

**PENGGUNAAN METODE TEXTURE  
BASED DENGAN SUPPORT VECTOR  
MACHINE DALAM MENDETEKSI AREA  
TEKS PADA GAMBAR**

**TUGAS AKHIR**

Diajukan sebagai syarat untuk menyelesaikan Program Studi Strata-1

Departemen Teknik Informatika

**Disusun oleh:**

**Kevin Hertanto 1111018**



**INSTITUT  
TEKNOLOGI  
HARAPAN  
BANGSA**

*School of Telematics*

**DEPARTEMEN TEKNIK INFORMATIKA  
INSTITUT TEKNOLOGI HARAPAN BANGSA  
2015**



**LEMBAR PENGESAHAN**

**PENGGUNAAN METODE TEXTURE BASED DENGAN  
SUPPORT VECTOR MACHINE DALAM MENDETEKSI  
AREA TEKS PADA GAMBAR**



Disusun Oleh:

Disusun Oleh :

Nama : Kevin Hertanto

NIM : 1111018

Telah disetujui dan disahkan sebagai laporan Tugas Akhir Program studi strata – 1

Departemen Teknik Informatika

INSTITUT TEKNOLOGI HARAPAN BANGSA

Bandung, Agustus 2015

Disetujui,  
Pembimbing I

Disetujui,  
Pembimbing II

(Elisafina Siswanto, S.T., M.T.)  
NIK.111033

(Ria Chaniago, S.T.)  
NIK.1114001



## **PERNYATAAN HASIL KARYA PRIBADI**

Saya yang bertanda tangan di bawah ini :

Nama : Kevin Hertanto

NIM : 1111018

Dengan ini menyatakan bahwa laporan Tugas Akhir dengan Judul : **"PENGGUNAAN METODE TEXTURE BASED DENGAN SUPPORT VECTOR MACHINE DALAM MENDETEKSI AREA TEKS PADA GAMBAR"** adalah hasil pekerjaan saya dan seluruh ide, pendapat atau materi dari sumber lain telah dikutip dengan cara penulisan referensi yang sesuai.

Pernyataan ini saya buat dengan sebenar-benarnya dan jika pernyataan ini tidak sesuai dengan kenyataan maka saya bersedia menanggung sanksi yang akan dikenakan pada saya.

Bandung, Juli 2015

Yang membuat pernyataan,

Kevin Hertanto

## ABSTRAK

Mendeteksi teks pada gambar merupakan teknologi yang akan sangat berguna bagi masa depan, misalnya: teknologi untuk robot, robot dapat mendeteksi letak teks pada gambar yang diambil, lalu aplikasi dapat mengembangkan hasil output lebih lanjut, seperti mengetahui isi teks tersebut, menerjemahkan teks tersebut ke dalam berbagai macam bahasa, dan sebagainya. *Texture-Based* salah satu teknik untuk mendeteksi gambar dengan cara membagi gambar menjadi bagian-bagian kecil dengan ukuran dan jumlah tertentu dengan tujuan untuk mencari bagian gambar yang terdapat teks. *Support Vector Machine* merupakan algoritma dengan metode *Supervised learning*, dimana program diberi data *training* sebagai bahan pembelajaran yang berfungsi menganalisa dan mengenali pola, dan akan digunakan untuk analisis klasifikasi dan regresi. *Support Vector Machine*(SVM) digunakan untuk mencari gambar yang mengandung teks dan tidak. *Feature Extraction Gray Level Co-occurrences Matrices* (GLCM) digunakan untuk membuat sebuah model diantaranya: kontras, homogenitas, energi, entropi, dan korelasi. Berdasarkan hasil pengujian yang telah dilakukan, *f-measure* rata-rata yang didapatkan sebesar 75,99%, dengan *f-measure* teks sebesar 67,97%, dan *f-measure* non-teks sebesar 84,01%.

Kata Kunci: *Texture-Based*, *Support Vector Machine* (SVM), *Gray Level Co-occurrences Matrices* (GLCM).

## **ABSTRACT**

Detect text in an image is a technology that will be very useful for the future, for example: technology for robot, the robot can detect the location of the text on the picture taken, then the application can develop further output results, such as knowing the content of the text, translate the text into a wide variety of languages, and so on. Texture-Based is one technique for detecting image by dividing the image into small parts with a size and a certain amount for the purpose of searching for images that are part of the text. Support Vector Machine is an algorithm with *Supervised learning* method, where the program given the training data as a learning material that serves to analyze and recognize patterns, and will be used for classification and regression analysis. Support Vector Machine (SVM) is used to search for text that contains images and not. Feature Extraction Gray Level Co-occurrences Matrices (GLCM) is used to create a model including: contrast, homogeneity, energy, entropy, and correlation. Based on the results of tests that have been carried out, the average f-measure is obtained by 75.99%, with 67.97% f-measure of the text and 84.01% f-measure of non-text.

Keywords: *Texture-Based*, *Support Vector Machine* (SVM), *Gray Level Co-occurrences Matrices* (GLCM).

## **PEDOMAN PENGGUNAAN LAPORAN TUGAS AKHIR**

Laporan tugas akhir yang tidak dipublikasikan terdaftar dan tersedia di perpustakaan Institut Teknologi Harapan Bangsa, dan terbuka untuk umum dengan ketentuan bahwa hak cipta ada pada pengarang dan pembimbing tugas akhir. Referensi kepustakaan diperkenankan dicatat, tetapi pengutipan atau peringkasan hanya dapat dilakukan seizin pengarang dan pembimbing tugas akhir disesuaikan dengan kebiasaan untuk menyebutkan sumbernya.

Tidak diperkenankan untuk memperbanyak atau menerbitkan sebagian atau seluruh laporan tugas akhir tanpa seizin dari pengarang dan pembimbing tugas akhir yang bersangkutan.

## KATA PENGANTAR

Terima kasih kepada Tuhan yang Maha Esa karena dengan bimbingan-Nya dan karunia-Nya penulis dapat melaksanakan Tugas Akhir yang berjudul “PENGGUNAAN METODE TEXTURE BASED DENGAN SUPPORT VECTOR MACHINE DALAM MENDETEKSI AREA TEKS PADA GAMBAR”. Laporan ini disusun sebagai salah satu syarat kelulusan di Institut Teknologi Harapan Bangsa. Pada kesempatan ini penulis menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Ibu Elisafina Siswanto, S.T., M.T., selaku pembimbing I tugas akhir yang senantiasa memberi saran dan dukungan kepada penulis selama tugas akhir berlangsung dan selama pembuatan laporan tugas akhir ini.
2. Ibu Ria Chaniago, S.T. , selaku pembimbing II tugas akhir yang senantiasa memberi saran dan dukungan kepada penulis selama tugas akhir berlangsung dan selama pembuatan laporan tugas akhir ini.
3. Bapak Victor Libtuselah Brilliam Manu, S.T., selaku penguji I dalam Tugas Akhir Terima kasih atas dukungan, semangat, ilmu-ilmu, dan masukan yang telah diberikan kepada penulis dalam menyelesaikan Laporan Tugas Akhir ini.
4. Ibu Inge Martina, M.T. , selaku penguji II dalam Tugas Akhir Terima kasih atas dukungan, semangat, ilmu-ilmu, dan masukan yang telah diberikan kepada penulis dalam menyelesaikan Laporan Tugas Akhir ini.
5. Ibu Ken Ratri Retno W, S.Kom., M.T., selaku koordinator kerja praktek yang sudah menerima dan menyetujui kerja praktek yang diajukan penulis sehingga penulis dapat memulai kerja praktek.
6. Seluruh dosen dan staff Departemen Teknik Informatika ITHB yang telah membantu dalam menyelesaikan Laporan Tugas Akhir ini.
7. Segenap jajaran staf dan karyawan ITHB yang turut membantu kelancaran dalam menyelesaikan Laporan Tugas Akhir ini.
8. Kedua orang tua tercinta yang selalu menyediakan waktu untuk memberikan doa, semangat dan dukungan yang tak habis-habisnya kepada penulis untuk

- menyelesaikan Laporan Tugas Akhir ini. Terima kasih untuk nasihat, masukan, perhatian, teguran dan kasih sayang yang diberikan hingga saat ini.
9. Keluarga besar penulis yang selalu memberikan dukungan, masukan, semangat dan doa kepada penulis sehingga penulis dapat menyelesaikan tugas akhir ini dengan baik.
  10. Teman-teman seperjuangan dari Teknik Informatika ITHB angkatan 2011, terima kasih untuk perhatian, doa, teguran, ejekan dan semangat yang selalu dilontarkan setiap saat. Terima kasih untuk waktu dan pengalaman berharga bersama kalian, empat tahun ini adalah masa yang berharga. Sukses untuk kalian semua.
  11. Semua pihak yang telah membantu dan memberikan semangat dan doanya bagi penulis yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna karena keterbatasan waktu dan pengetahuan yang dimiliki oleh penulis. Oleh karena itu, kritik dan saran untuk membangun kesempurnaan tugas akhir ini sangat diharapkan. Semoga tugas akhir ini dapat membantu pihak-pihak yang membutuhkannya.

Bandung, Juli 2015

Hormat Saya,

Penulis

## DAFTAR ISI

LEMBAR PENGESAHAN .....	i
PERNYATAAN HASIL KARYA PRIBADI .....	ii
ABSTRAK .....	iii
ABSTRACT .....	iv
PEDOMAN PENGGUNAAN LAPORAN TUGAS AKHIR .....	v
KATA PENGANTAR .....	vi
DAFTAR ISI.....	viii
DAFTAR GAMBAR .....	xi
DAFTAR TABEL.....	xiii
BAB I PENDAHULUAN .....	1
1.1    Latar Belakang Masalah .....	1
1.2    Ruang Lingkup Masalah .....	3
1.3    Tujuan Penelitian .....	3
1.4    Batasan Masalah .....	3
1.5    Kontribusi Penelitian.....	4
1.6    Sistematika Penulisan .....	4
BAB II LANDASAN TEORI .....	6
2.1    Pengolahan Citra Digital.....	6
2.1.1    Citra Biner .....	8
2.1.2    Citra <i>Grayscale</i> .....	8
2.2    Transformasi Citra .....	9
2.2.1    Transformasi Kosinus Diskrit( <i>Discrete Cosine Transform</i> )....	10
2.3    Pengenalan Pola.....	12
2.4    Operasi Spasial (Filtering) .....	13
2.4.1    Tapis High-pass .....	14
2.5    GLCM (Gray Level Co-occurrence Matrices).....	14

2.6	Support Vector Machine(SVM).....	18
2.7	Java Matrix (JAMA).....	22
2.8	Image Plus.....	23
2.9	Segmentasi Citra .....	23
2.9.1	Pengambangan ( <i>Thresholding</i> ).....	24
2.9.2	Threshold Otsu.....	24
2.10	F-Measure.....	25
2.11	Pesamaan Linear .....	25
2.12	Tinjauan Studi .....	28
<b>BAB III ANALISIS MASALAH DAN IMPLEMENTASI .....</b>		<b>30</b>
3.1	Analisis Masalah.....	30
3.2	Kerangka Kerja .....	33
3.3	Analisis Cara Kerja Sistem.....	34
3.3.1	Data.....	35
3.3.2	Image Processing.....	36
3.3.3	Feature Texture-based extraction .....	43
3.3.4	Support Vector Machine.....	51
3.3.5	Post-processing .....	56
<b>BAB IV IMPLEMENTASI DAN PENGUJIAN .....</b>		<b>59</b>
4.1	Lingkungan Implementasi .....	59
4.2	Implementasi Perangkat Lunak .....	59
4.2.1	Implementasi Image Processing .....	65
4.2.2	Implementasi Feature Extraction .....	66
4.2.3	Implementasi Support Vector Machine .....	67
4.3	Cara Penggunaan Aplikasi.....	68
4.4	Pengujian .....	71
4.4.1	Pengujian untuk Menentukan arah GLCM.....	72
4.4.2	Pengujian untuk Konstanta Sigma pada Kernel RBF.....	75

4.4.3	Pengujian pengaruh Image Processing pada Hasil Akurasi .....	76
4.2.5	Pengujian untuk Ukuran Blok pada Feature Extration .....	77
4.2.6	Pengujian Post Processing .....	79
4.2.7	Pengujian Jumlah Data Training .....	81
4.4.8	Pengujian Movie Poster .....	84
BAB V	KESIMPULAN DAN SARAN .....	87
5.1	Kesimpulan .....	87
5.2	Saran .....	88
DAFTAR PUSTAKA .....		90
LAMPIRAN A.....		A-1
LAMPIRAN B .....		B-1

## DAFTAR GAMBAR

Gambar 2.1 Koordinat Citra Digital .....	7
Gambar 2.2 Citra Digital dalam Bentuk Matrik. ....	7
Gambar 2.3 Ilustrasi Digitalisasi Citra.....	7
Gambar 2.4 Citra Biner .....	8
Gambar 2.5 Citra Grayscale.....	9
Gambar 2.6 Contoh pembagian pada gambar menjadi beberapa blok .....	12
Gambar 2.7 Citra Karakter (Fokunaga, 1990) .....	12
Gambar 2.8 Struktur Sistem Pengenalan Pola .....	13
Gambar 2.9 Arah dan Jarak GLCM.....	15
Gambar 2.10 Berbagai Garis hyperplan (discrimination boundaries). .....	18
Gambar 2.11 Rumus untuk Hyperplan. .....	18
Gambar 2.12 Rumus untuk Hyperplan. .....	19
Gambar 2.13 SVM non-linear.....	20
Gambar 2.14 Contoh Implemntasi library JAMA .....	23
Gambar 2.15 Contoh Implementasi library Image Plus untuk Grayscale .....	23
Gambar 2.16 Visualisasi Precesion dan Recall.....	25
Gambar 2.17 Bentuk Umum .....	26
Gambar 2.18 Rumus perhitungan persamaan linear 2x2 .....	26
Gambar 2.19 Contoh penyelesaian persamaan linear 2x2 .....	27
Gambar 2.20 Rumus perhitungan persamaan linear 3x3 .....	27
Gambar 2.21 Contoh penyelesaian persamaan linear 3x3 .....	28
Gambar 2.22 Pre-processing menggunakan metode Texture-Based .....	30
Gambar 3.1 Image with texture based feature .....	30
Gambar 3.2 Contoh text vectical atau memiliki angle.....	32
Gambar 3.3 Kerangka kerja aplikasi mendeteksi teks pada gambar.....	33
Gambar 3.4 Flow chart aplikasi deteksi teks pada gambar.....	34
Gambar 3.5 Contoh data training.....	36
Gambar 3.6 Flow chart proses image processing .....	36
Gambar 3.7 Gambar yang akan di grayscale .....	37

Gambar 3.8 Gambar grayscale .....	38
Gambar 3.9 Matriks koefisien DCT .....	39
Gambar 3.10 Kernel high pass filter [ANG10] .....	40
Gambar 3.11 Hasil perkalian Kernel high pass filter .....	41
Gambar 3.12 Matriks invers DCT .....	41
Gambar 3.13 Hasil dari proses normalisasi .....	43
Gambar 3.14 Flow chart proses Texture-based extract .....	43
Gambar 3.15 Matrik Kemunculan pola .....	46
Gambar 3.16 Matriks Normalisasi .....	46
Gambar 3.17 Matriks Transpose .....	46
Gambar 3.18 Flow chart proses Support Vector Machine .....	51
Gambar 3.19 Contoh Sistem Voting .....	56
Gambar 3.20 Flow chart proses Post-Processing .....	56
Gambar 3.21 Hasil gambar yang mencurigakan .....	57
Gambar 3.22 Proses pengulangan dengan syarat blok yang lebih kecil .....	57
Gambar 3.23 Hasil akhir .....	58
Gambar 4.1 Tampilan aplikasi bagian membuat model .....	69
Gambar 4.2 Contoh gambar yang sudah disave setiap blok .....	70
Gambar 4.3 Contoh hasil aplikasi .....	70

## DAFTAR TABEL

Tabel 3.1 Data training .....	52
Tabel 3.2 Hasil kernel RBF.....	53
Tabel 3.3 Hasil alpha dan bias .....	54
Tabel 3.4 Contoh Data .....	54
Tabel 3.5 Hasil kernel RBF.....	55
Tabel 4.1 Objek Classification.....	60
Tabel 4.2 Daftar Method Image Processing.....	60
Tabel 4.3 Daftar Method Feature Extraction .....	62
Tabel 4.4 Daftar Method Support Vector Machine .....	63
Tabel 4.5 Daftar Method Classification Data Training.....	64
Tabel 4.6 Hasil pengujian arah 135 derajat blok teks.....	73
Tabel 4.7 Tabel hasil pengujian arah 90 derajat .....	73
Tabel 4.8 Hasil pegujian arah 45 derajat.....	74
Tabel 4.9 Hasil pengujian menggunakan Ukuran Blok 60x60 .....	77
Tabel 4.10 Hasil pengujian menggunakan Ukuran Blok 70x70 .....	78
Tabel 4.11 Hasil pengujian dengan 3 model tanpa post-processing .....	80
Tabel 4.12 Hasil pengujian 3 model dengan menambah attribute pada SVM.....	81
Tabel 4.13 Hasil pengujian dengan 9 model tanpa post-processing .....	82
Tabel 4.14 Hasil pengujian 9 model dengan post-processing.....	83
Tabel 4.15 Hasil pengujian dengan data training Movie Poster .....	84

# BAB I

## PENDAHULUAN

Bab ini merupakan bab yang berisi latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, kontribusi penelitian, metodologi penelitian dan sistematika penulisan.

### 1.1 Latar Belakang Masalah

Mendeteksi teks pada gambar merupakan teknologi yang akan sangat berguna bagi masa depan. Dengan adanya teknologi ini, berbagai media dapat mengembangkan lebih lanjut dengan berbagai kegunaan, misalnya: teknologi untuk robot, robot dapat mendeteksi letak teks pada gambar yang diambil, lalu aplikasi dapat mengembangkan hasil output lebih lanjut, seperti mengetahui isi teks tersebut, menerjemahkan teks tersebut ke dalam berbagai macam bahasa, dan sebagainya.

Dengan adanya permasalahan seperti ini, penulis akan membuat sebuah aplikasi yang dapat menangkap teks langsung melalui gambar, sehingga pengguna dengan mudah dalam mendeteksi letak teks. Pengguna hanya perlu mengambil gambar yang terdapat teks, aplikasi tersebut akan mendeteksi letak kata tersebut, dengan menggunakan teknik pengolahan citra aplikasi dapat menentukan terdapat teks atau tidak pada gambar tersebut. Teknologi tersebut tentu saja dapat memudahkan seseorang yang mempunyai aktifitas dalam mengenali pola.

Dalam mendeteksi teks pada pada gambar, dibagi menjadi 2 teknik, yaitu: *Connected Component-Based* [XUC10] dan *Texture-Based* [ANG10].Teknik *Connected Component-Based* mencari calon karakter pada gambar, setelah ditemukan, teknik ini menganalisis bagian sekitar calon karakter sehingga dapat dikelompokan menjadi suatu kata. Teknik ini jika digabung dengan (MSER) *Maximally Stable Extremal Region* based methods [XUC10] memiliki ketelitian/keakuratan 86.29%. Teknik *Texture-Based* membagi gambar menjadi bagian-bagian kecil dengan ukuran dan jumlah tertentu dengan tujuan untuk

mencari bagian gambar yang terdapat teks. Teknik ini memiliki keakuratan 92.8%-96.6% lebih tinggi dari pada teknik *Connected Component-Based*. [ANG10] .Teknik ini juga dapat mendeteksi teks dengan *resolution* gambar yang rendah. Oleh karena itu, penulis lebih memilih Teknik *Texture-Based* untuk mendeteksi teks pada gambar.

Dalam menentukan apakah ada atau tidak teks pada gambar, penulis menggunakan *machine learning*. Pada *machine learning* ada beberapa algoritma yang dapat digunakan untuk klasifikasi, diantaranya adalah: *Learning Vector Quantization, Back Propagation, Support Vector Machine*, dan masih banyak lagi. Penulis akan menggunakan algoritma *Support Vector Machine*, karena SVM menggunakan metode *Supervised Learning* yang dapat digunakan untuk menganalisa data dan mengenali pola, dan digunakan juga untuk analisis klasifikasi dan regresi. Melihat dari karakter algoritma *Support Vector Machine* sangat cocok menjadi *machine learning* dalam mendeteksi letak teks dengan menggunakan metode *Supervised Learning*.

Alasan penulis menggunakan *Support Vector Machine* sebagai *machine learning*, karena diberbagai jurnal telah dibuktikan bahwa *Support Vector Machine* memiliki hasil akurasi yang tinggi dalam hal klasifikasi dibandingkan *machine learning* yang lainnya. Dalam penelitian [MIN10], jika dibandingkan *machine learning* *Support Vector Machine* dengan *Backpropagation* pada kasus klasifikasi data maka didapatkan kesimpulan :

Method	Number of sample	Type I error	Type II error	Error	Accuracy
SVM	25	0/25 (0.0%)	0/25 (0.0%)	0/25 (0.0%)	100%
BPN	25	0/25 (0.0%)	1/25(4%)	1/25 (4%)	96%

Dalam penelitian [SHO15], jika dibandingkan *machine learning* *Support Vector Machine* dengan *Multi Layer Perceptron* (MLP) maka didapatkan kesimpulan bahwa *Support Vector Machine* sangat cocok untuk digunakan menganalisa gambar atau klasifikasi citra dibandingkan *Multi Layer Perceptron*. Dalam hal mengelola data *training*, *Support Vector Machine* lebih cepat 10 kali dari pada *Multi Layer Perceptron*.

## **1.2 Ruang Lingkup Masalah**

Berikut rumusan masalah yang dihadapi dalam pembuatan aplikasi dalam penelitian penentuan letak teks pada gambar yaitu :

1. Bagaimana memproses gambar dengan menggunakan Transformasi Kosinus Diskrit sehingga gambar dapat diproses lebih lanjut.
2. Bagaimana mendeteksi teks pada gambar menggunakan metode *Texture-based*.
3. Bagaimana menggunakan *machine learning Support Vector Machine* dalam melakukan klasifikasi letak teks.
4. Bagaimana aplikasi akan mendapatkan hasil akurasi yang tinggi dalam menentukan letak teks, dengan menggunakan *machine learning Support Vector Machine*.

## **1.3 Tujuan Penelitian**

Tujuan dari penelitian ini adalah menentukan letak teks pada gambar dengan menggunakan metode *Texture-based* dan *Support Vector Machine* dalam mengklasifikasikan letak teks. Setelah aplikasi ini selesai, aplikasi ini dapat digunakan untuk pengembangan lebih lanjut, misalnya menerjemahkan teks dalam berbagai bahasa dengan menggunakan hasil *output* program ini.

## **1.4 Batasan Masalah**

Batasan masalah dalam pembuatan aplikasi ini adalah:

1. Input data berupa gambar, ukuran gambar harus habis dibagi dengan ukuran blok.
2. Ukuran blok pada aplikasi sebesar 50x50 pixel.

## **1.5 Konstribusi Penelitian**

Dalam jurnal *Text Region Extraction from Low Resolution Natural Scene Images using Texture Features* [ANG10] (Andi Lukman,2010) telah dilakukan penelitian bahwa mendeteksi letak teks pada gambar dengan menggunakan metode *Texture-based* dan beberapa *preprocessing* didapatkan hasil akurasi 96.6% pada gambar dengan resolusi yang rendah. Pada penelitian kali ini, penulis ingin melakukan konstribusi yaitu, menggabungkan hasil penelitian yang telah dilakukan dengan *machine learning* yang menggunakan algoritma *Support Vector Machine* (SVM). *Machine learning* akan menentukan letak teks tertentu pada gambar dengan data *training* yang sudah diproses (diberikan *label*) tertentu, sehingga aplikasi dapat lebih mudah untuk menentukan letak teks pada gambar. Penulis mengharapkan penelitian ini dapat dikembangkan lebih lanjut hingga dapat menerjemahkan teks tersebut.

