_{i=1}^{8} ci} \begin{matrix} c1 & c2 & c3 \\ c4 & 0 & c5 \\ c6 & c7 & c8 \end{matrix} = \frac{1}{0.9415} \begin{matrix} 0.0098 & 0.1216 & 0.0563 \\ 0.1969 & 0 & 0.1073 \\ 0.0060 & 0.1073 & 0.3372 \end{matrix}$$

$$= \begin{matrix} 0.0104 & 0.1291 & 0.0598 \\ 0.2901 & 0 & 0.1139 \\ 0.0064 & 0.1139 & 0.3581 \end{matrix}$$

Pada Gambar 3.3 menjelaskan tentang *flowchart* dari rancangan penerapan pengolahan citra menggunakan teknik *EPSF*



Gambar 3.3 *Flowchart* Proses *EPSF* (I)



Gambar 3.4 *Flowchart* Proses *EPSF* (II)

Setelah didapatkan hasil dicoba pada *pixel* yang telah ditentukan dengan mengalikannya dengan *convolution mask* yang telah didapatkan dengan menerapkan persamaan 2.5. Misal pada *pixel* dengan koordinat (0,0) maka niliai pixel baru adalah :

G(x) untuk nilai *Red*

200	100	200		0.0104	0.1291	0.0598	
140	100	160	x	0.2901	0	0.1139	= 153
230	110	150		0.0064	0.1139	0.3581	

G(x) untuk nilai *Green*

100	125	220		0.0104	0.1291	0.0598	
160	170	200	x	0.2901	0	0.1139	= 172
240	120	160		0.0064	0.1139	0.3581	

G(x) untuk nilai *Blue*

40	100	200		0.0104	0.1291	0.0598	
130	180	220	x	0.2901	0	0.1139	= 163
250	110	170		0.0064	0.1139	0.3581	

Hasil output dari citra yang telah difilter dengan EPSF ditunjukkan gambar 3.4

200 100 40	100 125 100	200 220 200	200 100 125
140 160 130	153 172 163	160 200 220	210 140 120
230 240 250	110 120 110	150 160 170	110 210 230
110 120 100	120 140 150	130 100 50	135 125 115

Gambar 3.5 Citra Hasil Filter *EPSF*

3.2.2 Deteksi *Sobel* untuk Mendapatkan Sample

Contoh citra yang akan dilakukan pendeteksian tepi dengan operator *sobel* sebagai berikut :

3	4	2	5	7
2	1	6	4	2
3	5	7	2	4
4	2	5	7	1
2	5	1	6	9

Gambar 3.6 Citra Awal

*	*	*	*	*
*	18	*		

Gambar 3.7 Citra Hasil Konvolusi

Dengan menggunakan persamaan *gradient magnitude* 2.11 maka didapatkan nilai *gradient magnitude* maka M = 18.

sy = 3*(1) + 2*(2)+ 3*(1) + 2*(-1) + 6*(-2) + 7*(-1) = -11
sx = 3*(1) + 4*(2) + 2*(1) + 3*(-1) + 5*(-2) + 7*(-1) = -7
G(x,y) = |-11| + |-7| = 18

Pada penelitian ini digunakan operator sobel sebagai langkah dalam tahap deteksi tepi. Tahap ini digunakan untuk melakukan pendeteksian terhadap interior dari objek yang akan dipilih sebagai objek utama.

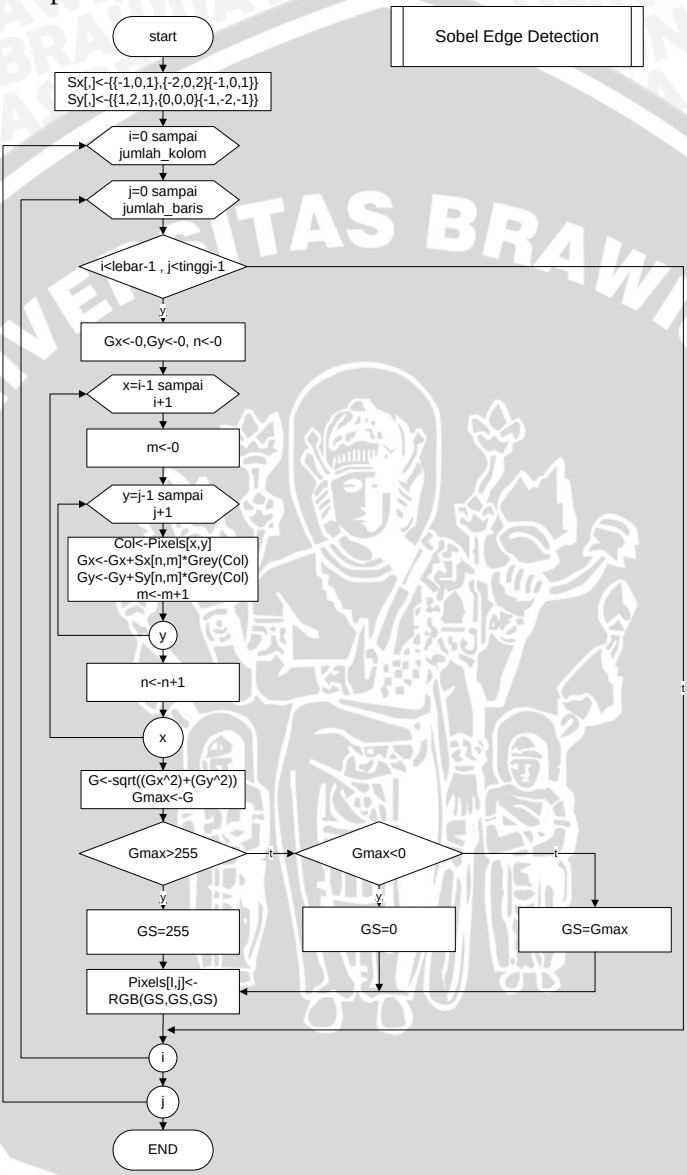
Dalam penelitian ini untuk melakukan tahap *edge detection* sehingga tidak mendapatkan warna *grayscale* yang hanya terdiri dari warna hitam putih maka dalam pengolahannya sehingga didapatkan warna RGB, maka ketika menghitung *gradient magnitude* dicari nilai RGB dan bukan diambil dari salah satu warna atau dari rata-rata warna. Dan kemudian nilai RGB tersebut dicari yang maksimal untuk melakukan perbandingan selanjutnya. Persamaan 2.13 sampai 2.15 merupakan persamaan umum dari deteksi tepi berupa warna RGB.

Dari persamaan 2.13 sampai 2.15 maka dapat dihitung nilai G pada masing-masing *pixel*. Dan dari hasil tersebut maka dicari G yang maksimal sebagai hasil *filter* dari citra yang asli untuk menentukan sampel dari citra. Dimana G didapatkan dengan persamaan 2.16.

Jika nilai G untuk semua *pixel image* sudah didapatkan maka tahap selanjutnya membandingkan nilai G dengan tetangganya dengan menerapkan persamaan yang terdapat pada 2.12.

Jika dalam proses tersebut tidak memenuhi syarat maka *pixel* citra baru tersebut diubah menjadi putih dimana warna ini bukanlah sample untuk tahap selanjutnya. Persamaan yang digunakan yaitu persamaan 2.17. Dimana dari persamaan tersebut didapatkan nilai yang menghasikan warna putih sebagai warna yang paling terang. Dan jika memenuhi syarat maka *pixel* tersebut diatur sesuai dengan warna aslinya dengan menggunaka persamaan 2.18 sampai 2.20.

Gambar 3.8 menggambarkan *flowchart* dari proses *sobel edge detection* pada reduksi warna.



Gambar 3.8 *Flowchart* Deteksi *Sobel* Pada Reduksi Warna

Contoh hasil dari sampel yang dihasilkan dari citra asli dengan parameter p = 10 untuk EPSF ditunjukkan Gambar 3.9:

Gambar 3.9 Citra hasil sobel edge detection setelah melalui tahap EPSF

3.2.3 *Initial Color* untuk Mereduksi jumlah Warna Sampel

Tahapan ini digunakan untuk mereduksi sejumlah warna yang terdapat pada sampel. Yang harus dilakukan dahulu yaitu melakukan *histogram* pada warna yang terdapat pada citra. Dalam penelitian ini *histogram* yang digunakan yaitu menggunakan warna *greyscale* pada gambar dan bukan warna rgb.

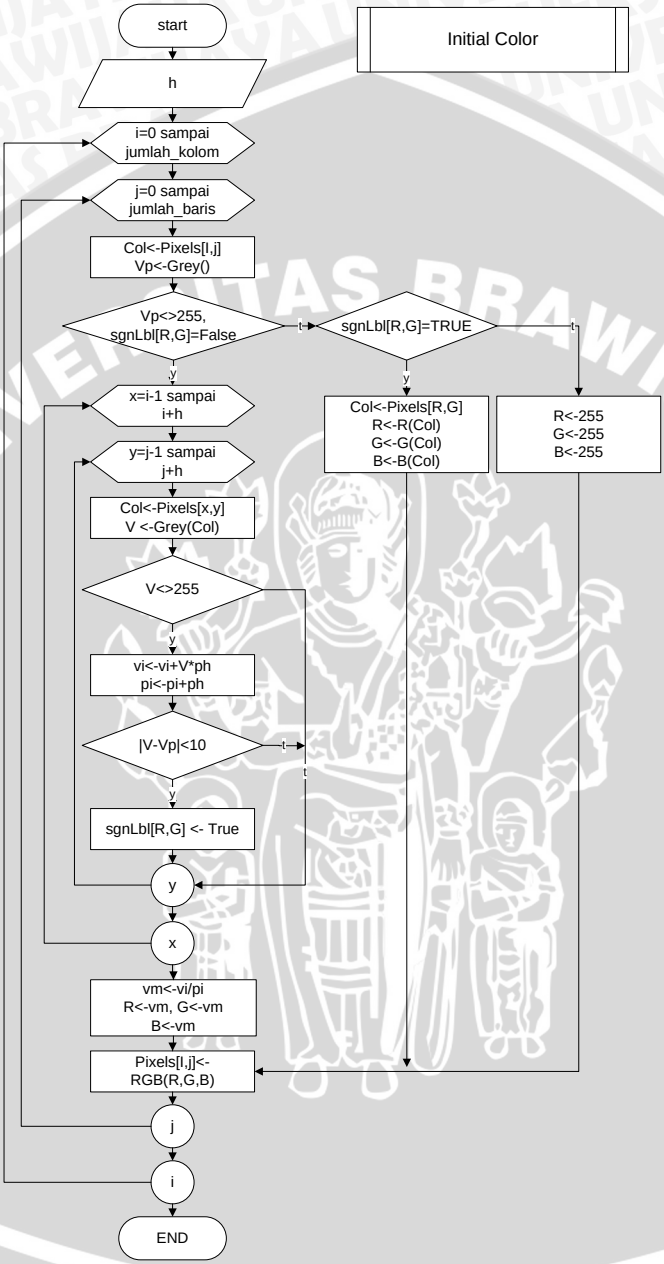
Sebagai contoh perhitungan *histogram* dimisalkan terdapat sebuah citra ukuran 5x5 window dengan L = 4 dan nilai *graylevel* pada Gambar 3.10:

0	1	0	0	0
3	3	1	2	2
1	0	0	0	1
3	3	3	2	1
3	3	0	0	0

Gambar 3.10 Citra berukuran *pixel* 5x5

Dari citra tersebut didapatkan nilai *histogram* dari :
$n_0 = 10$, $n_1 = 5$, $n_2 = 3$, $n_3 = 7$

Gambar 3.11 menunjukkan proses *initial color* sebagai bagian dari proses reduksi warna citra dokumen digital.



Gambar 3.11 *Flowchart* Proses *Initial Color*

Dengan menerapkan persamaan 2.21 maka dapat dilakukan perhitungan. sehingga *histogram* yang dihasilkan dari *image* tersebut yaitu :
p(0) = 10/25
p(1) = 5/25
p(2) = 3/25
p(3) = 7/25

Ditahap sebelumnya telah didapatkan kumpulan sampel dan dimisalkan kumpulan sampel tersebut adalah S. Di dalam tahap ini disebutkan bahwa s Є S (s adalah anggota dari S). Dan s akan dipertimbangkan sebagai kandidat *cluster* dan s akan dipilih acak yang digunakan sebagai titik awal dari sampel si sebagai sampel baru dari hasil *initial color*. Ada beberapa tahap yang harus dilakukan:

a. Tahap 1

Mendifinisikan h sebagai lebar dan tinggi dari area image yang akan diproses. Dengan mempertimbangkan si sebagai pusat *cluster*. Dari pusat *cluster* tersebut kemudian dicari nilai pixel baru s_{mi}= (r_{mi} , g_{mi}, b_{mi}) . Dimana r_{mi} , g_{mi}, b_{mi} didapatkan dari persamaan 2.22 sampai 2.24.

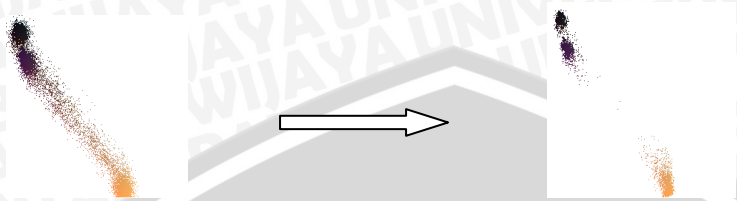
b. Tahap 2

Menandai semua poin *pixel* yang terdapat dalam area image yang telah diperiksa. Dimisalkan *pixel* yang telah diperiksa bernilai *True* sedangkan yang belum diperiksa bernilai *False*.

c. Tahap 3

Memilih sebuah sampel yang belum ditandai dan kemudian kembali menuju tahap pertama. Jika semua sampel telah ditandai maka hentikan pemeriksaan *pixel*.

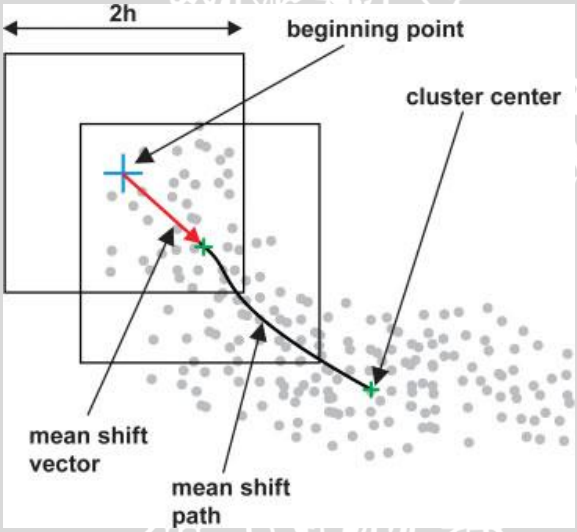
Berdasarkan hasil yang diperoleh semakin besar ukuran h maka jumlah sampel warna baru yang dihasilkan semakin sedikit. Gambar 3.12 adalah distribusi warna RGB pada ruang 2D yang dihasilkan pada suatu citra setelah dilakukan *initial color* .



