RANCANG BANGUN APLIKASI PEMBACA PARTITUR NOTASI BALOK

SKRIPSI

KONSENTRASI REKAYASA PERANGKAT LUNAK

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun oleh :

Ardhian Dharma Yudha Handoyo

NIM. 0910680006

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

PROGRAM STUDI TEKNIK INFORMATIKA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2013

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI PEMBACA PARTITUR NOTASI BALOK

SKRIPSI

KONSENTRASI REKAYASA PERANGKAT LUNAK

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun oleh :

Ardhian Dharma Yudha Handoyo

NIM. 0910680006

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Ismiarta Aknuranda, S.T., M.Sc., Ph.D	**Budi Darma Setiawan, S.Kom, M.Cs.**
NIK.740719 06 1 1 0079	**NIK. 841015 06 1 1 0090**

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI PEMBACA PARTITUR NOTASI BALOK

SKRIPSI

KONSENTRASI REKAYASA PERANGKAT LUNAK

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :

Ardhian Dharma Yudha Handoyo

NIM. 0910680006

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 17 Mei 2013

Penguji I

Denny Sagita Rusdianto, S.Kom., M.Kom.
NIK. 851124 06 1 1 0250

Penguji II

Fajar Pradana, S.ST., M.Eng.

Penguji III

Sabriansyah Rizqika Akbar, S.T., M.Eng.
NIK. 820809 06 1 1 0084

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 17 Mei 2013
Mahasiswa,

Ardhian Dharma Yudha Handoyo
NIM. 0910680006

KATA PENGANTAR

Puji Syukur kehadirat Tuhan Yang Maha Esa yang telah mencurahkan kasih dan berkat, sehingga skripsi yang berjudul "***Rancang Bangun Aplikasi Pembaca Partitur Notasi Balok***" ini dapat diselesaikan. Skripsi ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer di Program Studi Teknik Informatika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya Malang.

Melalui kesempatan ini, penulis ingin mengucapkan rasa hormat dan terima kasih yang sebesar-besarnya kepada seluruh pihak yang telah memberi bantuan materi maupun moril selama penyelesaian penulisan skripsi ini. Untuk itu penulis ingin menyampaikan penghargaan dan ucapan terima kasih kepada:

1. Ibu Debul Suliyati, Ayah Supriyadi Tri Handoyo, Nenek Tisiyah, Adik Purbaning Dharma Shanti Handoyo dan seluruh keluarga besar atas segala nasehat, kasih sayang, perhatian dan kesabarannya kepada penulis, serta yang senantiasa tiada henti – hentinya memberikan doa dan semangat demi terselesaikannya skripsi ini.
2. Bapak Ir. Sutrisno, M.T, Bapak Ir. Heru Nurwasito, M.Kom, Bapak Himawat Aryadita, S.T, M.Sc, dan Bapak Eddy Santoso, S.Kom selaku Ketua, Wakil Ketua 1, Wakil Ketua 2 dan Wakil Ketua 3 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
3. Bapak Drs. Marji, M.T dan Bapak Issa Arwani, S.Kom, M.Sc selaku Ketua dan Sekretaris Program Studi Teknik Informatika Universitas Brawijaya.
4. Bapak Ismiarta Aknuranda, ST., M.Sc., Ph.D dan Bapak Budi Darma Setiawan, S.Kom, M.Cs. selaku dosen pembimbing skripsi yang telah dengan sabar membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
5. Bapak Ir. Sutrisno, M.T selaku dosen penasehat akademik yang selalu memberikan nasehat kepada penulis selama menempuh masa studi.
6. Bapak Eriq Muhammad Adams, S.T, M.Kom, Bapak Denny Sagita Rusdianto, S.Kom., M.Kom, Bapak Afif Supianto, S.Si, M.Kom. yang

telah banyak membantu dan memberi penulis inspirasi dalam penulisan skripsi ini.

7. Seluruh Dosen Teknik Informatika Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis.
8. Seluruh Civitas Akademika Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.
9. Sahabat - sahabatku Angkatan 2009 Teknik Informatika, terimakasih atas segala bantuannya selama menempuh studi di Teknik Informatika Universitas Brawijaya.
10. Sahabat-sahabat Paduan Suara Mahasiswa Universitas Brawijaya (*Brawijaya University Student Choir*) yang memberikan bantuan, dukungan serta semangat.
11. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun yang tidak langsung demi terselesaikannya skripsi ini.

Hanya doa dan rasa terima kasih yang dapat penulis berikan semoga Tuhan membalas seluruh kebaikan dan berkat yang melimpah. Penulis menyadari bahwa skripsi ini jauh dari sempurna dan banyak kekurangan. Untuk itu, saran dan kritik yang membangun sangat penulis harapkan. Akhir kata, penulis berharap semoga skripsi ini dapat memberikan sesuatu yang bermanfaat bagi semua pihak yang membacanya.

Malang, Mei 2013

Penulis

ABSTRAK

Ardhian Dharma Yudha Handoyo. 2013. : Rancang Bangun Aplikasi Pembaca Partitur Notasi Balok. Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya. Dosen Pembimbing : Ismiarta Aknuranda, ST., M.Sc., Ph.D dan Budi Darma Setiawan, S.Kom, M.Cs.

Partitur musik merupakan dokumen standar dalam penulisan notasi dan simbol-simbol musik. Partitur musik yang beredar kebanyakan merupakan partitur notasi balok. Pembacaan partitur notasi balok memerlukan keahlian khusus dan relatif sulit, khususnya bagi orang awam dan pemusik pemula di Indonesia yang mana lebih terbiasa dengan notasi angka. Oleh karena itu, diperlukan pengembangan suatu aplikasi yang dapat membantu proses pembacaan partitur notasi balok.

Aplikasi Pembaca Partitur Notasi Balok merupakan aplikasi *desktop* yang bertujuan untuk mendukung orang awam dan pemula untuk membaca partitur notasi balok dalam bentuk notasi angka. Aplikasi ini dirancang dengan menggunakan *Reuse-Oriented Model* yang dimodelkan dengan notasi-notasi UML (*Unified Modelling Language*), diimplementasikan dengan bahasa pemrograman Java, dan didukung dengan *image processing library (JHLabs)* dan *audio programming library (JFugue)*. Aplikasi dikembangkan dengan menerapkan pola-pola perancangan (*design pattern*) dalam pembangunan arsitektur aplikasi. Pengujian aplikasi meliputi pengujian fungsional, pengujian akurasi dan pengujian performa. Pengujian fungsional menunjukkan bahwa implementasi dan fungsional aplikasi telah memenuhi kebutuhan yang telah dijabarkan pada tahap analisis kebutuhan. Pengujian akurasi menunjukkan akurasi sebesar 99% untuk citra partitur yang berasal dari PDF atau *Sibelius*, namun mengalami penurunan akurasi untuk citra partitur hasil *scanning* menjadi sebesar 85%. Pengujian performa menunjukkan bahwa waktu eksekusi aplikasi lebih cepat daripada waktu yang diperlukan oleh manusia.

Kata Kunci: *image processing*, partitur, notasi balok, *reuse-oriented model*, *JHLabs, JFugue*

ABSTRACT

Ardhian Dharma Yudha Handoyo. 2013. : Rancang Bangun Aplikasi Pembaca Partitur Notasi Balok. Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.
Supervisor : Ismiarta Aknuranda, ST., M.Sc., Ph.D and Budi Darma Setiawan, S.Kom, M.Cs.

Sheet music is a standard document in writing musical notation. Most sheet music in circulation today is written in modern (western) musical symbols. The reading of this sheet music requires special skills and is relatively difficult for laypersons and beginners in Indonesia, who are more familiar with chiper notation (numbered musical notation).Therefore, it is important to develop an application that can help them read sheet music.

Aplikasi Pembaca Partitur Notasi Balok is a desktop application aimed to support laypersons and beginners to read sheet music into chiper notation. It was designed using the Reuse-Oriented Model, modeled with UML (Unified Modeling Language) notations, implemented withthe Java programming language, and supported by both Java-based image processing library (JHLabs) and audio programming library (JFugue). The application applied several design patterns in the development of its architecture. The aplication was tested with functional testing, accuracy testing,and performance testing. Functional testing showed that the application functionalities fulfill the needs specified from the requirement analysis activity. Accuracy testing showed a high accuracy of 99% for PDF or *Sibelius* reading, but lower accuracy of 85% for scanned sheet music reading. Performance testing showed that the application's execution time is faster than the time needed by humans to run the same operations.

Keywords: image processing, sheet music, chipper notation, reuse-oriented models, *JHLabs*, *JFugue*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada mulanya musik merupakan pengiring sebuah upacara kepercayaan [HAR-01]. Hingga pada akhirnya musik dapat dijadikan salah satu bahasa universal yang digunakan di seluruh dunia. Para pencipta lagu menyebarkan informasi tentang lagu yang ia ciptakan kepada orang lain dengan cara menyanyikan lagu ciptaannya dan orang yang mendengarkan harus mengingat lagu tersebut. Cara seperti ini memiliki banyak kelemahan, karena tiap orang memiliki kemampuan yang berbeda. Hingga akhirnya, lagu yang dinyanyikan tidak sama dengan lagu aslinya.

Biarawan Benediktin, Guido d'Arezzo menemukan cara penulisan not-not musik dengan simbol-simbol yang dituliskan pada dokumen pada abad ke-11 [FEL-07] [HAR-01]. Dokumen standar ini kemudian disebut partitur musik. Partitur berisi lima garis horizontal yang saling berjarak sama. Garis ini disebut paranada. Selanjutnya garis ini berisi simbol-simbol yang menunjukkan tinggi rendahnya nada, durasi nada, serta waktu untuk berhenti. Sejak itu musik mengalami perkembangan yang signifikan.

Partitur mengalami transformasi di Indonesia. Hasil dari transformasi itu biasa disebut dengan notasi angka. Notasi angka merupakan representasi dari simbol-simbol partitur yang diubah menjadi angka. Notasi angka terdiri dari angka 1 – 7 (do re mi fa sol la si) dan untuk membedakan tinggi rendahnya oktaf diberikan tanda titik pada angka tersebut. Dengan adanya transformasi partitur, maka orang awam dapat membaca partitur dengan lebih mudah tanpa harus belajar tentang teori musik.

Citra partitur merupakan representasi digital dari partitur notasi balok. Citra partitur inilah yang nantinya akan diproses guna membantu proses pembacaan partitur bagi orang awam. Citra partitur memerlukan penanganan berbeda dibandingkan dengan citra lain. Garis paranada yang terdapat dalam partitur tidak dapat dihilangkan begitu saja, karena menyebabkan banyak informasi yang hilang (*loss of information*). Aplikasi yang menangani citra partitur untuk diambil informasinya disebut *Optical Music Recognition* [BEL-01].

Berdasarkan latar belakang adanya transformasi notasi balok menjadi notasi angaka, penulis mengambil judul "**Rancang Bangun Aplikasi Pembaca Partitur Notasi Balok"** dengan harapan aplikasi ini dapat membantu para penikmat musik, khususnya masyarakat awam maupun siswa untuk belajar membaca partitur dengan mudah.

1.2 Rumusan Masalah

Mengacu permasalahan yang diuraikan dalam latar belakang, maka rumusan masalah bisa ditekankan pada:

1. Menganalisis kebutuhan *Aplikasi Pembaca Partitur Notasi Balok.*
2. Merancang *Aplikasi Pembaca Partitur Notasi Balok.*
3. Mengimplementasikan *Aplikasi Pembaca Partitur Notasi Balok.*
4. Menguji *Aplikasi Pembaca Partitur Notasi Balok.*

1.3 Batasan Masalah

Dalam perencanaan dan pembuatan skripsi ini perlu dilakukan pembatasan masalah. Pembatasan masalah yang diajukan dalam skripsi ini antara lain:

1. Pembahasan difokuskan pada rekayasa perangkat lunak dari pembuatan *Aplikasi Pembaca Partitur Notasi Balok.*
2. Citra partitur masukan aplikasi yang digunakan merupakan partitur vokal
3. Citra partitur berasal dari partitur cetak (*printed document),* bukan partitur tulisan tangan (*handwritten document)*
4. Citra partitur yang digunakan memiliki nada dasar C
5. Citra partitur menggunakan *clef* G
6. Data keluaran berupa citra yang berisi keterangan notasi tanpa nilai ketukan dan audio notasi dari citra partitur
7. Simbol musik yang dapat dideteksi aplikasi, yaitu *clef, whole note, half note, quarter note, eight note* dan *rest note*
8. Bahasa pemrograman yang digunakan adalah *Java*
9. Sistem operasi yang digunakan adalah *Windows 7 Profesional*

1.4 Tujuan

Tujuan penyusunan skripsi ini adalah merancang, mengimplementasikan, dan menguji *Aplikasi Pembaca Partitur Notasi Balok* sehingga dapat membantu para penikmat musik maupun siswa untuk belajar membaca partitur dengan mudah.

1.5 Manfaat

Manfaat yang dapat diperoleh dari penelitian ini adalah :

a. Bagi penulis
   1. Menerapkan ilmu yang telah didapat pada Teknik Informatika Universitas Brawijaya.
   2. Mendapatkan pemahaman tentang *optical music recognition* sehingga dapat mengimplementasikannya sesuai dengan kebutuhan.

b. Bagi pengguna
   1. Mempermudah pengguna untuk membaca dan mempelajari partitur.

1.6 Sistematika Penulisan

Sistematika penulisan laporan skripsi ini adalah sebagai berikut :

1. BAB I Pendahuluan

   Memuat latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, sistematika penulisan
2. BAB II Dasar Teori

   Membahas teori-teori yang mendukung dalam perencanaan dan pembuatan aplikasi
3. BAB III Metodologi Penelitian

   Berisi tentang metodologi penelitian dan perencanaan aplikasi
4. BAB IV Perancangan

   Membahas analisis kebutuhan dan perancangan perangkat lunak sesuai dengan dasar teori dan literatur yang ada

5. BAB V Implementasi

   Membahas implementasi dari *Aplikasi Pembaca Partitur Notasi Balok* sesuai dengan perancangan perangkat lunak yang telah dibuat

6. BAB VI Pengujian dan Pembahasan

   Memuat hasil pengujian dan pembahasan terhadap perangkat lunak yang telah direalisasikan.

7. BAB VII Penutup

   Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak yang dikembangkan dalam skripsi ini serta saran – saran untuk pengembangan lebih lanjut

1.7 Jadwal Penulisan

Penulisan skripsi ini direncanakan dengan rincian sebagai berikut:

Tabel 1.1 Jadwal Penulisan

NO	PROSES PENULISAN :	BULAN DAN MINGGU KE:																			
		Bulan I				Bulan II				Bulan III				Bulan IV				Bulan V			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Studi literatur																				
2	Analisis Kebutuhan																				
3	Perancangan Perangkat Lunak																				
4	Implementasi Perangkat Lunak																				
5	Pengujian Perangkat Lunak																				
6	Penulisan penelitian																				

Sumber : [Pendahuluan]

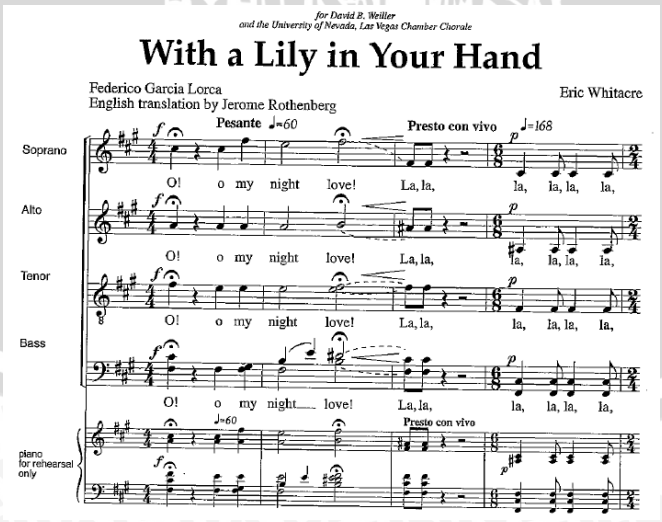
BAB II
DASAR TEORI

2.1 Partitur

Partitur merupakan bentuk tertulis pada komposisi musik [KAM-08]. Bagian dari sebuah partitur yang memiliki peranan penting dalam pembuatan proyek, antara lain paranada, tanda mula dan tangga nada.

2.1.1 Partitur Vokal

Partitur vokal merupakan pengurangan dari partitur lengkap sebuah karya musik (sebagai contoh opera, *musical*, *oratorio*, *cantata*, dll.) untuk menunjukkan bagian vokal (solo dan paduan suara) dalam garis biramanya dan bagian piano dari orkestra (biasanya dua tangan) di bawah bagian vokal; bagian orkestra yang murni dari partitur juga dikurangi untuk piano. Bila sebagian dari karya musik merupakan *a cappela*, pengurangan piano dari bagian vokal sering ditambahkan untuk membantu dalam latihan (sering terjadi dalam kasus partitur *a cappela* rohani). Walaupun tidak digunakan dalam pertunjukan, partitur vokal merupakan cara yang baik bagi seorang solois dan penyanyi paduan suara untuk mempelajari musik dan berlatih terpisah dari bagian instrumen. Partitur vokal umumnya tidak melibatkan dialog yang diucapkan, kecuali tanda-tanda tertentu.