## **1.6 Sistematika Penulisan**

Adapun metodologi dalam penelitian ini, yaitu :

Sistematika penulisan laporan penelitian ini adalah sebagai berikut :

1. Bab I Pendahuluan, berisi penjelasan mengenai latar belakang permasalahan yang terjadi, ruang lingkup masalah, tujuan penelitian, batasan masalah, konstribusi penelitian, serta sistematika penulisan yang digunakan untuk menyusun laporan penelitian.
2. Bab II Dasar Teori, berisi dasar teori yang akan digunakan dalam analisis, perancangan, dan implementasi perangkat lunak.
3. Bab III Analisis dan Perancangan Perangkat Lunak, berisi analisis kebutuhan dan spesifikasi perangkat lunak yang akan dibangun beserta perancangan perangkat lunak sesuai spesifikasi tersebut.
4. Bab IV Implementasi dan Pengujian Perangkat Lunak, berisi implementasi dari perancangan perangkat lunak yang telah dilakukan serta pengujian

untuk membangun perangkat lunak sesuai dengan spesifikasi yang telah ditentukan sebelumnya.

5. Bab V Penutup, berisi kesimpulan dan saran yang didapatkan selama proses penelitian.

## **BAB II**

### **LANDASAN TEORI**

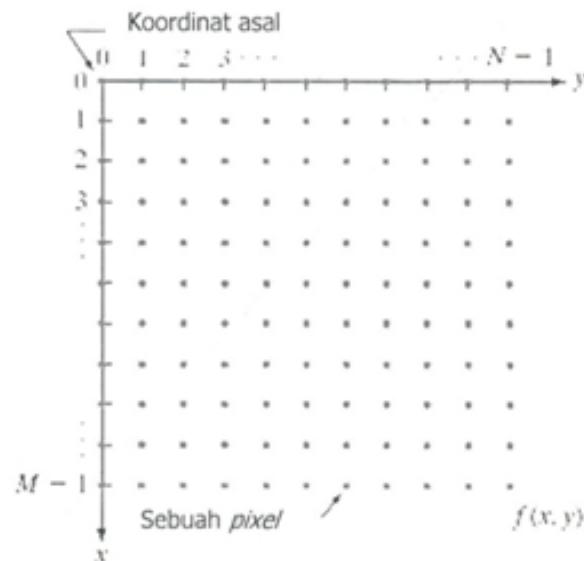
Pada bab ini akan dijelaskan mengenai teori dan metode yang digunakan oleh penulis dalam penelitian ini.

#### **2.1 Pengolahan Citra Digital**

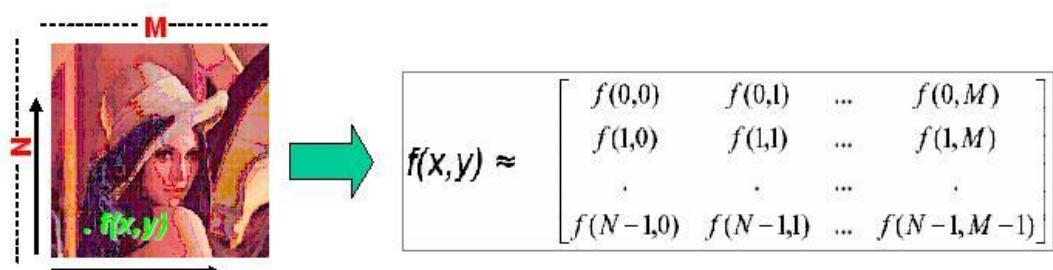
Secara umum, istilah pengolahan citra digital menyatakan “pemrosesan gambar berdimensi-dua melalui komputer digital” (Jain, 1989). Menurut Efford (2000), pengolahan citra adalah istilah umum untuk berbagai teknik yang keberadaannya untuk memanipulasi dan memodifikasi citra dengan berbagai cara. Foto adalah contoh gambar berdimensi dua yang bisa diolah dengan mudah. Setiap foto dalam bentuk citra digital (misalnya berasal dari kamera digital) dapat diolah melalui perangkat-lunak tertentu (Tukino, 2010).

Secara umum, pengolahan citra digital menunjuk pada pemrosesan gambar 2 dimensi menggunakan komputer. Dalam konteks yang lebih luas, pengolahan citra digital mengacu pada pemrosesan setiap data 2 dimensi. Citra digital merupakan sebuah larik (*array*) yang berisi nilai-nilai real maupun kompleks yang direpresentasikan dengan deratan bit tertentu. Suatu citra dapat didefinisikan sebagai fungsi  $f(x,y)$  berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitudo  $f$  dititik koordinat  $(x,y)$  dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai x, y, dan nilai amplitudo  $f$  secara keseluruhan berhingga (*finite*) dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra digital. Gambar 2.1 menunjukkan posisi korrdinat citra digital [DAR10].

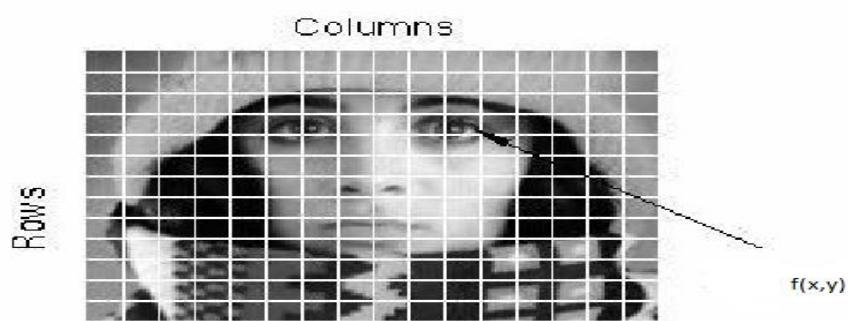
Nilai pada suatu irisan antara baris dan kolom (pada posisi x, y) disebut dengan *picture elements*, *image elements*, *pels*, atau *pixels*. Istilah pixel paling sering digunakan pada pengolahan citra digital.



**Gambar 2.1 Koordinat citra digital**



**Gambar 2.2 Citra digital dalam bentuk matrik.**



**Gambar 2.3 Ilustrasi digitalisasi citra**

Setiap pixel mewakili tidak hanya satu titik dalam sebuah citra melainkan sebuah bagian berupa kotak yang merupakan bagian terkecil (sel). Nilai dari sebuah pixel haruslah dapat menunjukkan nilai rata-rata yang sama untuk seluruh bagian dari sel tersebut.

### 2.1.1 Citra Biner

Citra biner adalah citra digital yang hanya memiliki dua kemungkinan nilai pixel yaitu hitam dan putih. Citra biner juga disebut sebagai citra B&W (*black and white*) atau citra monokrom. Hanya dibutuhkan 1 bit untuk mewakili nilai setiap pixel dari citra biner. Citra biner sering kali muncul sebagai hasil dari proses pengolahan seperti segmentasi, pengambangan, morfologi, ataupun *dithering..*



Gambar 2.4 Citra Biner

### 2.1.2 Citra *Grayscale*

Citra *grayscale* merupakan citra digital yang hanya memiliki satu niali kanal pada setiap pixelnya, dengan kata lain nilai bagian *RED* = *GREEN* = *BLUE*. Nilai tersebut digunakan untuk menunjukkan tingkat intensitas. Warna yang dimiliki adalah warna dari hitam, keabuan dan putih. Tingkatan keabuan di sini merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekati putih. Citra *grayscale* berikut memiliki kedalaman warna 8 bit (256 kombinasi warna keabuan).



**Gambar 2.5 Citra grayscale**

## 2.2 Transformasi Citra

Transformasi atau alih ragam citra dapat diartikan sebagai perubahan bentuk suatu citra. Perubahan bentuk tersebut dapat berupa perubahan geometri pixel seperti perputaran (rotasi), pergeseran (translasi), penskalaan, dan lain sebagainya atau dapat juga berupa perubahan ruang (domain) citra ke domain lainnya.

Citra hasil proses transformasi dapat dianalisis kembali, diinterpretasikan, dan dijadikan acuan untuk melakukan permrosesan selanjutnya. Tujuan diterapkannya transformasi citra adalah untuk memperoleh informasi (*feature extraction*) yang lebih jelas yang terkandung dalam suatu citra. Suatu citra dapat dinyatakan sebagai kombinasi linier dari sinyal dasar (*basic signals*) yang sering disebut dengan fungsi basis (*basis function*). Suatu citra yang telah mengalami transformasi dapat diperoleh kembali dengan menggunakan transformasi balik (*inverse transformation*).

### 2.2.1 Transformasi Kosinus Diskrit(*Discrete Cosine Transform*)

Transformasi kosinus diskrit (DCT) hampir mirip dengan Transformasi Fourier Diskrit yang berfungsi berupa fungsi sinus. Melalui transformasi Fourier, suatu citra (sinyal atau fungsi) dapat dinyatakan sebagai penjumlahan sinyal sinus atau kosinus dengan amplitude dan frekuensi yang bervariasi. Perbedaan Transformasi Kosinus Diskrit dengan Transformasi Fourier Diskrit adalah Transformasi Kosinus Diskrit menggunakan komponen kosinus saja. DCT telah menjadi pilihan sebagai dasar algoritma kompresi JPEG dan MPEG.

#### DCT 1 Dimensi

DCT 1 dimensi  $C(u)$  didefinisikan sebagai berikut.

$$C(u) = \frac{\sqrt{2}}{\sqrt{N}} \alpha(u) \sum_{x=0}^{N-1} f(x) \cos \left[ \frac{\pi(2x+1)u}{2N} \right] \quad (2.2)$$

Untuk  $u = 0, 1, 2, \dots, N-1$ .

Dengan cara yang sama, DCT balik (*invers*) dapat didefinisikan sebagai berikut:

$$f(x) = \frac{\sqrt{2}}{\sqrt{N}} \sum_{u=0}^{N-1} \alpha(u) C(u) \cos \left[ \frac{\pi(2x+1)u}{2N} \right] \quad (2.3)$$

Untuk  $u = 0, 1, 2, \dots, N-1$

Dengan  $\alpha(u)$  dinyatakan sebagai berikut.

$$\alpha(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{untuk } u = 0 \\ 1 & \text{untuk } u \neq 0 \end{cases} \quad (2.4)$$

Dimana:

$u$  : Iterasi

$N$  : Panjang dimensi

$x$  : Indeks pada matriks pada arah baris

$\alpha(u)$  : konstanta

## DCT 2 Dimensi

Persamaan DTC 2 dimensi untuk citra  $f(x,y)$  sebagai berikut:

$$C(u, v) = \frac{\sqrt{2}}{\sqrt{MN}} \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x, y) \cos\left[\frac{\pi(2x+1)u}{2N}\right] \cos\left[\frac{\pi(2y+1)v}{2M}\right] \quad (2.6)$$

Dengan  $u = 0, 1, 2, \dots, N-1$  dan  $v = 0, 1, 2, 3, \dots, M-1$  sedangkan

Transformasi kosinus diskrit 2D invers dapat dinyatakan sebagai:

$$f(x, y) = \frac{\sqrt{2}}{\sqrt{MN}} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} \alpha(u)\alpha(v) C(u, v) \cos\left[\frac{\pi(2x+1)u}{2N}\right] \cos\left[\frac{\pi(2y+1)v}{2M}\right] \quad (2.8)$$

Dengan  $x = 0, 1, 2, 3, \dots, N-1$  dan  $y = 0, 1, 2, 3, \dots, M-1$ .

Dengan  $\alpha(k)$  dinyatakan sebagai berikut.

$$\alpha(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{untuk } k = 0 \\ 1 & \text{untuk } k \neq 0 \end{cases} \quad (2.7)$$

Dimana:

$u$  : Iterasi

$M$  : Panjang dimensi

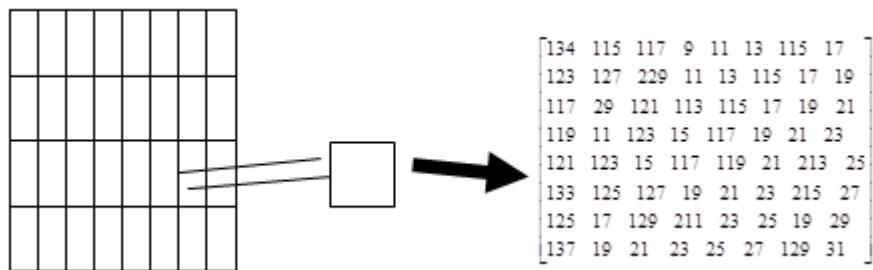
$N$  : Lebar dimensi

$x$  : Indeks pada matriks pada arah baris

$y$  : Indeks pada matriks pada arah kolom

$\alpha(k)$  : konstanta

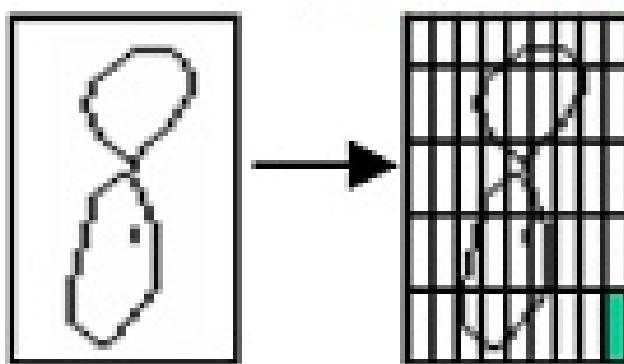
DCT 2 dimensi dapat juga dihitung dengan menggunakan DCT 1 dimensi dua tahap. Pertama, DCT 1 dimensi dilakukan dalam arah baris pada citra, setelah itu dilakukan DCT 1 dimensi dalam arah kolom pada citra. Gambar yang akan diproses DCT akan dibagi dalam blok-blok (8x8), masing-masing blok tersebut akan dikenai DCT dengan tujuan untuk mempercepat proses DCT.



**Gambar 2.6 Contoh pembagian pada gambar menjadi beberapa blok**

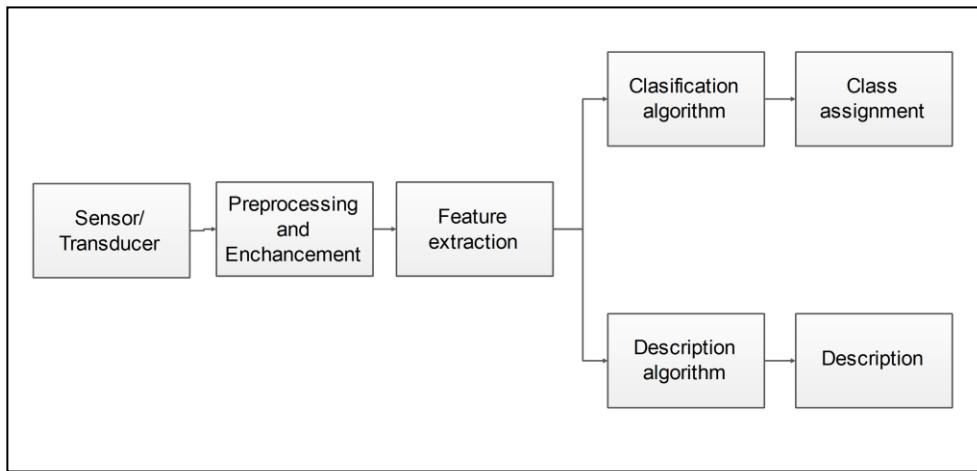
### 2.3 Pengenalan Pola

Pengenalan pola biasanya digunakan pada mesin-mesin dengan kecerdasan buatan (*Machine Learning*). Teknik pengenalan pola merupakan salah satu komponen penting dari mesin atau system cerdas tersebut yang digunakan baik untuk mengolah data maupun dalam pengambilan keputusan. Pengenalan pola adalah suatu ilmu untuk mengklasifikasikan atau menggambarkan sesuatu berdasarkan pengukuran kuantitatif fitur (ciri) atau sifat utama dari suatu objek. Pola sendiri adalah suatu entitas yang terdefinisi dan dapat didentifikasi serta diberi nama, contoh: sidik jari.



**Gambar 2.7 Citra Karakter (Fokunaga, 1990)**

Struktur pengenalan pola dapat dilihat dari gambar 2.7 Sistem terdiri atas sensor misalnya:kamera, suatu algoritma atau mekanisme pencari fitur, dan algoritma untuk klasifikasi atau pengenalan. Sebagai tambahan, biasanya beberapa data yang sudah diklasifikasian diasumsikan telah tersedia untuk melatih sistem.



**Gambar 2.8 Struktur sistem pengenalan pola**

1. **Sensor** berfungsi untuk menangkap objek dari dunia nyata dan selanjutnya diubah menjadi sinyal digital (sinyal yang terdiri atas sekumpulan biangan) melalui proses digitalisasi.
2. **Preprocessing** berfungsi mempersiapkan citra atau sinyal agar dapat menghasilkan ciri yang lebih baik pada tahap berikutnya. Pada tahap ini sinyal informasi ditonjolkan dan sinyal pengganggu (*noise*) diminimalisasi.
3. **Feature extraction** berfungsi menemukan karakteristik pembeda yang mewakili sifat utama sinyal dan sekaliagus mengurangi dimensi sinyal menjadi sekumpulan bilangan yang lebih sedikit tetapi representatif.
4. **Classification algorithm** berfungsi untuk mengelompokan fitur ke dalam kelas yang sesuai.
5. **Description algorithm** berfungsi memberikan deskripsi pada sinyal.

## 2.4 Operasi Spasial (Filtering)

Pentapisan pada pengolahan citra biasa disebut dengan pentapisan spasial (*spatial filtering*). Nilai pixel baru tersebut dapat dikelompokkan menjadi 2, yaitu pertama, pixel baru diperoleh melalui kombinasi linier pixel tetangga dan kedua, pixel baru diperoleh langsung dari salah satu nilai pixel tetangga. Berdasarkan kedua kelompok tersebut, maka tapis juga dikelompokkan menjadi 2, yaitu tapis linear dan tapis non-linear. Pada operasi spasial tidak terlepas dari teori *kernel*

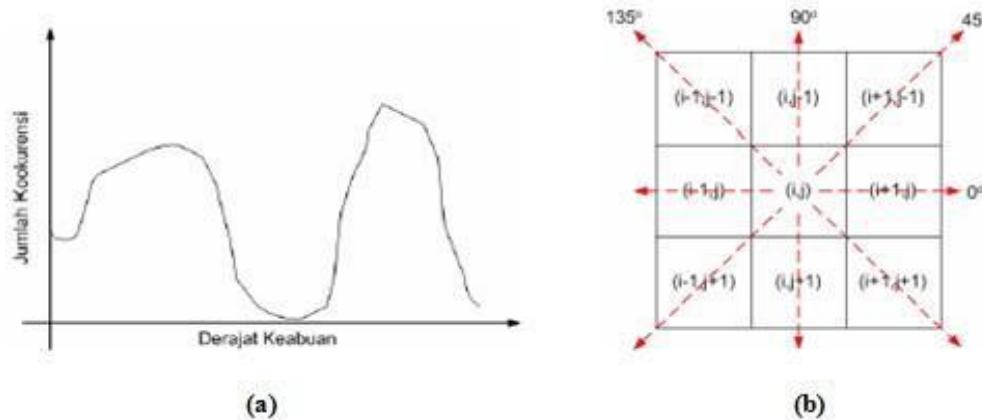
(mask). Kernel adalah matrik yang pada umumnya berukuran kecil dengan elemen-elemennya adalah berupa bilangan. Ukuran kernel dapat berbeda-beda 2x2, 3x3, 5x5, dan sebagainya. Elemen-elemen kernel yang juga disebut bobot merupakan bilangan-bilangan yang membentuk pola-pola tertentu.[ DAR10]

#### **2.4.1 Tapis High-pass**

Tapis *high-pass* adalah mempertahankan (mempertajam) komponen frekuensi tinggi dan menghilangkan (mengurangi) komponen frekuensi rendah sehingga tapis ini sangat cocok untuk penajaman tepi citra. Nilai koefisien tapis pada koordinat pusat bernilai positif dan koefisien kelilingnya bernilai negatif. Pada proses Tapis high pass ini, biasanya menggunakan kernel yang akan dikalikan dengan hasil koefisien DCT atau dari matriks pixel [DAR10].

### **2.5 GLCM (Gray Level Co-occurrence Matrices)**

Metode GLCM (gray-level co-occurrence matrix) adalah salah satu cara mengekstrak fitur tekstur statistik orde-kedua (Albregtsen, 1995). Metode GLCM termasuk dalam metode statistik dimana dalam perhitungan statistiknya menggunakan distribusi derajat keabuan (histogram) dengan mengukur tingkat kekontrasan, granularitas, dan kekasaran suatu daerah dari hubungan ketetanggaan antar piksel di dalam citra. Paradigma statistik ini penggunaannya tidak terbatas, sehingga sesuai untuk tekstur-tektur alami yang tidak terstruktur dari sub pola dan himpunan aturan (mikrostruktur). Ciri statistik orde kedua dilakukan dengan matriks kookurensi, yaitu suatu matriks antara yang merepresentasikan hubungan ketetanggaan antar piksel dalam citra pada berbagai arah orientasi dan jarak spasial.



**Gambar 2.9 Arah dan jarak GLCM**

Pada gambar 2.15(b) dapat diihat arah dan jarak GLCM yang dapat digunakan untuk mendapatkan matriks kemunculan pola. Matriks kemunculan pola akan dinormalisasi dengan cara membagi hasil setiap nilai dengan nilai rata-rata dari seluruh nilai matriks kemunculan pola. Setelah dilakukan normalisasi maka matriks tersebut akan ditranspose dengan cara mengubah bagian baris menjadi kolom dan kolom menjadi baris. Ciri atau fitur statistik GLCM dapat dihitung dengan menggunakan matriks yang sudah diproses tersebut. Ciri atau fitur statistik GLCM antara lain:

### 1. Kontras

Perhitungan kontras berkaitan dengan jumlah keberagaman intensitas keabuan dalam citra.

$$Kontras = \sum_{a,b=0}^{n-1} |a - b|^2 P_{a,b} \quad (2.9)$$

Dimana:

$P_{a,b}$ : Nilai pada baris ke-a dan kolom ke -b

a : Indeks pada baris

b : Indeks pada kolom

## 2. Homogenitas

Secara matematis, homogenitas GLCM adalah invers dari kontras GLCM, yaitu keseragaman intensitas keabuan pada citra.

$$\text{Homogenitas} = \sum_{a,b=0}^{n-1} \frac{P_{a,b}}{1 + (a - b)^2} \quad (2.10)$$

Dimana:

$P_{a,b}$  : Nilai pada baris ke-a dan kolom ke - b

a : Indeks pada baris

b : Indeks pada kolom

## 3. Energi

Energi menyatakan ukuran konsentrasi pasangan dengan intensitas keabuan tertentu pada matriks.

$$\text{Energi} = \sum_{a,b=0}^{n-1} P_{a,b}^2 \quad (2.11)$$

Dimana:

$P_{a,b}$  : Nilai pada baris ke-a dan kolom ke- b

a : Indeks pada baris

b : Indeks pada kolom

## 4. Entropi

Entropi digunakan untuk mengukur keteracakannya dari distribusi intensitas.

$$\text{Entropi} = \sum_{a,b=0}^{n-1} P_{a,b} \log_2(P_{a,b}) \quad (2.12)$$

Dimana:

$P_{a,b}$  : Nilai pada baris ke-a dan kolom ke - b

a : Indeks pada baris

b : Indeks pada kolom

## 5. Korelasi

Menyatakan ukuran hubungan dependen piksel terhadap piksel tetangga dalam citra. Pada persamaan (2.13 – 2.17),  $Mean(\mu_x, \mu_y)$  adalah rata-rata dari suatu sebaran nilai intensitas citra keabuan. Sedangkan standar *deviasi* ( $\sigma_x, \sigma_y$ ) adalah menunjukkan sebaran nilai piksel pada bidang citra [MUH13].

$$\text{Korelasi} = \sum_{a,b=0}^{n-1} P_{a,b} \left[ \frac{(a - \mu_x)(b - \mu_y)}{\sigma_x \sigma_y} \right] \quad (2.13)$$

Keterangan :

$$\mu_x = \sum_{a=0}^{n-1} a \sum_{b=0}^{n-1} P_{a,b} \quad (2.14)$$

$$\mu_y = \sum_{a=0}^{n-1} b \sum_{b=0}^{n-1} P_{a,b} \quad (2.15)$$

$$\sigma_x = \sum_{a=0}^{n-1} (a - \mu_x)^2 \sum_{b=0}^{n-1} P_{a,b} \quad (2.16)$$

$$\sigma_y = \sum_{a=0}^{n-1} (b - \mu_y)^2 \sum_{b=0}^{n-1} P_{a,b} \quad (2.17)$$

Dimana:

$P_{a,b}$  : Nilai pada baris ke-a dan kolom ke - b

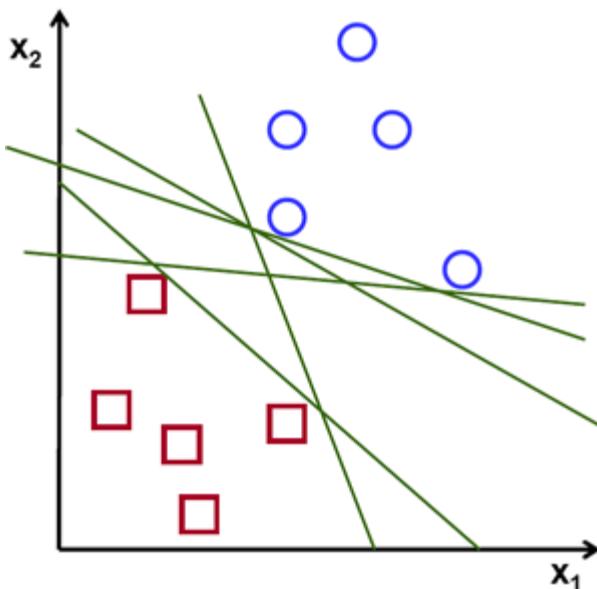
a : Indeks pada baris

b : Indeks pada kolom

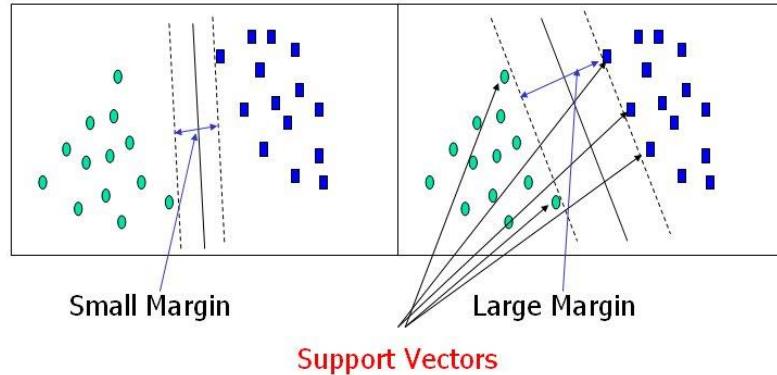
## 2.6 Support Vector Machine(SVM)

*Support Vector Machine* merupakan algoritma dengan metode *Supervised learning*, dimana program diberi data *training* sebagai bahan pembelajaran yang berfungsi menganalisa dan mengenali pola, dan akan digunakan untuk analisis klasifikasi dan regresi.