Gambar 3.12 Sampel Citra Setelah Initial Color

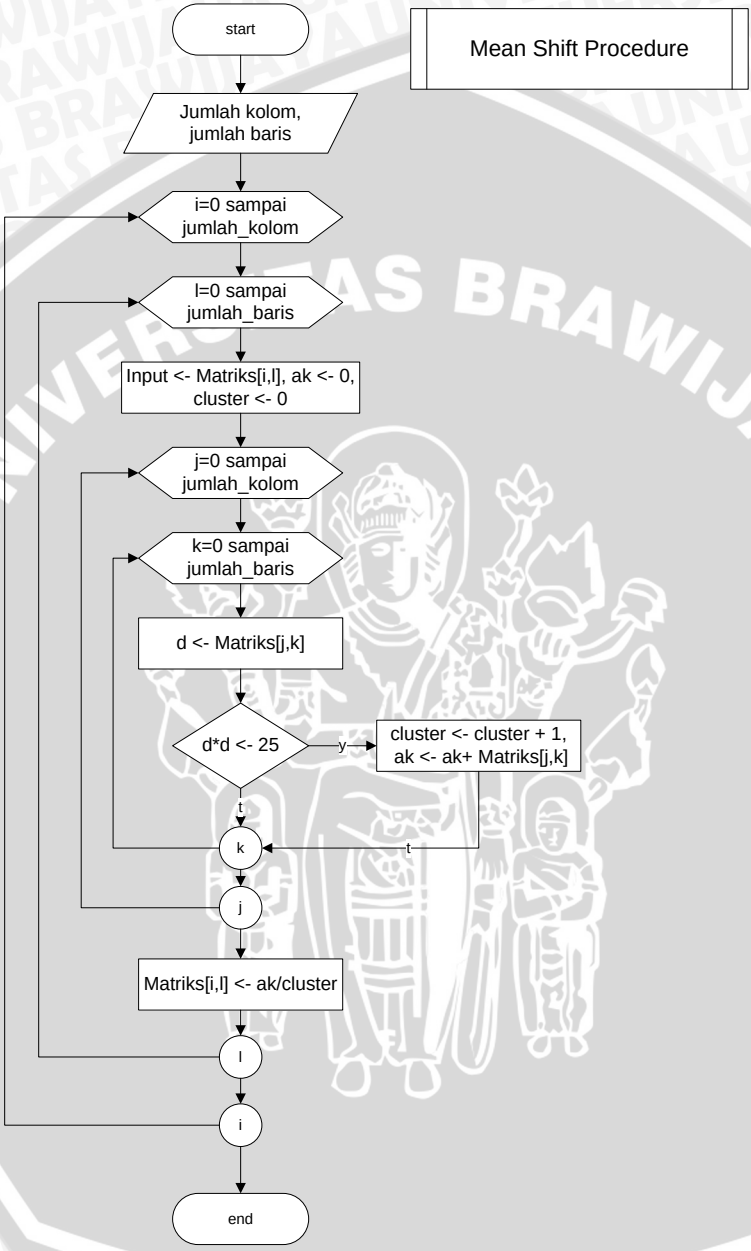
3.2.4 *Mean Shift Procedure* untuk Menentukan Pusat Warna

Dalam tahap ini *mean shift procedure* digunakan untuk mereduksi sejumlah warna yang sama dan kemudian diarahkan menuju warna yang mendekati warna tersebut. Untuk dapat mengarah ke warna yang dituju dibutuhkan pusat *cluster*. Yang nanti area dari sejumlah warna yang telah ditentukan akan mendekati ke area pusat *cluster*. Gambar 3.13 merupakan proses *mean shift* pada suatu titik.



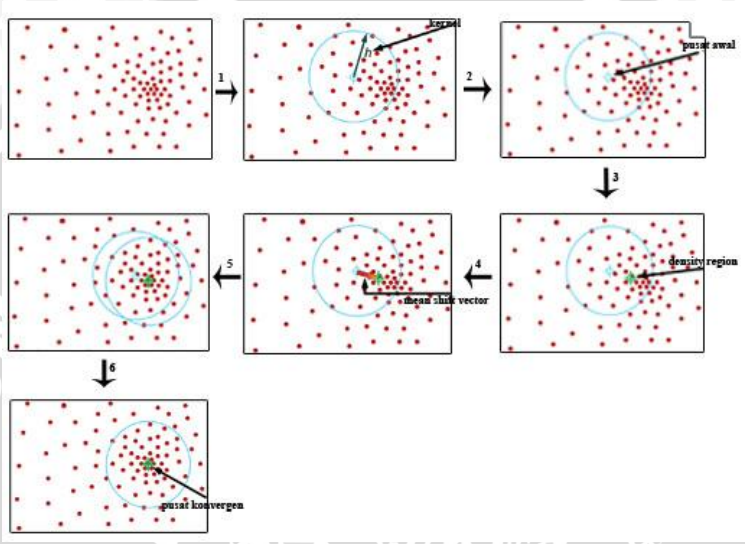
Gambar 3.13 Proses Pergerakan Titik Pada *Mean Shift*

Di misalkan poin awal dari adalah xi dan poin yang akan dituju sebagai point baru dari tahap selanjutnya adalah xj. Dan xi didapatkan dengan menentukan area titik awal secara acak sampai didapatkan syarat konvergen. Gambar 3.14 merupakan flowchart sedangkan Gambar 3.15 adalah gambar langkah proses *meanshift* procedure berdasarkan algoritma *meanshift* pada tinjauan pustaka



Gambar 3.14 *Flowchart Mean Shift*

untuk mencari area yang konvergen dan menentukan titik pusat dengan parameter h.



Gambar 3.15 Proses Pencarian Titik Sampai Konvergen

3.2.5 *Manhattan Distance* untuk Melakukan Final Color

Pada tahap ini akan dilakukan *merging* warna-warna pada gambar asli menuju klas yang lebih dekat dari pusat warna yang ada pada sampel. Dimana nilai *distance* yang memiliki nilai lebih kecil dari nilai pixel tetangga yang lain maka pixel tetangga tersebut yang lebih dekat dengan pusat pixel. Untuk mendapatkan nilai distance digunakan *manhattan distance* seperti yang terdapat dalam persamaan 2.35.

Dalam penelitian ini dimensi dari objek terdiri dari tiga nilai yaitu R, G dan B sedangkan objek yang digunakan yaitu membandingkan objek warna untuk gambar asli dan objek.

Contoh penggunaan *manhattan distance* dengan menggunakan matriks tetangga 3x3 seperi pada Gambar 3.16 yang

berwarna kuning. *Pixel* yang lebih dekat warna tersebut adalah yang memiliki *distance* paling kecil dibanding pixel lain. Penerapan persamaan *manhattan distance* pada penelitian ini ditunjukkan pada persamaan 2.36. Gambar 3.16 merupakan sebuah contoh citra berukuran 4x4.

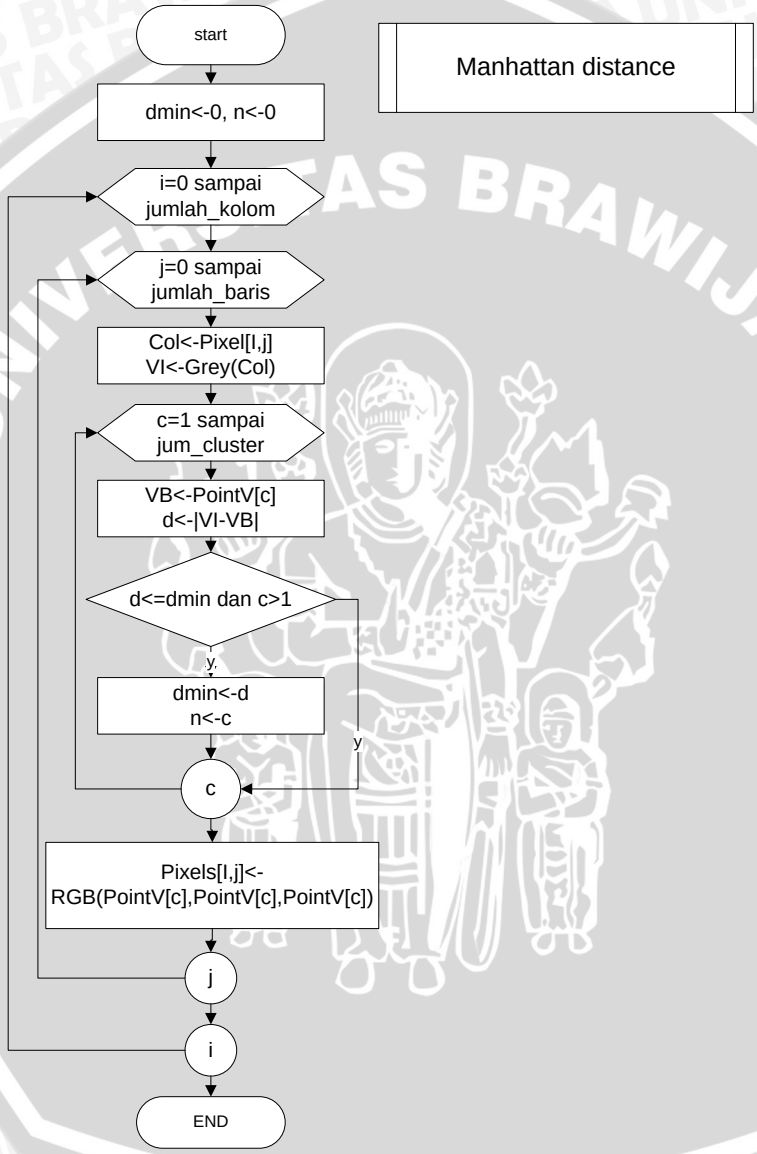
200 100 40	100 125 100	200 220 200	200 100 125
140 160 130	153 172 163	160 200 220	210 140 120
230 240 250	110 120 110	150 160 170	110 210 230
110 120 100	120 140 150	130 100 50	135 125 115

Gambar 3.16 Citra berukuran *pixel* 4x4

Berdasarkan persamaan 2.36 maka nilai pixel baru dari gambar 3.16 didapatkan dengan melakukan perhitungan berikut.

a. Untuk RGB(200,100,40)
   dm = |200-153+|100-172|+|40-163| = 242
b. Untuk RGB(100,125,100)
   dm = |100-153+|125-172|+|100-163| = 163
c. Untuk RGB(200,220,200)
   dm = |200-153+|220-172|+|200-163| = 132
d. Untuk RGB(140,160,130)
   dm = |140-153+|160-172|+|130-163| = 58
e. Untuk RGB(160,200,220)
   dm = |160-153+|200-172|+|220-163| = 92
f. Untuk RGB(230,240,250)
   dm = |230-153+|240-172|+|250-163| = 232
g. Untuk RGB(110,120,110)
   dm = |110-153+|120-172|+|110-163| = 148
h. Untuk RGB(150,160,170)
   dm = |150-153+|160-172|+|170-163| = 22

Gambar 3.17 merupakan flowchart dari manhattan distance sebagai proses akhir reduksi warna.



Gambar 3.17 *Flowchart* Proses *Manhattan Distance*

Berdasarkan perhitungan tersebut maka nilai *distance* terkecil adalah 22 yang terdapat pada perhitungan ke h. Sehingga pixel tetangga yang paling mendekati kemiripan dengan pixel pusat adalah pixel dengan nilai RGB (150, 160, 170).

3.3 Perancangan Uji Coba

Perancangan uji coba ini berupa input citra dengan warna RGB. Hal ini dilakukan untuk memeperlihatkan terjadinya reduksi warna pada gambar sehingga benar-benar terlihat hasilnya. Ukuran pixel yang digunakan uji coba digunakan ukuran yang tidak terlalu besar antara 200x200. Karena semakin besar ukuran pixel semakin lama waktu komputasi. Adapun hal-hal yang harus dilakukan :

a. Mempersiapkan citra uji dengan ukuran pixel yang telah ditetapkan untuk melakukan manipulasi ukuran pixel bisa menggunakan adobe photosop atau program pengolahan citra yang lain.
b. Menggunakan eksetensi file .bmp. Ketentuan ini hanya untuk memberikan standard keseluruhan file citra dengan ekstensi yang sama.
c. Parameter input untuk uji coba yang akan dilakukan dalam penelitian ini yaitu h untuk bagian *initial color* dan *mean shift procedure*. Serta jumlah iterasi untuk *mean shift procedure.* Sedangkan tolak ukur hasil citra digunakan jumlah warna dan *connected component*.
d. Uji coba dilakukan dengan citra yang sama tetapi dengan nilai parameter uji yang berbeda.
e. Hasil uji coba yang ditampilkan berupa jumlah warna yang dihasilkan dan *connected component*.

3.4 Pengujian citra

Citra yang telah dihasilkan dengan menggunakan 3 parameter input dan 2 parameter hasil akan digunakan untuk menguji

tingkat kesuksesan reduksi warna. Tabel 3.1 memberikan gambaran pengujian terhadap citra.

Tabel 3.1 Pengujian (I)

Nama citra	p	h	Iterasi	Jumlah Warna Akhir		Connected component	
				Citra dengan EPSF	Citra Tanpa EPSF	Citra dengan EPSF	Citra Tanpa EPSF

Untuk melakukan pengujian terhadap citra yang diperoleh dengan menggunakan reduksi warna, maka beberapa hal yang harus diketahui terlebih dahulu :

a. Mendapatkan jumlah warna akhir dan *connected component* pada citra yang dilakukan dengan pengolahan tehnik EPSF dan tanpa menggunakan tehnik EPSF.
b. Membandingkan citra dengan citra biner untuk memperjelas hasil segmentasi.
c. Kualitas citra diketahui dengan menggunakan *Mean Square Error* (MSE) menggunakan persamaan 2.37.

Sedangkan pengujian yang kedua didasarkan pada visualisasi setiap orang terhadap hasil dari citra yang diperlihatkan pada tabel 3.2.

Tabel 3.2 Pengujian (II)

Citra Asli	Citra Tanpa EPSF	Citra dengan EPSF
	V	
		V

Tanda (V) menunjukkan hasil yang diperoleh lebih baik pada citra tersebut.