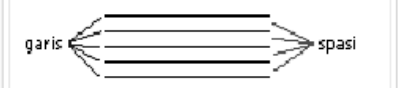


Gambar 2.1 Partitur vokal

Sumber : [ERI-01]

2.1.2 Paranada

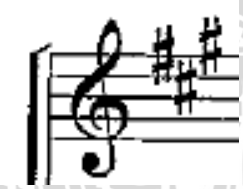
Dalam notasi musik balok, paranada adalah lima garis horisontal tempat not ditulis [HAR-01]. Not dapat diletakkan di garis atau di antara garis (spasi) paranada. Garis paranada diberi nomor dari bawah ke atas; garis paling bawah disebut garis kelima dan garis paling atas disebut garis pertama. Not yang terletak di garis atau spasi lebih tinggi berarti memiliki tinggi nada lebih tinggi. Tanda kruis, mol dan tanda musik lainnya juga ditulis dalam paranada.



Gambar 2.2 Garis paranada
Sumber : [PHI-07]

2.1.3 Tanda Mula

Tanda mula adalah kumpulan dari tanda kruis (♯) atau mol (♭) yang terdapat pada paranada yang terletak di depan *clef* suatu partitur [HAR-01]. Tanda mula berfungsi untuk menentukan nada dasar dari suatu lagu dan menentukan nama tangga nada. Ada beberapa macam tanda mula yang disajikan dalam bentuk Tabel 2.1 dan Tabel 2.2. Contoh tanda mula dapat diliha di Gambar 2.3



Gambar 2.3 Tanda mula
Sumber : [ERI-01]

Tabel 2.1 Daftar tanda mula kruis

Tanda mula	Jumlah tanda kruis
C mayor	0
G mayor	1
D mayor	2
A mayor	3
E mayor	4
B mayor	5
F♯ mayor	6
C♯ mayor	7

Sumber : [PHI-07]

Tabel 2.2 Daftar tanda mula mol

Tanda mula	Jumlah tanda mol
C mayor	0
F mayor	1
B♭ mayor	2
E♭ mayor	3
A♭ mayor	4
D♭ mayor	5
G♭ mayor	6
C♭ mayor	7

Sumber : [PHI-07]

Mengacu pada batasan masalah pada bab sebelumnya, citra partitur yang digunakan dalam skripsi ini memiliki tanda mula C, maka citra partitur yang akan diujikan adalah citra partitur yang tidak memilki tanda kruis dan mol pada tanda mula (Gambar 2.5).

2.1.4 Tangga Nada

Tangga nada merupakan urutan dari suatu notasi yang memiliki pola khusus [HAR-01]. Setiap nada dasar memiliki tangga nada yang berbeda. Berikut ini contoh tangga nada yang disajaikan dalam bentuk Gambar 2.4

Nada Dasar	Nada ke-							Nada chord mayor berdasar rumus 1-3-5	Chord Mayor	Notasi Balok
	1	2	3	4	5	6	7			
C	C	D	E	F	G	A	B	C - E - G	C	
C♯	C♯	D♯	F	F♯	G♯	A♯	C	C♯ - F - G♯	C♯	
D	D	E	F♯	G	A	B	C♯	D - F♯ - A	D	
D♯	D♯	F	G	G♯	A♯	C	D	D♯ - G - A♯	D♯	
E	E	F♯	G♯	A	B	C♯	D♯	E - G♯ - B	E	
F	F	G	A	A♯	C	D	E	F - A - C	F	
F♯	F♯	G♯	A♯	B	C♯	D♯	F	F♯ - A♯ - C♯	F♯	
G	G	A	B	C	D	E	F♯	G - B – D	G	
G♯	G♯	A♯	C	C♯	D♯	F	G	G♯ - C - D♯	G♯	
A	A	B	C♯	D	E	F♯	G♯	A - C♯ - E	A	
A♯	A♯	C	D	D♯	F	G	A	A♯ - D - F	A♯	
B	B	C♯	D♯	E	F♯	G♯	A♯	B - D♯ - F♯	B	

Gambar 2.4 Tangga nada pada nada dasar mayor

Sumber : [PHI-07]

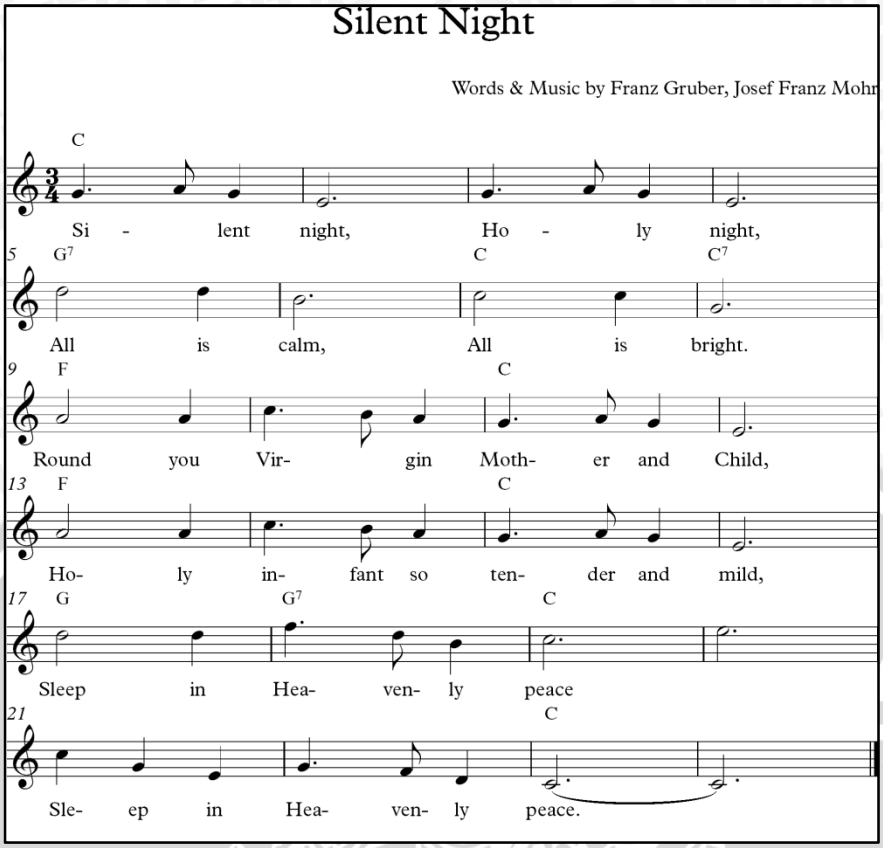
2.2 Citra

Definisi citra menurut kamus webster adalah “suatu representasi, kemiripan atau imitasi dari suatu obyek atau benda” [ACH-05].

Citra skala keabuan memberi kemungkinan warna yang lebih banyak daripada citra biner, karena ada nilai-nilai lain di antara nilai minimum (biasanya = 0) dan nilai maksimumnya. Banyaknya kemungkinan nilai dan nilai maksimumnya bergantung pada jumlah bit yang digunakan. Format citra ini disebut skala keabuan karena pada umumnya warna yang dipakai adalah antara hitam sebagai warna minimal dan warna putih sebagai warna maksimalnya, sehingga warna antaranya adalah abu-abu [ACH-05].

2.2.1 Citra Partitur

Citra partitur merupakan representasi dari partitur. Citra partitur inilah yang nantinya akan diujikan pada aplikasi. Contoh citra partitur dapat dilihat pada Gambar 2.5



Gambar 2.5 Citra partitur

Sumber : [Perancangan]

2.2.2 Grayscale

Proses *grayscale* ini merupakan proses perubahan model warna RGB menjadi model warna derajad keabuan [GON-07]. Proses ini diawali dengan pengambilan nilai RGB di tiap piksel gambar dan diuraikan menurut unsur *red*, *green* dan *blue*. Setelah mendapatkan nilai setiap unsur tersebut maka nilai ketiganya akan dijumlah dan dibagi 3. Hasil pembagian dikembalikan lagi sebagai nilai piksel. Untuk lebih jelasnya dapat dilihat pada Persamaan 2-1

$$\boldsymbol{grayscale}\ (\boldsymbol{x},\boldsymbol{y}) = \frac{\boldsymbol{r}(\boldsymbol{x},\boldsymbol{y})+\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y})+\boldsymbol{b}(\boldsymbol{x},\boldsymbol{y})}{\mathbf{3}} \qquad \textbf{(2-1)}$$

dengan :

grayscale = nilai *grayscale* pada piksel (x,y)

r = nilai warna merah pada piksel (x,y)

g = nilai warna hijau pada piksel (x,y)

b = nilai warna biru pada piksel (x,y)

2.2.3 Binerisasi / *Tresholding*

Thresholding adalah proses mengubah citra berderajat keabuan menjadi citra biner atau hitam putih sehingga dapat diketahui daerah mana yang termasuk obyek dan *background* dari citra secara jelas [GON-07]. Citra hasil *thresholding* biasanya digunakan lebih lanjut untuk proses pengenalan obyek serta ekstraksi fitur. Menurut [GON-07] metode *thresholding* secara umum dibagi menjadi dua, yaitu :

1. *Thresholding* global

   Thresholding dilakukan dengan mempartisi histogram dengan menggunakan sebuah *threshold* (batas ambang) global T, yang berlaku untuk seluruh bagian pada citra
2. *Thresholding* adaptif

   Thesholding dilakukan dengan membagi citra menggunakan beberapa sub citra. Lalu pada setiap sub citra, segmentasi dilakukan dengan menggunakan *threshold* yang berbeda.

Pada aplikasi yang akan dibuat menggunakan *thresholding* global, dimana batas nilai sudah ditentukan dan berlaku untuk semua citra. Persamaan 2-2 merupakan persamaan yang digunakan dalam proses *thresholding.*

$$g(x,y) = \begin{cases} 1 \; if \; f(x,y) > T \\ 0 \; if \; f(x,y) \leq T \end{cases} \quad \textbf{(2-2)}$$

dengan :

g = nilai akhir piksel (x,y) pada citra

f = nilai awal piksel (x,y) pada citra

T = nilai *threshold*

2.2.4 Deteksi garis paranada

Proses deteksi garis paranada digunakan untuk menginisialisasi posisi acuan notasi [CIU-10]. Proses deteksi garis paranada memiliki tahapan yang hampir sama seperti yang ditulis dalam [GON-07], namun terdapat beberapa modifikasi sesuai dengan yang ditulis pada jurnal [CIU-10]. Proses deteksi dilakukan dengan menghitung proyeksi horisontal suatu citra, jika jumlah piksel berwarna hitam melebihi dari setengah kali lebar citra maka garis itu merupakan garis paranada.

$$V(m) = \begin{cases} 1, P_{opt}(m) > T/2 \\ 0, otherwise \end{cases} \quad \textbf{(2-3)}$$

dengan :

P_{opt} = proyeksi horisontal tiap baris

V(m) = kandidat garis paranada

2.2.5 Pengahapusan garis paranada

Proses pengahpusan garis parana diperlukan agar setiap simbol musik dapat disegmentasi dengan baik [BOL-12]. Untuk setiap garis paranada yang terdeksi akan dihapus secara horisontal. Namun apabila garis melewati musik simbol, maka piksel garis paranada tidak dihapus.

$$S = \begin{cases} Staff\ Line\ len > T \\ Musical\ Note\ otherwise \end{cases} \quad \textbf{(2-4)}$$

dengan :

S = segmen dari garis paranada

len = *vertical run* pada segmen garis paranada

T = nilai batas pengahpusan, diambil dari nilai minimum antara 2 kali *staff height* atau *staff height + staff space*

2.2.6 Region Growing

Region growing merupakan pengelompokan piksel-piksel berdasarkan kriteria yang telah ditetapkan sebelumnya [GON-07]. Metode *region growing* yang digunakan adalah *region growing* dengan 8 piksel ketetanggan.

$$I(x,y) = \begin{cases} 0\ background\ pixel \\ 1\ foreground\ pixel \\ 2, 3, \ldots region\ label \end{cases} \quad \textbf{(2-5)}$$

dengan :

I(x,y) = piksel pada koordinat x,y dalam citra

2.2.7 Erosi

Erosi merupakan proses pengurangan piksel-piksel kecil pada citra di luar objek utama [ACH-05][GON-07]. Proses erosi pada aplikasi ini digunakan untuk menghilangkan tangkai notasi pada notasi balok. Untuk algoritma erosi dapat dilihat pada Persamaan 2-5.

$$A\ \theta\ B = \{z | (B)_z \subseteq A\} \quad \textbf{(2-6)}$$

dengan :
A = piksel suatu citra
B = matrix *filter*

2.2.8 Pengenalan simbol musik

Proses pengenalan simbol musik ditujukan untuk mengenali objek-objek yang ada pada citra. Pada skripsi ini, semua objek akan dikenali sebagai simbol musik. Proses pengenalan pada skripsi ini menggunakan metode *template matching.* Algoritma *template matching* yang digunakan dalam skripsi ini adalah *correlation coefficient* [BUR-07].

$$C_L(r,s) = \frac{\sum_{(i,j)\in R}\left(I(r+i,s+j).R(i,j)\right) - K.\bar{I}_{r,s}.\bar{R}}{\left[\sum_{(i,j)\in R} I^2(r+i,s+j) - K.\bar{I}^2_{r,s}\right]^{1/2}.S_R} \quad \textbf{(2-7)}$$

dengan :
R = citra refrensi
I = citra yang akan dikenali
K = jumlah piksel pada cira R
S_R = *variant* dari citra

Proses pengenalan ini diawali dengan pencarian nilai *mean* dan *variance* dari piksel-piksel dalam citra *template*. Selanjutnya tiap objek hasil segmentasi dilakukan pencarian nilai *mean* dan *variance* dari piksel-piksel objek tersebut. Selanjutnya dari nilai *mean* dan *variance* tiap objek akan dihitung tingkat kemiripannya dengan citra *template.* Besaran kemiripan citra berkisar antara 0 – 1. Pada skripsi ini, ambang kemiripan suatu objek sebesar 0,7 yang artinya apabila suatu objek memiliki kemiripan diatas 0,7 maka objek dikenali dengan citra *template* tersebut.

2.2.9 *Center of Mass*

Center of Mass merupakan proses untuk menentukan titik tengah atau pusat massa suatu objek pada citra. Metode *center of mass* yang digunakan memiliki kesamaan dengan metode pada fisika [SAR-09]. Proses ini digunakan untk mencari titik tengah dari suatu objek notasi. Nilai titik tengah inilah yang nantinya akan dibandingkan dengan posisi garis paranada untuk menentukan notasi. Proses ini diawalai dengan menghitung jumlah piksel berwarna hitam dalam suatu objek notasi. Proses dilanjutkan dengan menjumlah koordinat piksel

berwarna hitam dan diakhiri dengan membagi hasilnya dengan jumlah piksel berwarna hitam pada objek notasi.

$$x_o = \frac{\sum_{i=1}^{n} m_i x_i}{\sum_{i=1}^{n} m_i} \quad \textbf{(2-8)}$$

$$y_o = \frac{\sum_{i=1}^{n} m_i y_i}{\sum_{i=1}^{n} m_i} \quad \textbf{(2-9)}$$

dengan :

m = jumlah piksel setiap objek

x_o = koordinat titik tengah pada sumbu x

y_o = koordinat titik tengah pada sumbu y

2.3 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak adalah disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal spesifikasi sistem sampai pemeliharaan sistem setelah digunakan [SOM-11].

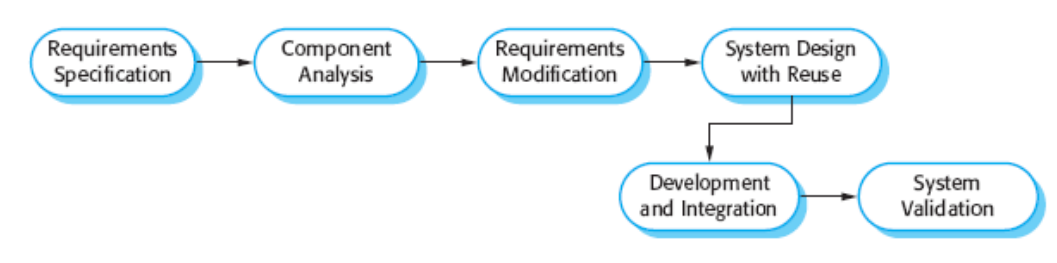
Rekayasa perangkat lunak lebih fokus pada bagaimana membuat perangkat lunak yang memenuhi karakteristik dasar berikut [SOM-11] :

- Perangkat lunak harus dapat terus dipelihara setelah perangkat lunak selesai dibuat seiring berubahnya kebutuhan sesuai proses bisnis dan lingkungan (*maintainability*)
- Perangkat lunak dapat diandalkan sehingga tidak mengakibatkan kerugian secara fisik maupun ekonomi. Perangkat lunak juga dapat menolak akses dari pengguna yang mencurigakan yang dapat membahayakan sistem (*dependability and security*)
- Perangkat lunak tidak melakukan pemborosan dalam penggunaan sumber daya sistem, seperti memori dan *processor cycles* (*efficiency*)
- Perangkat lunak harus dapat diterima oleh pengguna yang bersangkutan. Hal ini berarti perangkat lunak harus dapat dimengerti, digunakan dan sesuai dengan sistem lain yang mereka gunakan (*acceptability*)

Dari karakteristik di atas maka perangkat lunak yang baik adalah perangkat lunak yang dapat memenuhi kebutuhan pelanggan (*customer*) atau pemakai perangkat lunak (*user)* atau berorientasi pada pelanggan atau pemakai perangkat lunak, bukan berorientasi pada pembuat atau pengembang perangkat lunak.