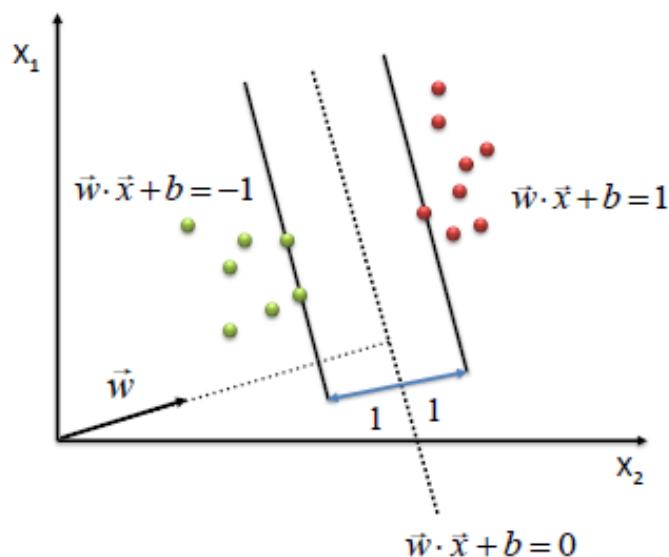
SVM mencari sebuah *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah class pada ruang input. *Hyperplan* dapat diartikan juga sebagai sebuah solusi untuk memisahkan/mengklasifikasikan kelas yang berbeda [AND13]. Pada Gambar 2.9. terdiri dari beberapa solusi (*hyperplan*) yang berfungsi untuk memisahkan/mengklasifikasikan kelas yang ada. *Hyperplan* yang terbaik adalah *hyperplan* yang memisahkan kelas yang ada dengan jarak *margin hyperplan* terbesar. *Margin* adalah jarak antara *hyperplan* tersebut dengan pola terdekat dari masing-masing kelas gambar 2.10.



Gambar 2.10 Berbagai garis hyperplan (discrimination boundaries).



**Gambar 2.10 Ukuran margin hyperplan.**



**Gambar 2.12 Rumus untuk hyperplan.**

*Support Vector Machine* akan membentuk *hyperplan*, yang membagi data menjadi 2 kelas yang berbeda. *Hyperplan* itu sendiri memiliki rumus:

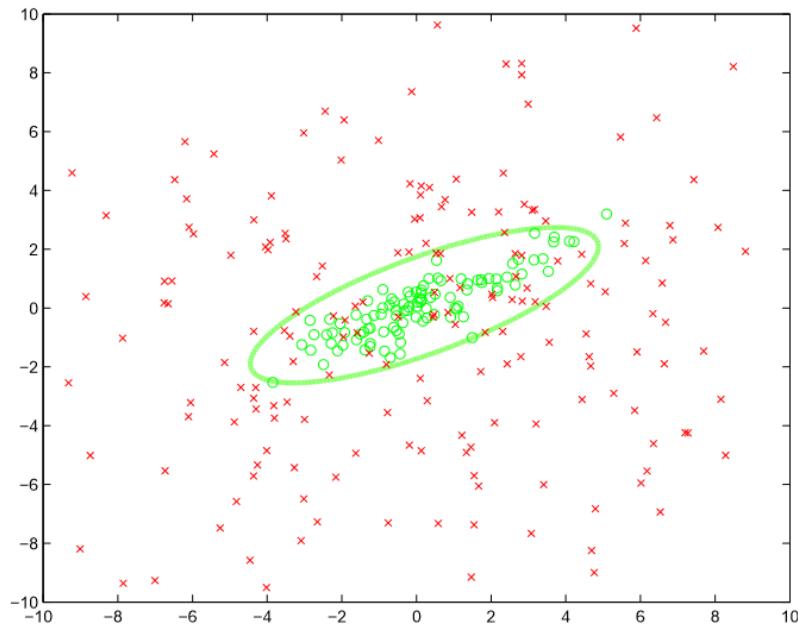
$$\vec{w} \cdot \vec{x} + b = 0 \quad (2.18)$$

$$\vec{w} \cdot \vec{x} + b = -1 \quad (2.19)$$

$$\vec{w} \cdot \vec{x} + b = 1 \quad (2.20)$$

Persamaan 2.19 adalah kelas -1 pada bagian kiri dan persamaan 2.20 adalah kelas +1 pada bagian kanan. Tetapi SVM bertipe *Linear* tidak dapat digunakan pada

semua kasus. Beberapa kasus, akan lebih baik jika menggunakan SVM bertipe *Non-Linear*.



**Gambar 2.13 SVM non-linear.**

Pada gambar 2.12 terlihat klasifikasi kelas tidak mungkin dilakukan dengan membagi menjadi 2 dengan cara menarik garis (*hyperplan*) secara lurus. Oleh karena itu, dibutuhkan garis yang tidak lurus untuk memisahkan kelas tersebut, hal ini disebut *non-linear*. SVM *non-linear* akan menggunakan kernel yang berfungsi untuk mempermudah perhitungan. Ada beberapa macam kernel yang dapat digunakan, penulis menggunakan Kernel Gaussian RBF. Kernel pada persamaan tersebut akan digunakan sehingga akan mendapat suatu persamaan yang akan disubstitusi persamaan satu dengan persamaan lainnya dengan tujuan untuk mendapatkan alpha dan bias (b).

Polynomial:

$$K(q1, q2) = \tanh(\alpha \cdot q1 \cdot q2 + b) \quad (2.21)$$

Gaussian RBF:

$$K(q1, q2) = \exp(-||q1 - q2||/\alpha^2) \quad (2.22)$$

Sigmoid:

$$K(q1, q2) = (1 + q1 \cdot q2)^k \quad (2.23)$$

Penjelasan Rumus Gaussian RBF dimana:

$q1, q2$  : Nilai *feature extraction*

Exp : Eksponensial (Contoh :  $\exp(x) = e^x$ )

$\alpha$  : Konstanta

Persamaan yang digunakan sebagai berikut:

Constraint 1:

$$\sum \alpha_i y_i = 0 \quad (2.24)$$

Constraint 2:

$$\sum \alpha_i y_i K(x_i, x_2) + b = +1 \quad (2.25)$$

Constraint 3:

$$\sum \alpha_i y_i K(x_i, x_2) + b = -1 \quad (2.26)$$

Dimana:

$K(x_1, x_2)$  : Nilai RBF pada baris ke  $x_1$ , dan kolom ke  $x_2$  dari tabel yang telah dihitung pada persamaan 2.22

**b** : bias

Constraint 1 : Persamaan untuk kelas +1

Constraint 2 : Persamaan untuk kelas -1

Untuk pengambilan keputusan (klasifikasi), menggunakan rumus dengan syarat alpha dan bias(b) harus diketahui:

$$f(x) = \text{sign} \left( \sum_{i=1}^l \alpha_i y_i K(x, x_i) + b \right) \quad (2.27)$$

Dimana:

$\alpha$  : Nilai alpha yang sudah dicari pada persamaan 2.24 – 2.26

$y_i$ : Kelas data training (+1 atau -1)

$(x, x_i)$  : Nilai RBF data uji

**b** : bias

Fungsi *sign* pada persamaan 2.27 berfungsi untuk mengubah nilai menjadi 1 jika hasil bernilai positif, dan mengubah nilai menjadi -1 jika hasil bernilai negatif.

## 2.7 Java Matrix (JAMA)

JAMA adalah sebuah *library* yang berisi paket linear aljabar dasar untuk Java. JAMA menyediakan kelas-kelas untuk yang berfungsi untuk membuat dan memanipulasi matriks dengan mudah. Pada penelitian kali ini, penulis menggunakan JAMA untuk menghitung determinant yang berguna untuk mensubtitusikan persamaan pada rumus SVM dengan persamaan 2.24, persamaan 2.25, dan persamaan 2.26. Dengan menggunakan *library* JAMA tersebut, waktu yang diperlukan untuk menghitung determinant menjadi lebih singkat dibandingkan menghitung determinant secara manual. Contoh implementasi untuk mendapatkan determinant:

```
Matrix temp = new Matrix (det);
detResult = temp.det();
```

**Gambar 2.14 Contoh implemntasi libraby JAMA**

## 2.8 Image Plus

*Image plus* adalah sebuah *library* java pengolah gambar Java. *Library* ini dapat menampilkan, mengedit, menganalisa, memproses, menyimpan dan mencetak 8-bit, 16-bit dan 32-bit tipe gambar dan dapat membaca banyak format gambar termasuk TIFF, GIF, JPEG, BMP, DICOM, FITS. Pada *libraby* ini juga terdapat fitur untuk menghitung luas, mengukur jarak dan sudut, dan terdapat fungsi pengolahan citra seperti manipulasi kontras, mempertajam gambar, *smoothing*, deteksi tepi dan *median filtering*. Penulis menggunakan *library* tersebut dengan tujuan mengubah citra gambar menjadi citra *grayscale* 8-bit dengan *Image* sebagai parameternya.

```
ImagePlus ip = new ImagePlus ("image about to gs",colorImage);
ImageConverter ic = new ImageConverter(ip);
Ic.convertToGray8();
Return ip.getImage();
```

**Gambar 2.15 Contoh Implementasi library Image Plus untuk Grayscale**

## 2.9 Segmentasi Citra

Segmentasi merupakan teknik untuk membagi suatu citra menjadi beberapa daerah (region) dimana setiap daerah memiliki kemiripan atribut antara tingkat keabuan suatu piksel dengan tingkat keabuan piksel-piksel tetangganya. Proses segmentasi memiliki tujuan yang hampir sama dengan proses klasifikasi tidak terpandu. Segmentasi sering dideskripsikan sebagai proses analogi terhadap proses pemisahan latar depanlatar belakang [DAR10].

### 2.9.1 Pengambangan (*Thresholding*)

Proses *Thresholding* akan menghasilkan citra biner, yaitu citra yang memiliki dua nilai tingkat keabuan yaitu hitam dan putih. Secara umum proses *thresholding* citra *grayscale* untuk menghasilkan citra biner adalah sebagai berikut.

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) \geq T \\ 0 & \text{if } f(x,y) < T \end{cases} \quad (2.28)$$

Dengan  $g(x,y)$  adalah citra biner dari citra *grayscale*  $f(x,y)$ , dan  $T$  menyatakan nilai ambang. Nilai  $T$  memegang peranan yang sangat penting dalam proses pengambangan. Kualitas hasil citra biner sangat tergantung pada nilai  $T$  yang digunakan.

Terdapat dua jenis pengambangan, yaitu pengambangan global (*global thresholding*) dan pengambangan secara local adaptif (*locally adaptive thresholding*). Pada pengambangan global, seluruh pixel pada citra dikonversikan menjadi hitam atau putih dengan satu nilai ambang  $T$ . Kemungkinan besar pada pengambangan global akan banyak informasi hilang karena hanya menggunakan satu nilai  $T$  untuk keseluruhan pixel. Untuk mengatasi masalah ini dapat digunakan pengambangan secara lokal adaptif. Pada pengambangan lokal, suatu citra dibagi menjadi blok-blok kecil dan kemudian dilakukan pengeambangan local pada setiap blok dengan nilai  $T$  yang berbeda.

### 2.9.2 Threshold Otsu

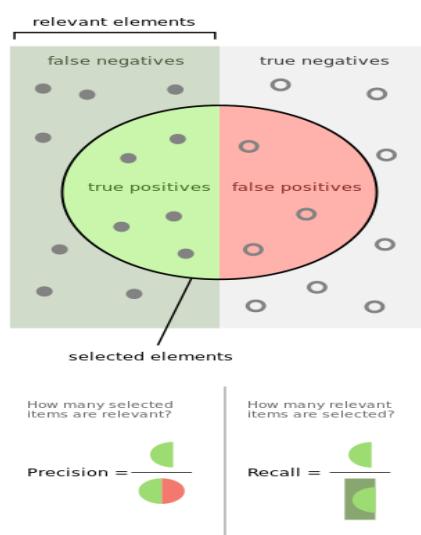
Tujuan dari metode *threshold otsu* tersebut adalah mencari nilai ambang ( $T$ ) secara otomatis dari sebuah gambar melalui histogram *citra gray level*, sehingga penulis tidak perlu menentukan nilai ambang *thresholding* secara manual. Pendekatan yang digunakan oleh metode Otsu adalah dengan melakukan analisis diskriminan yaitu menentukan suatu variabel yang muncul secara alami berfungsi untuk memaksimumkan variabel tersebut agar dapat memisahkan objek dengan latar belakang.

## 2.10 F-Measure

*F-measure* merupakan salah satu perhitungan evaluasi dalam menghitung akurasi informasi yang mengkombinasikan *recall* dan *precision*. Nilai *recall* dan *precision* pada suatu keadaan dapat memiliki bobot yang berbeda. *Precision* (p) adalah jumlah sampel berkategori positif diklasifikasi benar dibagi dengan total sampel yang diklasifikasi sebagai sample positif. *Recall* (r) adalah jumlah sampel diklasifikasi positif dibagi total sampel dalam testing set berkategori positif.

Untuk menghitung *Precision* dan *Recall* diperlukan jumlah/nilai dari *True Positive*, *True Negative*, *False Positive*, dan *False Negative*. Pada setiap kasus definisi tersebut dapat berbeda-beda. Berikut adalah keterangan-keterangan untuk *F-measure* pada kasus ini sebagai berikut:

1.  $Precision = \text{True Positive} / (\text{True Positive} + \text{False Positive})$ .
2.  $Recall = \text{True Positive} / (\text{True Positive} + \text{False Negative})$ .
3.  $F\text{-Measure} = 2 \times (Precision \times Recall) / (Precision + Recall)$ .



Gambar 2.16 Visualisasi Precesion dan Recall

## 2.11 Pesamaan Linear

Persamaan linear adalah suatu himpunan nilai yang memenuhi secara serentak (simultan) persamaan-persamaan dari sistem tersebut. Secara sederhana

penyelesaian sistem persamaan linear adalah menentukan titik potong dari dua persamaan linear.

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m$$

**Gambar 2.17 Bentuk Umum**

Ada tiga cara yang dapat digunakan untuk penyelesaian suatu sistem persamaan linear, yaitu: (1). Metode Substitusi, (2). Metode Eliminasi, dan (3). Metode Determinan. Pada penelitian kali ini, penulis menggunakan Metode Determinan untuk menyelesaikan suatu sistem persamaan linear.

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - cb$$

$$\begin{vmatrix} 4 & 1 \\ 2 & 3 \end{vmatrix} = 4 \times 3 - 2 \times 1 = 12 - 2 = 10$$

**Gambar 2.18 Rumus perhitungan persamaan linear 2x2**

Contoh persamaan linear 2x2:

$$x - 3y = 6$$

$$2x + 3y = 3.$$

Dari gambar 2.19, dapat didapatkan determinan = 9, x=3 dan y= -1.

$$\begin{array}{l} a_1 = 1 \\ a_2 = 2 \end{array} \quad \begin{array}{l} b_1 = -3 \\ b_2 = 3 \end{array} \quad \begin{array}{l} c_1 = 6 \\ c_2 = 3 \end{array}$$

$$x = \frac{\begin{vmatrix} 6 & -3 \\ 3 & 3 \end{vmatrix}}{\begin{vmatrix} 1 & -3 \\ 2 & 3 \end{vmatrix}} = \frac{18 + 9}{3 + 6} = 3$$

$$y = \frac{\begin{vmatrix} 1 & 6 \\ 2 & 3 \end{vmatrix}}{\begin{vmatrix} 1 & -3 \\ 2 & 3 \end{vmatrix}} = \frac{3 - 12}{3 + 6} = \frac{-9}{9} = -1$$

**Gambar 2.19 Contoh penyelesaian persamaan linear 2x2**

$$\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} = a_1 \begin{vmatrix} b_2 & c_2 \\ b_3 & c_3 \end{vmatrix} - a_2 \begin{vmatrix} b_1 & c_1 \\ b_3 & c_3 \end{vmatrix} + a_3 \begin{vmatrix} b_1 & c_1 \\ b_2 & c_2 \end{vmatrix}$$

**Gambar 2.20 Rumus perhitungan persamaan linear 3x3**

Contoh persamaan linear 3x3:

$$2x + 3y + z = 2$$

$$-x + 2y + 3z = -1$$

$$-3x - 3y + z = 0$$

$$\Delta = \begin{vmatrix} 2 & 3 & 1 \\ -1 & 2 & 3 \\ -3 & -3 & 1 \end{vmatrix} = 2(11) + 1(6) - 3(7) = 7$$

$$x = \frac{\begin{vmatrix} 2 & 3 & 1 \\ -1 & 2 & 3 \\ 0 & -3 & 1 \end{vmatrix}}{\Delta}$$

$$y = \frac{\begin{vmatrix} 2 & 2 & 1 \\ -1 & -1 & 3 \\ -3 & 0 & 1 \end{vmatrix}}{\Delta}$$

$$z = \frac{\begin{vmatrix} 2 & 3 & 2 \\ -1 & 2 & -1 \\ -3 & -3 & 0 \end{vmatrix}}{\Delta}$$

$$x = \frac{2(11) + 1(6) + 0}{7} = \frac{28}{7} = 4$$

$$y = \frac{2(-1) + 1(2) - 3(7)}{7} = -\frac{21}{7} = -3$$

$$z = \frac{2(-3) + 1(6) - 3(-7)}{7} = \frac{21}{7} = 3$$

**Gambar 2.21 Contoh penyelesaian persamaan linear 3x3**

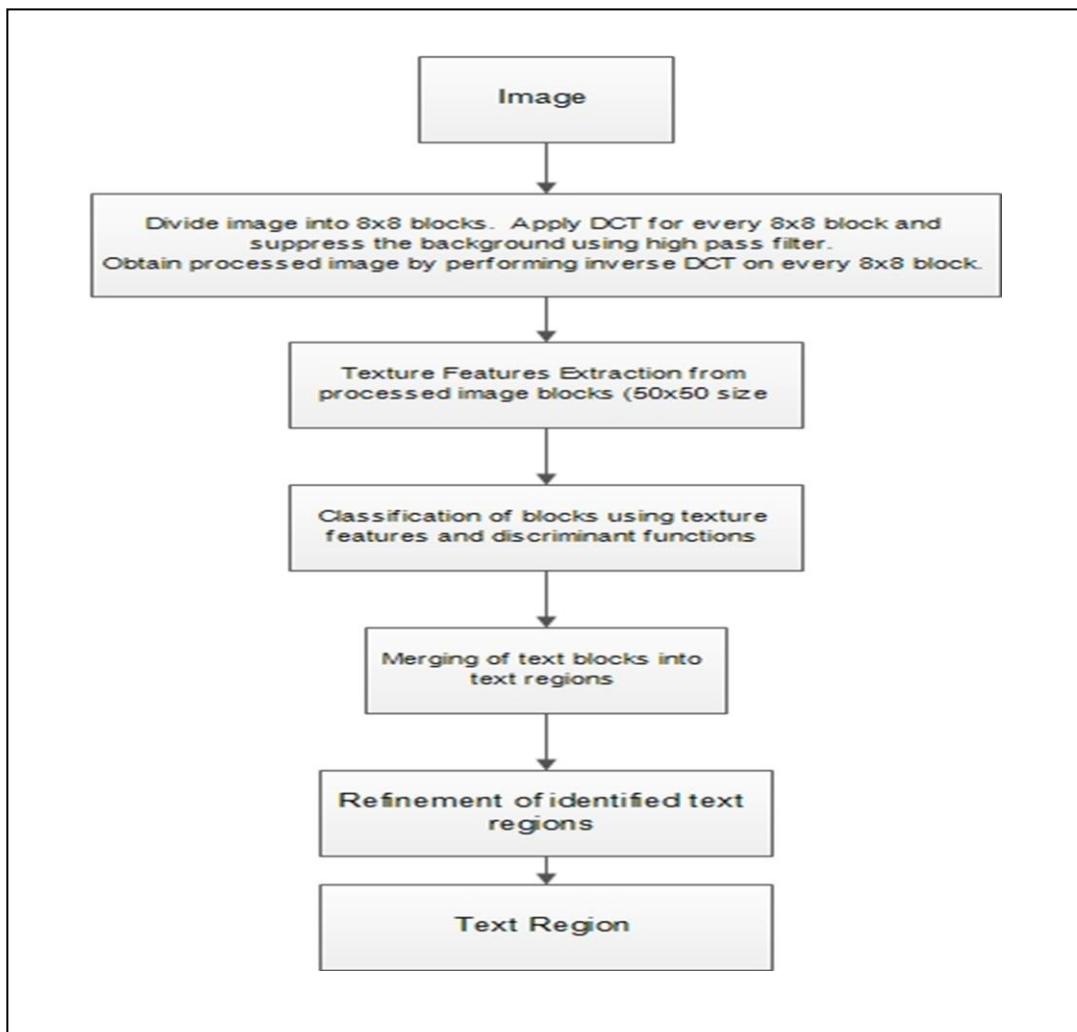
Dari gambar 2.21, didapatkan determinan = 7, x = 4, y = -3, dan z = 3.

## 2.12 Tinjauan Studi

Jurnal yang ditinjau oleh penulis berjudul “*Text Region Extraction from Low Resolution Natural Scene Images using Texture Features*”. Pada jurnal ini membahas mengenai, mendeteksi kata/kalimat pada gambar dengan kualitas gambar yang rendah yaitu menggunakan *handphone* sebagai alat untuk mengambil gambar tersebut. Dengan menggunakan metode Texture-based, membuat aplikasi tersebut dapat mendeteksi letak kata yang berada pada gambar tersebut. Pada jurnal ini juga membahas mengenai bagian *pre-processing* dalam mendeteksi kata/kalimat tersebut, Transformasi Kosinus Diskrit(*Discrete Cosine*

*Transform*) digunakan sebagai membuang bagian *background* pada gambar dan gangguan (*noise*) yang ada, sehingga dapat menentukan letak kata tersebut.