BAB IV
ANALISA DAN PEMBAHASAN

Pada bab ini akan dibahas mengenai implementasi dari representasi rancangan ke bahasa pemrograman yang dimengerti oleh komputer. Bab ini akan membahas hasil implementasi yang dihasilkan oleh perangkat lunak, untuk selanjutnya dilakukan evaluasi hasil kualitas citra setelah mengalami reduksi warna.

4.1 Lingkungan Implementasi

Lingkungan implementasi meliputi lingkungan perangkat keras serta lingkungan perangkat lunak.

4.1.1 Lingkungan perangkat keras

Perangkat keras yang digunakan dalam pengembangan sistem pembuatan citra reduksi warna ini adalah:

1. Prosesor Mobile Intel(R) Pentium(R) 4 – M CPU 1.73 GHz.
2. RAM 512 MB
3. *Harddisk* dengan kapasitas 160 GB
4. VGA Mobile Intel(R) 128 MB

4.1.2 Lingkungan perangkat lunak

Perangkat lunak yang digunakan dalam pengembangan sistem pembuatan citra reduksi warna ini adalah :

1. Sistem Operasi *Microsoft Windows XP*
2. *Borland Delphi 7*
3. *Adobe Photoshop 7*

4.2 Implementasi Program

Berdasarkan perancangan perangkat lunak pada subbab 3.2 maka pada subbab ini akan dibahas mengenai implementasi dari perancangan tersebut.

4.2.1 *Input*

Proses *input* dalam bentuk citra dengan tipe file bmp dan akan ditampilkan dua citra olahan (sampai *initial color*) dan hasil serta sebaran warna dari citra yang diambil. Dari sebaran warna tersebut

dapat diketahui frekuensi histogramnya. Source Code 4.1 merupakan kode program untuk melakukan *load image* sedangkan Source Code 4.2 adalah code dalam menampilkan sebaran warna dan menghitung histogram dari citra.

```
// jika load citra gagal
if (ODLoad.Execute = False) then exit

// jika load citra berhasil
  else
  begin
  if picture<>nil then
      picture.Destroy;
  if scrBitmap<>nil then
      scrBitmap.Destroy;
  if processedBitmap<>nil then
      processedBitmap.Destroy;

    EdLoad.Text := ODLoad.Files.Text;
    picture := TPicture.Create;
    SLoad := ODLoad.Files.Strings[0];
    picture.LoadFromFile(SLoad);

//penyimpanan citra ke dalam objek Bitmap
    scrBitmap := TBitmap.Create;
    processedBitmap := TBitmap.Create;
    Imagedistribution := TBitmap.Create;

    scrBitmap.Height := picture.Graphic.Height;
    scrBitmap.Width := picture.Graphic.Width;
    scrBitmap.Canvas.Draw(0,0,picture.Graphic);

    processedBitmap.Height := picture.Graphic.Height;
    processedBitmap.Width := picture.Graphic.Width;
    H := picture.Graphic.Height;
    W := picture.Graphic.Width;
    processedBitmap.Canvas.Draw(0,0,picture.Graphic);

    Imagedistribution.Height := 255;
    Imagedistribution.Width := 255;

//mengambil nilai objek picture dan menampilkannya ke
dalam Imgload dan ImgLoad2
    ImgLoad.Picture := picture;
    ImgLoad2.Picture := picture;
    noPoint := 0;
```

```
//mengetahui sebaran warna dari citra
    DistribusiSpace2;
```

Source Code 4.1 *Source Code Load* Citra

```
//mengatur nilai frekuensi histogram untuk tiap warna
yaitu 0
for i:=0 to imgdist2.Picture.Width do
  for j:=0 to imgdist2.Picture.Height do
  begin
    P[i,j]:=0;
  end;

//mendapatkan frekuensi histogram tiap warna pada
distribusi warna
for i := 1 to ImgLoad2.Picture.Width-1 do
  for j :=1 to ImgLoad2.Picture.Height-1 do
  begin
    Col := ImgLoad2.Canvas.Pixels[i,j];
    R := GetRValue(Col);
    G := GetGValue(Col);
    B := GetBValue(Col);

    //menampilkan distribusi warna pada ruang 2D
    imgdist2.Canvas.Pixels[R,G] := RGB(R,G,B);
    if((R<>255) and (G<>255)) then
    begin
      P[R,G]:=P[R,G]+1;
    end;
  end;

for i := 1 to imgdist2.Picture.Width-1 do
  for j :=1 to imgdist2.Picture.Height-1 do
  begin
    Col := imgdist2.Canvas.Pixels[i,j];
    Htgr2[i,j] := P[i,j] / Luas;
    if(GetRValue(Col)<>255) then
    begin
       Inc(jum_color);
    end;
  end;
```

Source Code 4.2 *Source Code* Distribusi Warna Ruang 2d Dan Frekuensi *Histogram*

Histogram didapatkan dari keseluruhan warna pada citra dan menghitung frekuensi *grayscale* pixel. Dan perwakilan tiap warna diletakkan pada area distribusi warna yaitu

`imgdist2.Canvas.Pixels[R,G]` dan memasukkannya pada array `P[R,G]` yang pada awalnya nilainya bernilai 0 untuuk menghitung frekuensi histogram.

a. R merupakan index *array* dengan mengambil nilai *red* pada pixel citra.
b. G merupakan index *array* dengan megambil nilai *grey* pada pixel citra.
c. Array `P[R,G]` nilai akan meningkat jika terdapat index yang sama pada citra.
d. Diketahui bahwa ukuran nilai *red*(R) dan *gre*y(G) untuk index tersebut maksimal memiliki nilai 255. Sehingga ukuran dari distribusi warna adalah 255 x 255.

Gambar 4.1 menunjukkan citra input dan hasil distribusi warna citra input dalam ruang 2D.

Gambar 4.1 Hasil *Input* Citra D.Bmp Dan Distribusi Warna Pada Ruang 2D

4.2.2 *Edge Preserving Smoothing Filter*

Pada proses ini parameter yang digunakan dalam pengolahan citra sehingga mempengaruhi hasil dari citra yaitu parameter dengan variable p. *Source Code* 4.3 merupakan kode program dalam proses EPSF dalam reduksi warna dokumen digital.

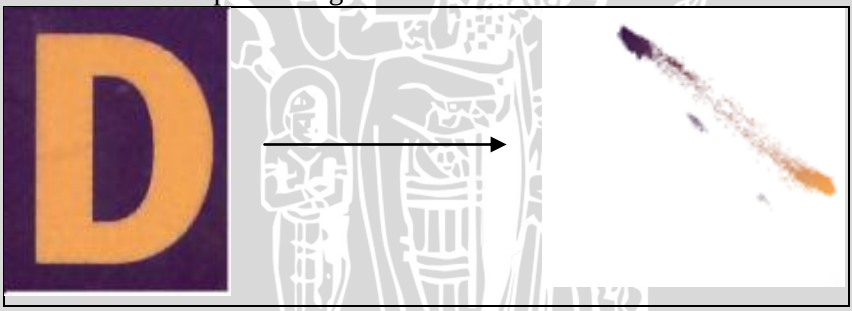
```
//nilai parameter p
 p := StrToInt(EdP.Text);

//mengolah citra dengan teknik EPSF
for i:= 1 to ImgLoad.Picture.Width-1 do
  for j:= 1 to ImgLoad.Picture.Height-1 do
```

```
  begin
   Col := ImgLoad.Canvas.Pixels[i, j];
   InRed := GetRValue(Col);
   InGreen := GetGValue(Col);
   InBlue := GetBValue(Col);

   totalc := 0;
   n := 0;

   if (i < W-1) or (j < H-1) then
   begin
    for x := i-1  to (i + 1) do
    begin
     m := 0;
     for y := j - 1 to (j + 1) do
      begin
       Col := ImgLoad.Canvas.Pixels[x,y];
       d[n,m] := ((InRed - GetRValue(Col))+(InGreen -
GetGValue(Col))+(InBlue - GetBValue(Col)))/(3*255);
       if (x=i) and (y=j) then
        c[n,m] := 0
       else
        c[n,m] := Power(1 - d[n,m], p);
       totalc := totalc + c[n,m];
       Inc(m);
      end;
      Inc(n);
     end;

     totalR := 0;
     totalG := 0;
     totalB := 0;

     n:=0;
     for x := i-1  to (i + 1) do
     begin
      m := 0;
      for y := j - 1 to (j + 1) do
       begin
       Col := ImgLoad.Canvas.Pixels[x,y];
       totalR := totalR + GetRValue(Col)*c[n,m];
       totalG := totalG + GetGValue(Col)*c[n,m];
       totalB := totalB + GetBValue(Col)*c[n,m];
       Inc(m);
      end;
      Inc(n);
     end;
```

```
      totalR := (totalR/totalc);
      totalG := (totalG/totalc);
      totalB := (totalB/totalc);
...

      processedBitmap.Canvas.Pixels[i,j] :=
RGB(Round(totalR), Round(totalG), Round(totalB));
    end;
   end;
```

Source Code 4.3 *Source Code EPSF*

Terjadinya *smoothing* pada tahapan ini karena yaitu terletak pada eksekusi kode program untuk variabel totalR, totalG, dan totalB pada Source Code 4.3. Dimana pixel pusat nilainya ditentukan pixel tetangga dan menarik rata-rata totalR, totalG, totalB. Sehingga memungkinkan pixel tersebut memiliki nilai yang hampir sama dengan *pixel* tetangganya. Dan proses reduksi warna juga terjadi pada tahap ini karena tiap *pixel* yang diolah bisa memiliki nilai yang sama dengan *pixel* tetangga. Nilai *index* R dan G yang sama akan disimpan dalam area distribusi warna yang sama sehingga mempengaruhi jumlah warna baru sebagai kandidat pusat *cluster* dan frekuensi *histogram*. Gambar 4.2 merupakan citra hasil EPSF dan distribusi warna pada ruang 2D.



Gambar 4.2 Hasil *EPSF* Citra D.Bmp Dan Distribusi Warna Pada Ruang 2d Dengan P = 10, H = 16 Dan Iterasi = 3

4.2.3 Sobel Edge Detection

Pada proses ini variabel Sx dan Sy adalah matriks yang digunakan dalam teknik deteksi tepi *sobel*. Sedangkan variabel Gxm, Gym, Gxh, Gyh, Gxb, Gyb merupakan gradien dengan arah x dan y yang digunakan untuk memperoleh gradien pada warna RGB. Dan

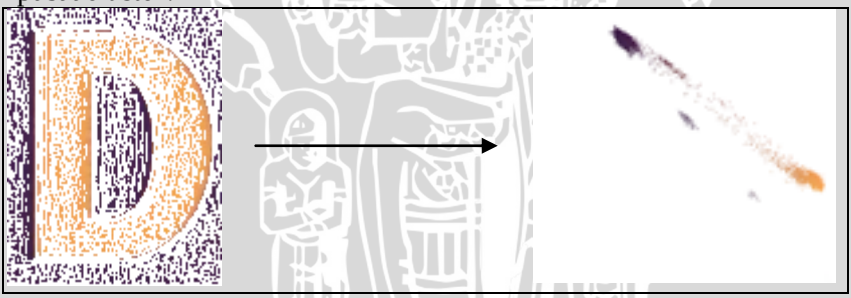
nilai gradien yang diambil adalah gradien yang memiliki nilai paling besar diantara tiga nilai gradien. Source Code 4.4 merupakan kode program dalam proses *sobel edge detection.*

```
//matriks sobel
Sx[0,0] := -1; Sx[0,1] := 0; Sx[0,2] := 1; Sx[1,0] := -
2; Sx[1,1] := 0; Sx[1,2] := 2; Sx[2,0] := -1; Sx[2,1] :=
0; Sx[2,2] := 1;
Sy[0,0] := 1; Sy[0,1] := 2; Sy[0,2] := 1; Sy[1,0] := 0;
Sy[1,1] := 0; Sy[1,2] := 0; Sy[2,0] := -1; Sy[2,1] := -
2; Sy[2,2] := -1;

...

for i:= 1 to ImgLoad.Picture.Width-1 do
 for j:=1 to ImgLoad.Picture.Height-1 do
  begin
   Gxm := 0; Gym := 0; Gxh := 0; Gyh := 0; Gxb := 0; Gyb
:= 0;
   n :=0;
   if (i < W-1) or (j < H-1) then
   begin
    for x := i-1  to (i + 1) do
    begin
     m := 0;
     for y := j - 1 to (j + 1) do
     begin
      Col := ImgLoad.Canvas.Pixels[x,y] ;
      Gxm := Gxm + Sx[n,m]*GetRValue(Col);
      Gym := Gym + Sy[n,m]*GetRValue(Col);
      Gxh := Gxh + Sx[n,m]*GetGValue(Col);
      Gyh := Gyh + Sy[n,m]*GetGValue(Col);
      Gxb := Gxb + Sx[n,m]*GetBValue(Col);
      Gyb := Gyb + Sy[n,m]*GetBValue(Col);
      Inc(m);
     end;
     Inc(n);
    end;

    Gr := sqrt(Power(Gxm,2) + Power(Gym,2));
    Gh := sqrt(Power(Gxh,2) + Power(Gyh,2));
    Gb := sqrt(Power(Gxb,2) + Power(Gyb,2));j

    Gmax := Max(Gr, Gh);
    Gmax := Max(Gmax, Gb);
```

```
    if (Gmax>255) then
      Gresult := 255
    else if (Gmax<0) then
      Gresult := 0
    else
      Gresult := Round(Gmax);
...
    processedBitmap.Canvas.Pixels[i,j]  :=  RGB(Gresult  ,
Gresult , Gresult );
   end;
  end;
...
     if (GetRValue(Col) > GetRValue(Col2)) then
     begin
       value := True;
       Break;
     end
...
```

Source Code 4.4 *Source Code Sobel Edge Detection*

Gambar 4.3 merupakan citra hasil proses *sobel edge detection* dan distribusi warna untuk mendapatkan sampel warna sebagai kandidat pusat *cluster*.



Gambar 4.3 Hasil *Sobel Edge Detection* Citra D.Bmp Dan Distribusi Warna Pada Ruang 2d Dengan P = 10, H = 16 Dan Iterasi = 3

Tahap ini bekerja untuk mendapatkan sampel baru sebagai kandidat pusat *cluster* baru yaitu dengan mencari nilai maksimum dari nilai gradien RGB pada masng-masing *pixel* seperti yang ditunjukkkan pada Source Code 4.4 . Dimana gradien masing-masing variable yang digunakan yairu Gr, Gh, dan Gb. Dan nilai maksimum yang

didapatkan ditampung pada variabel Gmax. Dan nilai Gmax akan dikondisikan menggunakan pernyataan (if) untuk mendapatkan Gresult. Jika telah didapatkan nilai Gresult dari keseluruhan *pixel* maka Gresult dibandingkan dengan Gresult pixel tetangga jika memnuhi kondisi if (GetRValue(Col) > GetRValue(Col2)) maka *pixel* tersebut diatur menjadi *pixel* berwana putih dan jika tidak diset dengan nilai asli. Sehingga distribusi yang dihasilkan sebagai kandidat pusat *cluster* menurun karena kondisi tersebut.