2.4 *Reuse-oriented Model*

Model proses untuk rekayasa perangkat lunak dipilih sesuai dengan sifat dari proyek dan aplikasi yang akan dibuat. Salah satu dari model proses yang digunakan adalah *Reuse-oriented. Reuse-oriented* merupakan rekayasa perangkat lunak yang menggunakan pendekatan berbasis tersedianyanya jumlah komponen *reusable* yang signifikan. Proses pengembangan sistem berfokus pada integrasi komponen-komponen ini ke dalam sistem daripada mengembangkan sistem dari dasar [SOM-11]. Model proses secara umum dari pengembangan berbasis *reuse* dapat dilihat pada Gambar 2.6.



Gambar 2.6 *Reuse-Oriented Model*

Sumber : [SOM-11]

Walaupun inisialisasi spesifikasi kebutuhuan dan validasi pada proses ini sama dengan proses pengembangan perangkat lunak yang lain, namun terdapat perbedaan pada pertengahan proses, yaitu dengan adanya proses *reuse-oriented*. Menurut *Sommerville* [SOM-11] proses ini antara lain:

1. Analisis komponen

   Dari spesifikasi kebutuhan yang telah ditentukan, dicari komponen yang dapat mengimplementasi spesifikasi tersebut. Biasanya, tidak ada komponen yang benar-benar mampu memenuhi spesifikasi dan mungkin hanya beberapa komponen yang memenuhi persyaratan fungsional dari spesifikasi kebutuhan
2. Modifikasi kebutuhan

   Pada proses ini, kebutuhan dianalisis menggunakan informasi tentang komponen yang telah ditemukan. Kemudian, kebutuhan ini dimodifikasi untuk merefleksi komponen yang tersedia. Jika terdapat modifikasi yang tidak mungkin untuk diterapkan, proses analisis komponen dapat dilakukan kembali untuk mencari solusi alternatif.

3. Desain sistem dengan *reuse*

   Pada proses ini, perancang mempertimbangkan komponen yang dapat digunakan ulang dan mengatur *framework* untuk mendukung komponen ini. Beberapa perangkat lunak baru mungkin perlu dirancang mulai awal apabila tidak ada komponen yang dapat digunakan ulang

4. Pengembangan dan integrasi

   Perangkat lunak yang tidak dapat diperoleh bebas dapat dikembangkan dengan komponen-komponen dan sistem COTS diintegrasikan untuk menciptakan suatu sistem yang baru. Integrasi sistem dalam model ini dapat menjadi bagian proses pengembangan daripada kegiatan yang terpisah.

Sommerville [SOM-11] juga menyebutkan tiga tipe komponen perangkat lunak yang dapat digunakan dalam proses *reuse-oriented*, yaitu :

1. *Web service* yang dikembangkan berdasarkan *service standard* dan tersedia untuk digunakan secara *remote*.
2. Koleksi *object* yang dikembangkan sebagai *package* untuk diintegrasikan dengan sebuah *framework* komponen seperti .NET dan J2EE.
3. Sistem perangkat lunak yang berdiri sendri yang dapat dikonfigurasi untuk digunakan dalam lingkungan tertentu.

Model *reuse-oriented* memiliki keuntungan jelas dalam mengurangi jumlah perangkat lunak yang dikembangkan dan mengurangi biaya dan resiko.

Cakupan *reuse-oriented* yang digunakan pada aplikasi dalam skripsi ini nantinya adalah *program library* dan *design pattern*. *Program library* digunakan untuk proses pengolahan citra partitur, sedangkan *design patern* digunakan untuk merancang arsitektur aplikasi agar dapat digunakan kembali pada tahap pengembangan.

2.5 Pola-pola Perancangan

Pola-pola perancangan (*design pattern*) merupakan suatu teknik yang dapat mempermudah pembuatan desain dan arsitektur aplikasi yang bersifat *reuse* [GAM-01]. Dengan adanya pola-pola perancangan, suatu aplikasi dapat dikembangkan dengan lebih mudah. Pola-pola perancangan bahkan dapat

meningkatkan proses dokumentasi dan pemeliharaan sistem yang ada dengan penyempurnaan spesifikasi kelas dan interaksi antar obejk yang mendasarinya.

Pola-pola perancangan sendiri dibagi menjadi 3 kelompok besar berdasarkan tujuannya (*purpose*), yaitu [GAM-01] :

1. *Creational*

   Pola-pola perancangan dalam kelompok ini berperan pada proses pembuatan objek. Adapun pola-pola perancangan yang masuk dalam kelompok ini antara lain *abstract method, abstract factory, builder, prototype,* dan *singleton.*

2. *Structural*

   Pola-pola perancangan dalam kelompok ini berurusan dalam komposisi kelas atau objek. Adapun pola-pola perancangan yang masuk dalam kelompok ini antara lain *adapter, bridge, composite, decorator, fascade,* dan *proxy.*

3. *Behavioral*

   Pola-pola perancangan dalam kelompok ini mengkarakterisasi cara suatu kelas atau objek yang saling berinteraksi dan mendistribusakan tanggung jawab. Adapun pola-pola perancangan yang masuk dalam kelompok ini antara lain *intepreter, template method, chain of responsibility, command, iterator, mediator, memento, flyweight, observer, state, strategy, visitor.*

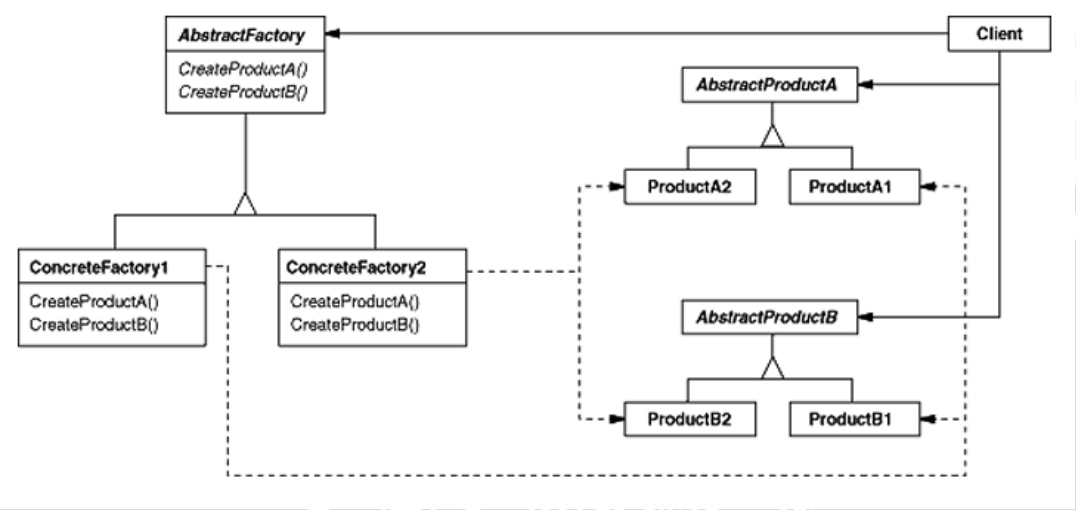
Pada proses skripsi ini, pola-pola perancangan yang digunakan dalam perancangan aplikasi adalah *abstarct factory*, *template method,* dan *strategy.*

2.5.1 *Abstract Factory Pattern*

Abstract Factory merupakan sebuah pola perancangan yang menyediakan suatu *interface* untuk membuat objek dari suatu kelas dalam keluarga yang sama tanpa menentukan nama kelas objek tersebut [GAM-01]. Struktur pola perancangan ini dapat dilihat pada Gambar 2.7

Pola perancangan ini digunakan apabila menemukan skenario perancangan aplikasi sebagai berikut :

1. Sistem harus bersifat *independent* terhadap bagaimana produk itu dibuat, disusun dan dipresentasikan
2. Sistem harus dikonfigurasi dengan satu keluarga dari beberapa produk
3. Keluarga dari produk harus dirancang untuk dapat digunakan bersama-sama
4. Perancang ingin menunjukan *interface* dari produk bukan menunjukkan implementasinya



Gambar 2.7 Struktur *Abstract Factory*

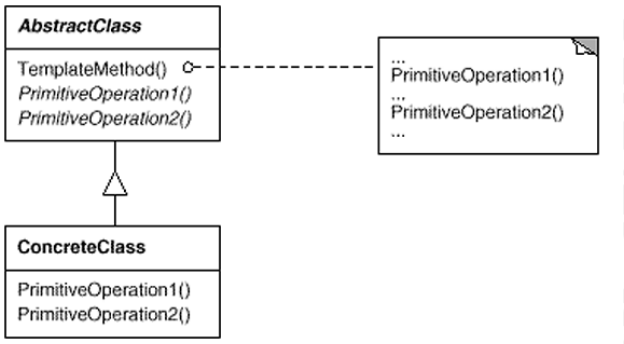
Sumber : [GAM-01]

2.5.2 *Template Method Pattern*

Template Method sebuah pola perancangan yang diperuntukan pendefinisian kerangka algoritma dalam sebuah operasi, menangguhkan beberapa langkah untuk *subclass*. Pola perancangan ini memungkinkan *subclass* untuk mendefinisikan langkah-langkah tertentu dari suatu algoritma tanpa mengubah struktur algoritma dasar. Struktur pola perancangan ini dapat dilihat pada Gambar 2.8.

Pola perancangan ini digunakan apabila menemukan skenario perancangan aplikasi sebagai berikut :

1. Untuk mengimplementasikan bagian yang sama dari suatu algoritma dan menyerahkan *subclass* untuk menerapkan perilaku yang bervariasi
2. Untuk menghindari duplikasi kode program
3. Untuk mengontrol perluasan *subclass*



Gambar 2.8 Struktur *Template Method*

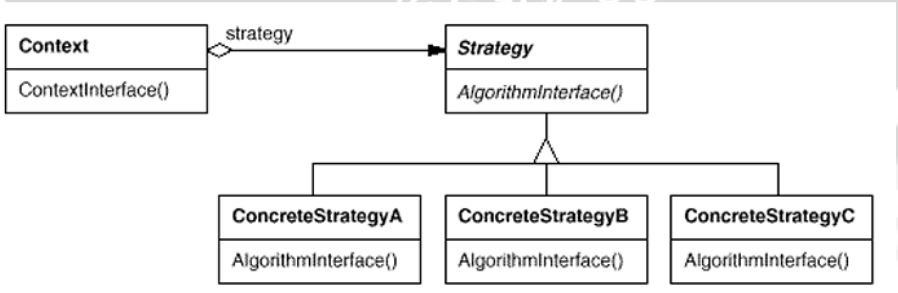
Sumber : [GAM-01]

2.5.3 *Strategy Pattern*

Strategy Pattern sebuah pola perancangan yang diperuntukkan mendefinisikan keluarga algoritma, membungkus masing-masing, dan membuat algoritma saling menggantikan. Strategi ini memungkinkan algoritma bervariasi secara independen dari klien yang menggunakannya. Struktur pola perancangan ini dapat dilihat pada Gambar 2.9.

Pola perancangan ini digunakan apabila menemukan skenario perancangan aplikasi sebagai berikut :

1. Terdapat kelas yang saling terkait memilki banyak perbedaan hanya dalam perilaku mereka
2. Perancang memerlukan beberapa algoritma yang berbeda pada proses yang sama
3. Untuk menghindari proses kondisi yang sangat banyak



Gambar 2.9 Struktur *Strategy*

Sumber : [GAM-01]

2.6 Unified Modelling Language

Unified Modelling Language (UML) merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek [ROS-11].

Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Berikut ini penjelasan singkat dari pembagian kategori tersebut.

- Structure diagrams yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
- Behavior diagrams yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
- Interaction diagrams yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antarsubsistem pada suatu sistem.

2.6.1 *Use Case Diagram*

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (behavior) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat [ROS-11].

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefisian apa yang disebut aktor dan *use case*.

- Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan di buat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

- *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antarunit atau aktor.

2.6.2 *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi [ROS-11].

- Atribut merupakan variabel-variabel yyang dimiliki oleh suatu kelas
- Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

- Kelas main

  Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan
- Kelas yang menangani tampilan sistem

  Kelas yang mendefinisikan dan mengatur tampilan ke pemakai
- Kelas yang diambil dari pendefisian use case

  Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefisian use case
- Kelas yang diambil dari pendefisian data

  Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Jenis-jenis kelas di atas juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada.

2.6.3 *Sequence Diagram*

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirmkan dan diterima antarobjek [ROS-11].

Banyaknya *sequence diagram* yang harus digambar adalah sebanyak pendefisian use case yang memiliki proses sendiri atau yang penting semua use

case yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada *sequence diagram* sehingga semakin banyak use case yang didefinisikan maka *sequence diagram* yang harus dibuat juga semakin banyak.

2.6.4 *Activity Diagram*

Activity diagram merupakan diagram tambahan *use case diagram* yang menggambarkan aliran interaksi dalam skenario tertentu dalam bentuk gambar [PRE-10]. Serupa dengan flowchart, suatu *activity diagram* menggunakan persegi panjang untuk mewakili fungsi sistem tertentu, panah untuk mewakili aliran sistem, belah ketupat untuk menggambarkan percabangan dan garis horisontal untuk menunjukkan bahwa kegiatan paralel sedang terjadi.

2.7 Bahasa Pemrograman Java

Java dikembangkan oleh perusahaan Sun Microsystem. Java menurut definisi dari Sun Microsystem adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer standalone ataupun pada lingkarang jaringan [ROS-11].

Java berdiri di atas sebuah mesin interpreter yang diberi nama Java Virtual Machine (JVM). JVM inilah yang akan membaca bytecode dalam file. *Class* dari suatu program sebagai representasi langsung program yang berisi bahasa mesin. Java merupakan bahasa pemrograman objek murni karena semua kode programnya dibungkus dalam kelas [ROS-11]. Setiap kelas dalam java terdiri dari beberapa *method* yang melakukan tugas-tugas dan mengembalikan informasi saat selesai dijalankan.

Bahasa pemrograman java merupakan pemrograman berorientasi objek [DEI-06]. Kebanyakan para pengembang mengambil keuntungan dari teknologi java tersebut, yaitu dengan memakai ulang kelas-kelas yang telah ada atau biasa disebut dengan *class library* dan lebih dikenal dengan Java API (*Application Programming Interfaces*) [DEI-06]. Java sendiri menyediakan banyak *library* yang tersedia dalam SDK Java. Selain *library* yang telah ada di dalam SDK, pengembang juga bisa mendapatkan *library* yang dikembangkan oleh *vendor* lain. *Library* yang digunakan dalam skripsi ini adalah *library* yang berhubungan

dengan pemrosesan citra digital dan pemrograman musik, yaitu *JHLabs Image Processing* dan *JFugue*. Kedua *library* tersebut merupakan *library* yang berasal dari *vendor* lain.

2.8 JHLabs

JHLabs merupakan sebuah *library* yang dibuat oleh Jerry Huxtable [HUX-12]. Pada situs JHLabs ini banyak terdapat hal yang berhubungan dengan *java*, salah satunya adalah kelas *java* yang berhubungan dengan pengolahan citra.

Java Image Procesessing merupakan sebuah *library* tambahan *java* yang dibuat oleh JHLabs. Dalam *library* ini terdapat beberapa kategori filter, antara lain *color adjustment filters, distortion and wrapping filters, effect filters, texturing filters, blurring and sharpening filters, edge detection, transitions*, dan *alpha channel filters* [HUX-12]. Filter yang digunakan untuk pemrosesan citra pada fase *prepocessing* adalah *grayscale filter* dan *thresholding filter* pada kategori *color adjustment filters.*

2.9 JFugue

JFugue merupakan API untuk pemrograman musik [JFU-13]. Versi JFugue yang dipakai pada skripsi ini adalah versi 4.0. API ini memiliki banyak fitur yang berhubungan dengan musik. Fitur yang akan dipakai dalam skripsi ini adalah *MIDI Player* dan *MIDI Creator.*

2.10 Pengujian

Pengembangan sistem perangkat lunak melibatkan serangkaian kegiatan produksi di mana peluang untuk keteledoran manusia sangat besar. Kesalahan dapat muncul pada awal proses dimana kemungkinan terjadi kekeliruan pada tujuan atau tujuan tidak terspesifikasi secara tepat. Karena ketidakmampuan manusia dalam membuat sesuatu yang sempurna, maka pengembangan perangkat lunak disertai dengan aktivitas penjaminan kualitas [PRE-10].