Hasil dari penelitian tersebut menyebutkan, bahwa penelitian mendeteksi kata/kalimat dengan *texture-based* mendapatkan akurasi sebesar 96.6% dan tingkat error sebesar 3.4% dari 100 foto yang diambil di dalam ruangan (*indoor*) maupun di luar ruangan (*outdoor*) dengan menggunakan nilai-nilai *feature extraction* dalam membuat model yang akan dibandingkan dengan data uji. Pada jurnal ini tidak disebutkan mengenai *feature extraction* apa saja yang digunakan untuk membuat model tersebut, sehingga penulis berasumsi untuk menggunakan *feature extraction* yang terdapat pada GLCM (Gray Level Co-occurrences Matrices) seperti kontrast, homogenitas, energi, entropi, dan kolerasi (Persamaan 2.9-2.17).



Gambar 2.22 Pre-processing menggunakan metode Texture-Based

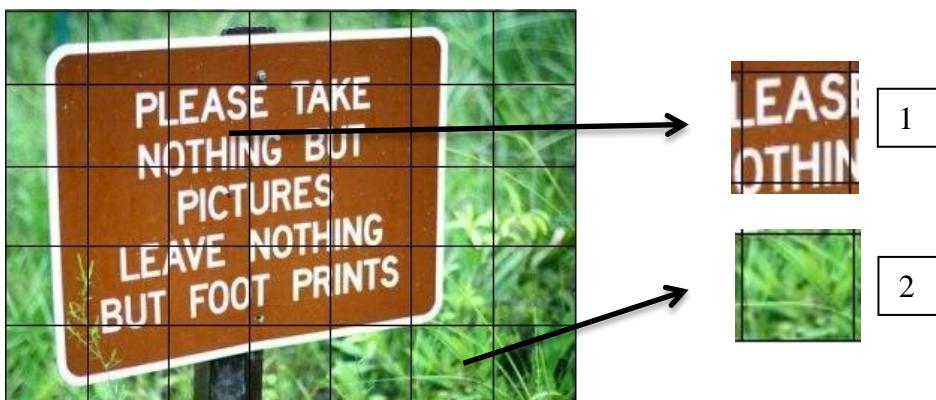
## BAB III

### ANALISIS MASALAH DAN IMPLEMENTASI

Pada bab ini menjelaskan analisis masalah dan perancangan sistem yang akan dikembangkan. Dalam bab ini juga akan dijelaskan gambaran/ alur dari perangkat lunak ini sendiri.

#### 3.1 Analisis Masalah

Mendeteksi teks pada gambar akan menjadi teknologi yang sangat berguna untuk masa depan ketika hal tersebut dikembangkan pada berbagai media, salah satunya seperti yang sudah dijelaskan pada sub-bab latar belakang. Untuk mengembangkan kasus tersebut, tentu saja ilmu dalam pengolahan citra atau *image processing* sangat berperan penting. Pada kasus ini, penulis menggunakan teknik *Texture-Based* yaitu dimana gambar dibagi menjadi beberapa bagian. Metode *Texture-Based* memiliki kelebihan, yaitu memiliki akurasi yang lebih tinggi dibandingkan metode *Connected Component-Based*, metode *Texture-Based* juga dapat memproses gambar dengan resolusi yang rendah, dan metode tersebut lebih mudah digunakan [ANG10].



Gambar 3.1 Image with texture based feature

Pada gambar 3.1 dapat terlihat perbedaan *texture* antara blok satu dengan blok dua, pada blok satu terdapat unsur teks dan akan diklasifikasikan sebagai

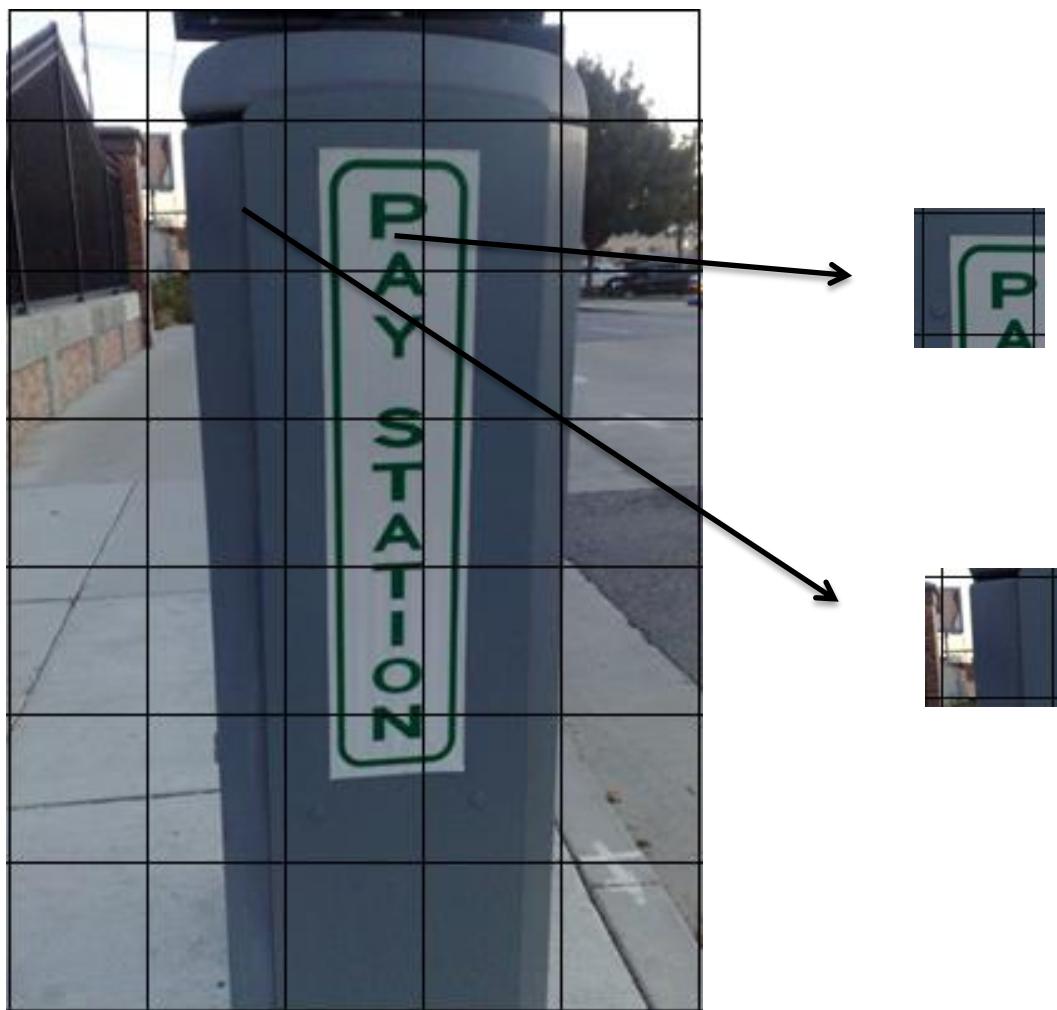
teks, tetapi pada blok kedua tidak memiliki unsur teks dan akan diklasifikasikan sebagai non-teks.

Setelah gambar diproses dengan metode *Texture-Based*, gambar akan diproses dengan menggunakan Transformasi Kosinus Diskrit (DCT) yang berfungsi untuk menghilangkan *background* atau *noise* (noda/kotoran pada gambar dan digunakan untuk melakukan kompresi pada gambar. Metode Transformasi Kosinus Diskrit (DCT) dipilih oleh penulis karena metode DCT lebih mudah diimplementasikan terhadap software dibandingkan metode yang lain [ANI11].

Tetapi tentu saja penelitian pada kasus ini memiliki masalah dalam hal penerapannya, oleh karena itu pada bab ini akan dilakukan analisis masalah dan analisis kerja pada sistem. Masalah dalam penerapan untuk membuat sebuah aplikasi untuk mengenali teks pada gambar disebabkan oleh faktor tertentu. Faktor-faktor itu adalah:

1. Jenis *font* teks yang beragam macam jenisnya dan memiliki warna yang sangat beragam.
2. Letak teks tidak *horizontal*, melainkan *vertical* atau memiliki *angle/sudut*.
3. Mendapatkan akurasi yang tinggi menggunakan *machine learning Support Vector Machine*.

Dengan adanya metode yang telah disebutkan pada bagian sebelumnya, permarsalaahan tersebut dapat dipecahkan. Dengan menggunakan metode *Texture-Based* maka aplikasi dapat mendekripsi beragam macam jenis font dan teks yang memiliki *angle*. Seperti yang sudah disebutkan sebelumnya, fungsi Transformasi Kosinus Diskrit (DCT) pada aplikasi adalah *background suppression/ menghilangkan hal yang tidak dibutuhkan (noise)*.

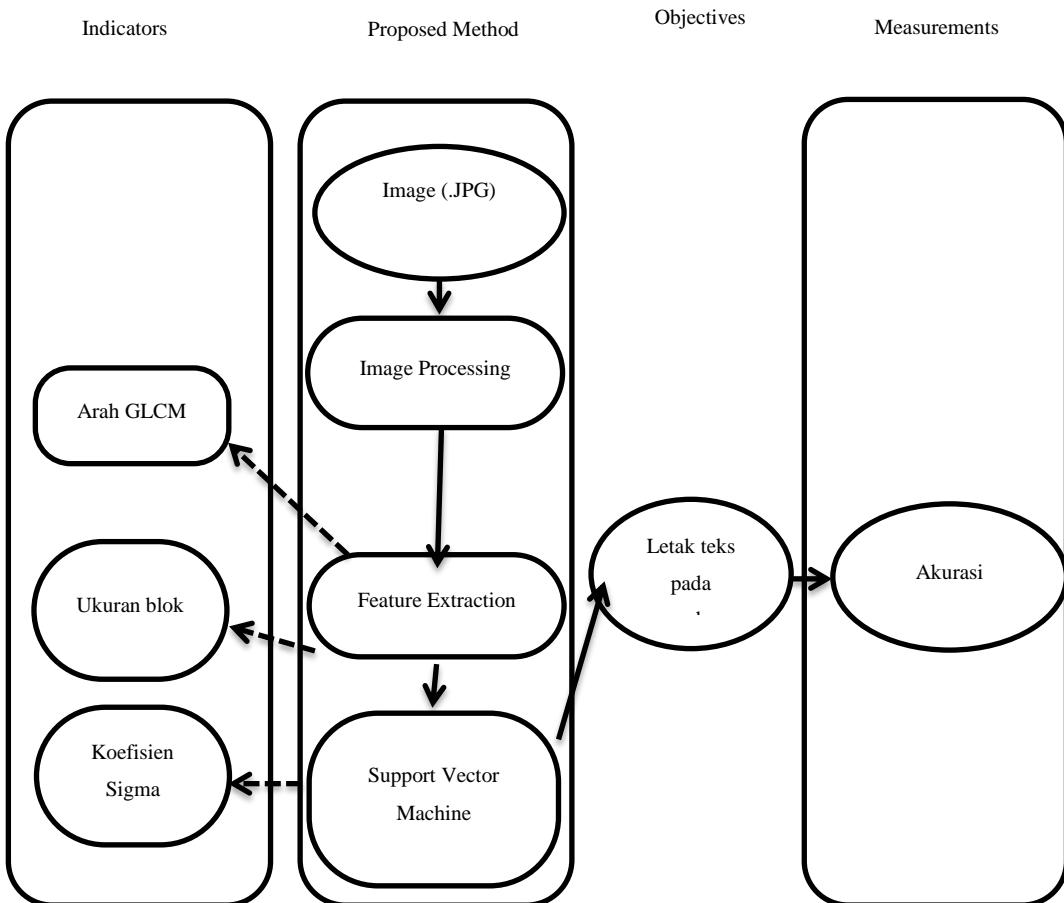


**Gambar 3.2 Contoh text vetical atau memiliki angle**

Gambar 3.2 adalah contoh dimana teks pada gambar tidak *horizontal* tetapi *vertical* dan teks tersebut berwarna hijau. Dengan metode *Texture-based*, teks seperti ini dapat dideteksi, dengan cara mengidentifikasi setiap kotak/*blok* yang ada, apakah kotak/*blok* tersebut mengandung unsur teks atau tidak.

Penelitian ini juga membutuhkan banyak data *training* sebagai data belajar untuk *machine learning*. Data *training* akan berupa gambar dimana letak teks sudah diketahui dan gambar dimana letak teks belum diketahui dengan tujuan *machine learning* dapat menentukan/mengklasifikasikan data yang mengandung teks dan tidak. Pada penelitian ini, penulis menggunakan algoritma *Support Vector Machine*, sebagai algoritma untuk *machine learning*.

### 3.2 Kerangka Kerja

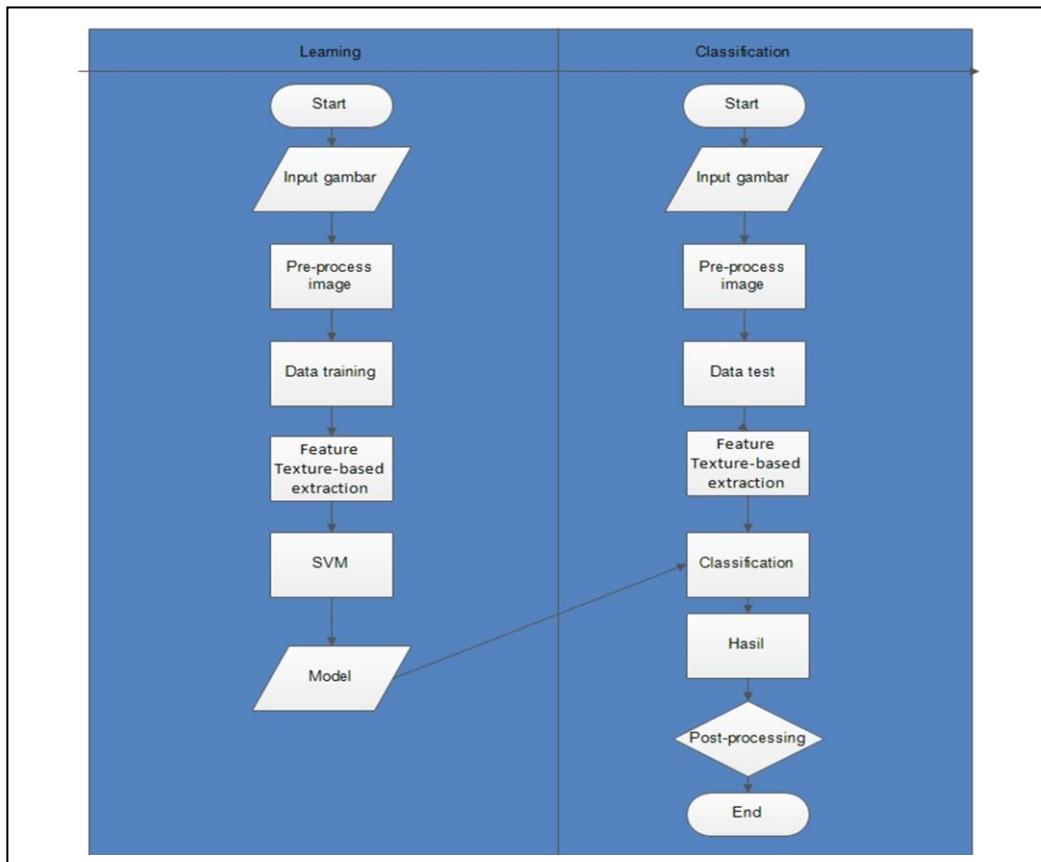


Gambar 3.3 Kerangka kerja aplikasi mendeteksi teks pada gambar

Pada gambar 3.3 dijelaskan bagaimana kerangka kerja pada aplikasi penulis. Aplikasi akan menerima input sebuah gambar, input akan di proses dengan serangkaian metode dengan memperhatikan faktor *Image Size, Data training* dan koefisien sigma. Setelah dilakukan serangkaian proses, maka aplikasi akan mencapai tujuannya, dengan cara hasil data yang sudah diproses akan diklasifikasikan menggunakan *machine learning* dengan algoritma *Support Vector Machine* yang akan dibandingkan dengan data *training* yang sudah ada.

### 3.3 Analisis Cara Kerja Sistem

Pada sub-bab ini akan dijelaskan mengenai cara kerja sistem dan alur proses pada sistem sehingga akan menghasilkan suatu gambar yang sudah diketahui letak teks berada.



**Gambar 3.4 Flow chart aplikasi deteksi teks pada gambar**

Flow chart dibagi menjadi 2 bagian yaitu, bagian *Learning* dan *Classification*. Ada beberapa bagian proses yang sama pada 2 bagian tersebut oleh karena itu, penulis akan menjelaskan pada 1 sub-bab saja.

Berikut adalah uraian proses flow chart Gambar 3.4 sistem yang dibuat dalam penelitian ini:

1. Tipe data gambar yang diinput dapat berupa JPG atau PNG.
2. Gambar akan diproses dengan serangkaian metode *image processing* pada proses *pre-process image*.

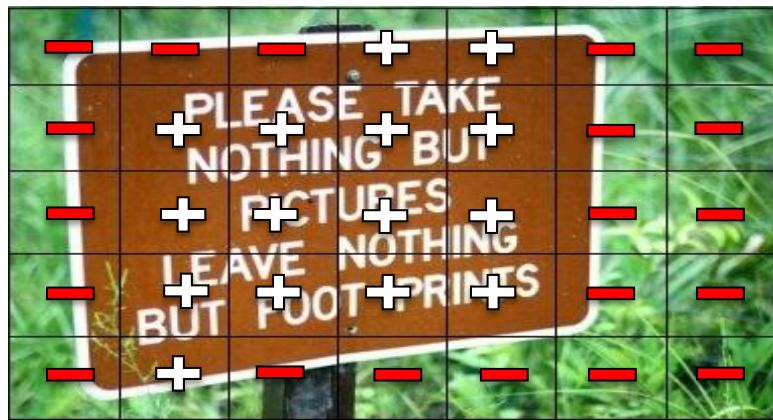
3. Pada bagian *learning*, gambar akan menjadi data *training*, sedangkan pada bagian *classification* gambar akan menjadi data test/data uji.
4. Gambar akan dibagi menjadi beberapa bagian berukuran 50x50 blok, lalu diterapkan *Texture features extraction*.
5. Pada bagian *learning*, gambar yang sudah diproses akan masuk *machine learning* yaitu *Support Vector Machine* yang berfungsi menciptakan model yang sudah diberi label dengan tujuan untuk membandingkan data test/data uji pada bagian *classification* dengan data *training* pada bagian *learning*.
6. Pada bagian *classification*, gambar yang sudah diproses sebelumnya akan dilakukan klasifikasi pada setiap bloknya, apakah data tersebut mengandung unsur teks atau tidak pada setiap blok yang ada dengan cara membandingkan dengan model pada bagian *learning*.
7. Gambar sudah teridentifikasi, apakah gambar tersebut mengandung unsur teks atau tidak.
8. Jika gambar masih belum teridentifikasi, maka akan dilakukan proses ulang dengan mengulang proses no.4, tetapi dengan syarat gambar akan dibagi menjadi beberapa bagian dengan ukuran blok yang lebih kecil dengan tujuan teks pada gambar dapat terdeteksi.

### 3.3.1 Data

Data input pada penelitian akan dibagi dua, yaitu data *training* dan data uji/test. Data *training* yang akan digunakan berupa blok-blok pada gambar yang sudah diberikan label.

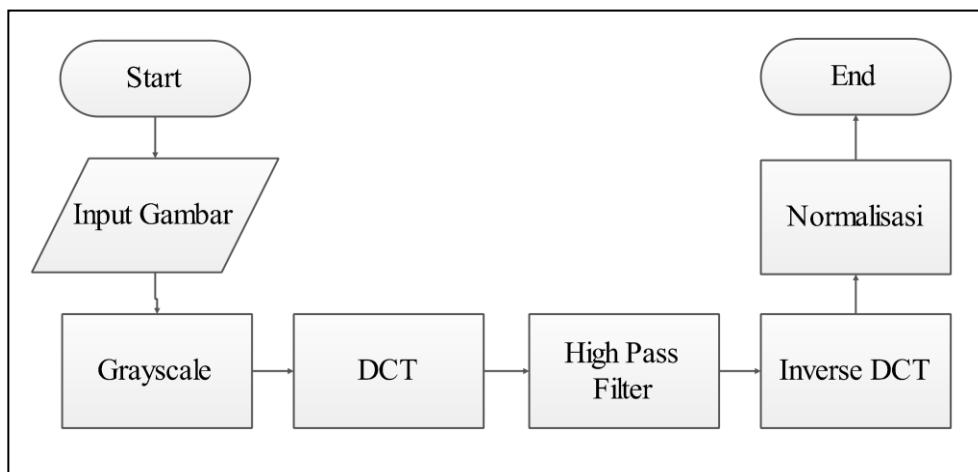
Pada gambar 3.5, gambar yang telah dibagi menjadi beberapa blok dengan ukuran 50x50, akan diidentifikasi pada setiap blok apakah blok tersebut mengandung unsur teks (+) atau tidak (-). Setiap blok pada gambar tersebut akan menjadi data *training* yang akan dijadikan input ke dalam proses *Support Vector Machine*. *Support Vector Machine* akan membentuk sebuah model dengan tujuan untuk dibandingkan dengan data uji/test yang akan diinput. Data uji/test juga akan

dilakukan hampir sama seperti data *training*, tetapi pada data uji/test tidak dilakukan identifikasi. Data uji/test akan dibandingkan dengan model yang sudah ada, sehingga proses *Support Vector Machine* yang akan menidentifikasi apakah blok tersebut mengandung unsur teks (+) atau tidak (-).



**Gambar 3.5 Contoh data training**

### 3.3.2 Image Processing



**Gambar 3.6 Flow chart proses image processing**

Urutan proses flow chart Gambar 3.6 sistem yang dibuat dalam penelitian ini:

1. Gambar input akan diubah menjadi citra *grayscale*, karena pada penelitian ini tidak memerlukan unsur warna RGB.
2. Gambar yang sudah menjadi citra *grayscale* akan dibagi menjadi beberapa bagian berbentuk kotak berukuran 8x8 blok dengan tujuan untuk

mempercepat perhitungan koefisien DCT. Penulis menggunakan DCT 2 dimensi, karena mengingat gambar memiliki panjang dan lebar, maka akan lebih mudah jika menggunakan rumus DCT 2 dimensi.

3. Setelah mendapatkan hasil matriks koefisien DCT tersebut, maka matriks koefisien DCT tersebut dikalikan dengan matriks *high pass filter* dengan tujuan mempertahankan (mempertajam) komponen frekuensi tinggi dan menghilangkan (mengurangi) komponen frekuensi rendah sehingga tepi citra dapat lebih terlihat.
4. Gambar akan dikembalikan menggunakan *inverse DCT* pada setiap 8x8 blok tersebut.

### 3.3.2.1 Grayscale

Berikut adalah contoh gambar yang akan diproses menggunakan *grayscale*:



**Gambar 3.7 Gambar yang akan di grayscale**

Gambar akan diubah menjadi *grayscale* dengan mengkonversi gambar menjadi abu-abu 8 bit dengan menggunakan *library Image-Plus*, setiap pixel pada gambar akan diubah menjadi 8-bit. Alasan penulis menggunakan *grayscale* 8-bit adalah pada penelitian ini menggunakan GLMC(Gray Level Co-occurrences Matrices), oleh karena itu, gambar diubah menjadi *grayscale* 8-bit.

Hasil dari proses *grayscale* sebagai berikut.