4.2.4 Initial Color

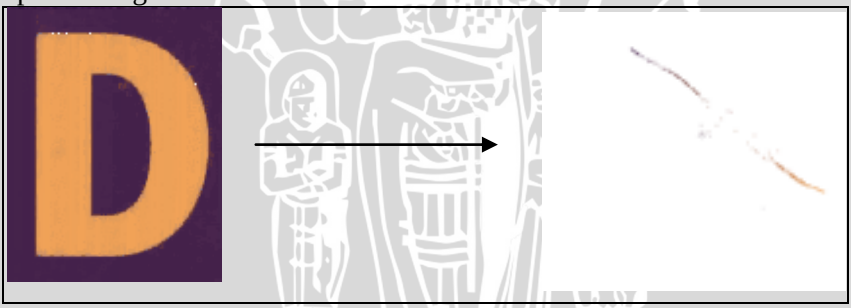
Variabel hin adalah area yang digunakan untuk melakukan proses ini. Dan ph merupakan variable untuk memperoleh frekuensi *histogram* dari warna yang dipilih. Sedangkan rm, gm, dan bm adalah variable yang didapatkan dari hasil bagi antara rmi, gmi, bmi dengan nilai variable pr, pg, pb. Dan variable rm, gm, dan bm adalaha nilai variable yang digunakan untuk nilai baru dari nilai RGB pada *pixel* yang diolah. Sedangkan sgnLbl digunakan untuk menandai bahwa *pixel* tersebut telah ditandai. Source Code 4.5 merupakan kode program untuk *initial color*.

```
Hin := StrToInt(edH.Text);
ph := 0;
…

for i := 1 to ImgLoad2.Picture.Width-1 do
 for j :=1 to ImgLoad2.Picture.Height-1 do
 begin
 Col := ImgLoad2.Canvas.Pixels[i,j];
 Rp := GetRValue(Col);
 Gp := GetGValue(Col);

if (Rp <> 255) and (Gp <> 255) and (sgnLbl[Rp,Gp]=False)
then
 begin
 …
 for x := Rp-hin to Rp+hin do
  for y := Gp-hin to Gp+hin do
  begin
  Col := imgdist2.Canvas.Pixels[x,y];
  R := GetRValue(Col);
  G := GetGValue(Col);
```

```
  B := GetBValue(Col);
 if (x>=1) and (y>=1) and (x<=Imgdist2.Picture.Width-1)
and (y<=Imgdist2.Picture.Height-1) then
  begin
  ph := Htgr2[R,G];
  rmi := rmi + R* ph;
  gmi := gmi + G* ph;
  bmi := bmi + B* ph;
  pr := pr + ph;
  pg := pg + ph;
  pb := pb + ph;
  if (R=Rp) and (sgnLbl[x,y]=False) then
   sgnLbl[x,y] := True;
  end;
 end;
 if ((pb<>0)or(pg<>0)or(pr<>0))then
  begin
  rm := Round(rmi / pr);
  gm := Round(gmi / pg);
  bm := Round(bmi / pb);

processedBitmap.Canvas.Pixels[i,j] := RGB(rm,gm,bm);
…
```

Source Code 4.5 *Source Code Initial Color*

Gambar 4.4 adalah gambar hasil initial color dan distribusi warna pada ruang 2D.



Gambar 4.4 Hasil *Initial Color* Citra D.Bmp Dan Distribusi Warna Pada Ruang 2d Dengan P = 10, H = 16 Dan Iterasi = 3

Untuk mendapatkan pusat *cluster* baru yang terlihat pada distribusi warna lebih sedikit dibandingkan sebelumnya. Jumlah warna pada distribusi semakin berkurang karena posisi tiap pixel pada area distribusi warna untuk index R dan G untuk perwakilan tiap *pixel*

pada citra memiliki nilai yang sama. Hal ini dipengaruhi adanya kode

```
rmi := rmi + R* ph;
gmi := gmi + G* ph;
pr := pr + ph;
pg := pg + ph;
```

Persamaan tersebut akan berulang pada pixel tetangga. Sehingga nilai akhir untuk posisi R dan G adalah

```
rm := Round(rmi / pr);
gm := Round(gmi / pg);
```

Kode tersebut akan memberikan nilai rata-rata rmi terhadap frekuensi *histogram* pada pixel tetangga. Dan nilai rata-rata inilah yang memungkinkan tiap *pixel* tetangga memiliki nilai yang sama satu sama lain dan menyebabkan jumlah warna (kandidat pusat cluster) menjadi lebih sedikit dibandingkan jumlah warna yang terdapat pada distribusi warna sebelumnya.

4.2.5 *Mean Shift Procedure*

Variabel x dan y menunjukkan koordinat titik letak *pixel* pada daerah distribusi warna. Variabel untuk menyimpan nilai RGB *pixel input* yaitu inR, inG, inB. Dan akan didapatkan nilai akumulasi dari masing-masing niali *input* untuk mendapatkan warna *input* yang baru. Ource Code 4.6 adalah kode program yang digunakan untuk proses *mean shift procedure*.

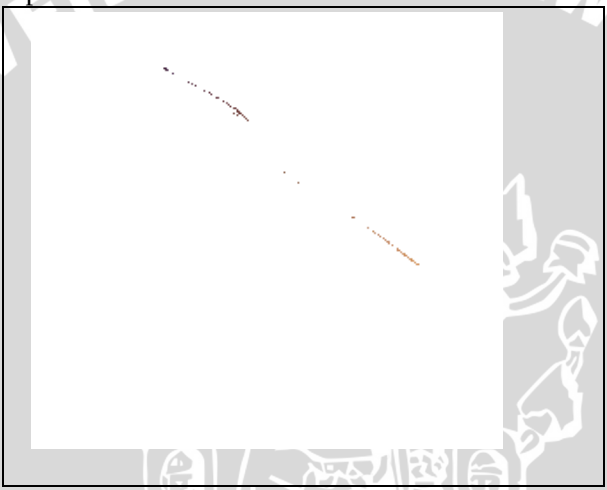
```
jum_color :=0;

...
dc := 0;
iterasi := StrToInt(edIterate.Text);
radious := 100;
hmean := StrToInt(edH.Text);
ck := hmean*hmean*hmean;
s:=0;
k:=10;
c:=RandomRange(1,noPoint);
```

```
d_wo:=0;
for c:=1 to noPoint do
 begin
 x := PointX[c];
 y := PointY[c];
 inputColor := imgdist.Canvas.Pixels[x,y];
 for f := 1 to iterasi do
  begin
  inR := GetRValue(inputColor);
  inG := GetGValue(inputColor);
  inB := GetBValue(inputColor);
  currentRDistance := 0;
  currentGDistance:=0;
  currentBDistance := 0;
  R:=0; G:=0; B:=0;
  accumulatedRvalues := 0;
  accumulatedGvalues :=0;
  accumulatedBvalues := 0;
  totalcluster := 1; dc:=0;
  JumXR:=0; JumXG:=0;; JumXB:=0;
  GiR:=0; GiG:=0; GiB:=0;
  JumGiR:=0; JumGiG:=0; JumGiB:=0;

  for i:=x-hmean to x+hmean do
   for j:=y-hmean to y+hmean do
   begin
   if (i>=1) and (j>=1) and (i<=imgdist.Picture.Width-1)
and (j<=imgdist.Picture.Height-1) then
   begin
    Col := imgdist.Canvas.Pixels[i,j];
    if (GetRValue(Col)<>255) then
    begin
    currentRDistance := GetRValue(Col) - inR;
    currentGDistance := GetGValue(Col) - inG;
    currentBDistance := GetBValue(Col) - inB;

    if  (currentRDistance<10)  and  (currentGDistance<10)
and (currentBDistance<10) then
    begin
    Inc(totalcluster);
  accumulatedRvalues:=accumulatedRvalues+GetRValue(Col);
  accumulatedGvalues:=accumulatedGvalues+GetGValue(Col);
  accumulatedBvalues:=accumulatedBvalues+GetBValue(Col);
  end;
  end;
  end;
end;
```

```
inputColor :=RGB(round(accumulatedRvalues/totalcluster),
round(accumulatedGvalues/totalcluster),
round(accumulatedBvalues/totalcluster));

end;

...
```

Source Code 4.6 *Source Code Mean Shift Procedure*

Gambar 4.5 merupakan gambar distribusi warna ruang 2D proses mean shift procedure



Gambar 4.5 Distribusi Warna *Mean Shift Procedure* Citra D.Bmp Pada Ruang 2d Dengan P = 10, H = 16 Dan Iterasi = 3

Permasalahn yang timbul pada tahapan ini jika melihat hasil distribusi warna yaitu jika pusat *cluster* baru yang diperoleh sedikitnya satu titik pada distribusi warna sehingga citra yang dihasilkan berupa warna homogen untuk seluruh citra. Hal yang menyebabkan terjadinya hal ini kemungkinan

a. Distribusi warna yang dihasilkan pada tahapan sebelumnya antar setiap pixel sangat berdekatan. Dan area h (tinggi) yang ditentukan pada mean shift memiliki jangkuan luas sehingga kumpulan cluster dari distribusi warna terfokus pada satu area tersebut.

b. Hasil *input* warna yang hampir sama dengan *input cluster* yang lain. Sehingga nilai yang dihasilkan menghasilkan nilai yang sama. Dimungkinkan karena kumpulan cluster yang saling berdekatan.

Mean shift terjadi karena adanya pencarian input baru berdasarkan area yang masih berada pada area *distance* yang ditentukan dengan menghitung menggunakan *manhattan distance* yang terletak pada variabel currentRDistance, currentGDistance, dan currentBDistance. Dan dari hasil *distance* yang memiliki *pixel* warna yang mirip berdasarkan kondisi `if (currentRDistance<10) and (currentGDistance<10) and (currentBDistance<10)` akan dimasukkan dalam *cluster* yang sama dan dihitung nilai akumulasi keseluruhan *pixel* yang masuk dalam *cluster* yang sama. Dan untuk mendapatkan *input* baru yang selanjutnya akan dihitung pada iterasi berikutnya yaitu dengan membagi nilai akumulasi yang telah dihasilkan dengan jumlah anggota *cluster* baru yang diproses dalam variabel inputColor.

4.2.6 *Manhattan Distance*

Pada proses ini terdapat variabel RI, GI, dan BI yang digunakan sebagai variabel nilai RGB citra untuk *pixel* pusat. Sedangkan untuk nilai RB, GB, dan BB merupakan variabel untuk nilai RGB citra untuk *pixel* tetangga. Dan d merupakan variabel untuk mendapatkan jarak terdekat antara *pixel* pusat dan *pixel* tetangga. Dan yang memiliki nilai paling kecil yang digunakan sebagai nilai RGB *pixel* yang baru. Source Code 4.7 adalah kode program untuk proses *manhattan distance*.

```
for i:= 1 to ImgLoad2.Picture.Width-1 do
 for j:=1 to ImgLoad2.Picture.Height-1 do
 begin
  ColI := ImgLoad2.Canvas.Pixels[i,j];
  RI := GetRValue(ColI);
  GI := GetGValue(ColI);
  BI := GetBValue(ColI);

  for c:= 1 to noPoint do
  begin
   RB := PointR[c];
```

```
    GB := PointG[c];
    BB := PointB[c];
    d := Abs(RI-RB)+Abs(GI-GB)+Abs(BI-BB);
    if (d<=dmin) and(c>1) then
    begin
    dmin := d;
    n := c;
    end
    else if(c=1) then
    begin
    dmin := d;
    n:=c
    end;
   end;
   ImgLoad2.Canvas.Pixels[i,j]:=
RGB(PointR[n],PointG[n],PointB[n]);
end;
```

Source Code 4.7 *Source Code Manhattan Distance*

Gambar 4.6 merupakan citra hasil dari reduksi warna setelah melalui proses *manhattan distance*.

Gambar 4.6 Citra Hasil Reduksi Warna Dengan *Manhattan Distance*

Hasil distribusi wana yang menggambarkan jumlah pusat *cluster* yang didapatkan dari *mean shift procedure* akan diproses pada citra *output* yang telah diproses pada tahapan sebelumnya. Dari keseluruhan wana yang terdapat pada citra didekatkan pada warna *pixel* yang diperoleh pada *mean shift* dengan menggunakan perhitungan jarak *manhattan* (d). Manhattan distance (d) akan diambil yang memiliki nilai paling kecil. Dan jika telah diperoleh maka pusat *cluster* tersebut yang lebih dekat dengan warna citra.

4.3 Implementasi Uji Coba

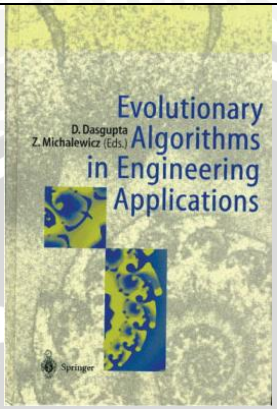
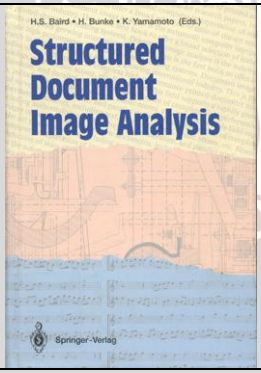
Pada subbab ini akan dilakukan pembahasan mengenai pengujian yang telah dilakukan pada sistem dan hasil evaluasi dari hasil yang dikeluarkan sistem.

4.3.1 Evaluasi Citra Uji

Citra uji ini memiliki 2 parameter yang akan dipertimbangkan untuk mengevaluasi citra hasil uji. Parameter tersebut adalah jumlah warna dan *connected component*. Parameter ini akan diuji apakah akan terjadi penurunan jumlah warna dan *connected component* ataukah akan terjadi peningkatan terhadap jumlah warna dan *connected component* pada citra hasil uji. Kedua parameter tersebut digunakan untuk mempertimbangkan citra hasil uji coba yang diukur menggunakan *Mean Square Error* (MSE). Tabel 4.1 menunjukkan citra awal yang akan diuji coba.