Pengujian perangkat lunak merupakan elemen penting dari penjaminan kualitas perangkat lunak dan merepresentasikan tinjauan utama spesifikasi, desain, dan pembuatan kode [PRE-10]. Pengujian yang akan digunakan dalam

skripsi ini antara lain pengujian fungsional, pengujian performa dan pengujian akurasi. Hampir seluruh pengujian akan dilakukan pada tahap akhir pembuatan aplikasi.

2.10.1 Pengujian Fungsional

Pengujian fungsional dalam skripsi ini menggunakan pendekatan pengujian *black box*. Pengujian *black-box* atau pengujian *behavioral* berfokus pada persyaratan fungsional dari perangkat lunak [PRE-10]. Dengan demikian, pengujian *black box* memungkinkan perekayasa perangkat lunak memperoleh rangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk sebuah program.

Menurut Pressman [PRE-10], pengujian *black-box* berusaha menemukan kesalahan dalam kategori berikut: (1) fungsi-fungsi yang tidak benar atau belum ada. (2) kesalahan antarmuka, (3) kesalahan dalam struktur data atau akses database eksternal, (4) kesalahan kinerja atau perilaku , dan (5) kesalahan inisialisasi dan terminasi. Pengujian *black-box* cenderung diaplikasikan selama tahap pengujian berikutnya. Karena pengujian *black-box* mengabaikan struktur kontrol, maka perhatian berfokus pada domain informasi. Pengujian didesain untuk menjawab pertanyaan-pertanyaan berikut :

1. Bagaimana validitas fungsional diuji?
2. Bagaimana perilaku dan kinerja sistem diuji?
3. Kelas input apa yang akan membuat kasus uji menjadi baik?
4. Apakah sistem sangat sensitif terhadap nilai input tertentu?
5. Bagaimana batas-batas kelas data yang terisolasi?
6. Kecepatan dan volume data apa yang dapat ditolerir oleh sistem?
7. Apa pengaruh kombinasi tertentu dari data terhadap operasi sistem?

Pengujian fungsional pada skripsi ini nantinya akan menjawab pertanyaan pada poin (1), (2), (3), dan (4) dari beberapa pertanyaan di atas. Pada skripsi ini, untuk menjawab secara detail pertanyaan poin (3) dan (4) digunakan pengujian akurasi yang merupakan bagian dari pengujian fungsional [NAI-08].

2.10.2 Pengujian Performa

Pengujian perfoma (*Performance Test)* merupakan sebuah pengujian yang ditujukan untuk mengetahui perfoma karakteristik sebuah aplikasi [NAI-08]. Beberapa parameter yang dapat digunakan untuk pengujian ini, antara lain :

1. Waktu respon, menunjukkan berapa lama waktu yang dibutuhkan sistem untuk memberikan respon permintaan pengguna
2. Waktu eksekusi, menunjukkan berapa lama waktu yang dibuthkan untuk mengeksekusi perintah yang diberikan pengguna
3. Pemanfaatan sumber daya, menunjukkan berapa banyak sumber daya yang diperlukan untuk menjalankan suatu proses
4. Tingkat lalu lintas, menunjukkan tingkat lalu lintas data dari sistem saat berjalan

Parameter pengujian performa yang digunakan dalam skripsi ini adalah waktu eksekusi dan waktu repon aplikasi. Khusus dalam kasus skripsi ini, waktu eksekusi dan waktu respon diasumsikan sama. Hal ini dikarenakan aplikasi yang akan dibuat merupakan aplikasi *desktop* dan tidak *multiuser*.

2.10.3 Pengujian Akurasi

Pengujian akurasi berfungsi untuk melakukan evaluasi terhadap presisi proses komputasi dan hasil keluaran [NAI-08]. Pada dasarnya pengujian akurasi termasuk dalam proses pengujian fungsional, namun pengujian akurasi dalam skripsi ini lebih spesifik menilai akurasi hasil dari proses pengenalan simbol musik.

Evaluasi performa dari sebuah model klasifikasi didasarkan pada perhitungan hasil pengujian yang prediksi secara benar dan salah oleh model klasifikasi tersebut [TAN-06]. Perhitungan tersebut disusun dalam tabel yang disebut dengan *confusion matrix*. Tabel 2.3 menggambarkan *confusion matrix* dalam permasalahan klasifikasi biner. Setiap masukan f_{ij} dalam tabel ini menunjukkan jumlah hasil dari *class i* yang prediksi sebagai *class j*. Sebagai contoh, f_{01} adalah jumlah hasil dari *class* 0 yang salah diprediksi sebagai *class* 1. Berdasarkan masukan dalam *confusion matrix*, jumlah prediksi yang benar adalah (f_{11}+ f_{00}) dan jumlah prediksi angka yang salah yaitu (f_{10}+ f_{01}).

Tabel 2.3 *Confusion matrix* untuk permasalahan 2 *class*

		Predicted Class	
		Class = 1	Class = 0
Actual Class	Class = 1	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

Sumber : [TAN-06]

Walaupun *confusion matrix* memberikan informasi yang dibutuhkan untuk menentukan seberapa baik akurasi model klasifikasi, meringkas informasi menjadi sebuah angka akan membuat perbandingan dengan model yang berbeda menjadi lebih baik. Hal ini dapat dilakukan dengan menggunakan sebuah *performance metric* seperti akurasi, yang didefinisikan sebagai berikut [TAN-06] :

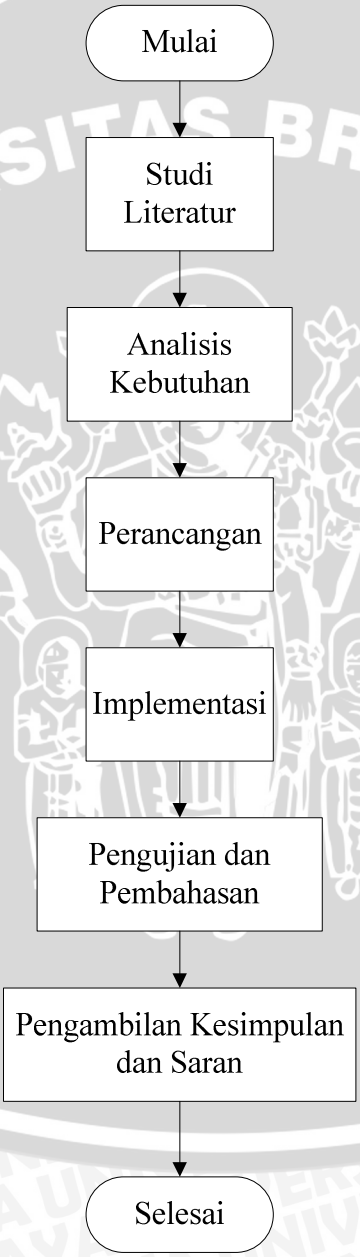
$$\boldsymbol{accuracy} = \frac{\textbf{number of correct predictions}}{\textbf{total number of predictions}} = \frac{f_{11}+f_{00}}{f_{11}+f_{10}+f_{01}+f_{00}} \qquad \textbf{(2-10)}$$

Sama halnya dengan akurasi, performa klasifikasi dapat dilihat dari *error rate*, yang ditunjukkan pada persamaan berikut [TAN-06:149] :

$$\boldsymbol{error\ rate} = \frac{\textbf{number of wrong predictions}}{\textbf{total number of predictions}} = \frac{f_{10}+f_{01}}{f_{11}+f_{10}+f_{01}+f_{00}} \qquad \textbf{(2-11)}$$

BAB III
METODOLOGI PENELITIAN

Pada bab ini dijelaskan langkah - langkah yang akan dilakukan dalam perancangan, implementasi dan pengujian dari *Aplikasi Pembaca Partitur Notasi Balok* yang akan dikembangkan. Kesimpulan dan saran disertakan sebagai catatan atas aplikasi dan kemungkinan arah pengembangan perangkat lunak selanjutnya.



Gambar 3.1 Diagram Alir Penelitian

Sumber : Adaptasi dari *Reuse-Oriented model*

3.1 Studi Literatur

Studi literatur mejelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori pendukung tersebut meliputi:

a. Citra Partitur
b. Pemrosesan Citra
c. *Reuse-Oriented Model*
d. Bahasa Pemrograman *Java*
e. JHLabs – *Java Image Processing*
f. JFugue – *Java API for Music Programming*
g. Pola-pola Perancangan
    - *Abstract Factory Pattern*
    - *Template Method Pattern*
    - *Strategy Pattern*
h. *Unified Modelling Language*
    - *Use Case Diagram*
    - *Class Diagram*
    - *Sequence Diagram*
    - *Activity Diagram*
i. Pengujian Perangkat Lunak
    - Pengujian Fungsional
    - Pengujian Performa
    - Pengujian Akurasi

3.2 Analisis Kebutuhan

Analisis kebutuhan dilakukan dengan menentukan kebutuhan apa saja yang dibutuhkan untuk membangun *Aplikasi Pembaca Partitur Notasi Balok*. Metode analisis yang digunakan adalah *Object Oriented Analysis* dengan menggunakan bahasa pemodelan UML (*Unified Modelling Language*). Fungsi-fungsi yang dapat dilakukan oleh aplikasi ini antara lain:

- Menerima masukan berupa citra partitur
- Memproses citra partitur notasi balok menajdi notasi angka
- Mengeluarkan bunyi dan informasi nada dari notasi yang terbaca

- Menyimpan hasil pemrosesan citra partitur, baik berupa citra maupun audio

Kebutuhan yang digunakan dalam pembuatan *Aplikasi Pembaca Partitur Notasi Balok* ini meliputi:

1. Kebutuhan Hardware, meliputi:
   - Komputer PC
2. Kebutuhan Software, meliputi:
   - Microsoft Windows 7 sebagai sistem operasi
   - Oracle NetBeans IDE 7.1 sebagai platform pengembangan
   - Java Development Kit 6 Update 26
   - Java Runtime Environment 6
   - StarUML sebagai *tool* pembuatan diagram pemodelan sistem
3. Data yang dibutuhkan meliputi:
   - Citra partitur berasal dari partitur cetak (*printed document)*, bukan partitur tulisan tangan (*handwritten document)*
   - Data *training* sebagai *template* dari simbol musik pada citra partitur

3.3 Perancangan

Setelah menentukan kebutuhan untuk membangun sistem, maka langkah selanjutnya yaitu desain atau perancangan sistem. Perancangan terdiri dari beberapa tahap yang sesuai dengan model perangkat lunak *Reuse-Oriented,* antara lain :

- Analisis Komponen

  Tahap ini bertujuan untuk menentukan komponen-komponen yang akan dipakai pada aplikasi. Diawali dengan mengumpulkan seluruh komponen yang berhubungan dengan kebutuhan yang ada. Dilanjutkan dengan proses pemilihan komponen-komponen yang dapat digunakan untuk memenuhi kebutuhan.
- Modifikasi Kebutuhan

  Tahap ini bertujuan untuk menyesuaikan atau memodifikasi setiap kebutuhan pada aplikasi agar sesuai dengn komponen-komponen yang telah tersedia.

- Mendesain sistem dengan konsep *reuse*

  Tahap ini direalisasikan dengan membuat sistem berdasarkan pola-pola perancangan (*design pattern),* dimana akan dihasilkan sebuah sistem yang dapat digunakan kembali (*reuse).*

Perancangan aplikasi berdasarkan *Object Oriented Analysis* dan *Object Oriented Design* yaitu menggunakan pemodelan UML. Pada tahap perancangan di skripsi ini, digunakan pemodelan dengan menggunakan dua macam diagram, yaitu *class diagram* dan *sequence diagram*. *Class diagram* digunakan untuk mengidentifikasi kelas-kelas dan *interface-interface* yang dibutuhkan pada sistem. Sedangkan *sequence diagram* digunakan untuk menggambarkan hubungan interaksi antar elemen (objek) yang telah diidentifikasi dan dimodelkan pada sistem yang disusun dalam urutan waktu. Perancangan aplikasi pembaca notasi balok ini secara garis besar meliputi:

- Pencarian komponen yang digunakan untuk memenuhi setiap kebutuhan
- Pemodelan *use case diagram*
- Pemodelan *class diagram*
- Pemodelan *sequence diagram*
- Pemodelan *activity diagram*
- Pemodelan *class* dan *interface* komponen
- Perancangan *user interface*

3.4 Implementasi

Implementasi perangkat lunak mengacu kepada perancangan perangkat lunak. Implementasi perangkat lunak diawali dengan penjabaran spesifikasi lingkungan pengembangan perangkat lunak. Selanjutnya dijabarkan melalui pemetaan antara *class* dengan *file* program saat implementasi perangkat lunak. Kemudian implementasi algoritma menjadi lanjutan dari tahap tersebut. Implementasi perangkat lunak dilakukan dengan menggunakan bahasa pemrograman Java. Tahap terakhir dari implementasi adalah implementasi antarmuka berdasarkan perancangan yang telah dilakukan.

3.5 Pengujian dan Pembahasan

Pengujian perangkat lunak dilakukan agar dapat menunjukkan bahwa perangkat lunak telah mampu bekerja sesuai dengan spesifikasi dari kebutuhan yang melandasinya. Proses pengujian dilakukan melalui tiga tahapan (strategi) yaitu pengujian fungsional, pengujian akurasi dan pengujian performa.

Pengujian dimulai dengan pengujian fungsional menggunakan metode *black box*. Setelah semua kebutuhan tervalidasi, maka dilanjutkan dengan pengujian akurasi untuk menghitung seberapa akurat aplikasi dalam menyelesaikan masalah. Pengujian yang terakhir adalah pengujian performa, pengujian ini ditujukan untuk menghitung waktu eksekusi aplikasi dalam menyelesaikan masalah.

3.6 Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan perangkat lunak, implementasi perangkat lunak, dan pengujian perangkat lunak telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan pembahasan terhadap sistem yang dibangun. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan – kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan perangkat lunak lebih lanjut.

BAB IV
PERANCANGAN

Bab ini membahas mengenai perancangan proses analisis kebutuhan dan proses perancangan *Aplikasi Pembaca Partitur Notasi Balok*. Analisis kebutuhan terdiri atas lima langkah yaitu melakukan penjabaran tentang gambaran umum perangkat lunak, melakukan proses identifikasi aktor yang terlibat dalam sistem perangkat lunak, melakukan proses analisis data yang diperlukan, membuat daftar kebutuhan pengguna berdasarkan penjabaran gambaran umum perangkat lunak dan menggunakan pemodelan diagram *use case* untuk menggambarkan kebutuhan tersebut. Proses perancangan perangkat lunak memiliki tiga tahap, yaitu analisis komponen, modifikasi kebutuhan dan pendesainan sistem dengan *reuse*.

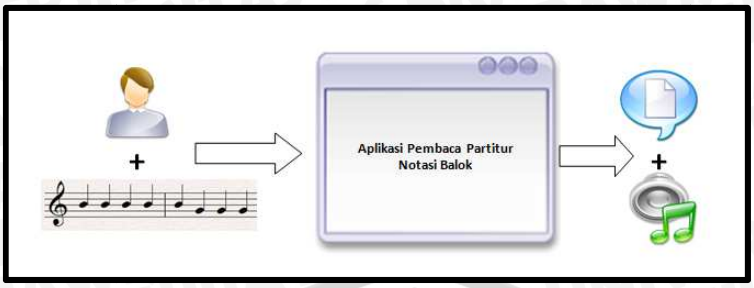
4.1 Analisis Kebutuhan

Proses analisis kebutuhan ini diawali dengan deskripsi umum perangkat lunak *Aplikasi Pembaca Partitur Notasi Balok*, identifikasi aktor yang terlibat, analisis data yang digunakan dalam aplikasi, penjabaran tentang daftar kebutuhan dan kemudian memodelkannya ke dalam diagram *use case*. Analisis kebutuhan ini bertujuan untuk menggambarkan kebutuhan – kebutuhan yang harus disediakan oleh sistem agar dapat memenuhi kebutuhan pengguna.

4.1.1 Deskripsi Umum Perangkat Lunak

Perangkat lunak yang akan dikembangkan pada proyek skripsi ini adalah *Aplikasi Pembaca Partitur Notasi Balok*. Aplikasi ini merupakan aplikasi pembantu bagi para pemula untuk membaca dan mendengarkan beberapa notasi yang ada pada partitur balok. Garis besar aplikasi ini dapat dilihat pada Gambar 4.1.

Pada Gambar 4.1 dapat dilihat bahwa pengguna memasukkan sebuah citra partitur sebagai data input aplikasi. Selanjutnya aplikasi akan memproses citra tersebut dan menghasilkan keluaran berupa citra partitur yang berisi notasi angka dan audio dari notasi yang terdeteksi.