**Gambar 3.8 Gambar grayscale**

### 3.3.2.2 DCT

Selanjutnya gambar yang sudah di *grayscale* akan dibagi menjadi beberapa bagian berbentuk kotak berukuran 8x8 blok untuk mempercepat proses DCT. Penulis menggunakan rumus DCT 2-dimensi pada penelitian ini. Pada penelitian ini, penulis menggunakan persamaan 2.6

107	246	141	19	44	74	215	49
14	147	50	209	100	75	99	233
52	243	199	16	68	91	102	254
69	52	72	228	244	39	222	41
52	179	131	101	35	53	187	133
177	100	174	38	79	16	239	195
13	95	232	41	90	246	38	110
145	124	5	134	172	152	142	183



953	-83	80	-122	-92	-44	-97	32
-15	41	28	-84	-42	-19	-115	-23
2	9	28	74	-27	-72	53	-78
-37	62	88	-14	-123	86	3	75
-6	29	-50	115	22	93	-159	12
-12	48	-99	60	-104	1	71	88
49	44	193	-0	-33	-7	-28	97
-27	48	69	-66	-180	-124	108	-51

**Gambar 3.9 Matriks koefisien DCT**

Dengan menggunakan persamaan 2.6 maka didapatkan hasil koefisien DCT pada gambar 3.6. Berikut contoh perhitungan :

**Tahap 1:**

$$C(0,0), f(0,0) = \cos \left[ \frac{\pi(2.0 + 1).0}{2.8} \right] \cos \left[ \frac{\pi(2.0 + 1).0}{2.8} \right] . matriks[0][0]$$

$$C(0,0), f(0,0) = 1 \times 1 \times 107$$

$$C(0,0), f(0,0) = 107$$

**Tahap 2:**

$$C(0,0)f(0,1) = \cos \left[ \frac{\pi(2.0 + 1).0}{2.8} \right] \cos \left[ \frac{\pi(2.1 + 1).0}{2.8} \right] . matriks[0][1]$$

$$C(0,0)f(0,1) = 1 \times 1 \times 246$$

$$C(0,0)f(0,1) = 246$$

Hasil dari tahap 1,2, sampai akhir akan dijumlahkan lalu dikalikan ke dalam persamaan 2.6 dengan syarat persamaan 2.7:

$$C(0,0) = \frac{\sqrt{2}}{\sqrt{8} \times 8} \frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} \times total$$

$$C(0,0) = 953$$

Persamaan di atas menjelaskan bahwa koefisien DCT pada kordinat matriks(0,0) adalah 953. Perhitungan tersebut akan diulang sampai semua matriks pixel berubah menjadi koefisien DCT.

### 3.3.3.3 High pass filter

Setelah digunakan mendapatkan koefisien DCT, maka matriks tersebut akan dikalikan dengan matrik *high pass filter* dengan koefesien:

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

Gambar 3.10 Kernel high pass filter [ANG10]



0	-83	80	-122	-92	-44	-97	32
-15	0	28	-84	-42	-19	-115	-23
2	9	0	74	-27	-72	53	-78
-37	62	88	-0	-123	86	3	75
-6	29	-50	115	0	93	-159	12
-12	48	-99	60	-104	0	71	88
49	44	193	-0	-33	-7	-0	97
-27	48	69	-66	-180	-124	108	-0

**Gambar 3.11 Hasil perkalian Kernel high pass filter**

Pada tahap ini, perkalian dilakukan seperti perkalian matriks yaitu seperti:

$$\text{Matriks}[0][0] = \text{koefisienDCT}[0][0] * \text{matriksHighPassFilter}[0][0]$$

Perhitungan tersebut akan diulang sampai perhitungan matriks koefiesien terakhir.

### 3.3.3.4 Invers DCT

Setelah koefisien dikalikan dengan matriks *high pass filter*, maka akan diterapkan metode *inverse DCT* untuk mengembalikan gambar pada setiap blok yang ada. Pada penelitian ini, penulis menggunakan persamaan 2.8:

-27	114	23	-104	-67	-31	104	-71
-118	27	-87	104	-17	-42	-15	122
-66	106	91	-115	-44	-35	-15	149
-54	-53	-59	114	104	-73	105	-70
-59	62	19	-39	-79	-78	82	10
72	-17	48	-74	-52	-92	102	77
-98	-19	115	-76	-15	109	-82	-22
25	13	-100	23	49	34	10	49

**Gambar 3.12 Matriks invers DCT**

Dengan menggunakan persamaan 2.8 dan syarat dengan persamaan 2.7 maka didapatkan hasil pixel pada gambar 3.9. Berikut contoh perhitungan:

**Tahap 1:**

$$f(0,0)C(0,0) = \cos\left[\frac{\pi(2.0 + 1).0}{2.8}\right] \cos\left[\frac{\pi(2.0 + 1).0}{2.8}\right] \cdot \frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} \cdot matriks[0][0]$$

$$C(0,0), f(0,0) = 1 \times 1 \times \frac{1}{\sqrt{2}} \times 0$$

$$C(0,0), f(0,0) = 0$$

**Tahap 2:**

$$f(0,0)C(0,1) = \cos\left[\frac{\pi(2.0 + 1).0}{2.8}\right] \cos\left[\frac{\pi(2.0 + 1).1}{2.8}\right] \cdot \frac{1}{\sqrt{2}} \cdot 1 \cdot matriks[0][1]$$

$$C(0,0), f(0,1) = 1 \times 0.98 \times \frac{1}{\sqrt{2}} \times 1 \times (-83)$$

$$C(0,0), f(0,1) = -14.4$$

Hasil perhitungan dari tahap 1,2, sampai akhir akan dijumlahkan lalu total tersebut akan dimasukkan ke matriks hasil invers dct pada koordinat (0,0).

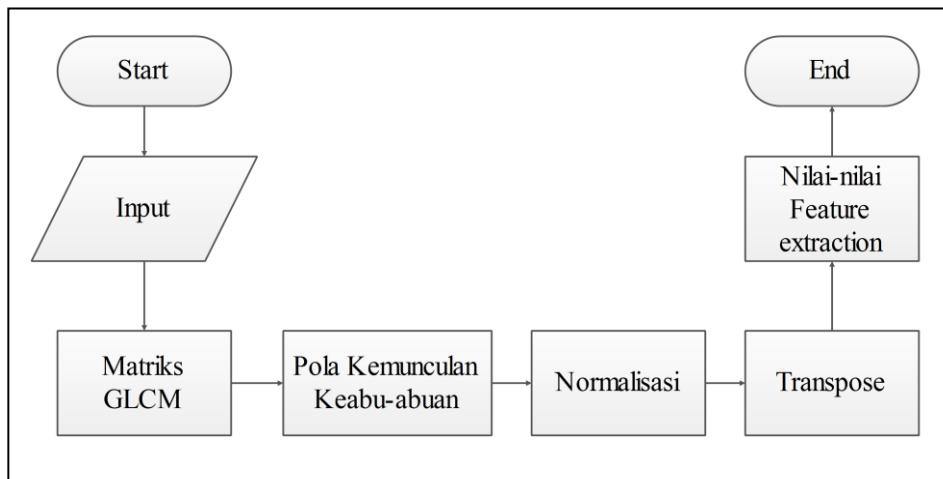
### 3.3.3.5 Normalisasi

Sebenarnya pixel pada gambar memiliki range antar 0 – 255, tetapi karena proses *high pass filter* membuat nilai pixel tersebut menjadi lebih kecil dari 0 atau masih tetap diantara 0 – 255 atau lebih besar dari pada 255. Pada tahap ini, penulis akan mengubah pixel yang lebih kecil 0 menjadi 0 dan yang lebih besar 255 akan menjadi 255 dengan proses normalisasi.

0	114	23	0	0	0	104	0
0	27	0	104	0	0	0	122
0	106	91	0	0	0	0	149
0	0	0	114	104	0	105	0
0	62	19	0	0	0	82	10
72	0	48	0	0	0	102	77
0	0	115	0	0	109	0	0
25	13	0	23	49	34	10	49

**Gambar 3.13 Hasil dari proses normalisasi**

### 3.3.3 Feature Texture-based extraction



**Gambar 3.14 Flow chart proses Texture-based extract**

Urutan proses flow chart Gambar 3.14 sistem yang dibuat dalam penelitian ini:

1. Gambar yang sudah diproses, akan menjadi input dalam flow chart ini. Gambar akan dibagi menjadi beberapa bagian dengan ukuran 50x50 blok.
2. Setiap blok akan diubah kembali menjadi matrik pixel. Lalu blok tersebut akan diubah menjadi matriks keabuan-abuan.
3. Matrik keabu-abuan akan diubah menjadi pola kemunculan keabu-abuan.
4. Matrik akan dilakukan normalisasi sehingga dapat dilakukan perhitungan Kontras, Homogenitas, Energi, Entropi, Korelasi.

- Setelah dilakukan normalisasi, matriks akan ditranspose yang berfungsi mengubah baris pada matriks tersebut menjadi kolom dan mengubah kolom menjadi baris.

Pada tahap awal, blok input akan diubah kembali menjadi matriks pixel. Penulis akan mengambil contoh pada gambar 3.15.

### 3.3.3.1 GLCM

Setelah blok diubah menjadi matriks pixel, maka matriks akan dikuantisasi 0-255, sehingga dapat terlihat jumlah pada setiap nilai.

100	180	201	55	72
203	108	200	145	232
32	188	110	190	189
30	150	79	70	119
167	125	201	132	60

Gambar 3.15 Matrik Pixel

<b>i</b>	<b>0</b>	<b>1</b>	<b>3</b>	...	<b>255</b>
<b>0</b>	1	2	3	...	1
<b>1</b>	3	1	3	...	3
<b>3</b>	2	3	1	...	2
...	...	...	...	...	...
<b>255</b>	1	3	2	...	1

Gambar 3.16 Matrik GLCM

### 3.3.3.2 Matrik Kemunculan Pola

Setelah matriks pixel dikuantisasi menjadi 0-255 level keabu-abuan, maka proses selanjutnya adalah matriks GLCM akan dibentuk menjadi matriks

kemunculan pola. Pada penelitian ini, penulis akan melakukan uji coba dengan mencoba semua jarak dan arah, tetapi pada perhitungan untuk bab ini penulis menggunakan perhitungan matriks GLCM memakai jarak satu piksel ( $d=1$ ) dan arah  $0^\circ$ . Arah  $0^\circ$  berarti melihat tetangganya ke sebelah kanan seperti yang dijelaskan pada gambar 2.x dengan jarak 1 pixel. Contoh pada gambar 3.17 dapat dilihat untuk mempermudah visualisasi.

Pada matriks kemunculan pola, pada koordinat (0,0) didapatkan dengan cara melihat a dan b pada matriks tersebut. Pada matriks kemunculan pola jika,  $a=0$  dan  $b=0$  yang berarti pada gambar 3.17 dicari jumlah yang memiliki  $0 \rightarrow 0=0$ . Jika  $a=0$  dan  $b=1$  maka pada gambar 3.17 dicari jumlah yang memiliki  $0 \rightarrow 1=1$ . Seterusnya akan dilakukan sampai akhir. Hasil dapat dilihat pada gambar 3.18.

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

**Gambar 3.17 Proses perhitungan Matrik Kemunculan Pola**

a/b	0	1	2	3
0	0	1	2	0
1	0	2	2	2
2	1	3	1	2
3	1	1	2	0

**Gambar 3.18 Matrik Kemunculan pola.**

### 3.3.3.3 Matrik Normalisasi

Setelah didapatkan matrik Kemunculan pola, matrik tersebut akan dinormalisasi dengan contoh sebagai berikut:

a/b	0	1	2	3
0	0	1	2	0
1	0	2	2	2
2	1	3	1	2
3	1	1	2	0

=

**Gambar 3.15 Matrik Kemunculan pola**

Jika total nilai dalam matriks kemunculan pola dijumlahkan maka akan mendapatkan nilai = 20. Maka setiap matrik kemunculan pola akan dilakukan pembagian dengan total, yaitu 20.

a/b	0	1	2	3
0	0.0	0.05	0.1	0.0
1	0.0	0.1	0.1	0.1
2	0.05	0.15	0.05	0.1
3	0.05	0.05	0.1	0.0

**Gambar 3.16 Matriks Normalisasi**

### 3.3.3.4 Matrik Transpose

Setelah mendapatkan matriks normalisasi, maka matriks akan ditranspose dengan cara mengubah bagian baris menjadi kolom dan kolom menjadi baris.

a/b	0	1	2	3
0	0.0	0.0	0.05	0.05
1	0.05	0.1	0.15	0.05
2	0.1	0.1	0.05	0.1
3	0.0	0.1	0.1	0.0

**Gambar 3.17 Matriks Transpose**

Nilai-nilai pada matrik transpose yang kemudian dimasukkan ke dalam rumus-rumus untuk mencari Kontras, Homogenitas, Energi, Entropi, Korelasi.

**Kontrast:**

a=0 b=0

Kontrast=0.0

a=0 b=1

Kontrast=0.05

a=0 b=2

Kontrast=0.45

a=0 b=3

Kontrast=0.45

a=1 b=0

Kontrast=0.45

a=1 b=1

Kontrast=0.45

a=1 b=2

Kontrast=0.55

a=1 b=3

Kontrast=0.9500000000000000

a=2 b=0

Kontrast=1.1500000000000000

a=2 b=1

Kontrast=1.3

a=2 b=2

Kontrast=1.3

a=2 b=3

Kontrast=1.4000000000000000

a=3 b=0

Kontrast=1.85

a=3 b=1

Kontrast=2.0500000000000000

a=3 b=2

Kontrast=2.1500000000000000

a=3 b=3

Kontrast=**2.1500000000000000** Perhitungan Kontrast dengan menggunakan persamaan 2.9. Dengan memasukan nilai a dan b ke dalam rumus dan matrik normalisasi. Dari iterasi di atas, didapatkan nilai kontrast dari blok tersebut adalah sebesar **2.1500000000000000**.

**Homogenitas:**

a=0 b=0

a=0 b=2

Homogenitas=0.0

Homogenitas=0.045

a=0 b=1

a=0 b=3

Homogenitas=0.025

Homogenitas=0.045

		Homogenitas=0.35
a=1 b=0		a=2 b=3
Homogenitas=0.045		Homogenitas=0.3999999999999997
a=1 b=1		
Homogenitas=0.1450000000000000	a=3 b=0	
2		Homogenitas=0.4049999999999999
a=1 b=2	7	
Homogenitas=0.195	a=3 b=1	
a=1 b=3		Homogenitas=0.415
Homogenitas=0.215	a=3 b=2	
a=2 b=0		Homogenitas=0.4649999999999999
Homogenitas=0.225	7	
a=2 b=1	a=3 b=3	
Homogenitas=0.3		Homogenitas= <b>0.4649999999999997</b>
a=2 b=2		

Perhitungan Homogenitas dengan menggunakan persamaan 2.10. Dengan memasukan nilai a dan b ke dalam rumus dan matrik normalisasi. Dari iterasi di atas, didapatkan nilai homogenitas dari blok tersebut adalah sebesar **0.4649999999999997**.

### Energi:

a=0 b=0	a=1 b=0
Energi=0.0	Energi=0.012500000000000002
a=0 b=1	a=1 b=1
Energi=0.002500000000000005	Energi=0.022500000000000006
a=0 b=2	a=1 b=2
Energi=0.012500000000000002	Energi=0.03250000000000001
a=0 b=3	a=1 b=3
Energi=0.012500000000000002	Energi=0.04250000000000001

a=2 b=0	a=3 b=0
Energi=0.04500000000000001	Energi=0.08250000000000002
a=2 b=1	a=3 b=1
Energi=0.0675	Energi=0.08500000000000002
a=2 b=2	a=3 b=2
Energi=0.07	Energi=0.09500000000000003
a=2 b=3	a=3 b=3
Energi=0.08000000000000002	Energi= <b>0.09500000000000003</b>

Perhitungan energi dengan menggunakan persamaan 2.11. Dengan memasukan nilai a dan b ke dalam rumus dan matrik normalisasi. Dari iterasi di atas, didapatkan nilai energi dari blok tersebut adalah sebesar **0.09500000000000003**.

### Entropi:

a=0 b=1	a=2 b=1
Entropi=-0.21609640474436814	Entropi=-2.1715088865686125
a=0 b=2	a=2 b=2
Entropi=-0.5482892142331044	Entropi=-2.3876052913129806
a=1 b=1	a=2 b=3
Entropi=-0.8804820237218407	Entropi=-2.7197981008017167
a=1 b=2	a=3 b=0
Entropi=-1.212674833210577	Entropi=-2.9358945055460848
a=1 b=3	a=3 b=1
Entropi=-1.5448676426993133	Entropi=-3.151990910290453
a=2 b=0	a=3 b=2
Entropi=-1.7609640474436814	Entropi= <b>-3.484183719779189</b>

Perhitungan energi dengan menggunakan persamaan 2.12. Dengan memasukan nilai a dan b ke dalam rumus dan matrik normalisasi. Dari iterasi di atas, didapatkan nilai energi dari blok tersebut adalah sebesar **-3.484183719779189**.

**Korelasi:**Pada proses kolerasi, diperlukan *mean* dan *deviasi* seperti telah dijelaskan pada bab 2, oleh karena itu pertama-tama dalam proses kolerasi adalah mencari *mean* dan *deviasi* terlebih dahulu.

**$\mu_x$ :**

$$\mu_x = \mathbf{5.05}$$

Perhitungan  $\mu_x$  dengan menggunakan persamaan 2.14.

**$\mu_y$ :**

$$\mu_x = \mathbf{5.05000000000001}$$

Perhitungan  $\mu_x$  dengan menggunakan persamaan 2.15.

**$\sigma_x$  :**

$$\sigma_x = \mathbf{29.69999999999996}$$

Perhitungan  $\mu_x$  dengan menggunakan persamaan 2.16.

**$\sigma_y$  :**

$$\sigma_y = \mathbf{21.57587500000014}$$

Perhitungan  $\mu_x$  dengan menggunakan persamaan 2.17.

**Korelasi:**

$$a=0 \ b=0$$

$$a=1 \ b=0$$

$$\text{Kolerasi}=0.0$$

$$\text{Kolerasi}=0.0039994710546889765$$

$$a=0 \ b=1$$

$$a=1 \ b=1$$

$$\text{Kolerasi}=0.0015958480563044684$$

$$\text{Kolerasi}=0.006559148135098124$$

$$a=0 \ b=2$$

$$a=1 \ b=2$$

$$\text{Kolerasi}=0.0039994710546889765$$

$$\text{Kolerasi}=0.008486806183307483$$

$$a=0 \ b=3$$

$$a=1 \ b=3$$

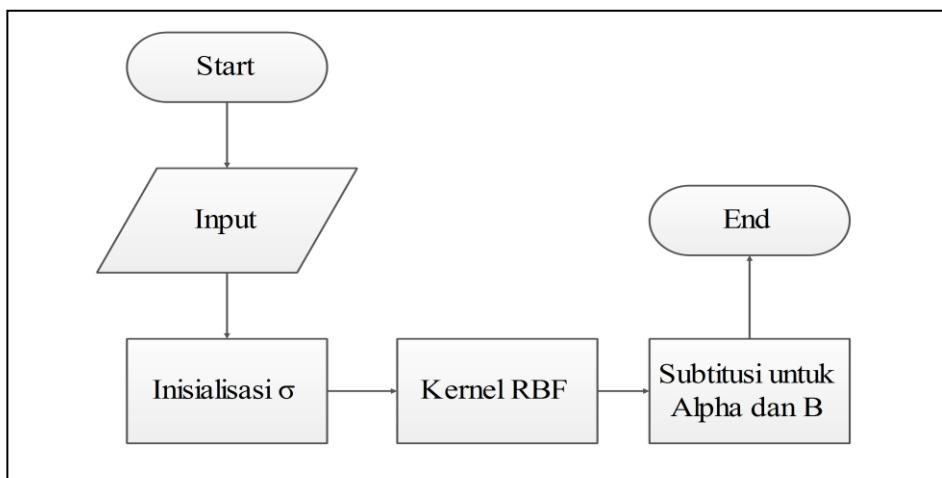
$$\text{Kolerasi}=0.0039994710546889765$$

$$\text{Kolerasi}=0.009782445199317051$$

$a=2 b=0$	$a=3 b=0$
Kolerasi=0.010984256698509305	Kolerasi=0.016385093409779544
$a=2 b=1$	$a=3 b=1$
Kolerasi=0.013875743770823342	Kolerasi=0.01703291291778433
$a=2 b=2$	$a=3 b=2$
Kolerasi=0.014601590319840447	Kolerasi=0.018008641065643385
$a=2 b=3$	$a=3 b=3$
Kolerasi=0.015577318467699504	Kolerasi= <b>0.018008641065643385</b>

Perhitungan energi dengan menggunakan persamaan 2.13. Dengan memasukan nilai  $a, b, mean$ , dan *deviasi* yang sudah didapatkan ke dalam rumus dan matrik normalisasi. Dari iterasi di atas, didapatkan nilai energi dari blok tersebut adalah sebesar **0.018008641065643385**.

### 3.3.4 Support Vector Machine



**Gambar 3.18 Flow chart proses Support Vector Machine**

Urutan proses flow chart Gambar 3.21 sistem yang dibuat dalam penelitian ini:

1. Nilai-nilai yang sudah diproses pada proses *Feature texture based* akan menjadi dimensi pada *Support Vector Machine*.
2. Inisialisasi  $\sigma$  pada awal untuk mencari alpha dan b.

3. Masukan ke dalam rumus kernel, pada penelitian ini penulis menggunakan kernel RBF pada persamaan 2.23, penulis menggunakan kernel RBF karena kernel tersebut hanya menggunakan 1 parameter saja yaitu  $\sigma$ , dibandingkan dengan kernel yang lain.
4. Lakukan substitusi sehingga mendapatkan nilai Alpha dan Bias.

Pada proses sebelumnya telah didapatkan hasil kontras, homogenitas, energi, entropi, dan korelasi pada setiap blok yang sudah dibagi dengan ukuran 50x50. Nilai – nilai tersebut akan digunakan sebagai dimensi pada vector *Support Vector Machine*.

Pada table 3.1 jika klasifikasi bernilai +1 maka, data tersebut mengandung unsur teks, tetapi jika klasifikasi bernilai -1 maka, data tersebut tidak mengandung unsur teks.

Setelah data training terkumpul, proses selanjutnya adalah membuat model dengan mencari nilai alpha dan b. Penulis menginisialisasi  $\sigma = 0,1$ , lalu setiap data *training* yang ada akan dihitung menggunakan kernel RBF dengan persamaan 2.23.

**Tabel 3.1 Data training**

Data	Kontras <b>t</b>	Homogenita <b>s</b>	Energ <b>i</b>	Entrop <b>i</b>	Korelas <b>i</b>	Klasifikas <b>i</b>
<b>D1</b>	0.1	0.5	0.3	0.6	0.2	+1
<b>D2</b>	0.2	0.3	0.2	0.8	0.3	+1
<b>D3</b>	0.3	0.2	0.1	0.5	0.3	+1
<b>D4</b>	0.4	0.8	0.2	0.2	0.2	-1
<b>D5</b>	0.5	0.1	0.2	0.1	0.8	-1

$$\begin{aligned}
 K(D1, D1) &= \exp(-\|D1 - D1\|^2 / (2\sigma^2)) \\
 &= \exp(-50 (\|0.1-0.1\|^2 + \|0.5-0.5\|^2 + \|0.3-0.3\|^2 + \|0.6-0.6\|^2 + \\
 &\quad \|0.2-0.2\|^2)) \\
 &= 1 \\
 K(D1, D2) &= \exp(-\|D1 - D2\|^2 / (2\sigma^2))
 \end{aligned}$$

$$\begin{aligned}
&= \exp (-50 (\|0.1-0.2\|^2 + \|0.5-0.3\|^2 + \|0.3-0.2\|^2 + \|0.6-0.5\|^2 + \\
&\quad \|0.2-0.3\|^2)) \\
&= 0.00408 \\
K(D1, D3) &= \exp (-\|D1 - D3\|^2 / (2\sigma^2)) \\
&= \exp (-50 (\|0.1-0.3\|^2 + \|0.5-0.2\|^2 + \|0.3-0.1\|^2 + \|0.6-0.5\|^2 + \\
&\quad \|0.2-0.3\|^2)) \\
&= 7.485 \\
K(D1, D4) &= \exp (-\|D1 - D4\|^2 / (2\sigma^2)) \\
&= \exp (-50 (\|0.1-0.4\|^2 + \|0.5-0.8\|^2 + \|0.3-0.2\|^2 + \|0.6-0.2\|^2 + \\
&\quad \|0.2-0.2\|^2)) \\
&= 2.510 \\
K(D1, D5) &= \exp (-\|D1 - D5\|^2 / (2\sigma^2)) \\
&= \exp (-50 (\|0.1-0.5\|^2 + \|0.5-0.1\|^2 + \|0.3-0.2\|^2 + \|0.6-0.1\|^2 + \\
&\quad \|0.2-0.8\|^2)) \\
&= 3.8739
\end{aligned}$$

Proses tersebut akan dilakukan sampai data *training* terakhir, sehingga akan menghasilkan Table 3.2.