Tabel 4.1 Citra Awal Dengan Jumlah Warna Dan Connected Component

Nama file	Citra	Jumlah Warna I	Connected Component
D.bmp		2720	13488
Asli.bmp		5868	15242

Ma.bmp		7599	88864
Dead.bmp		4587	12347
Plus.bmp		5127	53187
Struct.bmp		4507	62310

4.3.2 Evaluasi Citra Hasil Uji Coba Berdasarkan Parameter I (Iterasi), H (Tinggi Area) Dan Mse (*Mean Square Error*)

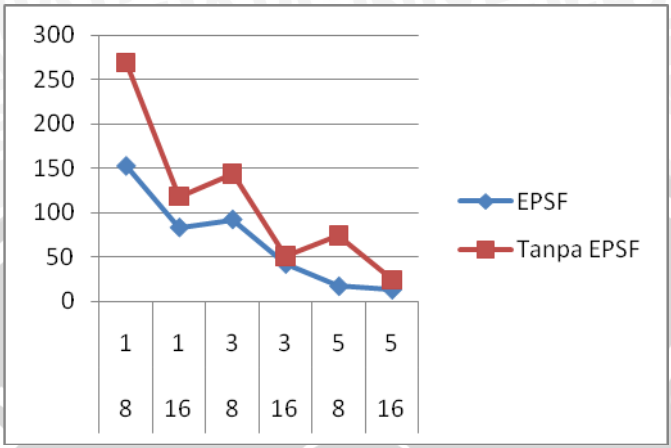
Dari citra yang terdapat pada Tabel 4.2 akan diberikan perlakuan yang berbeda dengan memberikan parameter input yang berbeda dan akan dilakukan pengujian terhadap dua citra hasil dari tiap parameter yaitu citra yang menerapkan EPSF dan tanpa EPSF. Tujuan dari pemberian parameter yang berbeda pada tiap pengujian yaitu untuk membandingkan hasil citra yang dihasilkan. Tabel 4.2 Menunjukkan hasil dari pengujian terhadap beberapa citra.

Tabel 4.2 Hasil Uji Coba

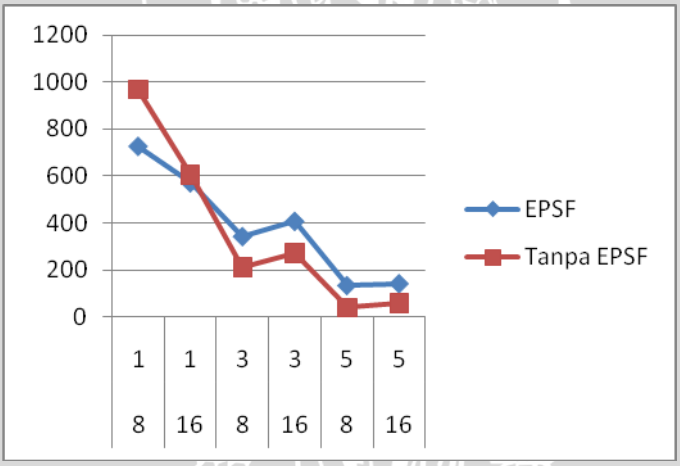
File (bmp)	Parameter Input			Jumlah Warna		Connected Component (CC)		Mean Squarer Error (MSE)	
	p	h	i	r1	r2	CC1	CC2	MSE1	MSE2
D	10	8	1	152	269	2042	6049	6.2824	1.0037
D	10	16	1	83	118	1198	3268	7.0504	1.3244
D	10	8	3	92	144	1290	5009	7.43577	1.6089
D	10	16	3	42	51	709	1969	10.0889	2.9054
D	10	8	5	17	74	801	3207	81.4042	102.390
D	10	16	5	13	24	103	1494	81.557	64.1195
Asli	10	8	1	726	968	5300	8510	14.897	3.7455
Asli	10	16	1	570	604	3645	6023	14.6747	5.4498
Asli	10	8	3	341	210	3217	3365	13.9656	6.0885
Asli	10	16	3	406	270	3049	3316	14.2353	5.4498
Asli	10	8	5	132	39	819	834	17.4556	72.5813
Asli	10	16	5	140	59	2369	1693	14.446	9.9722
Ma	10	8	1	460	632	31403	52202	5.7648	0.1981
Ma	10	16	1	288	313	19306	30421	6.2358	0.5364
Ma	10	8	3	363	419	16907	42411	8.2752	0.8838
Ma	10	16	3	211	257	13706	25415	7.596	1.8318
Ma	10	8	5	235	162	8313	13132	50.5912	19.5872
Ma	10	16	5	150	153	7991	10993	31.3547	18.8897

Dead	10	8	1	589	620	4194	5483	14.9621	0.936
Dead	10	16	1	305	306	2756	3082	15.2179	1.2874
Dead	10	8	3	451	353	3703	4847	14.5548	1.1667
Dead	10	16	3	251	252	2505	2573	14.8751	2.9534
Dead	10	8	5	379	176	3583	3128	14.5566	29.5466
Dead	10	16	5	235	134	2600	1982	14.5803	20.0466
Plus	10	8	1	290	374	8659	28578	18.5193	0.3799
Plus	10	16	1	219	187	5835	15962	18.3475	0.7607
Plus	10	8	3	194	264	7419	27459	15.0988	0.5375
Plus	10	16	3	163	123	4394	14352	15.1681	10.3041
Plus	10	8	5	128	187	5788	24341	15.0988	20.0256
Plus	10	16	5	109	68	3642	11713	16.285	21.8952
Struct	10	8	1	259	360	6092	9218	6.8732	2.0792
Struct	10	16	1	192	211	3764	8546	6.8721	1.0701
Struct	10	8	3	95	60	2597	2882	22.4147	29.9291
Struct	10	16	3	147	105	5069	2467	9.4581	9.2543
Struct	10	8	5	14	0	873	521	91.0825	451.481
Struct	10	16	5	1	5	521	917	302.829	117.9318

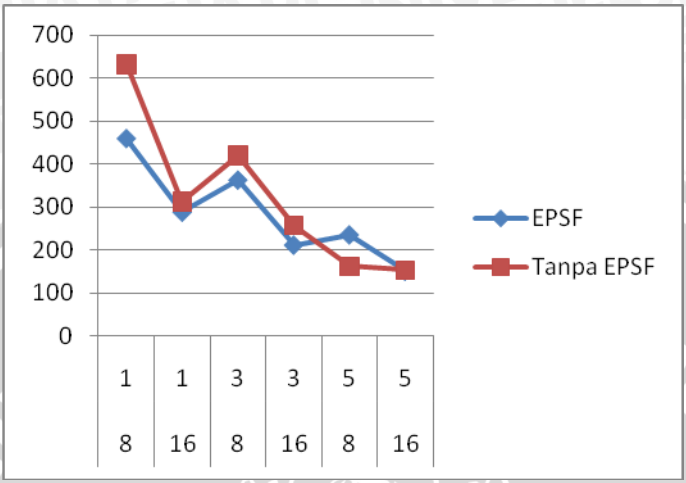
Tabel 4.2 menunjukkan pengaruh parameter p, h, dan i terhadap jumlah warna, *connected component* yang dihasilkan dan kualitas citra dengan melakukan pengukuran menggunakan MSE. Di Tabel 4.3 akan ditunjukkan nilai rata-rata dari ketiga parameter yang dihasilkan. Dan memperkuat pendapat digambarkan grafik masing-masing citra. Hal ini dilakukan untuk mengetahui perbedaan atau pengaruh antara citra dengan parameter *input* yang berbeda serta pengaruh penggunaan EPSF. Gambar 4.7, 4.8, dan 4.9 menggambarkan perbandingan grafik antara citra D.bmp, asli.bmp dan ma.bmp berdasarkan parameter jumlah warna dan *connected component* yang dihasilkan oleh citra hasil uji coba serta perbandingan kualitas citra. Gambar 4.7 adalah perbandingan jumlah warna citra tersebut.


300
250
200
150
100
50
0
1 8
1 16
3 8
3 16
5 8
5 16
EPSF
Tanpa EPSF

(a)


1200
1000
800
600
400
200
0
1 8
1 16
3 8
3 16
5 8
5 16
EPSF
Tanpa EPSF

(b)

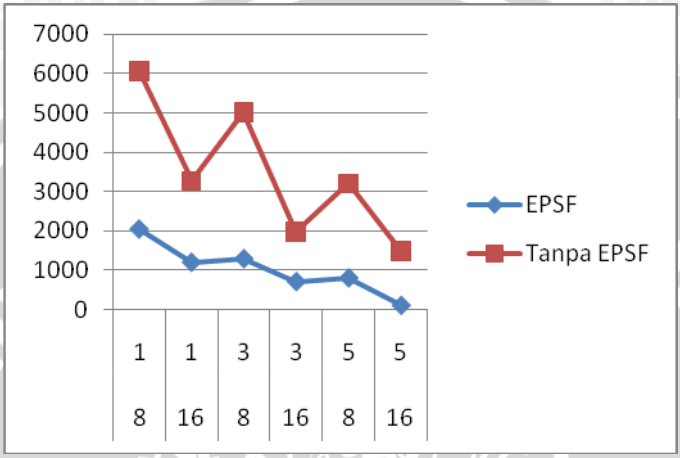


(c)

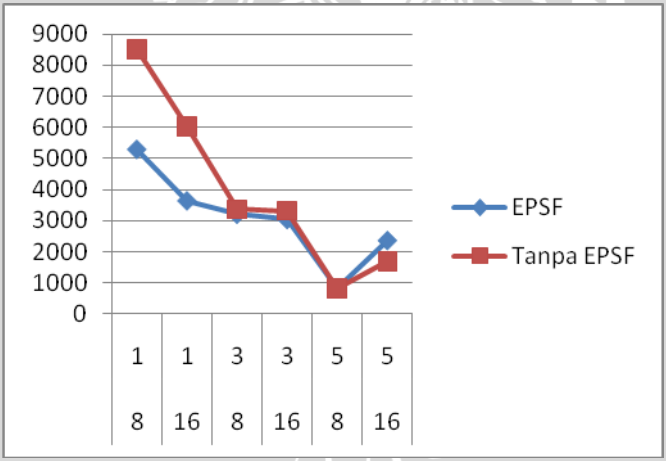
Gambar 4.7 (a) Jumlah Warna Citra D.Bmp (b) Jumlah Warna Citra Asli.Bmp (c) Jumlah Warna Citra Ma.Bmp

Grafik Gambar 4.7 citra D.bmp, asli.bmp dan ma.bmp menggunakan EPSF memperlihatkan pada i=1, i=3, dan i=5 dan h=8 nilai jumlah warna mengalami penurunan. Dan jika i=1, i=3, dan i=5 dan h=16 nilai jumlah warna juga menurun. Dan dapat dikatakan jika parameter i semakin tinggi dan h juga mengalami peningkatan maka terjadi penurunan jumlah warna. Dan berdasarkan pola grafik teknik tanpa EPSF secara bertahap mengurangi jumlah warna secara signifikan dibandingkan dengan EPSF. Hal ini ditunjukkan garis grafik merah yang pada kondisi i dan h meningkat secara terus menurus akan memotong garis grafik warna biru. Dan analisa yang dapat dijelaskan dari naiknay nilai h terhadap jumlah warna yaitu jumlah warna menurun karena pixel tetangga yang digunakan oleh pixel pusat untuk mendapatkan nilai pixel baru semakin luas sehingga memungkinkan pixel pusat untuk pixel selanjutnya memiliki rata-rata nilai pixel baru yang sama. Karena luasan area dari pixel tetangga yang semakin besar. Sedangkan pada iterasi yang semakin banyak akan mempengaruhi hasil input baru untuk kandidat pusat cluster. Semakin banyak interasi maka kemungkinan untuk mendapatkan pusat cluster baru yang sama dengan pusat cluster sebelumnya semakin besar. Dan inilah yang menyebabkan jumlah

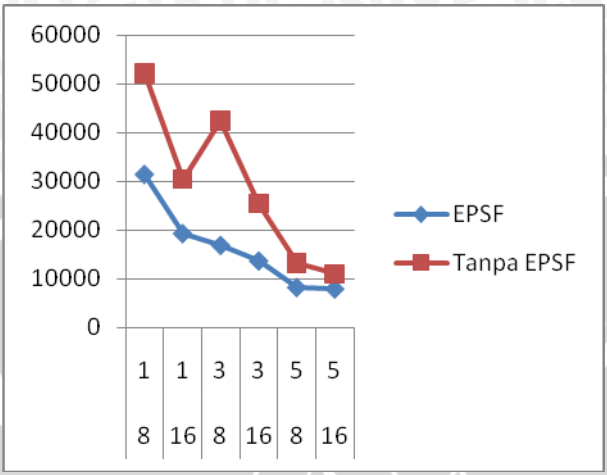
warna semakin menurun. Gambar 4.8 adalah perbandingan jumlah *connected component* ketiga citra tersebut.