Gambar 4.1 Cara Kerja Aplikasi

Sumber : [Perancangan]

4.1.2 Identifikasi Aktor

Tahap ini mempunyai tujuan untuk melakukan identifikasi terhadap aktor yang akan berinteraksi dengan sistem. Penjelasan dari masing-masing identifikasi aktor dapat dilihat pada Tabel 4.1.

Tabel 4.1 Identifikasi Aktor

Aktor	Deskripsi
Pengguna	Pengguna biasa yang ingin mengubah partitur notasi balok menjadi notasi angka dan mendengarkan nada dari partitur

Sumber : [Perancangan]

4.1.3 Analisis Data

Analisis data bertujuan untuk menyesuaikan data citra masukan dengan batasan masalah dari skripsi ini yang nantinya akan diimplementasikan ke dalam aplikasi. Adapun batasan dari citra masukan, yaitu :

1. Citra partitur berasal dari partitur cetak (*printed document)*, bukan partitur tulisan tangan (*handwritten document)*
2. Citra masukan berupa citra dari partitur balok
3. Citra partitur memakai tanda mula C
4. Citra partitur menggunakan *clef* G
5. Citra partitur berisikan simbol musik yang terdapat pada Tabel 4.2

Tabel 4.2 Daftar simbol musik

No	Nama	Gambar
1	*Clef*	
2	*Whole note*	
3	*Half note*	
4	*Quarter note*	
5	*Eight note*	
6	*Rest note*	

Sumber : [Perancangan]

Sedangkan batasan untuk data *training*, yaitu :

1. Citra yang digunakan sebagai data *training* berukuran 100 x 100 piksel
2. Data *training* disimpan dalam bentuk citra dengan ekstensi .png
3. Sumber data *training* berasal dari potongan acak simbol musik pada citra partitur

4.1.4 Daftar Kebutuhan

Daftar kebutuhan terdiri dari kebutuhan fungsional dan kebutuhan non-fungsional yang harus disediakan oleh sistem. Daftar kebutuhan ini disebut dengan *Software Requirement Specification* (SRS). Daftar kebutuhan fungsional dapat dilihat pada Tabel 4.3.

Tabel 4.3 Daftar kebutuhan fungsional

Nomor SRS	Kebutuhan	*Use Case*
SRS-001-01	Sistem mampu menerima masukan berupa citra partitur	Memasukkan citra partitur

SRS-001-02	Sistem mampu mengolah citra partitur sesuai aturan yang telah ditentukan	Memproses citra partitur
SRS-001-03	Sistem mampu menampilkan hasil pemrosesan citra partitur	Melihat hasil pemrosesan citra
SRS-001-04	Sistem mampu memutar audio yang berisikan nada sesuai hasil pemrosesan citra	Mendengarkan audio hasil pemrosesan citra
SRS-001-05	Sistem mampu menyimpan / menghasilkan citra hasil pemrosesan	Menyimpan citra
SRS-001-06	Sistem mampu menyimpan audio (MIDI) yang berisi nada sesuai hasil pemrosesan citra	Menyimpan MIDI

Sumber : [Perancangan]

Kebutuhan non-fungsional juga harus disediakan oleh sistem. Kebutuhan non-fungsional merupakan batasan layanan atau fungsi yang ditawarkan sistem seperti batasan waktu, batasan pengembangan proses, dan standarisasi. Daftar kebutuhan non-fungsional dapat dilihat pada Tabel 4.4.

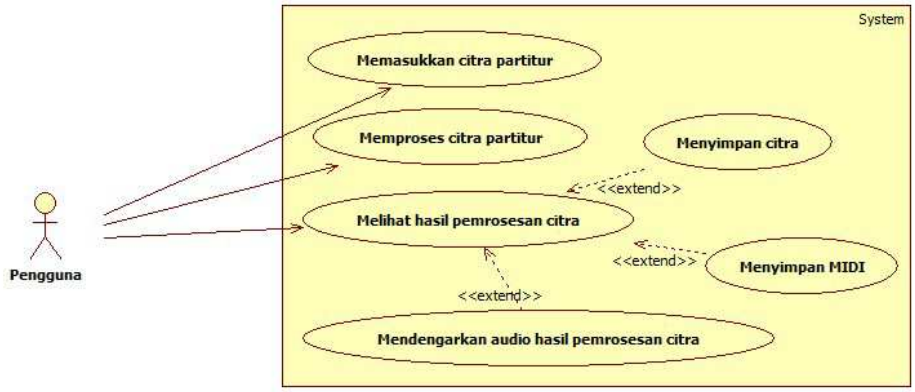
Tabel 4.4 Daftar kebutuhan non-fungsional

Parameter	Deskripsi Kebutuhan
Performance	Perangkat lunak harus memiliki waktu untuk melakukan pemrosesan citra tidak lebih atau sama dengan waktu yang diperlukan manusia untuk merubah notasi balok menjadi notasi angka

Sumber : [Perancangan]

4.1.5 *Use Case Diagram*

Diagram *use case* adalah salah satu diagram untuk memodelkan aspek perilaku sistem. Diagram *use case* menunjukkan sekumpulan *use case*, aktor, dan hubungannya. *Use case* merupakan fungsionalitas dari sistem yang diinisialisasi oleh aktor. Aplikasi pembaca notasi partitur balok hanya memiliki sebuah *use case*, yaitu *use case* pengguna yang dapat dilihat pada Gambar 4.2.



Gambar 4.2 Diagram *use case* pengguna

Sumber : [Perancangan]

4.1.6 Skenario *Use Case*

Masing-masing *use case* yang terdapat pada diagram *use case*, dijabarkan dalam skenario *use case* secara detail. Skenario *use case* akan memuat nama *use case*, aktor di dalam *use case* tersebut, tujuan dari *use case*, deskripsi global tentang *use case*, kondisi awal yang harus dipenuhi, dan kondisi akhir yang diharapkan setelah berjalannya fungsional *use case*. Selain itu juga akan diberikan ulasan yang berkaitan dengan tanggapan dari sistem atas suatu aksi yang diberikan oleh aktor (aliran utama atau *main flow)*, serta kejadian alternatif yang akan terjadi jika suatu kondisi tidak bisa terpenuhi (aliran alternatif atau *alternatif flow)*.

Tabel 4.5 Skenario *use case* memasukkan citra partitur

Nomor *Use Case*	SRS-001-01
Nama *Use Case*	Memasukkan citra partitur
Aktor	Pengguna
Tujuan	Untuk memasukkan citra partitur yang valid ke dalam aplikasi
Deskripsi	*Use case* ini digunakan untuk memasukkan citra partitur ke dalam aplikasi
Kondisi Awal	Aplikasi menampilkan antarmuka utama aplikasi
Kondisi Akhir	Citra partitur yang dimasukkan oleh aktor tampil pada tempat yang disediakan pada aplikasi

Main Flow	1. Aplikasi menampilkan antarmuka utama. 2. Aktor memilih menu buka citra partitur 3. Aplikasi menampilkan dialog *browse files* 4. Aktor memilih citra partitur yang dikehendaki 5. Aplikasi menampilkan citra partitur yang telah dipilih aktor
Alternatif Flow	a. Eksepsi file yang dipilih aktor tidak bertipe *jpg, jpeg, png* Aplikasi akan menampilkan *alert dialog* b. Eksepsi ukuran file yang terlalu besar Aplikasi akan menampilkan *alert dialog*

Sumber : [Perancangan]

Tabel 4.6 Skenario *use case* memproses citra partitur

Nomor *Use Case*	SRS-001-02
Nama *Use Case*	Memproses citra partitur
Aktor	Pengguna
Tujuan	Untuk mendapatkan informasi dari citra partitur
Deskripsi	*Use case* ini digunakan untuk memproses citra partitur
Kondisi Awal	Aplikasi menampilkan citra partitur yang telah dipilih pengguna
Kondisi Akhir	Menuliskan notasi pada citra partitur
Main Flow	1. Aplikasi menampilkan citra partitur yang dipilih oleh pengguna 2. Pengguna menekan tombol untuk memproses partitur 3. Aplikasi melakukan proses penggalian data dari citra partitur 4. Aplikasi menuliskan notasi pada citra partitur
Alternatif Flow	a. Eksepsi notasi dari citra melebihi batas Aplikasi akan menampilkan *alert dialog* b. Eksepsi citra partitur belum ada Aplikasi akan menampilkan *alert dialog*

Sumber : [Perancangan]

Tabel 4.7 Skenario *use case* melihat hasil pemrosesan citra

Nomor *Use Case*	SRS-001-03
Nama *Use Case*	Melihat hasil pemrosesan citra
Aktor	Pengguna
Tujuan	Untuk melihat citra hasil pemrosesan citra
Deskripsi	*Use case* ini digunakan melihat citra hasil pemrosesan citra
Kondisi Awal	Aplikasi belum melihat citra hasil pemrosesan citra Aplikasi telah melakukan pemrosesan citra
Kondisi Akhir	Aplikasi menampilkan citra hasil pemrosesan citra
Main Flow	1. Aplikasi belum menampilkan citra hasil pemrosesan citra 2. Aplikasi menampilkan citra hasil pemrosesan citra
Alternatif Flow	a. Eksepsi citra belum diproses Aplikasi akan menampilkan *alert dialog*

Sumber : [Perancangan]

Tabel 4.8 Skenario *use case* mendengarkan audio hasil pemrosesan citra

Nomor *Use Case*	SRS-001-04
Nama *Use Case*	Mendengarkan audio hasil pemrosesan citra
Aktor	Pengguna
Tujuan	Untuk mendengarkan audio hasil pemrosesan citra
Deskripsi	*Use case* ini digunakan mendengarkan audio notasi hasil pemrosesan citra
Kondisi Awal	Aplikasi belum memutar audio notasi hasil pemrosesan citra Aplikasi telah melakukan pemrosesan citra
Kondisi Akhir	Aplikasi memutar audio notasi hasil pemrosesan citra
Main Flow	1. Aplikasi belum memutar audio notasi hasil pemrosesan citra 2. Pengguna menekan tombol untuk memutar audio MIDI 3. Aplikasi memutar audio notasi hasil pemrosesan citra
Alternatif Flow	a. Eksepsi citra belum diproses Aplikasi akan menampilkan *alert dialog*

	b. Eksepsi audio tidak ditemukan Aplikasi akan menampilkan *alert dialog*

Sumber : [Perancangan]

Tabel 4.9 Skenario *use case* menyimpan citra

Nomor *Use Case*	SRS-001-05
Nama *Use Case*	Menyimpan citra
Aktor	Pengguna
Tujuan	Untuk menyimpan citra
Deskripsi	*Use case* ini digunakan untuk menyimpan citra
Kondisi Awal	Aplikasi belum menyimpan citra Aplikasi telah melakukan pemrosesan citra
Kondisi Akhir	Aplikasi berhasil menyimpan citra
Main Flow	1. Aplikasi belum menyimpan citra 2. Pengguna menekan tombol untuk menyimpan citra 3. Pengguna memilih lokasi penyimpanan 4. Aplikasi mengambil citra dari *image container* 5. Aplikasi mebuat file dari citra 6. Aplikasi memindahkan file citra menuju lokasi pilihan aktor 7. Aplikasi berhasil menyimpan citra
Alternatif Flow	a. Eksepsi tidak ada citra dalam *image container* Aplikasi akan menampilkan *alert dialog*

Sumber : [Perancangan]

Tabel 4.10 Skenario *use case* menyimpan MIDI

Nomor *Use Case*	SRS-001-06
Nama *Use Case*	Menyimpan MIDI
Aktor	Pengguna
Tujuan	Untuk menyimpan file MIDI
Deskripsi	*Use case* ini digunakan untuk menyimpan MIDI
Kondisi Awal	Aplikasi belum menyimpan MIDI Aplikasi telah melakukan pemrosesan citra
Kondisi Akhir	Aplikasi berhasil menyimpan MIDI
Main Flow	1. Aplikasi belum menyimpan MIDI

	2. Pengguna menekan tombol untuk menyimpan MIDI 3. Pengguna memilih lokasi penyimpanan 4. Aplikasi mengambil *pattern* MIDI 5. Aplikasi mebuat file MIDI dari *pattern* yang sudah ada dengan MIDICreator 6. Aplikasi memindah dile MIDI menuju lokasi pilihan aktor 7. Aplikasi berhasil menyimpan citra
Alternatif Flow	a. Eksepsi tidak ada citra dalam *image container* Aplikasi akan menampilkan *alert dialog*

Sumber : [Perancangan]

4.2 Perancangan Perangkat Lunak

Perancangan perangkat lunak pada skripsi ini menggunakan pendekatan desain berorientasi objek yang direpresentasikan dengan menggunakan UML (*Unified Modeling Language*). Perancangan terdiri dari beberapa tahap yang sesuai dengan model perangkat lunak *Reuse-Oriented*, antara lain analisi komponen, modifikasi kebutuhan dan desain sistem dengan *reuse*.

4.2.1 Analisis Komponen

Proses analisis komponen diawali dengan mencari semua komponen yang berhubungan dengan pokok bahasan skripsi. Langkah selanjutnya adalah memilih beberapa komponen yang sesuai dengan kebutuhan yang diperlukan dalam aplikasi. Komponen yang dapat memenuhi kebutuhan akan langsung dipakai dalam aplikasi, sedangkan komponen yang tidak sesuai akan mengalami tahap modifikasi kebutuhan. Daftar komponen yang digunakan pada aplikasi disajikan dalam bentuk Tabel 4.11.

Tabel 4.11 Daftar komponen yang digunakan dalam aplikasi

Nama *Library*	Nama Komponen	Fungsi Komponen
JHLabs	Grayscale Filter	Merubah citra berwarna manjadi citra abu-abu
	Threshold Filter	Melakukan proses *thresholding* pada citra

	Erode Filter	Melakukan proses erosi pada citra
	Dilation Filter	Melakukan proses dilasi pada citra
JFugue	MIDI Player	Memainkan file midi
	MIDI Creator	Membuat file midi

Sumber : [Perancangan]

4.2.2 Modifikasi Kebutuhan

Proses modifikasi kebutuhan tidak dilakukan pada perancangan aplikasi. Hal ini dikarenakan seluruh kebutuhan sudah terpenuhi dengan komponen-komponen yang ada.

4.2.3 Desain Sistem dengan *Reuse*

Pembuatan desain aplikasi dimodelkan dengan beberapa diagram. Pada proses desain dilakukan identifikasi terhadap *class-class* yang dibutuhkan yang dimodelkan dalam *class diagram*. Pada proses rinci yang terjadi di dalam sistem secara rinci dimodelkan dalam *activity diagram.* Hubungan interaksi antar elemen (objek) yang telah diidentifikasi, dimodelkan dalam *sequence diagram*.