**Tabel 3.2 Hasil kernel RBF**

Data	D1	D2	D3	D4	D5
<b>D1</b>	1	0.00408	7.485	2.510	3.8739
<b>D2</b>	0.00408	1	0.00247	4.658	1.282
<b>D3</b>	7.485	0.00247	1	3.775	6.224
<b>D4</b>	2.510	4.658	3.775	1	1.282
<b>D5</b>	3.873	1.282	6.224	1.282	1

Setelah mendapatkan hasil kernel RBF, proses selanjutnya adalah mencari alpha dan bias (*b*) dari masing-masing data *training*. Persamaan akan dibentuk sesuai dengan nilai klasifikasi pada persamaan 2.24, jika nilai klasifikasi +1 maka akan menggunakan persamaan 2.25 dan jika nilai klasifikasi -1 maka akan menggunakan persamaan 2.26.

Berikut proses substitusi hasil kernel RBF:

$$(1) \alpha_1 + (1) \alpha_2 + (1) \alpha_3 + (-1) \alpha_4 + (-1) \alpha_5 + (0)b = 0$$

$$(1) \alpha_1 + (0.00408) \alpha_2 + (7.485) \alpha_3 + (2.510) \alpha_4 + (3.8739) \alpha_5 + b = 1$$

$$(0.00408) \alpha_1 + (1) \alpha_2 + (0.00247) \alpha_3 + (4.658) \alpha_4 + (1.282) \alpha_5 + b = 1$$

$$(7.485) \alpha_1 + (0.00247) \alpha_2 + (1) \alpha_3 + (3.775) \alpha_4 + (6.224) \alpha_5 + b = 1$$

$$(2.510) \alpha_1 + (4.658) \alpha_2 + (3.775) \alpha_3 + (-1) \alpha_4 + (1.282) \alpha_5 + b = -1$$

$$(3.873) \alpha_1 + (1.282) \alpha_2 + (6.224) \alpha_3 + (1.282) \alpha_4 + (-1) \alpha_5 + b = -1$$

Dilakukan substitusi sehingga mendapatkan alpha 1-5 dan bias(b), sebagai berikut:

**Tabel 3.3 Hasil alpha dan bias**

$\alpha_1$	$\alpha_2$	$\alpha_3$	$\alpha_4$	$\alpha_5$	b
0.8176687	1.9149722	1.60687	4.6790448	-0.3395339	22.2819856

Setelah mendapatkan hasil alpha dan b maka SVM sudah membentuk sebuah model yang akan dibandingkan untuk proses klaisifikasi. Sebagai contoh, penulis mencoba dengan membandingkan dengan data test/uji.

**Tabel 3.4 Contoh Data**

Data	Kontras t	Homogenit as	Energ i	Entrop i	Korela si	Klasifika si
<b>TU1</b>	0.2	0.5	0.7	0.2	0.1	?

Data uji yang telah memiliki nilai kontras, homogenitas, energi, entropi, dan korelasi akan dibandingkan dengan model yang sudah jadi.

$$\begin{aligned} K(TU1, D1) &= \exp(-\|TU1 - D1\|^2 / (2\sigma^2)) \\ &= \exp(-50 (\|0.2-0.1\|^2 + \|0.5-0.5\|^2 + \|0.7-0.3\|^2 + \|0.2-0.6\|^2 + \|0.1-0.2\|^2)) \\ &= 1.670 \end{aligned}$$

$$\begin{aligned} K(TU1, D2) &= \exp(-\|TU1 - D2\|^2 / (2\sigma^2)) \\ &= \exp(-50 (\|0.2-0.2\|^2 + \|0.5-0.3\|^2 + \|0.7-0.2\|^2 + \|0.2-0.8\|^2 + \|0.1-0.3\|^2)) \end{aligned}$$

$$\begin{aligned}
&= 1.039 \\
K(TU1, D3) &= \exp(-\|TU1 - D3\|^2 / (2\sigma^2)) \\
&= \exp(-50 (\|0.2-0.3\|^2 + \|0.5-0.2\|^2 + \|0.7-0.1\|^2 + \|0.2-0.5\|^2 + \|0.1-0.3\|^2)) \\
&= 2.543 \\
K(TU1, D4) &= \exp(-\|TU1 - D4\|^2 / (2\sigma^2)) \\
&= \exp(-50 (\|0.2-0.4\|^2 + \|0.5-0.8\|^2 + \|0.7-0.2\|^2 + \|0.2-0.2\|^2 + \|0.1-0.2\|^2)) \\
&= 5.7495 \\
K(TU1, D5) &= \exp(-\|TU1 - D5\|^2 / (2\sigma^2)) \\
&= \exp(-50 (\|0.2-0.5\|^2 + \|0.5-0.1\|^2 + \|0.7-0.2\|^2 + \|0.2-0.1\|^2 + \|0.1-0.8\|^2)) \\
&= 1.928
\end{aligned}$$

Perhitungan tersebut akan mendapatkan hasil kernel RBF sebagai berikut:

**Tabel 3.5 Hasil kernel RBF**

	D1	D2	D3	D4	D5
TU1	1.670	1.039	2.543	5.7495	1.928

Setelah mendapatkan hasil kernel, proses selanjutnya adalah mengklasifikasi dengan menggunakan persamaan 2.27.

$$\begin{aligned}
(x) &= \text{sign} ((0.8176687 * 1 * 1.670) + (1.9149722 * 1 * 1.039) + (1.60687 * 1 * 2.543) + (4.6790448 * (-1) * 5.7495) + (-0.3395339 * (-1) * 1.928) + -22.2819856) \\
&= -1
\end{aligned}$$

Berdasarkan perhitungan di atas, TU1/ data test uji 1 yang dibandingkan dengan model mendapatkan nilai -1, oleh karena itu TU1 dapat diklasifikasikan kelas -1 atau gambar yang tidak mengandung unsur teks.

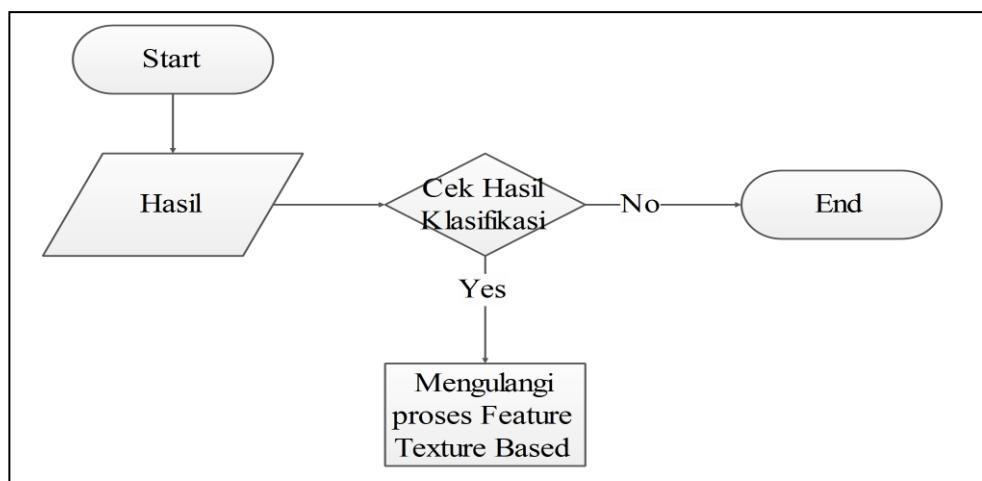
Pada penelitian ini, penulis menggunakan 3 model (MultiClass), dari 3 model tersebut dilakukan sistem voting untuk menentukan unsur teks pada setiap blok. Contoh menggunakan sistem voting:



**Gambar 3.19 Contoh Sistem Voting**

Pada contoh tersebut misalnya model ke-1 menyebutkan bahwa blok tersebut mengandung unsur teks (+1), model ke-2 menyebutkan bahwa blok tersebut mengandung unsur teks (+1), dan model ke-3 menyebutkan bahwa blok tersebut tidak mengandung unsur teks (-1). Maka aplikasi akan melakukan voting dengan jumlah terbanyak yaitu mengandung unsur teks (+1), oleh karena itu aplikasi akan menentukan bahwa blok tersebut mengandung unsur teks.

### 3.3.5 Post-processing



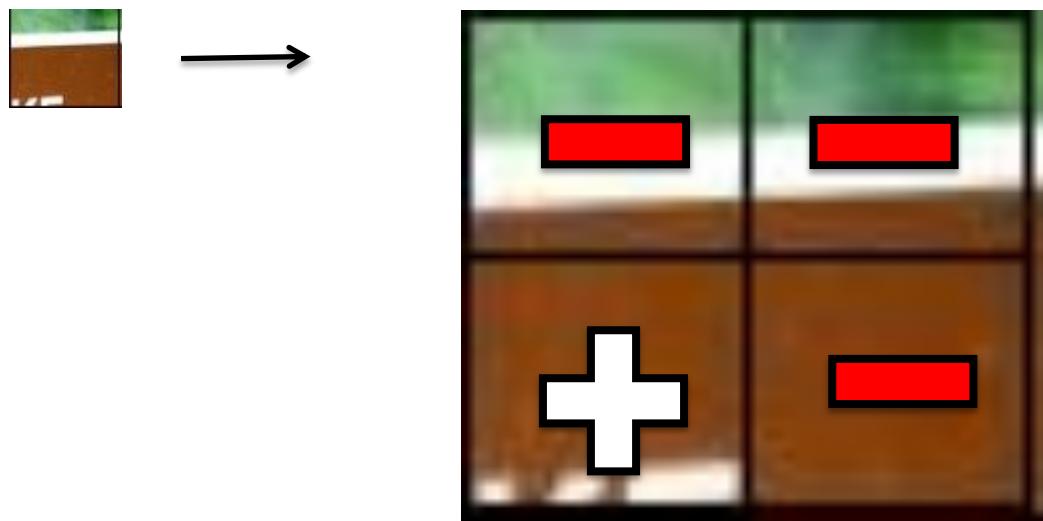
**Gambar 3.20 Flow chart proses Post-Processing**

Hasil/gambar akan dilakukan pengecekan ulang, proses akan dilakukan dari *Feature Texture-based extraction*. Berikut adalah contoh hasil/gambar yang mencurigakan



**Gambar 3.21 Hasil gambar yang mencurigakan**

Pada contoh di atas, pada penglihatan manusia sebenarnya blok tersebut mengandung unsur teks tetapi hanya sedikit saja, tetapi pada *machine learning* tidak mengklasifikasikan blok tersebut sebagai blok yang mengandung unsur teks atau *machine learning* mengalami kesalahan dalam mengenali unsur teks yang ada, oleh sebab itu di lakukan proses ulang sehingga blok tersebut dapat diklasifikasikan dengan benar.



**Gambar 3.22 Proses pengulangan dengan syarat blok yang lebih kecil**



**Gambar 3.23 Hasil akhir**

Blok yang melakukan proses ulang akan diperbesar (*scaling*) lalu akan dibagi menjadi 4 dan setiap bagian akan dibagi lagi menjadi beberapa bagian berbentuk blok dengan ukurang yang lebih kecil dari pada sebelumnya, sehingga unsur teks pada blok tersebut dapat terdeteksi dengan baik.

## **BAB IV**

### **IMPLEMENTASI DAN PENGUJIAN**

Pada bab ini menjelaskan mengenai bagaimana aplikasi diimplementasikan pada hardware yang diperlukan, software yang dipergunakan dan berisi mengenai penjelasan aplikasi ini.

#### **4.1 Lingkungan Implementasi**

Dalam aplikasi tersebut terbagi ke dalam dua bagian lingkungan implementasi, yaitu lingkungan implementasi perangkat keras dan lingkungan implementasi perangkat lunak. Di dalam pembuatan aplikasi penelitian ini, perangkat keras yang digunakan ialah:

1. Notebook ASUS dengan Model K43SV.
2. Processor Intel ® Core™ i5 – 2410M dengan kecepatan 2.3 GHz.
3. RAM 6GB.
4. VGA 2GB.

Spesifikasi perangkat lunak yang diperlukan untuk mengimplementasikan aplikasi terdiri dari:

1. Sistem Operasi Windows 8.1 64-bit.
2. Tools Pengembangan: Java Development Kit 1.7.0 32-bit, Netbeans IDE 7.3.
3. *Library*: JAMA 1.0.3

#### **4.2 Implementasi Perangkat Lunak**

Pada bagian ini akan dijelaskan mengenai implementasi aplikasi mendeteksi teks pada gambar, method pemrosesan, alur pemrosesan dan cara penggunaan aplikasi.

**Tabel 4.1 Objek Classification**

<b>Attribute:</b>					
<b>No</b>	<b>Nama Method</b>	<b>Input</b>		<b>Output</b>	<b>Keterangan</b>
		<b>Tipe</b>	<b>Atribut</b>		
1.	Classification()	-.	-.	-.	Constructor kosong
2.	Classification()	double	Contras, homogenitas , energi, entropi, korelasi,	-.	Constructor
		Int	klasifikasi		

**Tabel 4.2 Daftar Method Image Processing**

<b>Attribute:</b>					
<b>No</b>	<b>Nama Method</b>	<b>Input</b>		<b>Output</b>	<b>Keterangan</b>
		<b>Tipe</b>	<b>Atribut</b>		
1.	Grayscale	BufferedImage	inputImage	Image	Mengubah citra asli menjadi citra hitam,putih, abu (Grayscale).
2.	applyDCT	double[][]	matrixImage	double[][]	Menerapkan persamaan 2.6 pada citra gambar.

No	Nama Method	Input		Output	Keterangan
		Tipe	Atribut		
3.	applyHighPass Filter	double[][]	matrixImage	double[][]	Melakukan perkalian matrix pada matrix citra.
4.	applyIDCT	double[][]	matrixImage	double[][]	Menerapkan persamaan 2.8 pada citra gambar.
5.	Normalisasi	double[][]	matrixImage	double[][]	Mengubah nilai pixel jika < 0 menjadi 0 dan >255 menjadi 255.
6.	otsuThreshold	Buffered Image	inputImage	Buffered Image	Mengubah citra asli, menjadi citra biner, dengan menggunakan batasan nilai ambang (T) yang didapatkan dengan menggunakan metode otsu.

**Tabel 4.3 Daftar Method Feature Extraction**

<b>Attribute:</b> private static final int KONSTAN = 50;					
No	Nama <i>Method</i>	Input		Output	Keterangan
		Tipe	Atribut		
1.	GLCM	Buffered Image	inputImage	Classification (DAO)	Proses perhitungan nilai statistic Matriks GLCM pada gambar. Pada method ini juga terdapat persamaan (2.9 – 2.12)
2.	takeKolerasi	double[][]	matrixImage	double	Menerapkan persamaan 2.13 – 2.17.
3.	getGLCM0	Buffered Image	inputImage	ArrayList	Menghitung kemunculan pola nilai pixel ke arah 0 derajat atau ke arah timur.
4.	getGLCM45	Buffered Image	inputImage	ArrayList	Menghitung kemunculan pola nilai pixel ke arah 45 derajat atau ke arah timur laut.
5.	getGLCM90	Buffered Image	inputImage	ArrayList	Menghitung kemunculan pola nilai pixel ke arah 90 derajat atau ke arah utara.

No	Nama Method	Input		Output	Keterangan
		Tipe	Atribut		
6.	getGLCM135	Buffered Image	inputImage	ArrayList	Menghitung kemunculan pola nilai pixel ke arah 135 derajat atau ke arah barat laut.

**Tabel 4.4 Daftar Method Support Vector Machine**

No	Nama Method	Input		Output	Keterangan
		Tipe	Atribut		
1.	RBF	double[][]	temp Feature	double[][]	Menerapkan persamaan 2.23, dengan tujuan untuk mendapatkan sebuah model
		double	sigma		
2.	getAlpha	double[][]	tempRBF	double[]	Menerapkan persamaan 2.24-2.26, lalu dilakukan substitusi untuk mendapatkan alpha dan bias (b) dengan menggunakan library JAMA
		double[][]	tempFeat reDataTest		
3.	getRBFModel	double[][]	sigma	double[][]	Membentuk model data uji dengan alpha dan b yang sudah dibuat sebelumnya
		double			

No	Nama Method	Input		Output	Keterangan
		Tipe	Atribut		
4.	identifikasiDataUji	double[][]	tempRBF Model	double[]	<i>Method</i> ini menggunakan persamaan 2.27 yang berfungsi untuk menentukan kelas data uji apakah termasuk klasifikasi kelas -1 atau +1, dengan menggunakan model yang sudah dibuat

**Tabel 4.5 Daftar Method Classification Data Training**

<b>Attribute:</b> private static final int KONSTAN = 50;					
No	Nama Method	Input		Output	Keterangan
		Tipe	Atribut		
1.	saveBlockTo Image	Buffered Image	inputImage	-.	Menyimpan setiap blok menjadi *PNG
2.	listFolders	int	directory Name	int	Membuat directory baru
3.	listFiles	String	Directory Name	Array List	Mengambil semua file sesuai directory
		int	noData		
4.	WriteFile	Classification	my Classification	-.	Menulis data-data yang ada ke dalam bentuk .txt

No	Nama Method	Input		Output	Keterangan
		Tipe	Atribut		
5.	classificationTo Matrix	Classification [][]	Classification Image	Double [][]	Membaca data.txt menjadi matrix
6.	resultClasification	Buffered Image	inputImage	Buffered Image	Menampilkan hasil yang sudah diklasifikasikan
		double[]	result klasifikasi		
		boolean	Syarat		

#### 4.2.1 Implementasi Image Processing

Untuk mengubah suatu citra/*image* menjadi citra *grayscale*, DCT, dan menerapkan metode *Otsu Threshold*, penulis membuat satu kelas yang berfungsi melakukan proses tersebut. Tahap-tahap dalam proses ini yang akan dilakukan adalah:

1. Gambar akan diubah menjadi citra *grayscale*.
  1. Gambar yang sudah diinput akan ditampung ke dalam *buffered image*.
  2. *Buffered Image* akan diresize menjadi 500x500 seperti yang sudah ditulis menjadi batasan masalah.
  3. *Buffered Image* tersebut akan diubah menjadi citra *grayscale* 8-bit menggunakan method dari *library Image Plus* yang sudah dijelaskan pada bab 2. Method tersebut menghasilkan *image* yang akan diubah kembali menjadi *buffered image*. Berikut adalah contoh implementasi pada aplikasi:

```
public Image bufImageToGrayscale(BufferedImage colorImage) {
    ImagePlus ip = new ImagePlus("image about to gs", colorImage);
    ImageConverter ic = new ImageConverter(ip);
    ic.convertToGray8();
    return ip.getImage();
}
```

2. Citra *grayscale* dapat diimplementasikan rumus DCT.
  1. *Buffered image* citra *grayscale* akan dibagi dengan blok dengan ukuran 8x8, lalu akan diubah menjadi *matrix of pixel* ke dalam tipe double[][][]. Setiap blok akan ditampung ke dalam array 3 dimensi bertipe double[][][].
  2. Setelah menjadi matriks, setiap blok yang berisi matriks tersebut dapat langsung diimplementasikan rumus DCT (persamaan 2.6) yang akan menghasilkan koefisien DCT, output *matrix of pixel*.
  3. Koefisien DCT akan diimplementasikan rumus *high pass filter*, output berbentuk *matrix of pixel*.
  4. Hasil dari rumus *high pass filter* akan dikembalikan lagi menggunakan rumus *Invers DCT*.
  5. Hasil *Invers DCT* akan dinormalisasikan sehingga setiap pixel dapat sesuai dengan range yang ada
  6. Proses ini dilakukan pada setiap blok yang ada. Setelah selesai, blok-blok tersebut akan diubah menjadi *buffered image* kembali
3. Citra *grayscale* juga dapat diimplementasikan dengan *otsu threshold*.
  1. Citra *grayscale* akan diubah menjadi *matrix of pixel*, lalu dimasukkan ke dalam method histogram dan menghasilkan array int[]. Array int[] berfungsi pada method otsu untuk menentukan nilai threshold.
  2. Setelah mendapatkan nilai otsu pada method otsu, nilai tersebut akan menjadi pegangan untuk mengubah pixel. Setiap nilai pixel yang lebih besar dari nilai otsu, nilai pixel akan diubah menjadi 255, begitu sebaliknya, nilai pixel akan diubah menjadi 0.
  3. *Matrix of pixel* dari hasil *threshold* tersebut akan diubah kembali menjadi *buffered image*.

#### 4.2.2 Implementasi Feature Extraction

Untuk mendapatkan parameter *machine learning* SVM, maka gambar akan dilakukan *Feature Extraction*, sehingga gambar akan mendapatkan nilai-

nilai yang diperlukan untuk membuat sebuah model dalam *machine learning* SVM.. Tahap-tahap dalam proses ini yang akan dilakukan adalah:

1. Buffered image yang sudah diproses akan dibagi oleh blok berukuran 50x50,60x60,70x70 untuk pengujian. Setiap blok tersebut akan diubah menjadi *matrix of pixel* kembali yang akan ditampung pada array 3 dimensi bertipe double[][][].
2. Setiap blok tersebut akan diimplementasikan method GLCM yang akan menghasilkan nilai-nilai untuk membuat model pada SVM. Nilai-nilai tersebut akan disimpan ke dalam objek *Classification*, tetapi nilai klasifikasi masih 0.
3. Setelah mendapatkan nilai-nilai tersebut, secara manual penulis menyimpan blok-blok tersebut ke dalam suatu folder, lalu mengklasifikasikan mana yang mengandung unsur text dan yang tidak mengandung unsur text. Proses ini dilakukan untuk membuat model.
4. Setelah diklasifikasikan secara manual, nilai klasifikasi yang terdapat pada objek *classification* akan diisi sesuai dengan hasil klasifikasi.
5. Setiap objek *classification* yang berisi nilai-nilai tersebut disimpan pada file bertipe \*.txt.
6. Jika yang dimasukkan adalah data uji, maka proses no.3,4, dan 5 tidak akan dilakukan.

#### **4.2.3 Implementasi Support Vector Machine**

Setelah mendapatkan nilai – nilai *feature extraction*, maka akan dibuat sebuah model dari nilai-nilai *feature extraction* yang akan menjadi parameter *machine learning* SVM. Tahap-tahap dalam proses ini yang akan dilakukan adalah:

1. Nilai-nilai *feature extraction* yang ada pada file \*txt, akan diambil untuk dijadikan model. Nilai-nilai tersebut akan ditampung pada array 2 dimensi bertipe double[][].

2. Setelah ditampung pada double[][], maka akan diimplementasikan method RBF dengan konstan yang penulis tentukan antara 0.002, 0.02, 2, 6, 8. Output berupa array 2 dimensi double[][].
3. Hasil dari method RBF, akan dilakukan substitusi sehingga akan mendapatkan nilai alpha dan bias (b) yang akan disimpan pada file bertipe \*.txt.
4. Pada pengujian kali ini, penulis menggunakan 3 model yang berjumlah 800 data training untuk menentukan blok-blok yang terdapat unsur teks atau tidak pada data uji.
5. Jika data uji yang dimasukkan, maka proses 1-3 tidak dilakukan, tetapi aplikasi akan mengambil data alpha dan bias (b) pada \*.txt yang sudah ada, lalu diimplementasikan persamaan 2.27, sehingga dapat menghasilkan sebuah nilai klasifikasi apakah mengandung unsur teks atau tidak.
6. Setelah menentukan klasifikasi pada gambar apakah blok tersebut mengandung unsur teks atau tidak, maka tahap selanjutnya adalah *post-processing* yang berfungsi untuk mengecek ulang kembali unsur teks dengan tujuan meningkatkan akurasi. Blok yang tidak mengandung unsur teks (-1) akan mengecek pada bagian kiri blok dan atas blok apakah blok tersebut mengandung unsur teks atau tidak, jika mengandung unsur teks maka blok yang tidak mengandung unsur teks akan diklasifikasikan menjadi teks (+1).
7. Lalu data uji, akan ditampilkan mana yang memiliki klasifikasi teks dan non-teks.

### 4.3 Cara Penggunaan Aplikasi

Pada bagian ini akan dijelaskan mengenai penggunaan aplikasi mendeteksi teks pada gambar. Aplikasi dibagi menjadi 2, yaitu bagian membuat model dan bagian pengujian. Berikut ini adalah langkah-langkah penggunaan aplikasi:

- **Bagian membuat model.**

1. Pada tampilan awal gambar 4.1, klik *Browse* untuk memilih gambar yang akan dijadikan data *training*.
2. Tombol *grayscale*, *otsu threshold*, dan *image processing DCT* dapat digunakan sebagai *image processing*.
3. Tombol *feature extraction* berfungsi untuk mendapatkan nilai-nilai pada setiap blok.
4. Tombol *save data training* berfungsi untuk menyimpan blok-blok dalam bentuk gambar (\*PNG). Contoh dapat dilihat pada gambar 4.2.
5. Folder klasifikasi pada gambar 4.2, berfungsi untuk menentukan blok mana saja yang mengandung unsur teks yang dilakukan secara manual.
6. Tombol *Classification Data training* berfungsi untuk menentukan blok mana saja yang mengandung unsur teks yang akan disimpan sebagai data (\*txt).
7. Tombol *Make Model with SVM* berfungsi untuk membuat model dengan data-data yang sudah dikumpulkan, yaitu untuk mendapatkan alpha dan bias (b) yang akan disimpan sebagai (\*txt).