(a)



(b)

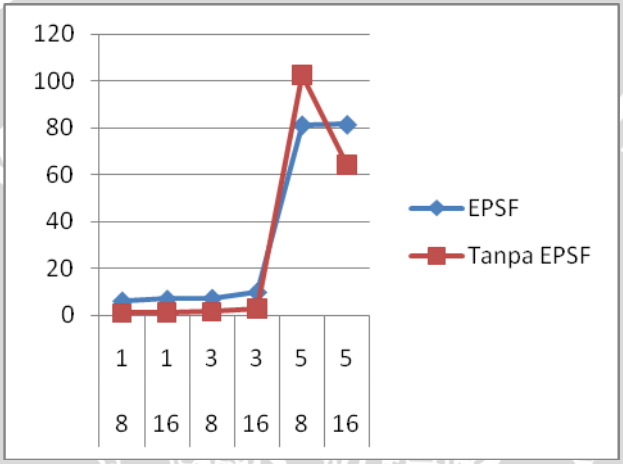


(c)

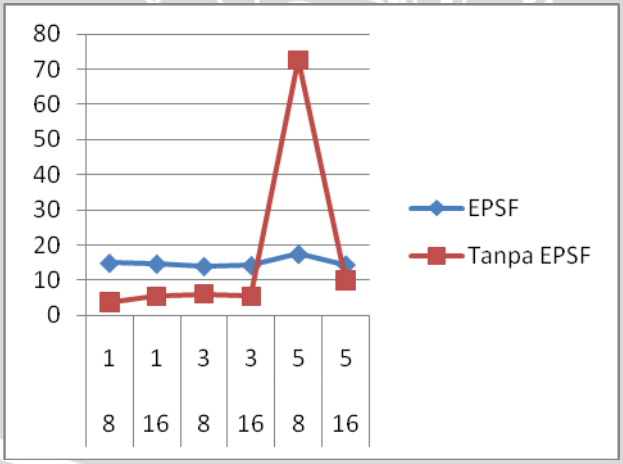
Gambar 4.8 (a) *Connected Component* Citra D.Bmp (b) *Connected Component* Citra Asli.Bmp (c) *Connected Component* Citra Ma.Bmp

Connected component pada reduksi warna dengan melihat 3 hasil grafik 4.8 menunjukkan pada i=1, i=3, dan i=5 dan h=8 nilai CC mengalami penurun setiap tahapnya. Begitu pula jika i=1, i=3, i=5 dan h=16 nilai CC mengalami penurunan. Sehingga dapat dikatakan nilai CC menurun jika nilai h dan i naik. Dan terlihat bahwa penurunan nilai CC secara tajam terjadi pada hasil reduksi dengan menggunakan teknik tanpa EPSF jika dibandingkan EPSF. Citra dikatakan masih berkualitas baik jika nilai CC tinggi. Pada grafik 4.8 menggambarkan jika citra dengan 6 kali uji coba citra tanpa EPSF memiliki kualitas citra yang lebih baik dibandingkan dengan EPSF. Ditunjukkan dengan adanya nilai CC yang tinggi pada beberapa citra hasil uji coba. Seperti yang dijelaskan sebelumnya dalam analisa h and i terhadap jumlah warna. Nilai *Connected Component* menurun sebanding dengan jumlah warna karena CC menghubungkan tiap pixel dengan pixel tetangga yang memiliki nilai pixel yang sama. Dan jika nilai h semakin besar maka kemungkinan pixel tetangga yang selanjutnya menjadi pixel pusat memiliki nilai yang sama dengan pixel pusat sebelumnya semakin besar karena memiliki pixel tetangga yang hampir sama. Sedangkan pengaruh i yaitu kemungkinan dihasilkannya input baru yang sama dengan hasil

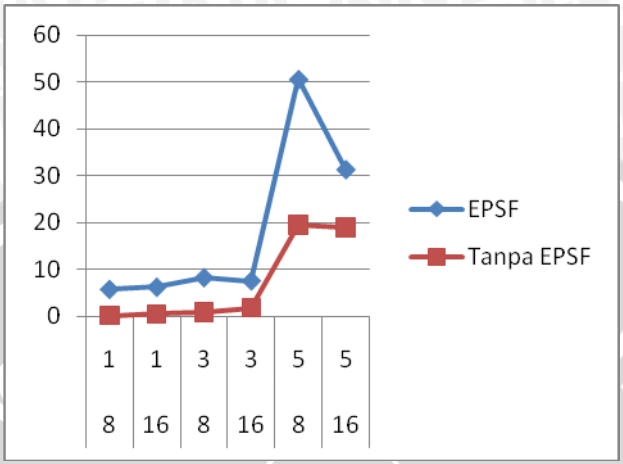
input baru untuk pixel selanjutnya karena semakin banyak perulangan maka kemungkinan untuk mendapatkan nilai pixel yang sama pada pixel tetangga semakin besar. Dan inilah yang menjadikan hubungan antar tiap pixel tetangga terdapat dalam satu koneksi CC. Dan Gambar 4.9 menunjukkan perbandingan kualitas citra yang diukur dengan menggunakan MSE.



(a)



(b)



(c)

Gambar 4.9 (a) Mse Citra D.Bmp (b) Mse Citra Asli.Bmp (c) Mse Citra Ma.Bmp

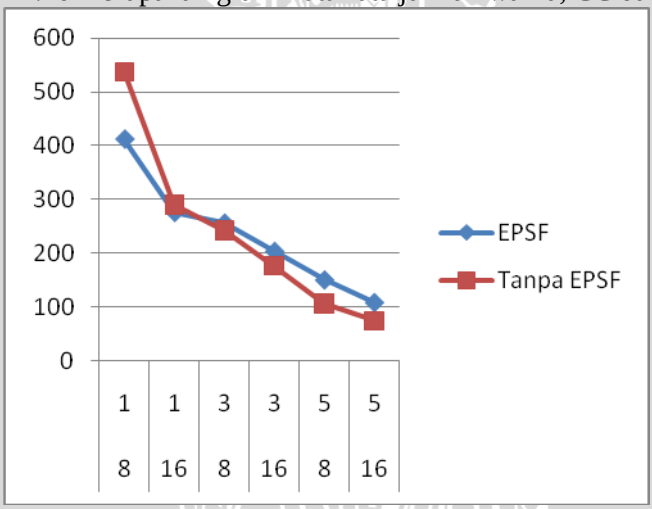
Grafik pada Gambar 4.9 menunjukkan kualitas citra dengan nilai MSE. Dari ketiga gambar untuk nilai i=1, i=3, i= 5 dan h= 8 nilai *error* meningkat. Begitu juga untuk nilai i=1, i=3, i=5 dan h=16 nilai *error* akan meningkat secara bertahap. Dapat dikatakan jika nilai h dan i semakin tinggi maka kualitas citra semakin menurun. *Error* tinggi dipengaruhi jumlah warna dan *connencted component* yang rendah karena kedua parameter inilah yang menunjukkan kualitas citra dan tingkat kemiripan dengan citra asli jika untuk reduksi warna ini. Dan untuk 6 percobaan masing-masing pada teknik EPSF dan tanpa EPSF, *error* semakin tinggi jika h dan i juga naik dan apabila nilai jumlah warna dan *connected component* menurun.

Dari penjelasan sebelumnya, grafik dari masing-masing citra belum bisa dijadikan dasar pengambilan kesimpulan karena hasil masing-masing citra belum dapat mewakili keseluruhan hasil citra yang diuji. Untuk itu dalam penjelasan berikutnya dilakukan pengujian dengan menggunakan rata-rata hasil dari keseluruhan citra yang diuji. Pada Tabel 4.3 menunjukkan hasil statistik dari rata-rata keseluruhan citra.

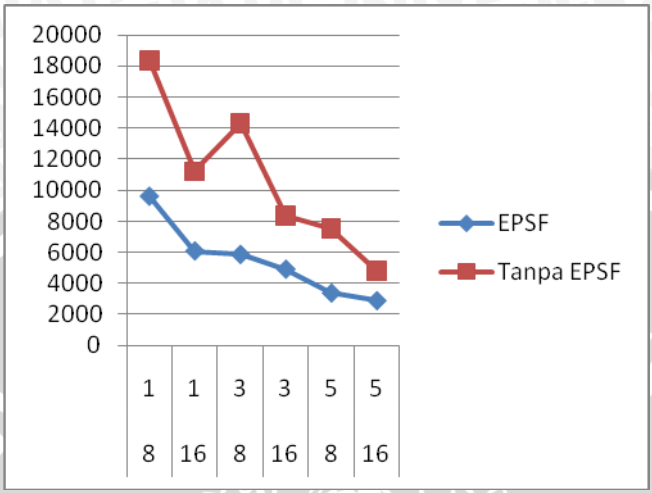
Tabel 4.3 Rata-Rata Citra Uji

Parameter Input			Jumlah Warna (r)		Connected Component (CC)		Mean Squarer Error (MSE)	
p	h	i	r1	r2	CC1	CC2	MSE1	MSE2
10	8	1	412.6	537.1	9615	18340	11.216	1.390
10	16	1	276.1	289.8	6084	11217	11.399	1.738
10	8	3	256	241.6	5855.5	14328.833	13.624	6.702
10	16	3	203.3	176.3	4905.333	8348.666	11.903	5.4498
10	8	5	150.1	106.3	3362.833	7527.166	45.031	115.935
10	16	5	108	73.83	2871	4798.666	76.842	42.1425

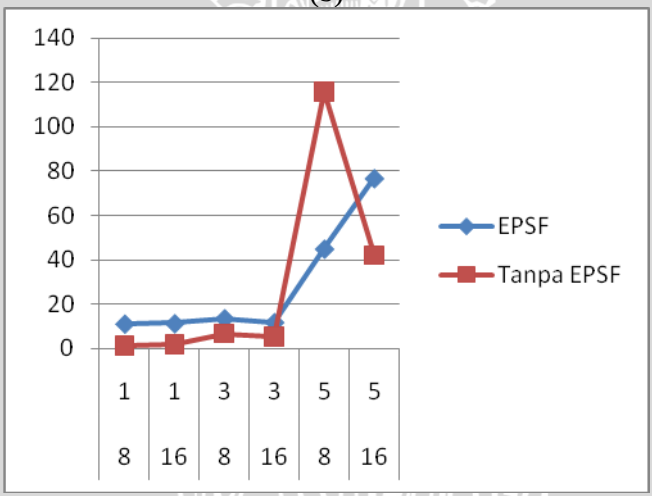
Gambar 4.10 merupakan grafik rata-rata jumlah warna, CC dan error.



(a)



(b)



(c)

Gambar 4.10 (a) Rata-Rata Jumlah Warna (b) Rata-Rata CC (c) Rata-Rata *Error*

Dari hasil rata-rata nilai jumlah warna pada Tabel 4.3 dan gambaran grafik Gambar 4.10, *Connected Component* (CC) dan ukuran kualitas citra menggunakan MSE menunjukkan :

1. Pada grafik (a) jumlah warna mengalami penurunan mengikuti kenaikan nilai i (iterasi) dan h. Dan jumlah warna citra tanpa EPSF memiliki penurunan yang tajam dibandingkan dengan jumlah warna pada reduksi menggunakan EPSF.
2. Padagrafik (b) menggunakan *Connected Component* (CC) secara umum mengalami penurunan jika mengikuti kenaikan nilai i (iterasi) dan h. Dan CC pada reduksi tanpa EPSF memiliki penurunan yang tajam dibandingkan dengan EPSF.
3. Pada grafik (c) memperlihatkan kualitas citra dengan menggunakan nilai MSE secara umum menunjukkan kenaikan nilai h dan i dapat menurunkan kualitas citra karena error semakin tinggi. Dari grafik dapat dibandingkan bahwa kualitas citra pada awal percobaan tanpa EPSF lebih baik dibandingkan EPSF. Karena penurunan yang lebih tajam kualitas tanpa EPSF pada suatu saat akan menemukan titik dimana citra tanpa EPSF lebih buruk dibandingkan EPSF.

Berdasarkan point tersebut menggambarkan bahwa kualitas dari citra yang mengalami reduksi warna semakin menurun jika nilai jumlah warna dan *connected component*(CC) menurun.

4.3.3 Hasil Uji Coba Menggunakan Citra Hasil dan Citra *Biner* Berdasarkan *Threshold* yang Diperoleh dari *Histogram*

Hasil citra D.bmp 6 kali uji coba mengunakan EPSF dan tanpa EPSF pada Tabel 4.4

Tabel 4.4 6 Kali Hasil Uji Coba Citra D.bmp

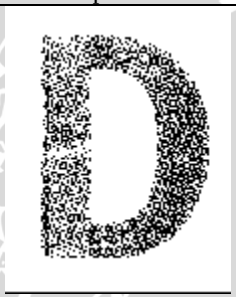
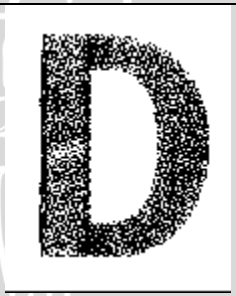
Parameter input	EPSF	Tanpa EPSF
p = 10, h = 8, i = 1		

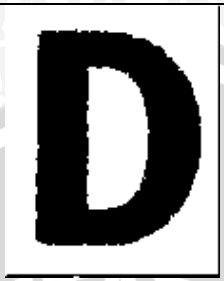
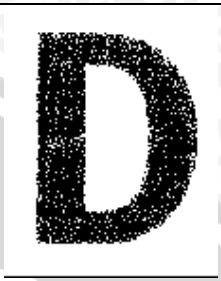
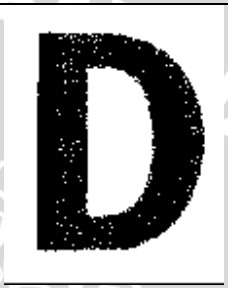
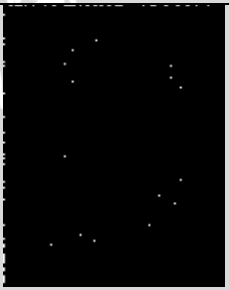
p = 10, h = 16, i = 1		
p = 10, h = 8, i = 3		
p = 10, h = 16, i = 3		
p = 10, h = 8, i = 5		

p = 10, h = 16, i = 5		

Hasil citra biner dari D.bmp pada Tabel 4.5

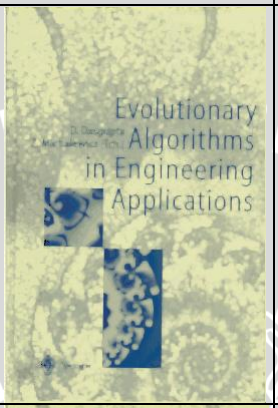
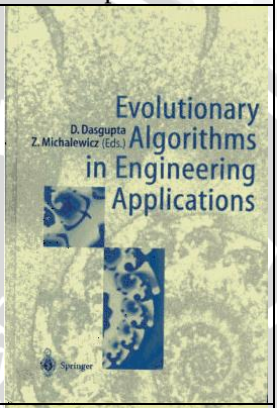
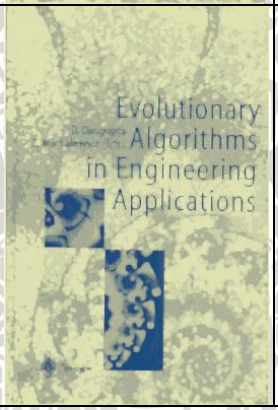
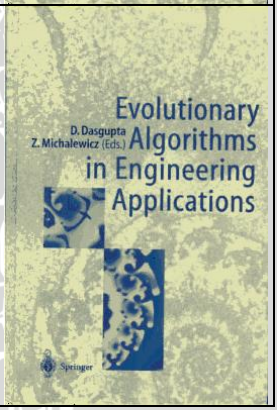
Tabel 4.5 Citra Hasil Uji D.bmp Yang Dibinerkan