4.2.3.1 Perancangan *Class* Aplikasi

Perancangan class aplikasi dipresentasikan dengan menggunakan *class diagram*. *Class diagram* dirancang menggunakan permodelan pada pola-pola perancangan. Pola-pola perancangan yang dipakai dalam memodelkan class diagram pada aplikasi, yaitu *abstract factory pattern*, *template method pattern,* dan *strategy pattern*. *Class diagram* sistem dapat dilihat pada Gambar 4.3. *Class diagram* sistem dibagi menjadi beberapa *package*, yaitu :

1. *Package Core*

   Pada *package* ini terdapat semua *class* yang bertanggung jawab pada fungsi utama dari aplikasi
2. *Package GUI*

   Pada *package* ini terdapat semua *class* yang bertanggung jawab untuk mengatur tampilan aplikasi

3. *Package Action*

   Pada *package* ini terdapat semua *class* yang bertanggung jawab untuk mengatur *action listener* pada komponen tampilan. Pada perancangan *class action listener* digunakan pola-pola perancangan *template method*

4. *Package Util*

   Pada *package* ini terdapat semua *class* yang mendukung fungsi utama pada aplikasi

5. *Package Factory*

   Pada *package* ini terdapat semua *class* yang menerapkan pola-pola perancangan *abstract factory pattern.* Penggunaan pola-pola perancangan ini bertujuan untuk mengatur penginstansian objek *filter* pada *library JHLabs*

6. *Package Strategy* dan *Interfaces*

   Pada *package* ini terdapat semua *class* yang menerapkan pola-pola perancangan *strategy pattern.* Penggunaan pola-pola perancangan ini bertujuan untuk mengatur proses segmentasi dan pengenalan objek dalam citra

7. *Package Objects*

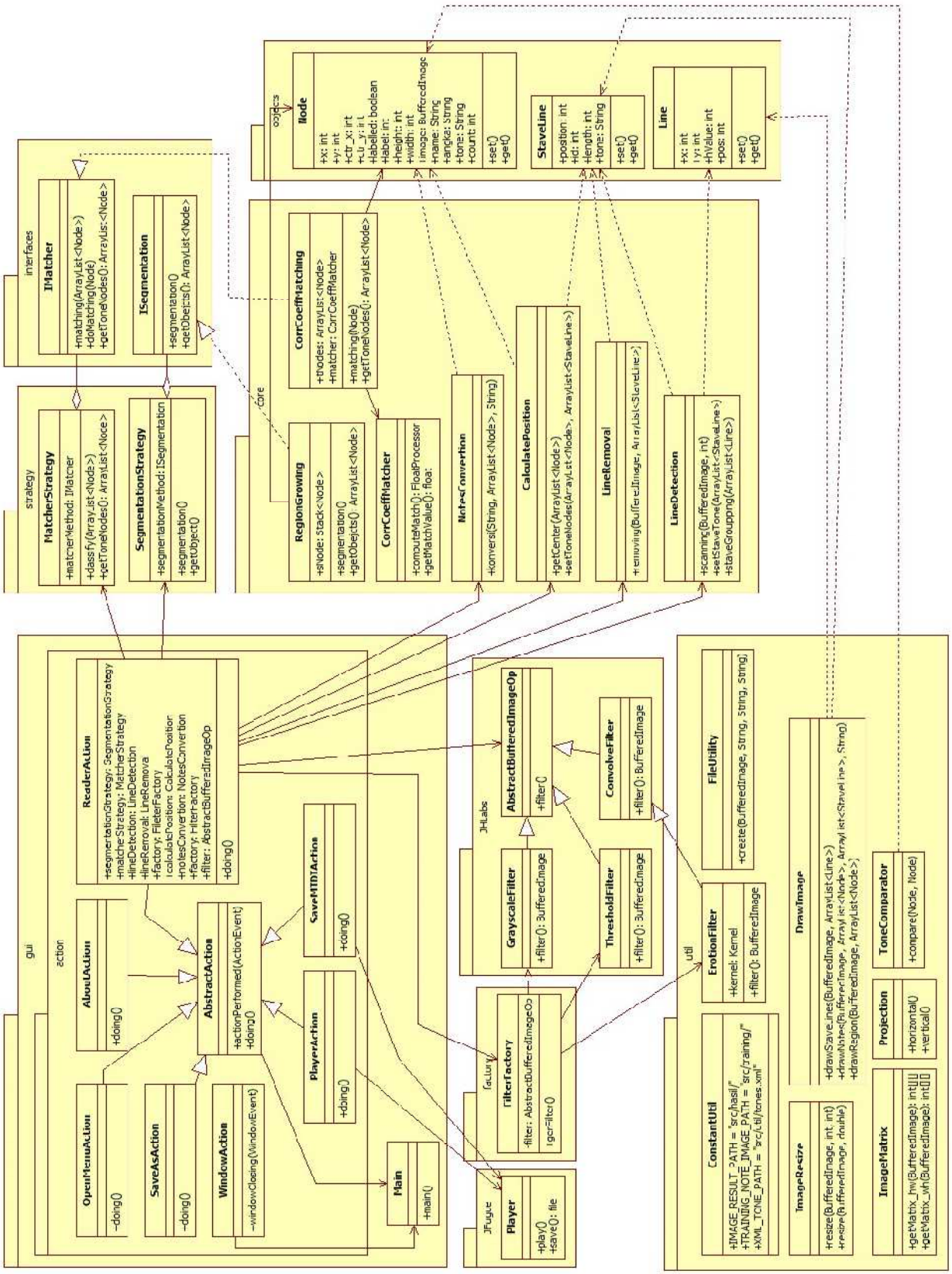
   Pada *package* ini terdapat semua *class* yang digunakan untuk mendeskripsikan objek yang ada dalam citra. Dalam skripsi ini, objek yang dimaksud adalah garis paranada dan simbol musik

8. *Package JHLabs*

   Package ini menggambarkan tentang *class* dalam *library JHLabs* yang digunakan dalam pembuatan aplikasi

9. *Package JFugue*

   Package ini menggambarkan tentang *class* dalam *library JFugue* yang digunakan dalam pembuatan aplikasi

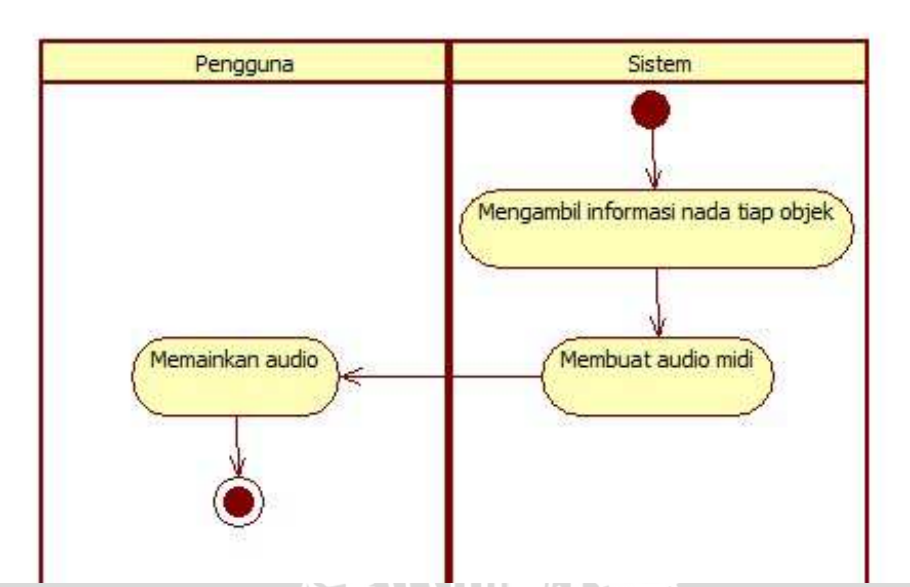


Gambar 4.3 *Class Diagram* Aplikasi

Sumber : [Perancangan]

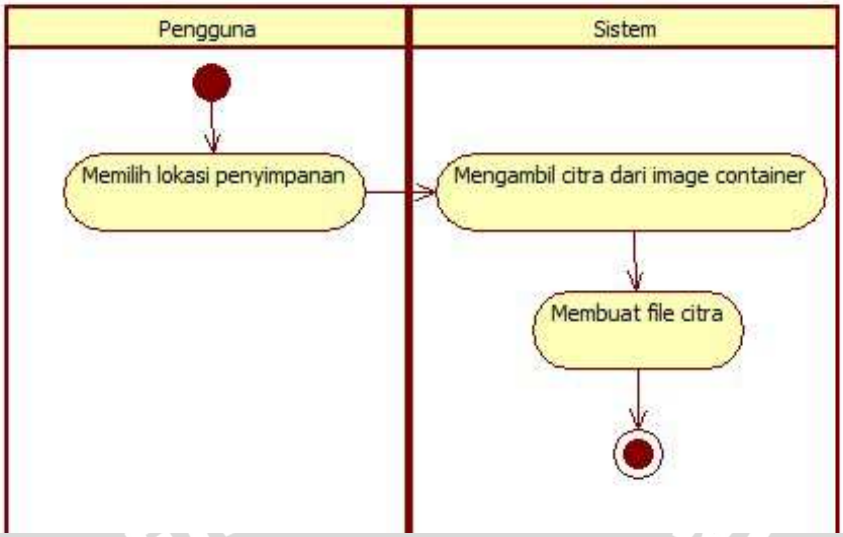
4.2.3.2 Perancangan Aktivitas Aplikasi

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, kemungkinan yang mungkin terjadi, dan bagaimana proses berakhir. Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* pada aplikasi disesuaikan dengan banyaknya kebutuhan fungsional dari aplikasi. Adapun *activity diagram* yang dibuat adalah *activity diagram* pemrosesan citra (Gambar 4.7), *activity diagram* memutar audio notasi (Gambar 4.4), *activity diagram* menyimpan citra (Gambar 4.5) dan *activity diagram* menyimpan MIDI (Gambar 4.6).



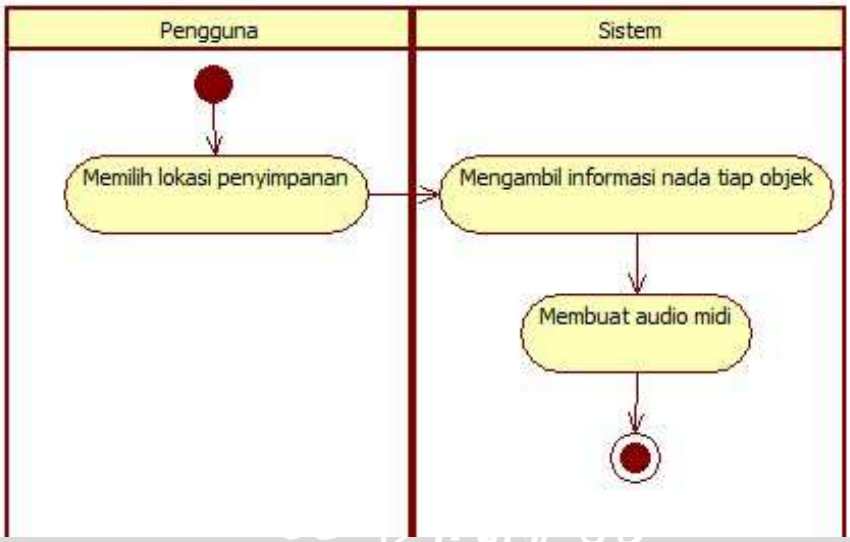
Gambar 4.4 *Activity diagram* memutar audio notasi

Sumber : [Perancangan]



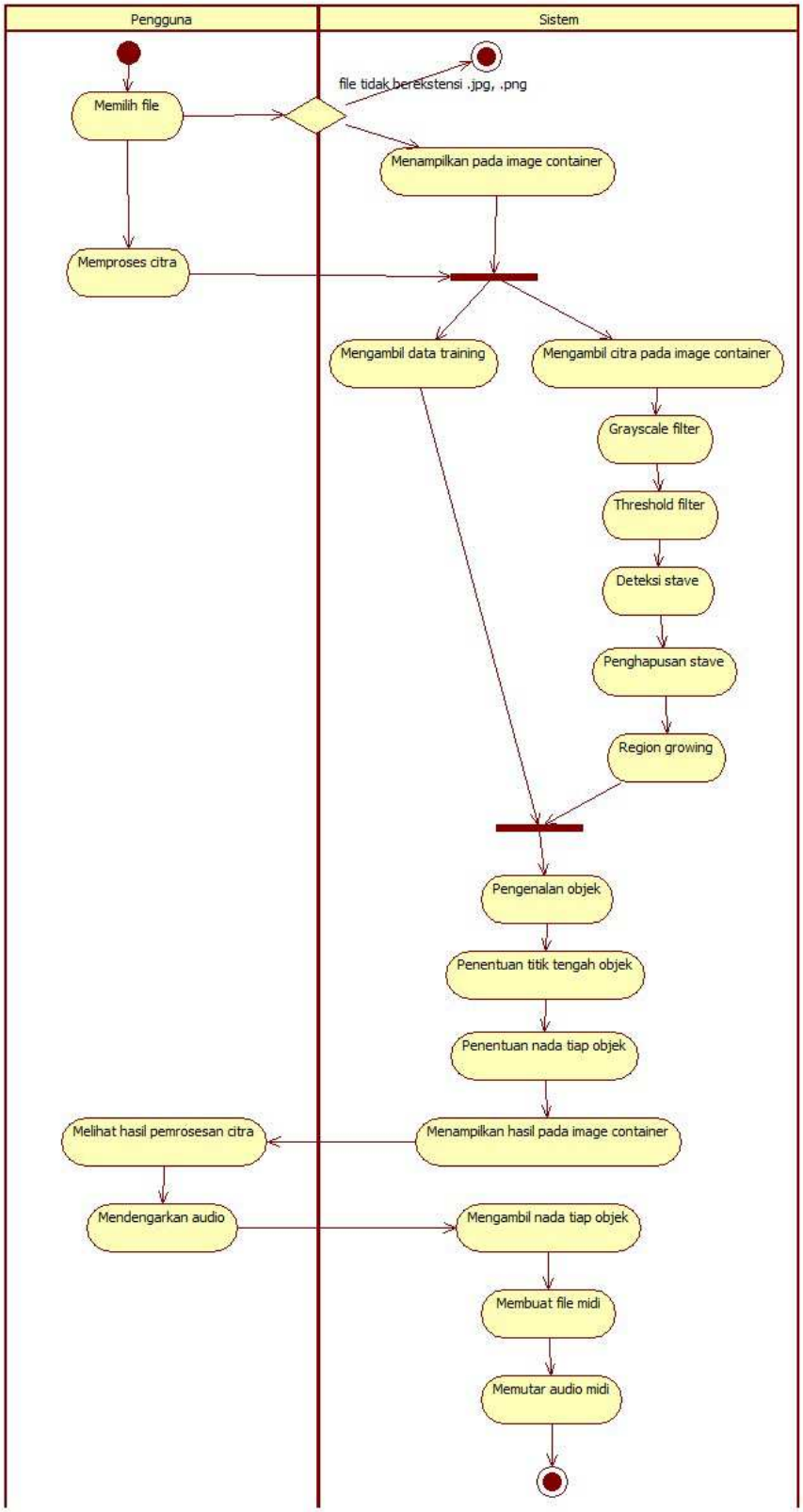
Gambar 4.5 *Activity diagram* menyimpan citra

Sumber : [Perancangan]



Gambar 4.6 *Activity diagram* menyimpan audio midi

Sumber : [Perancangan]

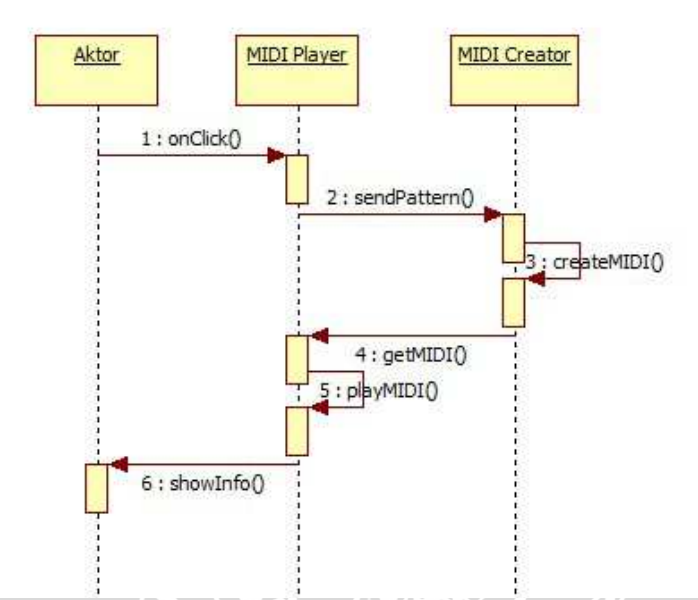


Gambar 4.7 *Activity diagram* pemrosesan citra

Sumber : [Perancangan]

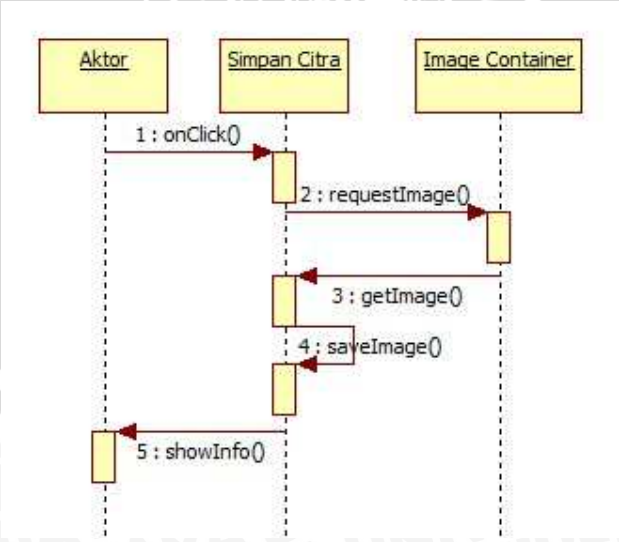
4.2.3.3 Perancangan Interaksi Antar Objek Aplikasi

Sequence diagram berisikan menggambarkan kelakuan objek yang digunakan dalam aplikasi. Adapun *sequence diagram* yang dibuat adalah *sequence diagram* pemrosesan citra (Gambar 4.10), *sequence diagram* memutar audio notasi (Gambar 4.8), *sequence diagram* menyimpan citra (Gambar 4.9) dan *sequence diagram* menyimpan MIDI (Gambar 4.11).