Gambar 4.1 Tampilan aplikasi bagian membuat model



**Gambar 4.2 Contoh gambar yang sudah disave setiap blok**

- **Bagian pengujian**

1. Tampilan aplikasi pada bagian ini hampir sama hanya saja tidak ada tombol yang berfungsi untuk membuat model.
2. Tombol *Classification data test* berfungsi untuk menentukan bagian mana saja yang mengandung unsur text. Contoh dapat dilihat pada gambar 4.3.

Gambar pada bagian gambar 4.3 adalah hasil dimana bagian gelap dianggap bukan text, dan bagian yang berwarna dianggap text.



**Gambar 4.3 Contoh hasil aplikasi**

#### 4.4 Pengujian

Pada sub-bab ini berisi tentang pengujian yang telah dilakukan. Penulis menggunakan 3 model sebanyak 800 data training.

Penulis menggunakan teori *F-Measure* untuk menghitung akurasi pada setiap pengujian. Setiap pengujian akan ditampilkan *True Positive*, *True Negative*, *False Positive*, *False Negative*, *Precision*, dan *Recall*. Pada pengujian kali ini, penulis membuat 2 F-Measure pada setiap pengujian, yaitu blok text yang terdeteksi dan non-blok yang terdeteksi. Keterangan-keterangan untuk blok text terdeteksi pada kasus ini sebagai berikut:

1. *True Positive* = blok text yg terdeteksi.
2. *True Negative* = non-blok text yg tidak terdeteksi.
3. *False Positive* =non-blok text yang terdeteksi.
4. *False Negative* = blok text yg tidak terdeteksi.

Keterangan-keterangan untuk non-blok text terdeteksi pada kasus ini sebagai berikut:

1. *True Positive* = non-blok text yg tidak terdeteksi.
2. *True Negative* = blok text yg terdeteksi.
3. *False Positive* =non-blok text yang terdeteksi.
4. *False Negative* = blok text yg tidak terdeteksi.

Keterangan-keterangan rumus *F-measure* dapat dilihat pada bab 2. Semua proses pengujian dilakukan pada perangkat keras dan perangkat lunak dengan spesifikasi yang telah disebutkan dalam subbab 4.1.

Tujuan dilakukannya pengujian terhadap aplikasi ini, di antaranya:

1. Untuk menentukan arah dari GLCM (Gray Level Co-occurrence Matrices) 0 derajat, 45 derajat, 90 derajat, 135 derajat yang terbaik.
2. Untuk menentukan konstanta sigma yang terbaik pada rumus kernel RBF *Support Vector Machine*.
3. Menggunakan *Image Processing* seperti *threshold otsu*, *DCT*, *grayscale*, dan tanpa proses.

4. Untuk menukan ukuran blok pada *Feature Extraction* yang terbaik, ukurannya antara lain 50x50 pixel, 60x60 pixel, 70x70 pixel.

#### **4.4.1 Pengujian untuk Menentukan arah GLCM**

Pengujian ini dimaksudkan untuk menentukan arah GLCM yang terbaik dalam mendapatkan nilai-nilai yang diperlukan dalam proses pembuatan model pada proses *Feature Extraction*. Hasil yang baik jika memiliki nilai akurasi yang tinggi. Arah yang akan diujikan : 0 derajat, 45 derajat, 90 derajat dan 135 derajat. Berikut hasil pengujian arah GLCM.dan data-data yang akan digunakan:

1. Ukuran gambar 500x500.
2. Image Processing : *Grayscale*.
3. GLCM: arah akan diuji.
4. Jumlah data training = 800 buah.
5. Sigma konstanta = 2.

Keterangan table:

1. TP(*True Positive*) = blok text yg terdeteksi.
2. TN(*True Negative*) = non-blok text yg tidak terdeteksi.
3. FP(*False Positive*) =non-blok text yang terdeteksi.
4. FN(*False Negative*) = blok text yg tidak terdeteksi.

##### **1. Arah 0 derajat:**

Hasil gagal, dikarenakan determinant 0.

**2. Arah 135 derajat:**

**Tabel 4.6 Hasil pengujian arah 135 derajat blok teks.**

No	TP	FP	FN	TN	Precision Teks	Recall Teks	Fmeasure Teks	Precision Non-Teks	Recall Non-Teks	Fmeasure Non-Teks	Rata-rata Fmeasure Total
1	2	24	6	68	7,69%	25,00%	11,76%	73,91%	91,89%	81,93%	46,85%
2	1	5	5	89	16,67%	16,67%	16,67%	94,68%	94,68%	94,68%	55,67%
3	0	0	14	86	0,00%	0,00%	0,00%	100,00%	86,00%	92,47%	46,24%
4	3	18	5	74	14,29%	37,50%	20,69%	80,43%	93,67%	86,55%	53,62%
5	8	3	17	72	72,73%	32,00%	44,44%	96,00%	80,90%	87,80%	66,12%
6	26	20	25	29	56,52%	50,98%	53,61%	59,18%	53,70%	56,31%	54,96%
7	6	29	0	65	17,14%	100,00%	29,27%	69,15%	100,00%	81,76%	55,51%
8	0	8	14	78	0,00%	0,00%	0,00%	90,70%	84,78%	87,64%	43,82%
9	15	22	26	37	40,54%	36,59%	38,46%	62,71%	58,73%	60,66%	49,56%
10	1	2	11	86	33,33%	8,33%	13,33%	97,73%	88,66%	92,97%	53,15%
<b>Rata-rata F-Measure Teks</b>						22,82%	<b>Rata-rata F-Measure Non-Teks</b>		82,28%		52,55%

**3. Arah 90 derajat:**

**Tabel 4.7 Tabel hasil pengujian arah 90 derajat**

No	TP	FP	FN	TN	Precision Teks	Recall Teks	Fmeasure Teks	Precision Non-Teks	Recall Non-Teks	Fmeasure Non-Teks	Rata-rata Fmeasure Total
1	5	10	3	82	33,33%	62,50%	43,48%	89,13%	96,47%	92,66%	68,07%
2	4	5	2	89	44,44%	66,67%	53,33%	94,68%	97,80%	96,22%	74,77%
3	3	8	11	78	27,27%	21,43%	24,00%	90,70%	87,64%	89,14%	56,57%
4	6	27	2	65	18,18%	75,00%	29,27%	70,65%	97,01%	81,76%	55,51%
5	5	6	20	69	45,45%	20,00%	27,78%	92,00%	77,53%	84,15%	55,96%
6	34	22	17	27	60,71%	66,67%	63,55%	55,10%	61,36%	58,06%	60,81%

No	TP	FP	FN	TN	Precision Teks	Recall Teks	Fmeasure Teks	Precision Non-Teks	Recall Non-Teks	Fmeasure Non-Teks	Rata-rata Fmeasure Total
7	4	34	2	60	10,53%	66,67%	18,18%	63,83%	96,77%	76,92%	47,55%
8	9	13	5	73	40,91%	64,29%	50,00%	84,88%	93,59%	89,02%	69,51%
9	18	13	23	46	58,06%	43,90%	50,00%	77,97%	66,67%	71,88%	60,94%
10	7	7	5	81	50,00%	58,33%	53,85%	92,05%	94,19%	93,10%	73,47%
<b>Rata-rata F-Measure Teks</b>					41,34%		<b>Rata-rata F-Measure Non-Teks</b>		83,29%		62,32%

#### 4. Arah 45 derajat:

**Tabel 4.8 Hasil pegujian arah 45 derajat**

No	TP	FP	FN	TN	Precision Teks	Recall Teks	Fmeasure Teks	Precision Non-Teks	Recall Non-Teks	Fmeasure Non-Teks	Rata-rata Fmeasure Total
1	5	9	3	83	35,71%	62,50%	45,45%	90,22%	96,51%	93,26%	69,36%
2	2	6	4	88	25,00%	33,33%	28,57%	93,62%	95,65%	94,62%	61,60%
3	4	5	10	81	44,44%	28,57%	34,78%	94,19%	89,01%	91,53%	63,15%
4	4	19	4	73	17,39%	50,00%	25,81%	79,35%	94,81%	86,39%	56,10%
5	11	8	14	67	57,89%	44,00%	50,00%	89,33%	82,72%	85,90%	67,95%
6	25	19	26	30	56,82%	49,02%	52,63%	61,22%	53,57%	57,14%	54,89%
7	3	33	3	61	8,33%	50,00%	14,29%	64,89%	95,31%	77,22%	45,75%
8	3	3	11	83	50,00%	21,43%	30,00%	96,51%	88,30%	92,22%	61,11%
9	22	17	19	42	56,41%	53,66%	55,00%	71,19%	68,85%	70,00%	62,50%
10	5	9	7	79	35,71%	41,67%	38,46%	89,77%	91,86%	90,80%	64,63%
<b>Rata-rata F-Measure Teks</b>					37,50%		<b>Rata-rata F-Measure Non-Teks</b>		83,91%		60,70%

Dari pengujian diatas dapat hasil akurasi yang terbaik adalah arah 90 derajat yang menghasilkan akurasi yang tebaik.

#### **4.4.2 Pengujian untuk Konstanta Sigma pada Kernel RBF**

Pengujian ini dimaksudkan untuk menentukan konstanta Sigma pada rumus kernel RBF yang terbaik dalam membentuk sebuah model. Hasil yang baik jika memiliki nilai akurasi yang tinggi. Niali sigma yang akan diuji adalah 0.002, 0.02, 2, 6 8. Berikut hasil pengujian .dan data-data yang akan digunakan:.

1. Ukuran gambar 500x500.
2. Image Processing : *Grayscale*.
3. GLCM: 90 derajat.
4. Jumlah data training = 800 buah.
5. Sigma konstanta = akan diuji.

##### **1. Nilai Sigma 8**

Hasil gagal, dikarenakan determinant 0.

##### **2. Nilai Sigma 6**

Hasil gagal, dikarenakan determinant 0.

##### **3. Nilai Sigma 0.2**

Hasil gagal, dikarenakan determinant 0.

##### **4. Nilai Sigma 0.002**

Hasil gagal, dikarenakan determinant 0.

Dari pengujian diatas dapat hasil akurasi yang terbaik adalah menggunakan nilai sigma 2 pada tabel 4.7, yang menghasilkan akurasi terbaik.

#### **4.4.3 Pengujian pengaruh Image Processing pada Hasil Akurasi**

Pengujian ini dimaksudkan untuk mendapatkan hasil akurasi yang terbaik dengan mengimplementasikan beberapa metode *image processing* yaitu: *Grayscale*, DCT, *Threshold otsu*. Hasil yang baik jika memiliki nilai akurasi yang tinggi. Proses yang akan diuji adalah *grayscale*, DCT, *Otsu threshold*. Berikut hasil pengujian dan data-data yang akan digunakan.:

1. Ukuran gambar 500x500.
2. Image Processing : akan diuji.
3. GLCM: 90 derajat.
4. Jumlah data training = 800 buah.
5. Sigma konstanta = 2.

##### **1. Menggunakan proses Grayscale + DCT**

Hasil gagal, dikarenakan determinant 0.

##### **2. Menggunakan proses Grayscale + OTSU**

Hasil gagal, dikarenakan determinant 0.

##### **3. Tidak menggunakan image processing**

1. Ukuran gambar 500x500.
2. Image Processing : -.
3. GLCM arah 90 derajat.
4. Jumlah data training = 300 buah.
5. Sigma konstanta = 2.
6. Determinant = 0.

Hasil tidak bisa dilkakukan karena gambar memiliki 3 nilai pixel (RGB) sedangkan *feature extraction* hanya menerima 1 nilai pixel.

Dari pengujian diatas didapatkan hasil akurasi yang terbaik adalah menggunakan hanya *grayscale* pada tabel 4.7 yang menghasilkan akurasi yang tebaik. Hal ini

dikarenakan oleh metode GLCM pada *feature extraction*, GLCM hanya cocok jika gambar berupa citra *grayscale*.

#### **4.2.5 Pengujian untuk Ukuran Blok pada Feature Extraction**

Pengujian ini dimaksudkan untuk menentukan ukuran blok yang terbaik untuk mendapatkan hasil akurasi yang terbaik dalam *Feature Extraction*. Hasil yang baik jika memiliki nilai akurasi yang tinggi. Ukuran blok yang akan diuji adalah 50x50, 60x60, 70x70. Berikut hasil pengujian .dan data-data yang akan digunakan:.

1. Ukuran gambar 500x500.
2. Image Processing : *grayscale*.
3. GLCM: 90 derajat.
4. Jumlah data training = 800 buah.
5. Sigma konstanta = 2.

Keterangan table:

1. TP(*True Positive*) = blok text yg terdeteksi.
2. TN(*True Negative*) = non-blok text yg tidak terdeteksi.
3. FP(*False Positive*) =non-blok text yang terdeteksi.
4. FN(*False Negative*) = blok text yg tidak terdeteksi.

##### **1. Ukuran blok 60x60**

**Tabel 4.9 Hasil pengujian menggunakan Ukuran Blok 60x60**

No	TP	FP	FN	TN	Precision Teks	Recall Teks	Fmeasure Teks	Precision Non-Teks	Recall Non-Teks	Fmeasure Non-Teks	Rata-rata Fmeasure Total
1	5	10	3	63	33,33%	62,50%	43,48%	86,30%	95,45%	90,65%	67,06%
2	3	12	3	63	20,00%	50,00%	28,57%	84,00%	95,45%	89,36%	58,97%
3	0	0	6	75	0,00%	0,00%	0,00%	100,00%	92,59%	96,15%	48,08%

No	TP	FP	FN	TN	Precision Teks	Recall Teks	Fmeasure Teks	Precision Non-Teks	Recall Non-Teks	Fmeasure Non-Teks	Rata-rata Fmeasure Total
4	2	19	2	58	9,52%	50,00%	16,00%	75,32%	96,67%	84,67%	50,34%
5	9	8	16	48	52,94%	36,00%	42,86%	85,71%	75,00%	80,00%	61,43%
6	26	7	16	32	78,79%	61,90%	69,33%	82,05%	66,67%	73,56%	71,45%
7	4	27	2	48	12,90%	66,67%	21,62%	64,00%	96,00%	76,80%	49,21%
8	4	4	8	65	50,00%	33,33%	40,00%	94,20%	89,04%	91,55%	65,77%
9	13	4	20	44	76,47%	39,39%	52,00%	91,67%	68,75%	78,57%	65,29%
10	5	5	7	64	50,00%	41,67%	45,45%	92,75%	90,14%	91,43%	68,44%
<b>Rata-rata F-Measure Teks</b>							35,93%	<b>Rata-rata F-Measure Non-Teks</b>		85,27%	60,60%

## 2. Ukuran blok 70x70

**Tabel 4.10 Hasil pengujian menggunakan Ukuran Blok 70x70**

No	TP	FP	FN	TN	Precision Teks	Recall Teks	Fmeasure Teks	Precision Non-Teks	Recall Non-Teks	Fmeasure Non-Teks	Rata-rata Fmeasure Total
1	2	11	3	48	15,38%	40,00%	22,22%	81,36%	94,12%	87,27%	54,75%
2	0	0	5	59	0,00%	0,00%	0,00%	100,00%	92,19%	95,93%	47,97%
3	0	0	5	59	0,00%	0,00%	0,00%	100,00%	92,19%	95,93%	47,97%
4	2	3	6	53	40,00%	25,00%	30,77%	94,64%	89,83%	92,17%	61,47%
5	2	1	20	41	66,67%	9,09%	16,00%	97,62%	67,21%	79,61%	47,81%
6	15	17	15	17	46,88%	50,00%	48,39%	50,00%	53,13%	51,52%	49,95%
7	3	11	3	47	21,43%	50,00%	30,00%	81,03%	94,00%	87,04%	58,52%
8	3	3	8	50	50,00%	27,27%	35,29%	94,34%	86,21%	90,09%	62,69%
9	18	3	11	32	85,71%	62,07%	72,00%	91,43%	74,42%	82,05%	77,03%
10	2	6	9	47	25,00%	18,18%	21,05%	88,68%	83,93%	86,24%	53,65%

<b>Rata-rata F-Measure Teks</b>	27,57%	<b>Rata-rata F-Measure Non-Teks</b>	84,79%	56,18%
---------------------------------	--------	-------------------------------------	--------	--------

Dari pengujian diatas dapat dilihat bahwa ukuran 50x50 pada tabel 4.7 yang menghasilkan akurasi yang tebaik.

#### 4.2.6 Pengujian Post Processing

Penulis melakukan pengujian terhadap *post-processing*, tujuan pengujian ini untuk melihat apakah *post processing* mempengaruhi nilai akurasi. Hasil pengujian dengan *post-processing* dapat di lihat pada tabel 4.7. Berikut hasil dari penelitian :

1. Ukuran gambar 500x500.
2. Image Processing : *grayscale*.
3. GLCM: 90 derajat.
4. Jumlah data training = 800 buah.
5. Sigma konstanta = 2.
6. Tanpa *post-processing*.

Keterangan table:

1. TP(*True Positive*) = blok text yg terdeteksi.
2. TN(*True Negative*) = non-blok text yg tidak terdeteksi.
3. FP(*False Positive*) =non-blok text yang terdeteksi.
4. FN(*False Negative*) = blok text yg tidak terdeteksi.

**1. Pengujian dengan 3 model tanpa post-processing.**

**Tabel 4.11 Hasil pengujian dengan 3 model tanpa post-processing**

No	TP	FP	FN	TN	Precision Teks	Recall Teks	Fmeasure Teks	Precision Non- Teks	Recall Non- Teks	Fmeasure Non-Teks	Rata-rata Fmeasure Total
1	2	4	6	88	33,33%	25,00%	28,57%	95,65%	93,62%	94,62%	61,60%
2	2	1	4	93	66,67%	33,33%	44,44%	98,94%	95,88%	97,38%	70,91%
3	1	3	13	83	25,00%	7,14%	11,11%	96,51%	86,46%	91,21%	51,16%
4	2	11	6	81	15,38%	25,00%	19,05%	88,04%	93,10%	90,50%	54,78%
5	2	2	23	73	50,00%	8,00%	13,79%	97,33%	76,04%	85,38%	49,59%
6	17	11	34	38	60,71%	33,33%	43,04%	77,55%	52,78%	62,81%	52,92%
7	2	13	4	81	13,33%	33,33%	19,05%	86,17%	95,29%	90,50%	54,78%
8	5	4	9	82	55,56%	35,71%	43,48%	95,35%	90,11%	92,66%	68,07%
9	8	4	33	55	66,67%	19,51%	30,19%	93,22%	62,50%	74,83%	52,51%
10	4	1	8	87	80,00%	33,33%	47,06%	98,86%	91,58%	95,08%	71,07%
<b>Rata-rata F-Measure Teks</b>					29,98%		<b>Rata-rata F- Measure Non- Teks</b>		87,50%		58,74%

**2. Pengujian dengan 3 model dan menambah attribute pada Feature Extraction.**

Pengujian ini dilakukan dengan cara menambah attribute pada *Support Vector Machine* yaitu dengan memperhatikan bagian kiri dan atas dari blok tersebut apakah blok tersebut teks atau non-teks. Dengan menambah attribute pada *Feature Extraction*, berikut hasil pengujian:

**Tabel 4.12 Hasil pengujian 3 model dengan menambah attribute pada SVM**

No	TP	FP	FN	TN	Precision Teks	Recall Teks	Fmeasur e Teks	Precision Non-Teks	Recall Non-Teks	Fmeasure Non-Teks	Rata-rata Fmeasure Total
1	2	3	6	89	40,00%	25,00%	30,77%	96,74%	93,68%	95,19%	62,98%
2	1	0	5	94	100,00%	16,67%	28,57%	100,00%	94,95%	97,41%	62,99%
3	2	2	12	84	50,00%	14,29%	22,22%	97,67%	87,50%	92,31%	57,26%
4	1	7	7	85	12,50%	12,50%	12,50%	92,39%	92,39%	92,39%	52,45%
5	2	2	23	73	50,00%	8,00%	13,79%	97,33%	76,04%	85,38%	49,59%
6	17	12	34	37	58,62%	33,33%	42,50%	75,51%	52,11%	61,67%	52,08%
7	2	13	4	81	13,33%	33,33%	19,05%	86,17%	95,29%	90,50%	54,78%
8	4	5	10	81	44,44%	28,57%	34,78%	94,19%	89,01%	91,53%	63,15%
9	7	3	34	56	70,00%	17,07%	27,45%	94,92%	62,22%	75,17%	51,31%
10	3	1	9	87	75,00%	25,00%	37,50%	98,86%	90,63%	94,57%	66,03%
<b>Rata-rata F-Measure Teks</b>						26,91%	<b>Rata-rata F- Measure Non-Teks</b>		87,61%		57,26%

Dari pengujian diatas dapat dilihat bahwa dengan *post-processing* didapatkan hasil yang lebih baik, hasil dapat dilihat pada tabel 4.7.

#### 4.2.7 Pengujian Jumlah Data Training

Penulis melakukan pengujian dengan memperbanyak jumlah training. Penulis memperbanyak menjadi 9 model 2000 data training. Berikut hasil pengujian :

1. Ukuran gambar 500x500.
2. Image Processing : *grayscale*.
3. GLCM: 90 derajat.
4. Jumlah data training = 2000 buah.
5. Sigma konstanta = 2.