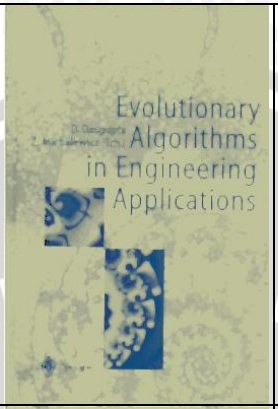
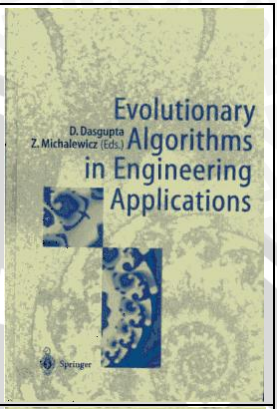
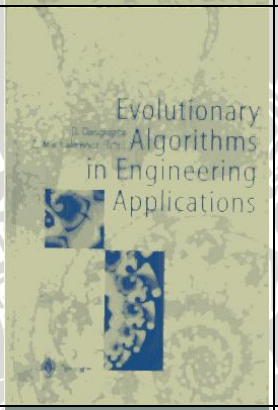
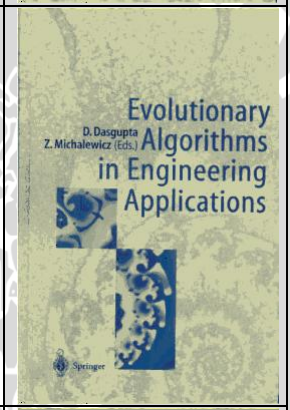
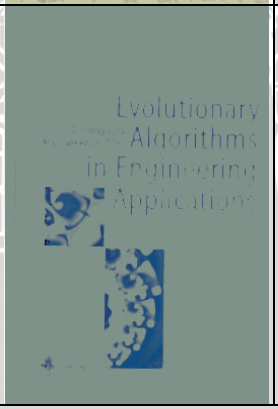
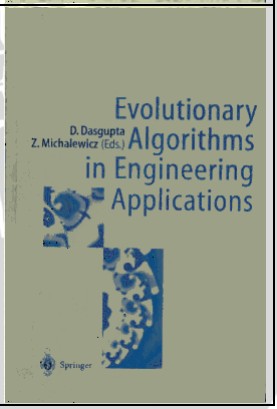
Parameter input	EPSF	Tanpa EPSF
p = 10, h = 8, I = 1	 Threshold = 230	 Threshold = 238
p = 10, h = 16, I = 1	 Threshold = 220	 Threshold = 231

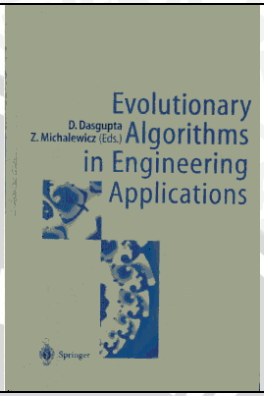
p = 10, h = 8, I = 3	 Threshold = 219	 Threshold = 226
p = 10, h = 16, I = 3	 Threshold = 202	 Threshold = 217
p = 10, h = 8, I = 5	 Threshold = 96	 Threshold = 79
p = 10, h = 16, I = 5	 Threshold = 96	 Threshold = 63

Hasil citra ma.bmp 6 kali uji coba mengunakan EPSF dan tanpa EPSF pada tabel 4.6

Tabel 4.6 6 Kali Hasil Uji Coba Citra Ma.bmp

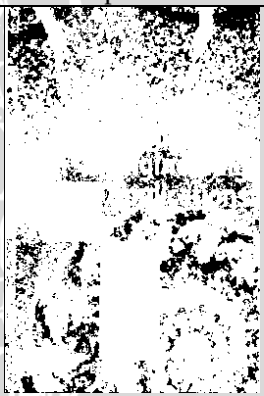
Parameter input	EPSF	Tanpa EPSF
p = 10, h = 8, i = 1		
p = 10, h = 16, i = 1		

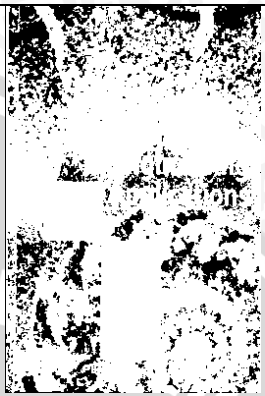
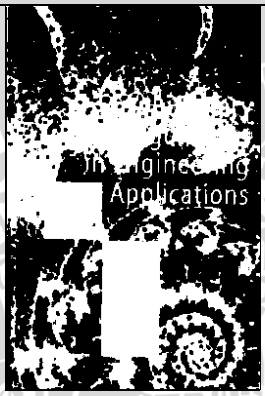
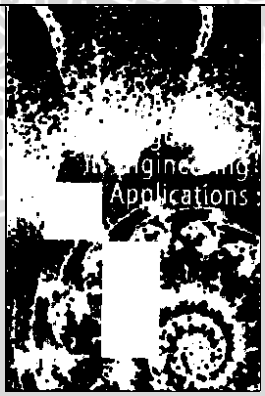
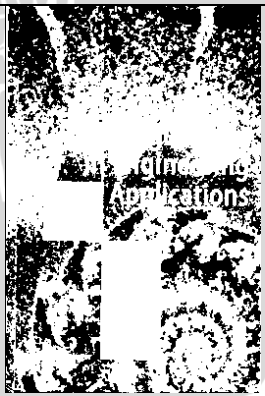
p = 10, h = 8, i = 3		
p = 10, h = 16, i = 3		
p = 10, h = 8, i = 5		

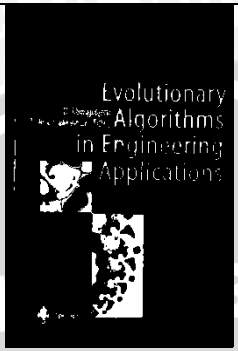
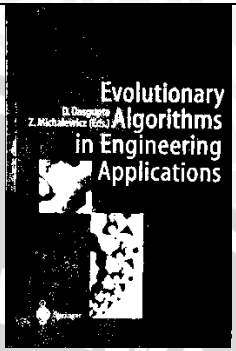
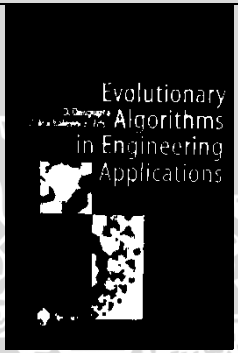
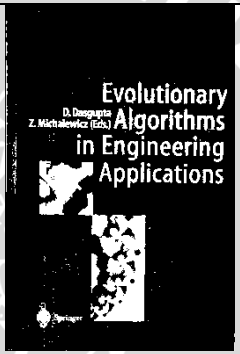
p = 10, h = 16, i = 5		

Hasil citra biner dari ma.bmp pada tabel 4.7

Tabel 4.7 Citra Hasil Uji Ma.bmp Yang Dibinerkan

Parameter input	EPSF	Tanpa EPSF
p = 10, h = 8, i = 1	 Threshold = 224	 Threshold = 227

p = 10, h = 16, i = 1	 Threshold = 205	 Threshold = 224
p = 10, h = 8, i = 3	 Threshold = 192	 Threshold = 212
p = 10, h = 16, i = 3	 Threshold = 198	 Threshold = 204

p = 10, h = 8, i = 5	 Threshold = 126	 Threshold = 156
p = 10, h = 16, i = 5	 Threshold = 145	 Threshold = 157

Untuk mendapatkan citra *biner* dari hasil citra reduksi warna yaitu mencari nilai maksimal dari histogram warna merah dan kemudian menggunakannya sebagai *threshold* untuk diterapkan dalam citra *biner*.

4.3.4 Hasil Uji Coba Berdasarkan Visualisasi Setiap Orang Terhadap Hasil Dari Citra

Pada uji coba ini diberikan kuisoner kepada 20 orang pengamat terhadap hasil citra yang telah diuji pada tabel 4.8 hal ini ditujukan untuk memperoleh kualitas citra yang baik dan buruk jika didasarkan pada visualisasi pengamat citra. Dan ini digunakan sebagai pembanding dari hasil yang diperoleh dari hasil uji coba dengan melihat parameter hasil yaitu jumlah warna, *connected component* (CC) dan MSE. Tabel 4.8 menunjukkan jumlah pengamat yang menganggap citra yang dipilih adalah baik.

Tabel 4.8 Jumlah Pengamat Menganggap Citra Yang Dipilih Baik

Citra	Parameter Input			Jumlah Pengamat	
	p	h	i (iterasi)	EPSF	Tanpa EPSF
D	10	8	1	16	4
D	10	16	1	17	3
D	10	8	3	16	4
D	10	16	3	16	4
D	10	8	5	15	5
D	10	16	5	0	20
Asli	10	8	1	7	13
Asli	10	16	1	8	12
Asli	10	8	3	9	11
Asli	10	16	3	9	11
Asli	10	8	5	13	7
Asli	10	16	5	14	6
Ma	10	8	1	2	18
Ma	10	16	1	2	18
Ma	10	8	3	2	18
Ma	10	16	3	0	20
Ma	10	8	5	0	20
Ma	10	16	5	2	18
Dead	10	8	1	20	0
Dead	10	16	1	19	1
Dead	10	8	3	20	0
Dead	10	16	3	20	0
Dead	10	8	5	19	1
Dead	10	16	5	19	1
Plus	10	8	1	2	18
Plus	10	16	1	1	19
Plus	10	8	3	1	19
Plus	10	16	3	12	8
Plus	10	8	5	19	1
Plus	10	16	5	1	19
Struct	10	8	1	1	19
Struct	10	16	1	1	19
Struct	10	8	3	0	20
Struct	10	16	3	1	19
Struct	10	8	5	3	17
Struct	10	16	5	3	17
Jumlah				310	410

Menghitung persentase perbandingan kualitas warna jika dilihat dari visualisasi orang dengan menggunakan persamaan 2.38.

Dapat dilihat dalam tabel 4.8 untuk citra file ”D” dengan parameter h=8 dan i=1 jumlah pengamat lebih banyak menggap citra EPSF lebih baik dibandingkan tanpa EPSF. Sedangkan pada citra file ”Asli” dengan parameter yang sama jumlah pengamat lebih banyak menggap citra tanpa EPSF lebih baik dibandingkan dengan citra EPSF. Dan hal ini dikarenakan citra tanpa EPSF pada awal-awal percobaan memang memiliki kualitas yang lebih baik tetapi dengan naiknya nilai h dan i pada suatu saaat citra tanpa EPSF akan menghasilkan citra yang lebih buruk dibandingkan dengan citra EPSF. Hal ini dikarenakan penurunan jumlah warna dan CC pada citra tanpa EPSF relatif ebih cepat dibandingkan dengan citra EPSF.

$$\text{EPSF} = \frac{(\text{Total pengamatan EPSF dari keseluruhan citra})}{(\text{Total pengamatan dari keseluruhan citra})} \text{ x } 100\%$$

$$= \frac{310}{720} \, x \, 100\% = 43\%$$

$$\text{Tanpa EPSF} = \frac{(\text{Total pengamatan t.EPSF dari keseluruhan citra})}{(\text{Total pengamatan dari keseluruhan citra})} \text{x} 100\%$$

$$= \frac{410}{720} \, x \, 100\% = 57\%$$

Berdasarkan hasil kuisoner dan perolehan persentase hasil pengamatan citra dari 20 orang pengamat sebanyak 43% menganggap citra hasil EPSF lebih baik dibandingkan dengan tanpa EPSF dan 57% menganggap hasil citra tanpa EPSF lebih baik dibandingkan dengan EPSF.

4.3.5 Analisa Hasil

Dari beberapa uji coba yang dihasilkan sebelumnya dicoba dilakukan analisa terhadap hasil yang diperoleh :

1. Pada tabel 4.3 menunjukkan jumlah warna mengalami penurunan jika parameter i dan h naik. Jumlah warna mengalami penurunan seperti yang dijelaskan pada *procedure* subbab 4.2. Dimana pusat cluster terakhir yang melibatkan parameter i dan h mempengaruhi hasil *error*. Dan *error lebih besar 0* dihasilkan karena warna yang dihasilkan berbeda dengan warna aslinya. Lebih jauh dijelaskan pada analisa berikutnya.
2. *Connected component* pada tabel 4.3 juga menunjukkan jika h dan i naik. Semakin sedikit pusat *cluster* yang dihasilkan maka CC yang terdapat pada citra hasil semakin sedikit. Hal ini terjadi karena CC menghubungkan tiap *pixel* pada citra yang terhubung dengan nilai *pixel* yang sama. Jika melalui beberapa proses subbab 4.2 dihasilkan jumlah warna yang sedikit maka warna citra hasil akan mengikuti warna yang menjadi pusat *cluster*. Semakin sedikit pusat cluster memungkinan *pixel* tiap tetangga pada citra memiliki keterkaitan hubungan sangat besar. Dan inilah yang menjadikan CC menurun. Keterkaitan CC dengan reduksi warna yaitu dokumen masih memiliki kualitas yang baik jika memiliki nilai CC yang tinggi walaupun hanya memiliki jumlah warna yang relatif sedikit. Untuk itu jumlah warna belum dapat digunakan sebagai parameter dasar untuk menguji kualitas citra dari reduksi warna ini. Oleh karena itu, CC inilah yang digunakan untuk membandingkan hasil citra menggunakan EPSF dan tanpa EPSF. Dan dari hasil rata-rata pada penurunan CC pada EPSF dibanding tanpa EPSF adalah

$$\frac{EPSF}{Tanpa\ EPSF} = \frac{|parameter(h{=}16,i{=}5) - parameter(h{=}16,i{=}3)|}{|parameter(h{=}16,i{=}5) - parameter(h{=}16,i{=}3)|}$$

$$= \frac{|2871-4905|}{|4798-8348|} = \frac{2034}{3550} = \frac{1}{1{,}7} = \frac{1}{2}$$

$$\frac{EPSF}{Tanpa\ EPSF} = \frac{|parameter(h=16,i=5) - parameter(h=8,i=5)|}{|parameter(h=16,i=5) - parameter(h=8,i=5)|}$$

$$= \frac{|2871-3362|}{|4798-7527|} = \frac{491}{2729} = \frac{1}{5}$$

Berdasarkan perbandingan tersebut membuktikan bahwa penurunan CC pada teknik EPSF lebih sedikit dibandingkan dengan tanpa EPSF.