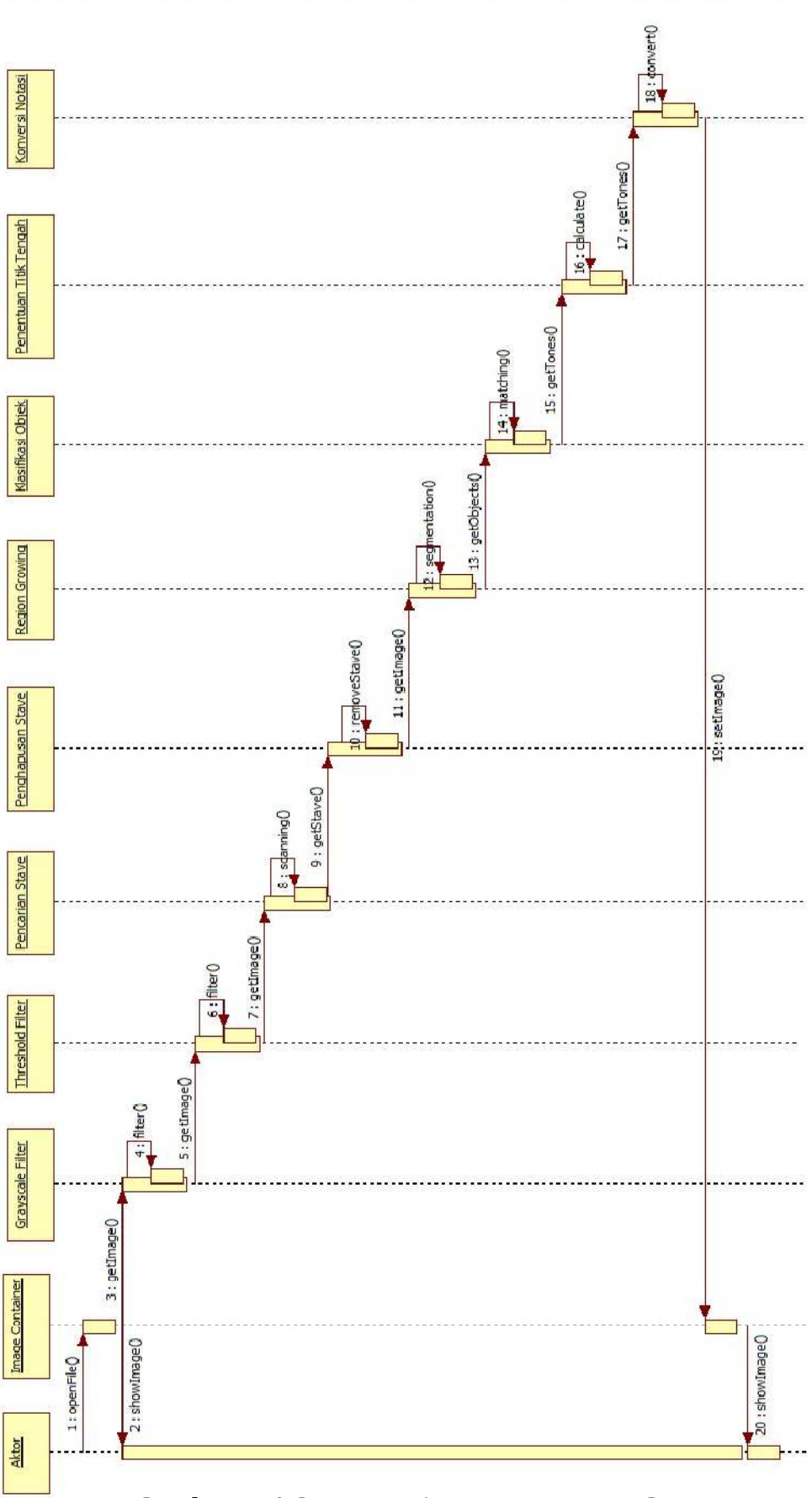
Gambar 4.8 *Sequence Diagram* Memutar Audio Notasi

Sumber : [Perancangan]



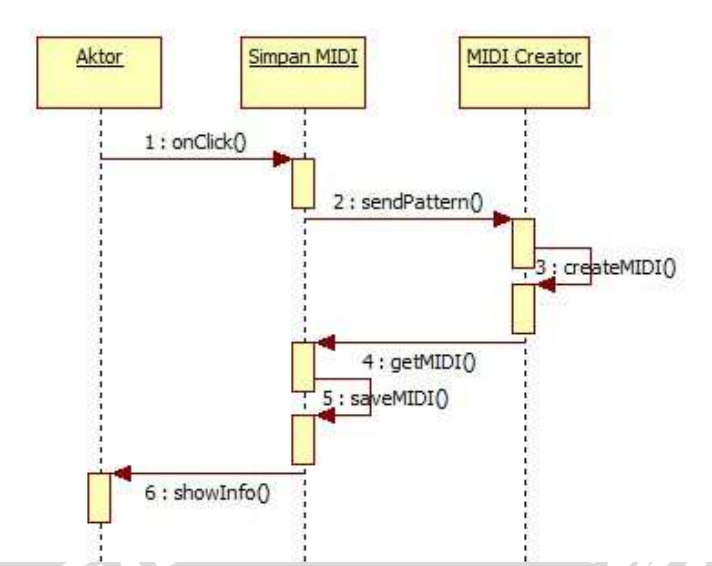
Gambar 4.9 *Sequence Diagram* Menyimpan Citra

Sumber : [Perancangan]



Gambar 4.10 *Sequence Diagram* Pemrosesan Citra

Sumber : [Perancangan]



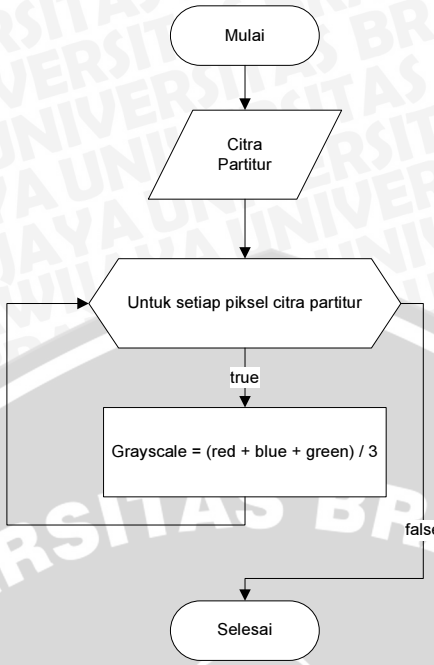
Gambar 4.11 *Sequence Diagram* Menyimpan MIDI

Sumber : [Perancangan]

4.2.3.4 Perancangan Algoritma Aplikasi

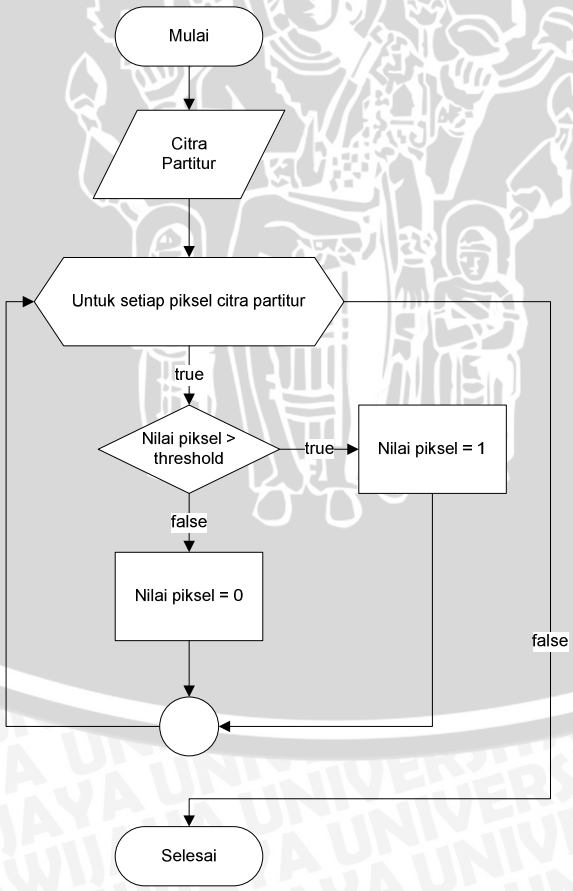
Perancangan algortima pada aplikasi dimodelkan dalam bentuk *flowchart.* Adapun algoritma yang dimodelkan adalah beberapa algoritma yang merupakan algoritma utama pada aplikasi, yaitu algoritma *preprocessing*, algoritma pencarian *staves*, algoritma penghapusan *staves*, algoritma *region growing*, algoritma pengenalan simbol musik, algoritma perhitungan titik tengah simbol musik, algoritma notasi.

Algoritma *preprocessing* terdiri dari proses *grayscale* dan *thresholding /* binerisasi. Algoritma *preprocessing* ini diambil dari *library* JHLabs yang mengimplementasikan Persamaan 2-1 dan 2-2. *Flowchart* proses *preprocessing* dapat dilihat pada Gambar 4.11 dan Gambar 4.12.



Gambar 4.12 *Flowchart* algoritma *grayscale*

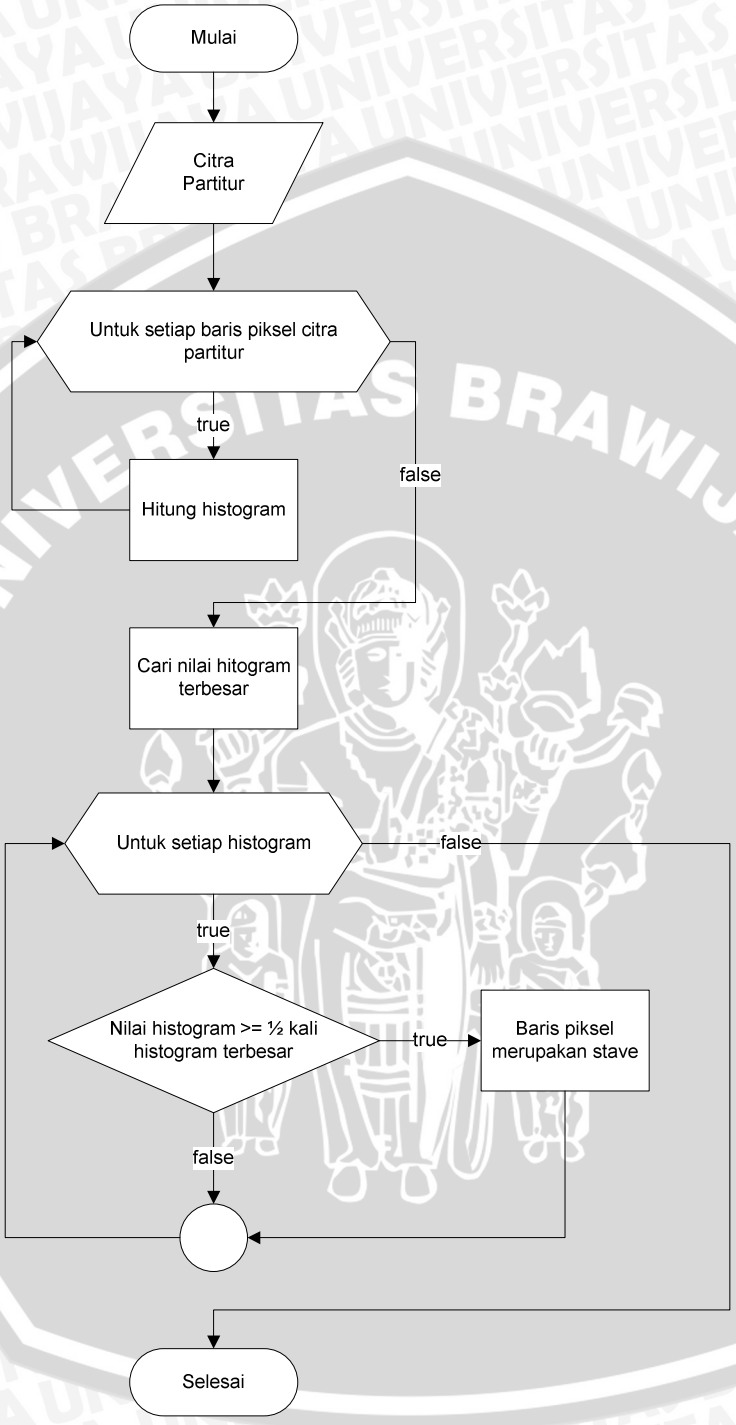
Sumber : [Perancangan]



Gambar 4.13 *Flowchart* algoritma *thresholding /* binerisasi

Sumber : [Perancangan]

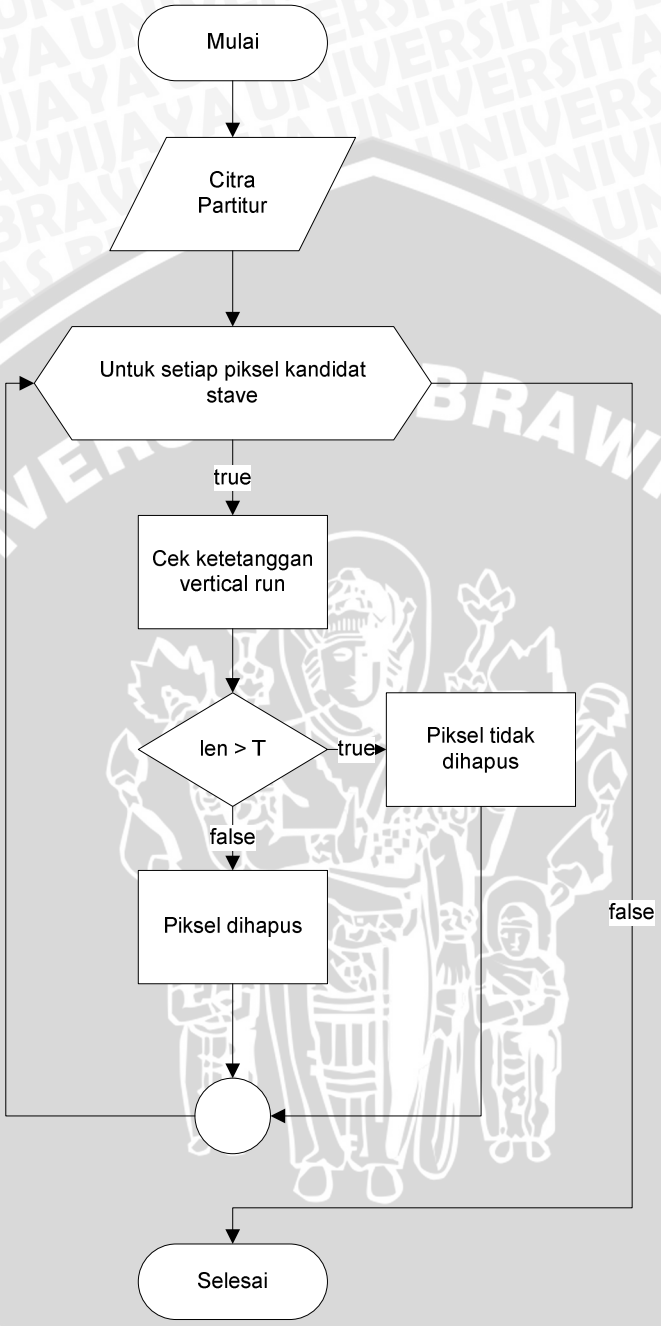
Proses pencarian garis paranada mengimplementasikan Persamaan 2-3. *Flowchart* algoritma dapat dilihat pada Gambar 4.13.



Gambar 4.14 *Flowchart* algoritma pencarian garis paranada

Sumber : [Perancangan]

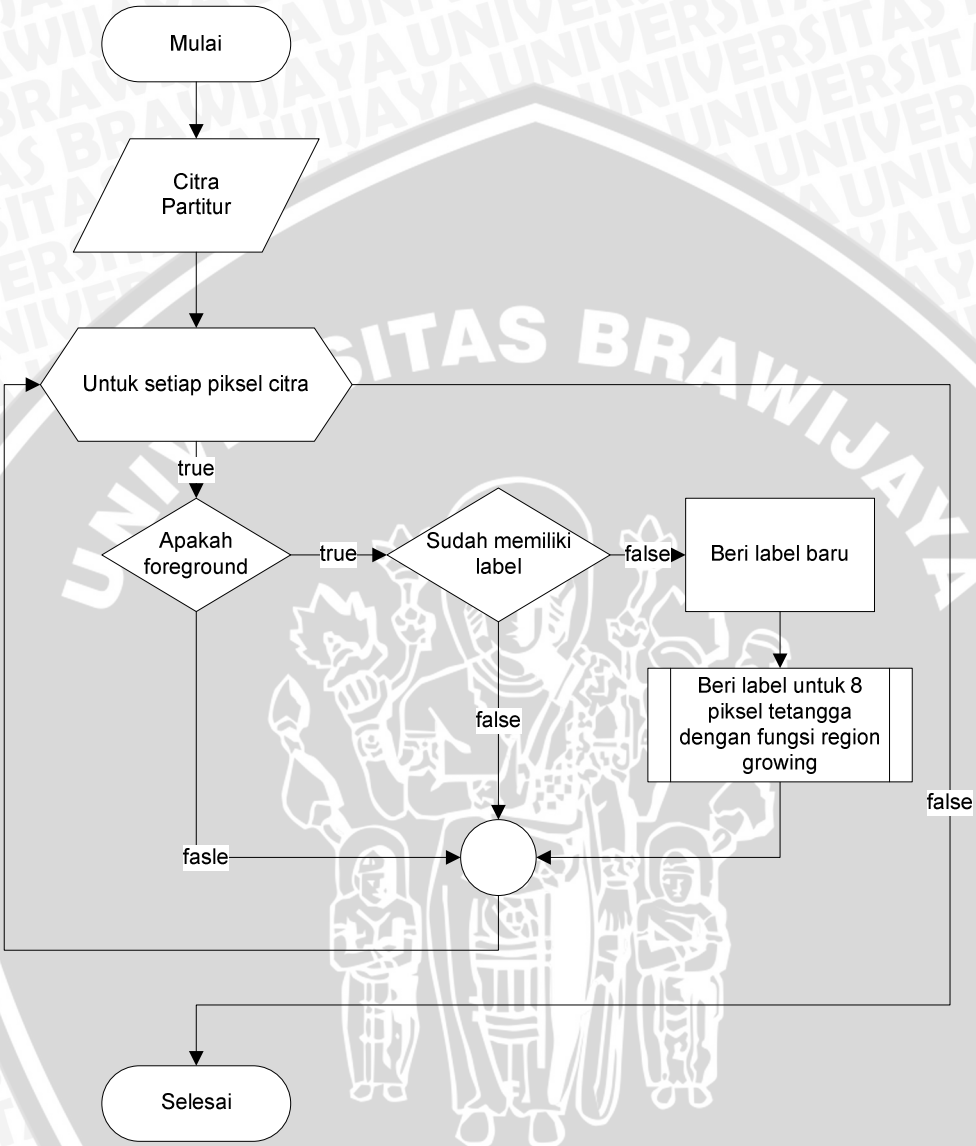
Proses penghapusan garis paranada mengimplementasikan Persamaan 2-4. *Flowchart* algoritma dapat dilihat pada Gambar 4.14.