Keterangan table:

1. TP(*True Positive*) = blok text yg terdeteksi.
2. TN(*True Negative*) = non-blok text yg tidak terdeteksi.
3. FP(*False Positive*) =non-blok text yang terdeteksi.
4. FN(*False Negative*) = blok text yg tidak terdeteksi

### 1. Pengujian dengan 9 model tanpa post-processing.

**Tabel 4.13 Hasil pengujian dengan 9 model tanpa post-processing**

No	TP	FP	FN	TN	Precision Teks	Recall Teks	Fmeasure Teks	Precision Non- Teks	Recall Non- Teks	Fmeasure Non-Teks	Rata-rata Fmeasure Total
1	1	0	7	92	100,00%	12,50%	22,22%	100,00%	92,93%	96,34%	59,28%
2	0	0	6	94	0,00%	0,00%	0,00%	100,00%	94,00%	96,91%	48,45%
3	0	1	14	85	0,00%	0,00%	0,00%	98,84%	85,86%	91,89%	45,95%
4	0	3	8	89	0,00%	0,00%	0,00%	96,74%	91,75%	94,18%	47,09%
5	1	0	24	75	100,00%	4,00%	7,69%	100,00%	75,76%	86,21%	46,95%
6	3	1	48	48	75,00%	5,88%	10,91%	97,96%	50,00%	66,21%	38,56%
7	0	1	6	93	0,00%	0,00%	0,00%	98,94%	93,94%	96,37%	48,19%
8	2	0	12	86	100,00%	14,29%	25,00%	100,00%	87,76%	93,48%	59,24%
9	1	0	40	59	100,00%	2,44%	4,76%	100,00%	59,60%	74,68%	39,72%
10	1	1	11	87	50,00%	8,33%	14,29%	98,86%	88,78%	93,55%	53,92%
<b>Rata-rata F-Measure Teks</b>					8,49%	<b>Rata-rata F- Measure Non- Teks</b>			88,98%	48,73%	

### 3. Pengujian 9 model dengan post-proceesing

Tabel 4.14 Hasil pengujian 9 model dengan post-processing

No	TP	FP	FN	TN	Precision Teks	Recall Teks	Fmeasure Teks	Precision Non-Teks	Recall Non-Teks	Fmeasure Non-Teks	Rata-rata Fmeasure Total
1	1	2	7	90	33,33%	12,50%	18,18%	97,83%	92,78%	95,24%	56,71%
2	0	0	6	94	0,00%	0,00%	0,00%	100,00%	94,00%	96,91%	48,45%
3	0	9	14	77	0,00%	0,00%	0,00%	89,53%	84,62%	87,01%	43,50%
4	0	16	8	76	0,00%	0,00%	0,00%	82,61%	90,48%	86,36%	43,18%
5	3	3	22	72	50,00%	12,00%	19,35%	96,00%	76,60%	85,21%	52,28%
6	9	5	42	44	64,29%	17,65%	27,69%	89,80%	51,16%	65,19%	46,44%
7	1	5	5	89	16,67%	16,67%	16,67%	94,68%	94,68%	94,68%	55,67%
8	5	7	9	79	41,67%	35,71%	38,46%	91,86%	89,77%	90,80%	64,63%
9	1	1	40	58	50,00%	2,44%	4,65%	98,31%	59,18%	73,89%	39,27%
10	1	7	11	81	12,50%	8,33%	10,00%	92,05%	88,04%	90,00%	50,00%
<b>Rata-rata F-Measure Teks</b>						13,50%	<b>Rata-rata F-Measure Non-Teks</b>	86,53%	50,01%		

### 4. Pengujian 9 model dengan menambahkan attribute pada Feature Extraction

Tabel 4.15 Hasil pengujian 9 model menambahkan attribute pada SVM

No	TP	FP	FN	TN	Precision Teks	Recall Teks	Fmeasure Teks	Precision Non-Teks	Recall Non-Teks	Fmeasure Non-Teks	Rata-rata Fmeasure Total
1	1	0	7	92	100,00%	12,50%	22,22%	100,00%	92,93%	96,34%	59,28%
2	0	0	6	94	0,00%	0,00%	0,00%	100,00%	94,00%	96,91%	48,45%
3	0	1	14	85	0,00%	0,00%	0,00%	98,84%	85,86%	91,89%	45,95%
4	0	3	8	89	0,00%	0,00%	0,00%	96,74%	91,75%	94,18%	47,09%
5	1	0	24	75	100,00%	4,00%	7,69%	100,00%	75,76%	86,21%	46,95%

No	TP	FP	FN	TN	Precision Teks	Recall Teks	Fmeasure Teks	Precision Non-Teks	Recall Non-Teks	Fmeasure Non-Teks	Rata-rata Fmeasure Total
6	3	1	48	48	75,00%	5,88%	10,91%	97,96%	50,00%	66,21%	38,56%
7	0	1	6	93	0,00%	0,00%	0,00%	98,94%	93,94%	96,37%	48,19%
8	2	0	12	86	100,00%	14,29%	25,00%	100,00%	87,76%	93,48%	59,24%
9	1	0	40	59	100,00%	2,44%	4,76%	100,00%	59,60%	74,68%	39,72%
10	1	1	11	87	50,00%	8,33%	14,29%	98,86%	88,78%	93,55%	53,92%
<b>Rata-rata F-Measure Teks</b>					8,49%	<b>Rata-rata F-Measure Non-Teks</b>		88,98%	48,73%		

Dari pengujian diatas dapat di lihat bahwa akurasi teks sangat kecil, dan akurasi non-teksnya cukup tinggi, hal ini di karenakan karena jumlah teks dan non-teks pada data training tidak sebanding.

#### 4.4.8 Pengujian Movie Poster

Penulis melakukan pengujian dengan mengganti data training menjadi *movie poster* dan menyamakan jumlah data training antar teks dan non-teks. Penulis menggunakan 3 model (700 blok teks dan 700 blok non-teks data training) dan 5 model (900 blok teks dan 900 blok non-teks data training). Berikut hasil pengujian :

**Tabel 4.15 Hasil pengujian dengan data training Movie Poster**

No	Skenario	F-Measure
1.	Jumlah Model : 3 model Sigma:2 Arah GLCM : 135 derajat Ukuran blok: 50x50 Tanpa <i>post processing</i>	73,33%

No	Skenario	F-Measure
2.	Jumlah Model : 3 model Sigma:2 Arah GLCM : 90 derajat Ukuran blok: 50x50 <i>Tanpa post processing</i>	<b>73,84%</b>
3.	Jumlah Model : 3 model Sigma:2 Arah GLCM : 45 derajat Ukuran blok: 50x50 <i>Tanpa post processing</i>	<b>71,18%</b>
4.	Jumlah Model : 3 model Sigma:2 Arah GLCM : 0 derajat Ukuran blok: 50x50 <i>Tanpa post processing</i>	<b>73,17%</b>
5.	Jumlah Model : 3 model Sigma: 2 Arah GLCM : 90 derajat Ukuran blok: 50x50 <i>Post processing</i> dengan menambah <i>feature extraction</i> pada SVM	<b>75,99%</b>
6.	Jumlah Model : 5 model Sigma: 2 Arah GLCM : 135 derajat Ukuran blok: 50x50 <i>Tanpa post processing</i>	<b>72,08%</b>
7.	Jumlah Model : 5 model Sigma: 2 Arah GLCM : 90 derajat Ukuran blok: 50x50 <i>Tanpa post processing</i>	<b>72,80%</b>
8.	Jumlah Model : 5 model Sigma: 2 Arah GLCM : 45 derajat Ukuran blok: 50x50 <i>Tanpa post processing</i>	<b>72,07%</b>
9.	Jumlah Model : 5 model Sigma: 2 Arah GLCM : 0 derajat Ukuran blok: 50x50 <i>Tanpa post processing</i>	<b>71,53%</b>

10.	Jumlah Model : 5 model Sigma: 2 Arah GLCM : 90 derajat Ukuran blok: 50x50 <i>Post processing</i> dengan menambah <i>feature extraction</i> pada SVM	<b>75,86%</b>
-----	---	---------------

Dari pengujian diatas dapat dilihat bahwa dengan menggunakan data training *Movie Poster* dan penggunaan *post-processing* dengan menambahkan *feature extraction* pada SVM didapatkan hasil yang jauh lebih baik, hasil dapat dilihat pada tabel 4.15.

## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Kesimpulan yang dapat diambil dari pembuatan aplikasi dan pengujian-pengujian yang telah dilakukan, penulis dapat menarik kesimpulan yaitu:

1. Pada pengujian kali ini, pengaruh dari *image processing* DCT dan *Otsu threshold* tidak memberikan hasil yang baik. Hal tersebut dikarenakan dalam bagian *feature extraction*, penulis menggunakan metode GLCM yang hanya dapat memberi hasil yang baik jika gambar dalam citra *grayscale*.
2. Pengaruh dari ukuran blok pada *feature extraction* sangat berpengaruh pada hasil akurasi, semakin kecil ukuran blok semakin baik hasil akurasi.
3. *Machine learning Support Vector Machine* dapat mengklasifikasikan teks dengan sangat baik jika jumlah data training semakin banyak. Jika jumlah data training semakin banyak maka *machine learning* dapat menghasilkan model yang semakin baik. Model yang sudah dibuat akan dibandingkan dengan data uji sehingga akan menghasilkan akurasi yang tinggi.
4. Hal-hal yang mempengaruhi akurasi menjadi tinggi adalah:
  - a. Arah dari GLCM. Pada pengujian arah yang terbaik adalah 90 derajat pada tabel 4.7 dan tabel 4.15.
  - b. Konstanta sigma pada rumus RBF yang terdapat pada *Support Vector Machine*. Pada pengujian sigma yang terbaik untuk mendapatkan akurasi yang tinggi adalah 2 pada tabel 4.7 dan tabel 4.15.
  - c. Hasil pengujian didapatkan hasil ukuran blok pada *feature extraction* yang terbaik adalah 50x50 pada tabel 4.7 dan tabel 4.15.

- d. Data training *movie poster* menghasilkan klasifikasi yang lebih baik dibandingkan data training *natural image*, hal ini dikarenakan teks pada *movie poster* sudah memiliki nilai kontrast, kolerasi, homogenitas, energi, entropi yang baik dibandingkan *natural image*. Hasil pengujian dapat dilihat dari tabel 4.15.
- e. Jumlah model yang banyak tidak menjamin hasil akurasi menjadi lebih baik. Hasil dapat dilihat dari tabel 4.15.

## 5.2 Saran

Saran penulis yang dapat dijadikan masukan dalam pengembangan sistem yang akan datang:

1. Mengembangkan *Support Vector Machine* lebih lanjut sehingga mendapatkan hasil akurasi yang lebih tinggi, terutama pada bagian mencari alpha dan bias (b), karena pengujian kali ini, penulis menggunakan metode substitusi untuk mencari alpha dan bias (b) untuk membuat model. Kekurangan dari metode substitusi pada pengujian kali ini adalah, jumlah data training yang terbatas karena semakin banyak persamaan, semakin sulit untuk mensubtitusikan yang mengakibatkan nilai determinant menjadi 0. Seharusnya *machine learning* wajib memiliki banyak data training, tetapi karena kekurangan hal tersebut membuat jumlah data training terbatas. Hal ini sangat merugikan karena jumlah data training yang sedikit mempengaruhi hasil akurasi. Pada pengujian kali ini, penulis membuat beberapa model yang terdiri dari data-data training sesuai batas maksimal, agar determinant tidak 0. Penulis menyarankan membuat 1 model saja dengan jumlah data training yang banyak, hal tersebut dapat membuat model menjadi baik. Jika ingin mencoba dengan beberapa model, penulis menyarankan untuk dilakukan penelitian lagi mengenai pembagian model.

2. Penulis juga menyarankan untuk mencoba kernel *polynomial*, dan *Sigmoid* pada *machine learning Support Vector Machine*.
3. Pada *image processing* sebaiknya tidak menggunakan DCT, tetapi menggunakan DWT karena proses DCT memakan waktu yang cukup lama sedangkan DWT memakan waktu yang lebih cepat dalam pemrosesannya dan menghasilkan hasil yang lebih baik.

## DAFTAR PUSTAKA

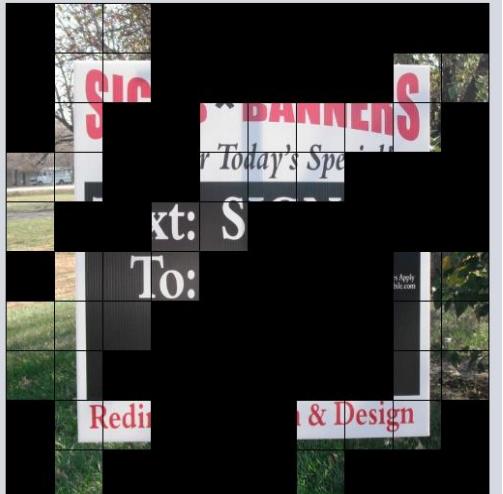
- [ANG10] S. A. Angadi, M. M. Kodabagi, " *Text Region Extraction from Low Resolution Natural Scene Images using Texture Features*",2010
- [SIM05] Simeon J.Simoof, Michael H. Bohlen, Aturas Mazeika, " *Visul Data Mining, theory ,techniques and tools for Visual Analitics*",2005
- [LUK10] Lukas Neumann and Jiri Matas," *A method for text localization and recognition in real-world images*",2010
- [LIP05] Lipo Wang," *Support Vector Machines: Theory and Applications*",2005
- [XUC10] Xu-Cheng Yin, Xuwang Yin, Kaizhu Huang, and Hong-Wei Hao," *Robust Text Detection in Natural Scene Images*",2010
- [MAH10] Maharani Desy Wuryandari, Irawan Afrianto " *Perbandingan Metode Jaringan Syaraf Tiruan Backpropagation dan Learning Vector Quantization pada Pengenalan Wajah*",2012
- [PAN10] Panca Mudji Rahardjo," *Pengenalan Ekspresi Wajah berbasis Filter Gabor dan Backpropagation Neural Network*",2010
- [ETT13] Ettyc Juharwidyningsih, Chastine Faticah, dan Wijayanti Nurul Khotimah," *Pengenalan Karakter Tulisan Tangan Angka dan Operator Matematika Berdasarkan Zernike Moments Menggunakan Support Vector Machine*",2013
- [AND13] Andi Lukman, Syafaruddin, Merna Baharuddin," *Perancangan Machine Learning Klasifikasi Citra Digital Berbasis Software As A Service (SaaS)*",2013
- [DAR10] Darma Putra," *Pengolahan Citra Digital*",2010
- [TUK10] Tukino, S.Kom, M.SI," *Pengantar Pengolahan Citra Digital*",2010.
- [ARI06] Arief Hermawan," *Jaringan Syaraf Tiruan Teori dan Aplikasi*",2006.
- [SHO15] Shovasis Kumar Biswas," *Image Reconstruction Using Multi Layer Perceptron (MLP) And Support Vector Machine (SVM) Classifier And Study Of Classification Accuracy*",2015.

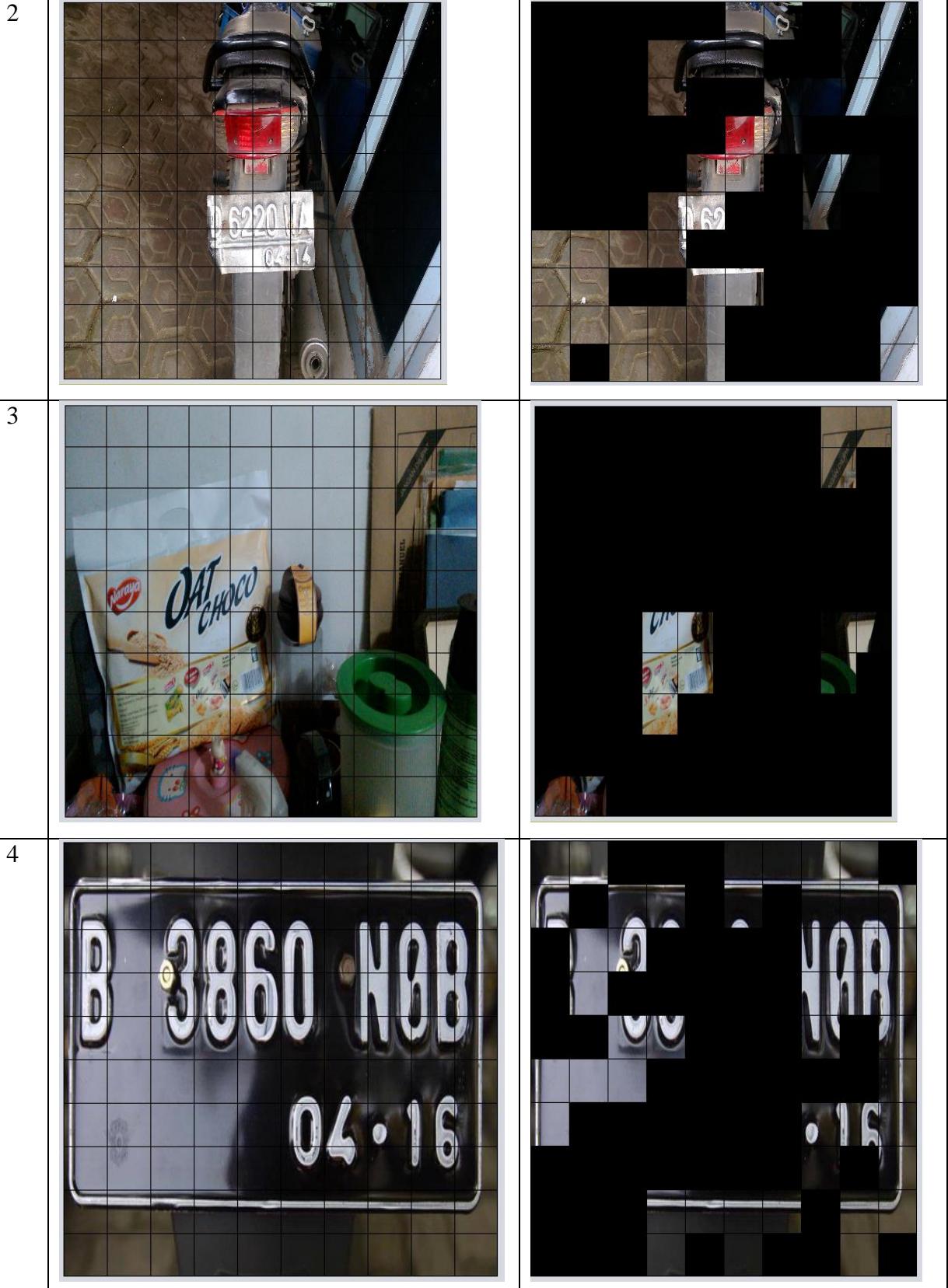
- [ANI11] Anilkumar Katharotiya,” *Comparative Analysis between DCT & DWT Techniques of Image Compression*”,2015.
- [ROB73] Robbert M Harlick,” *Textural Features for Image Classification*”,1973.
- [MUH13] Muhammad Dendy Agaputra,” *PENCARIAN CITRA DIGITAL BERBASISKAN KONTEN DENGAN EKSTRAKSI FITUR HSV, ACD DAN GLCM*”,2013.
- .

## LAMPIRAN A

Hasil pengujian menggunakan:

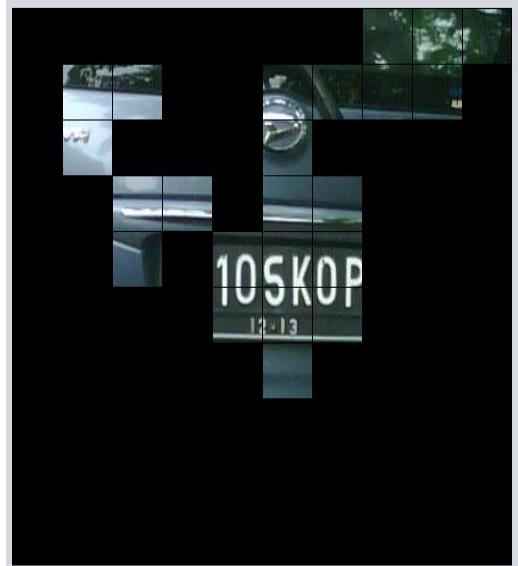
1. Ukuran gambar 500x500.
2. Data Training : *Natural Image*.
3. Image Processing : *Grayscale*.
4. GLCM: 90 derajat.
5. Jumlah data training = 800 buah.
6. Sigma konstanta = 2.
7. Ukuran block = 50x50.

No	Input	Hasil Pengujian
1		



A-2

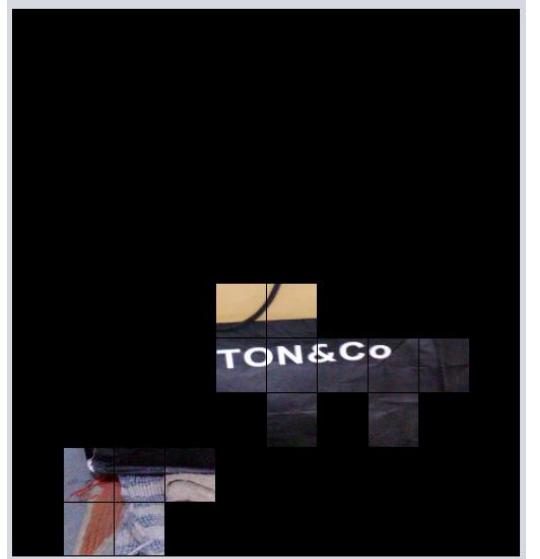
5



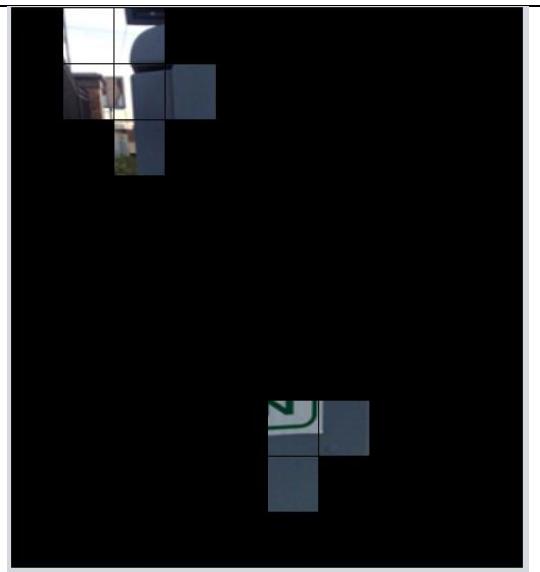
6



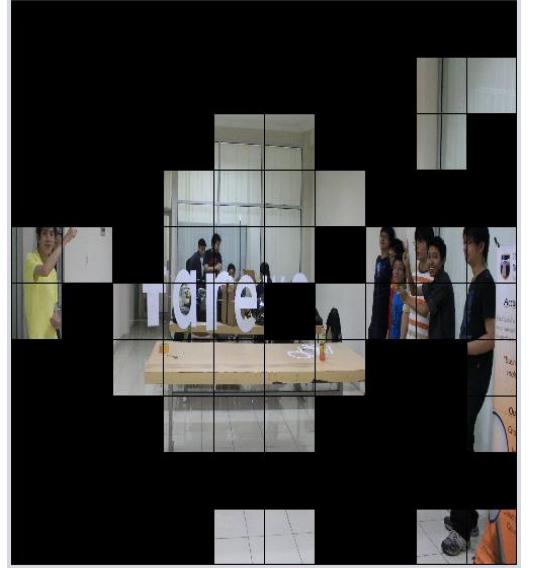
7



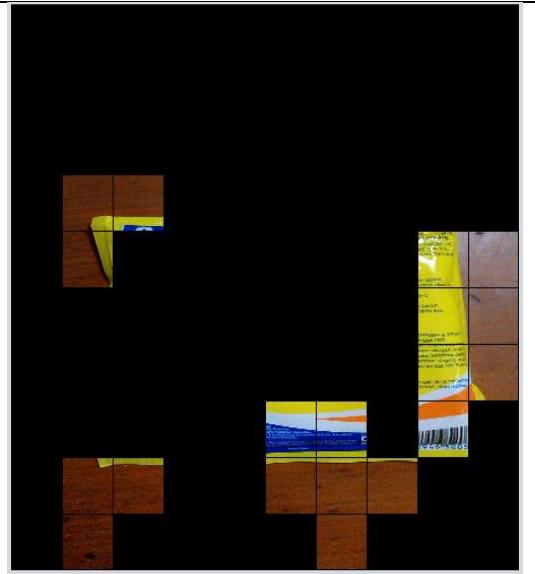
8



9



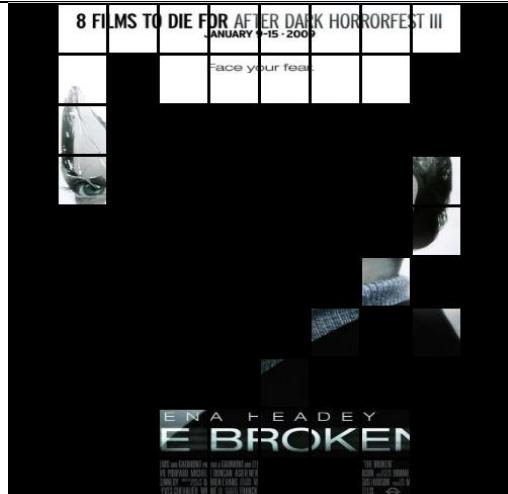
10



## LAMPIRAN B

Hasil pengujian menggunakan:

1. Ukuran gambar 500x500.
2. Data Training : *Movie Poster*.
3. Image Processing : *Grayscale*.
4. GLCM: 90 derajat.
5. Jumlah data training = 1400 buah(700 blok teks dan 700 blok non-teks).
6. Sigma konstanta = 2.
7. Ukuran block = 50x50.
8. Menggunakan *Post-processing* dengan menambahkan *feature extraction* pada SVM.

No	Input	Hasil Pengujian
1		



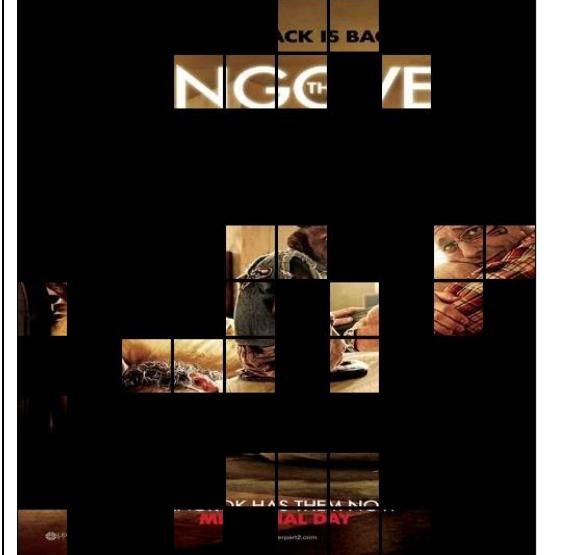
5



6



7



8