3. Parameter h dan i akan mempengaruhi jumlah warna, sedangkan jumlah warna digunakan dalam penentuan jumlah pusat *cluster*. Jika jumlah warna semakin sedikit *error* semakin besar. Karena penentuan MSE merupakan kuadrat dari pengurangan nilai pixel citra asli dengan pixel citra hasil. Jika tingkat kesamaan berbeda jauh maka dapat dipastikan *error* besar. Dan pada rata-rata tabel 4.3 nilai MSE untuk EPSF lebih tinggi dibandingkan dengan *error* yang dihasilkan oleh citra tanpa EPSF. Ini menunjukkan citra EPSF lebih baik untuk digunakan dalam pengolahan reduksi warna.
4. Dapat dilihat pada hasil citra struct.bmp pada tabel 4.2 untuk jumlah warna yang dihasilkan pada tahap EPSF yaitu 1 warna sedangkan *error* yang dihasilkan 302,829. Dan error tersebut termasuk yang paling besar dibandingkan error yang lain. Seperti yang dibahas pada subbab 4.2.5 tentang *mean shift procedure* yaitu mengenai permasalahan yang ditimbulkan pada tahap *mean shift*. *Error* terjadi bisa diakibatkan sebaran warna yang sangat dekat antara tiap *pixel*. Sehingga input baru yang diperoleh memiliki nilai akumulasi yang relatif sama. Sehinggga menyebabkan tiap *pixel* mengarah pada *cluster* yang sama. Ada beberapa kemungkinan menurut analisa peneliti yang mempengaruhi
   a. Tinggi area dalam penentuan *mean shift* sehingga area yang diambil relatif luas sehingga memungkinkan *cluster* yang didapatkan lebih sedikit.
   b. *Pixel* yang terletak pada area yang ditentukan memiliki kedekatan yang sangat tinggi sehingga mempengaruhi nilai akumulasi dari nilai input baru. Karena kepadatan dari nilai *pixel* yang hampir sama berkumpul pada di satu area.

5. Dari uji coba pada 4.3.2 tetang perlakuan citra hasil dengan mengubah ke citra *biner* terlihat bahwa citra D.bmp tanpa EPSF masih terdapat *noise* yag lebih jelas dibandingkan dengan citra dengan EPSF. Sehingga untuk dilakukan segmentasi akan lebih baik menggunakan EPSF.
6. Pada uji coba hasil pada subbab 4.3.3 menunjukan pengujian citra hasil melalui visualisasi manusia dari 20 orang pengamat 43% menganggap citra EPSF lebih baik sedangkan 57% manganggap citra tanpa EPSF lebih baik. Berdasarkan data ini menunjukkan bahwa hasil reduksi warna tanpa EPSF masih menghasilkan kualitas baik dari sisi visualisasi manusia.
7. Berdasarkan analisa no 1-6 menunjukkan kualitas citra hasil tanpa EPSF lebih baik daripada EPSF. Namun berdasarkan analisa no 5 bahwa citra dengan kualitas baik belum menjamin dalam mempermudah segmentasi karena masih terdapat *noise* yang menggangu sehingga pada dokumen citra digital ini tidak bisa dibedakan antara objek dan background. Dan pada citra tanpa EPSF pada suatu saat karena mengalami penurunan jumlah warna dan CC yang lebih tajam memungkinkan citra EPSF lebih baik dibandingkan citra tanpa EPSF dikarenakan penurunan yang relatif stabil untuk jumlah warna dan CC.

UNIVERSITAS BRAWIJAYA

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan dari penulisan ini menyatakan bahwa

a. Telah diterapkan teknik EPSF untuk melakukan reduksi warna pada dokumen digital menggunakan deteksi tepi *Sobel* dan *Mean Shift Procedure* yang diimplementasikan ke dalam sebuah program komputer.
b. Kualitas citra dari reduksi warna semakin menurun jika parameter h dan i naik. Pengukuran penurunan kualitas citra didasarkan pada *error* yang tinggi jika nilai h dan i mengalami kenaikan. Berlaku juga untuk parameter yang dihasilkan yaitu jumlah warna dan *connected component* yang juga mengalami penurunan.

5.2 Saran

Pada pengujian berdasarkan visualisasi manusia jumlah responden menganggap citra hasil EPSF memiliki kualitas sangat buruk dengan adanya jumlah responden kurang dari satu untuk beberapa citra pengujian. Oleh karena itu disarankan melakukan reduksi warna citra menggunakan metode yang lebih baik agar hasil penyederhanaan antara objek dan *background* lebih jelas.

UNIVERSITAS BRAWIJAYA

DAFTAR PUSTAKA

Bungin, Burhan. 2005. *Metode Penelitian Kuantitatif*. Jakarta: Kencana

Comanixiu, Dorin. 2002. *Mean Shift A Robust Approach Toward Feature Space Analysis*. IEEE

Fadova, D., Le Bourganis, F., dan Emproz, H. 2006. *A Modified Mean Shift Algorithm For Efficienci Document Image Destoration*. INSA de Lyon. French

Fisher, R., Perkins S., Walker A., Wolfart E. 2003. (online). (*http://homepages.inf.ed.ac.uk/r&f/HIPR2/conduce.htm, diakses tanggal* 22 Agustus 2010)

Garnica, C., Boochs F., Twardochlib M. 2000. *A New Approach to Edge-Preserving Smoothing For Edge Extraction and Image Segmentation*. Institute for Spatial Information and Surveying Technology, University of Applied Science Holtztrasse. Germany.

Jebara, T. 2000. (online). (*http://www.cs.coloumbia.edu/~jebara/ htmlpapers/ UTHELS/node15.html, diakses tanggal 22 Juli 2010*)

Jue Wang, Yingqing Xiu, Heung-Yung Shum dan Michael Chon. *Video Tooning*. University of Washington and Microsoft Research. America

Nikolou, N., Papamarkos, N. 2008. *Color reduction for Complex Document Image*. Democritus University of Thrace. Yunani

Nugroho, S. 2005. *Implementasi Metode Edge Linking untuk Mendeteksi Garis Tepi Pada Citra Digitali*. Balikpapan : STIKOM Balikpapan

Sonka, M., Hlavac V., Boyle . 1994. *Image Processing, Analysis and Machine Vision*. London, Glasgow, Weinhem, new york, Tokyo, Melbourne, Madras : Chapman & Hall Computing

Taylor, Walter F. 1991. *The Geometry Of Computer Graphics*. Pacific Grove : Wadsworth & Books/Cole Advanced Books & Software.

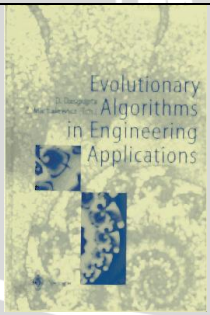
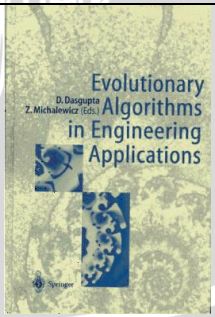
Yamaouchi, H., Lee, S., Lee, Y., Ohtake, Y., Belyaev, A., Seidel, H. *Feature Sensitive with Meanshift.* MPI informatik, POSTECH, RIKEN. Korea

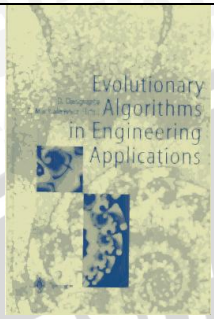
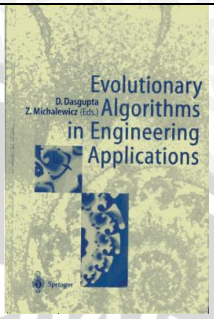
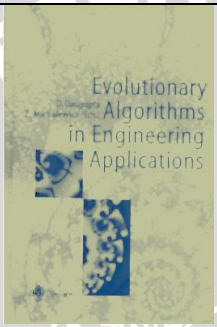
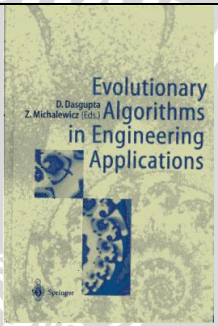
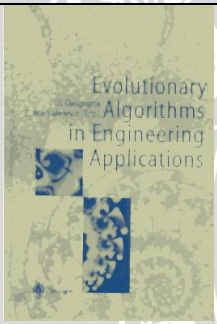
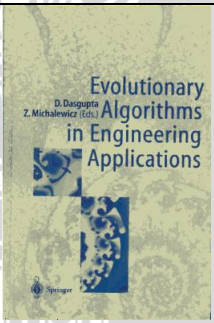
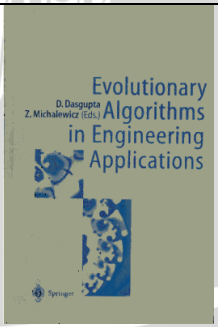
LAMPIRAN

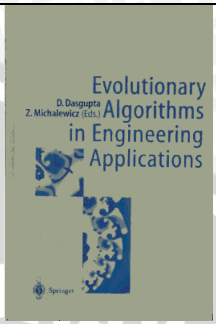
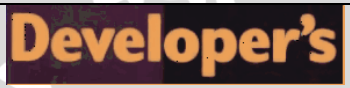
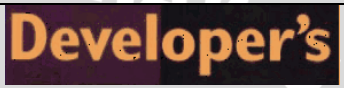
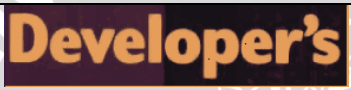
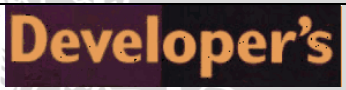
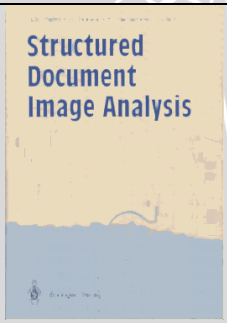
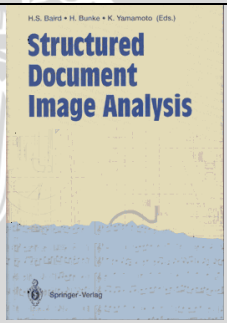
Kuisoner untuk penilaian citra pengamat terhadap citra dengan membandingkan 2 citra hasil penerapan EPSF (Gambar 1) dan tanpa EPSF (Gambar 2)

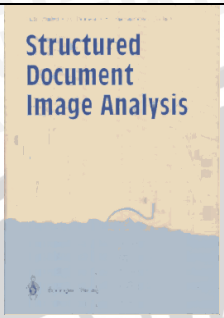
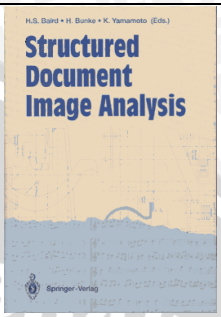
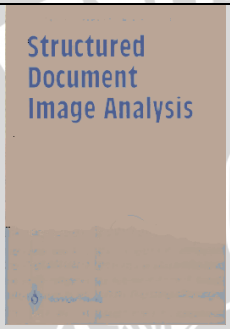
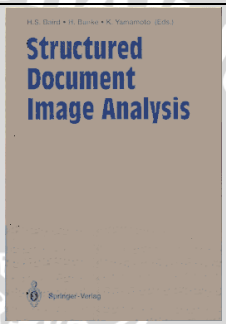
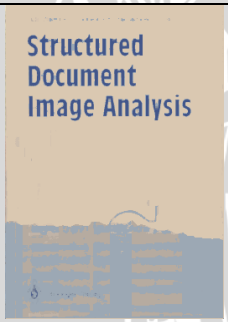
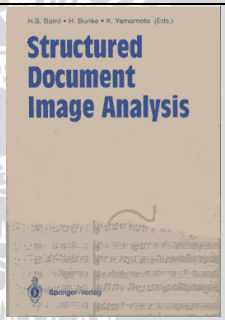
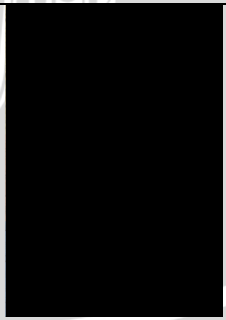
Tabel Daftar Gambar

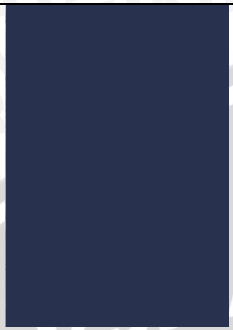
No	Gambar 1	Gambar 2
1		
2		
3		
4		

5		
6		
7		
8		
9		
10		
11		
12		
13		

14	 Evolutionary Algorithms in Engineering Applications	 Evolutionary D. Dasgupta Z. Michalewicz (Eds.) Algorithms in Engineering Applications
15	 Evolutionary Algorithms in Engineering Applications	 Evolutionary D. Dasgupta Z. Michalewicz (Eds.) Algorithms in Engineering Applications
16	 Evolutionary Algorithms in Engineering Applications	 Evolutionary D. Dasgupta Z. Michalewicz (Eds.) Algorithms in Engineering Applications
17	 Evolutionary Algorithms in Engineering Applications	 Evolutionary D. Dasgupta Z. Michalewicz (Eds.) Algorithms in Engineering Applications

18		
19		
20		
21		
22		
23		
24		
25		

26	 Structured Document Image Analysis	 H.S. Baird • H. Bunke • K. Yamamoto (Eds.) Structured Document Image Analysis Springer-Verlag
27	 Structured Document Image Analysis	 H.S. Baird • H. Bunke • K. Yamamoto (Eds.) Structured Document Image Analysis Springer-Verlag
28	 Structured Document Image Analysis	 H.S. Baird • H. Bunke • K. Yamamoto (Eds.) Structured Document Image Analysis Springer-Verlag
29	 Structured Document Image Analysis	

30		 Structured Document Image Analysis
31	 PLUS: • Monitors, Displays and Projectors • Robot Vision • Cost-Justifying Imaging	 PLUS: • Monitors, Displays and Projectors • Robot Vision • Cost-Justifying Imaging
32	 PLUS: • Monitors, Displays and Projectors • Robot Vision • Cost-Justifying Imaging	 PLUS: • Monitors, Displays and Projectors • Robot Vision • Cost-Justifying Imaging
33	 PLUS: • Monitors, Displays and Projectors • Robot Vision • Cost-Justifying Imaging	 PLUS: • Monitors, Displays and Projectors • Robot Vision • Cost-Justifying Imaging
34	 PLUS: • Monitors, Displays and Projectors • Robot Vision • Cost-Justifying Imaging	 PLUS: • Monitors, Displays and Projectors • Robot Vision • Cost-Justifying Imaging

35	 PLUS: • Monitors, Displays and Projectors • Robot Vision • Cost-Justifying Imaging	 PLUS: • Monitors, Displays and Projectors • Robot Vision • Cost-Justifying Imaging
36	 PLUS: • Monitors, Displays and Projectors • Robot Vision • Cost-Justifying Imaging	 PLUS: • Monitors, Displays and Projectors • Robot Vision • Cost-Justifying Imaging

Berikan tanda (V) untuk gambar yang anda anggap memiliki kualitas bagus. Untuk perbandingan 2 gambar tersebut.

Tabel Form Penilaian

No	Gambar 1	Gambar 2
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
30		
31		
32		
33		
34		
35		
36		

UNIVERSITAS BRAWIJAYA