Gambar 4.15 *Flowchart* algoritma penghapusan garis paranada

Sumber : [Perancangan]

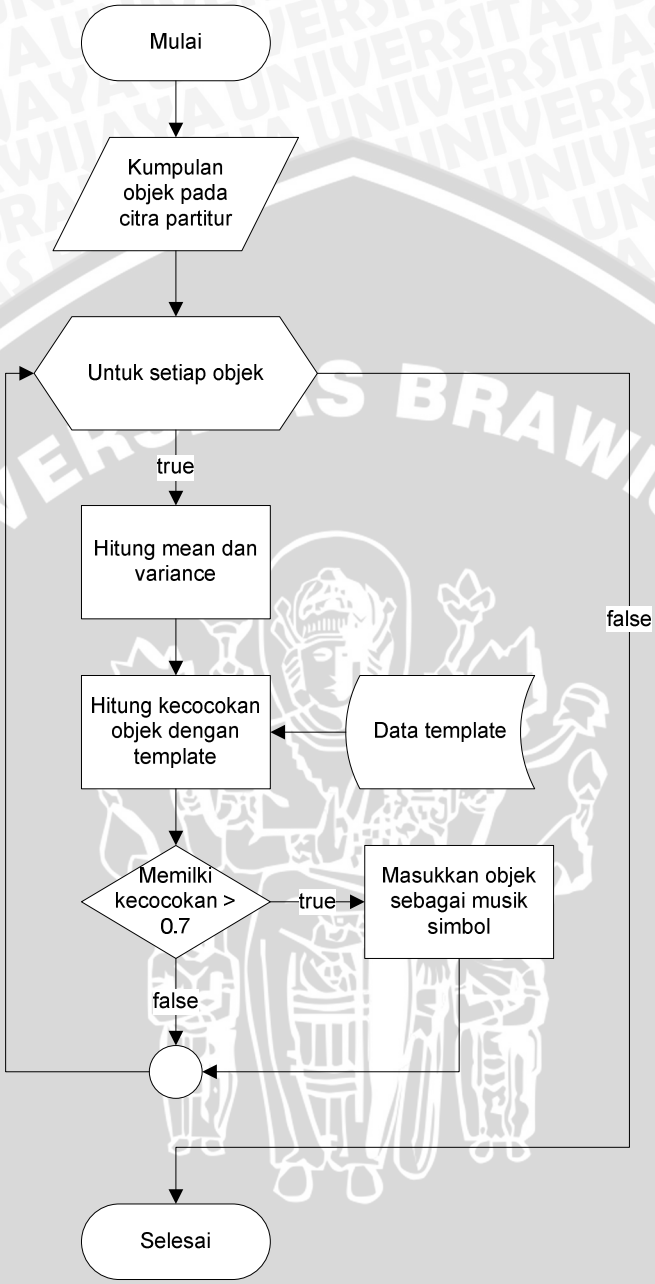
Proses *region growing* mengimplementasikan Persamaan 2-5. *Flowchart* algoritma dapat dilihat pada Gambar 4.15.



Gambar 4.16 *Flowchart* algoritma *region growing*

Sumber : [Perancangan]

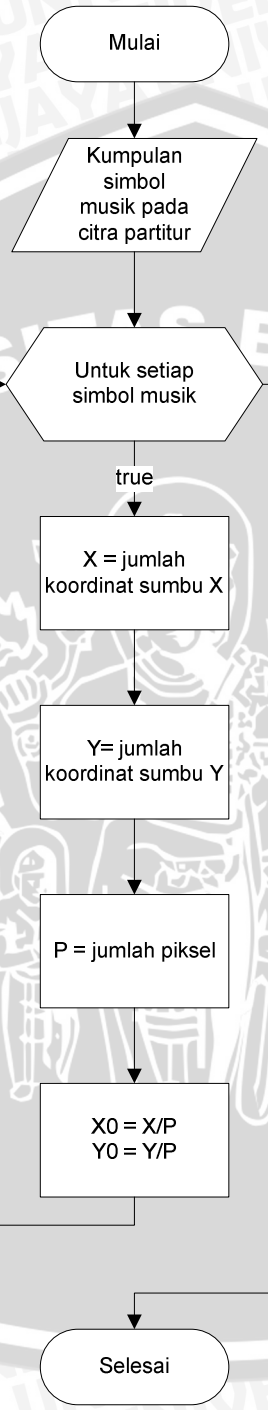
Proses pengenalan simbol musik mengimplementasikan Persamaan 2-7. *Flowchart* algoritma dapat dilihat pada Gambar 4.16.



Gambar 4.17 *Flowchart* algoritma pengenalan simbol musik

Sumber : [Perancangan]

Proses perhitungan titik tengah simbol musik mengimplementasikan Persamaan 2-8 dan Persamaan 2-9. *Flowchart* algoritma dapat dilihat pada Gambar 4.17.

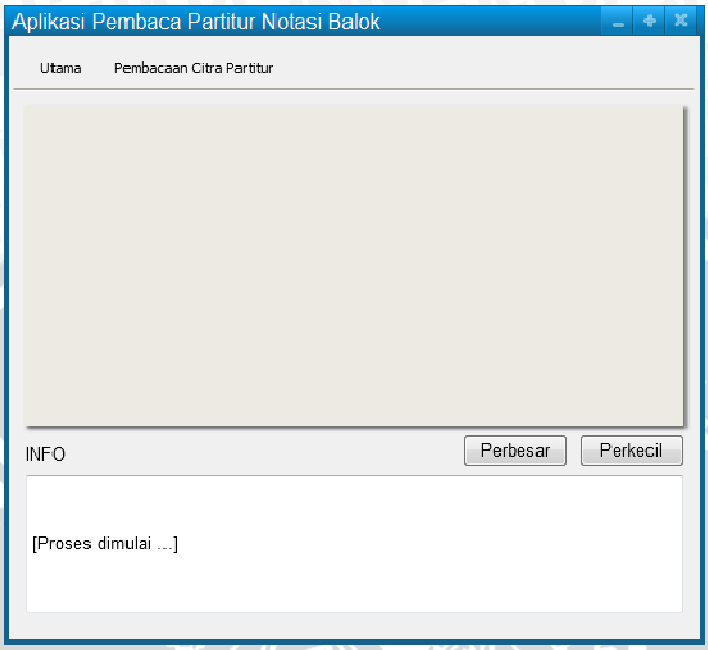


Gambar 4.18 *Flowchart* algoritma perhitungan titik tengah simbol musik

Sumber : [Perancangan]

4.2.3.5 Perancangan Antarmuka

Pada bagian ini akan dijelaskan tentang antarmuka aplikasi yang akan digunakan oleh aktor untuk berkomunikasi dengan aplikasi.



Gambar 4.19 Antarmuka Aplikasi

Sumber : [Perancangan]

Antarmuka aplikasi memilki 3 bagian penting yang digunakan aktor untuk berkomunikasi dengan aplikasi, antar lain :

1. *Menu Bar*

   Pada bagian ini aktor diberi 2 buah pilihan menu, yaitu utama dan pemrosesan citra. Pada pilihan utama, aktor dapat memberikan aksi untuk membuka file citra, menyimpan file citra dan menyimpan file audio MIDI. Sedangkan pada pilihan pemrosesan citra, aktor dapat memberikan aksi memroses citra dan mendengarkan audio notasi

2. *Image Container*

   Bagian ini berfungsi untuk menampilkan citra yang dipilih oleh aktor. Pada bagian ini pula, aktor dapat melihat hasil pemrosesan citra

3. Info

   Bagian ini berfungsi untuk menyimpan segala bentuk aktivitas, maupun informasi yang diperlukan oleh aktor.

BAB V
IMPLEMENTASI

Bab ini membahas mengenai tahapan implementasi perangkat lunak *Aplikasi Pembaca Partitur Notasi Balok* berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan perangkat lunak. Pembahasan terdiri atas penjelasan tentang spesifikasi sistem, batasan – batasan dalam implementasi, implementasi tiap *class* dan *interface* pada *file* program, implementasi algoritma, dan implementasi antarmuka.

5.1 Spesifikasi Sistem

Aplikasi Pembaca Partitur Notasi Balok dikembangkan dalam lingkungan implementasi yang terdiri dari perangkat keras dan perangkat lunak.

5.1.1 Spesifikasi Lingkungan Perangkat Keras

Spesifikasi perangkat keras yang dipakai dalam proses pengembangan dijelaskan pada Tabel 5.1.

Tabel 5.1 Spesifikasi lingkungan perangkat keras komputer

Desktop PC	
Processor	AMD Phenom II X4 955 3,20 GHz
Memory (RAM)	8.00 GB
Harddisk	Seagate Barracuda 1 TB
Motherboard	ASUS M5A99X EVO
Graphic Card	ATI RADEON HD 5750
Monitor	ASUS LED Monitor VS197

Sumber : [Implementasi]

5.1.2 Spesifikasi Lingkungan Perangkat Lunak

Spesifikasi perangkat lunak yang dipakai dalam proses pengembangan *Aplikasi Pembaca Partitur Notasi Balok* dijelaskan pada Tabel 5.2.

Tabel 5.2 Spesifikasi lingkungan perangkat lunak komputer

Desktop PC	
Operating System	Microsoft Windows 7 Ultimate 64-bit
Programming Language	Java

Software Development Kit	JDK 1.6
Integrated Development Environment	Oracle NetBeans IDE 7.1

Sumber : [Implementasi]

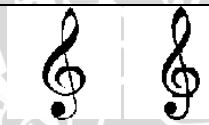
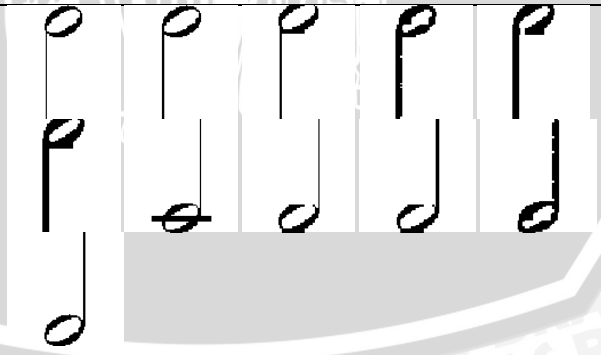
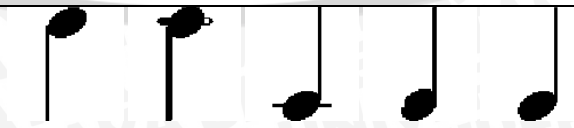
5.2 Batasan Implementasi

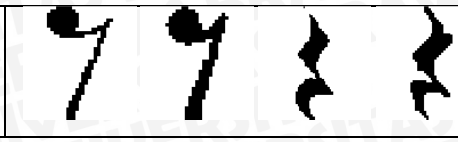
Batasan dalam mengimplementasikan *Aplikasi Pembaca Partitur Notasi Balok* adalah aplikasi dirancang dan dijalankan dengan menggunakan *Java Desktop Application.*

5.3 Implementasi Data *Training*

Implementasi data *training* ditujukan untuk menjelaskan data *training* yang digunakan. Seluruh data *taining* disesuaikan dengan perancangan data *training* pada bab sebelumnya. Adapun data *training* yang digunakan dalam skripsi ini dapat dilihat pada tabel 5.3

Tabel 5.3 Daftar data *training*

No	Nama	Gambar
1	*Clef*	
2	*Whole note*	
3	*Half note*	
4	*Quarter note*	
5	*Eight note*	

6	*Rest note*	

Sumber : [Implementasi]

5.4 Implementasi *Class* dan *Interface*

Setiap *class* yang telah dirancang pada proses perancangan direalisasikan pada sebuah *file* program dengan ekstensi *.java. Tabel 5.3 menjelaskan mengenai pasangan antara *class* dengan *file* program yang digunakan untuk mengimplementasikannya

Tabel 5.4 Implementasi *Class* dan *Interface*

No.	*Package*	Nama *Class* atau *Interface*	Nama *File* Program
1	core	CalculatePosition	CalculatePosition.java
2	core	CorrCoeffMatcher	CorrCoeffMatcher.java
3	core	CorrCoeffMatching	CorrCoeffMatching.java
4	core	LineDetection	LineDetection.java
5	core	LineRemoval	LineRemoval.java
6	core	NotesConversion	NotesConversion.java
7	core	RegionGrowing	RegionGrowing.java
8	factory	FilterFactory	FilterFactory.java
9	gui	Main	Main.java
10	gui	FileExtFilter	FileExtFilter.java
11	gui.action	AboutAction	AboutAction.java
12	gui.action	AbstractAction	AbstractAction.java
13	gui.action	OpenMenuAction	OpenMenuAction.java
14	gui.action	PlayerAction	PlayerAction.java
15	gui.action	ReaderAction	ReaderAction.java
16	gui.action	SaveAsAction	SaveAsAction.java
17	gui.action	SaveMIDIAction	SaveMIDIAction.java
18	gui.action	WindowAction	WindowAction.java
19	interfaces	IMatcher	IMatcher.java
20	interfaces	ISegmentation	ISegmentation.java
21	objects	Line	Line.java
22	objects	Node	Node.java
23	objects	StaveLine	StaveLine.java
24	strategy	MatcherStrategy	MatcherStrategy.java
25	strategy	SegmentationStrategy	SegmentationStrategy.java
26	util	ConstantUtil	ConstantUtil.java
27	util	DrawImage	DrawImage.java
28	util	ErotionFilter	ErotionFilter.java
29	util	FileUtility	FileUtility.java
30	util	ImageMatrix	ImageMatrix.java

31	util	ImageResize	ImageResize.java
32	util	RenderingImage	RenderingImage.java
33	util	ToneComparator	ToneComparator.java
34	util	Projection	Projection.java

Sumber : [Implementasi]

5.5 Implementasi Algoritma

Aplikasi Pembaca Partitur Notasi Balok mempunyai beberapa proses (*method*) utama yang terbagi dalam beberapa *class*. Pada penulisan skripsi ini hanya dicantumkan algoritma dari beberapa proses saja sehingga tidak semua implementasi algoritma *method* akan dicantumkan. Implementasi algoritma ini akan direpresentasikan dalam bentuk *source code* dengan bahasa pemrograman Java.

5.5.1 Implementasi Algoritma Pencarian *Stave*

Proses pencarian *stave* bertujuan untuk mendeteksi seluruh *stave* yang ada dalam citra partitur. Proses ini merupakan langkah awal dalam pembacaan citra partitur. Proses ini memiliki peranan yang sangat penting, karena *stave* yang terdeteksi dipakai sebagai acuan dalam menentukan nada pada tiap notasi.

```
1.  public static ArrayList<Line> scanning(BufferedImage src, int
2.        lineThreshold) {
3.          ArrayList<Line> lines = new ArrayList<Line>();
4.          int height = src.getHeight();
5.          int width = src.getWidth();
6.
7.          int pixels[][] = ImageMatrix.getMatrix_hw(src);
8.
9.          Histogram histogram;
10.
11.         for (int i = 0; i < height; i++) {
12.             histogram = new Histogram(pixels[i], width, 1, 0,
13.       width);
14.             if (histogram.getFrequency(0) >= lineThreshold) {
15.                 lines.add(new Line(0, i, (int)
16.       histogram.getFrequency(0)));
17.             }
18.         }
19.         return lines;
20.     }
21.
```

Gambar 5.1 Implementasi algoritma Pencarian *Stave*

Sumber : [Implementasi]

Penjelasan implementasi algoritma Pencarian *Stave* pada Gambar 5.1 yaitu:

1. Baris 3 – 5 menjelaskan tentang variabel yang dibutuhkan dalam proses ini
2. Baris 7 menjelaskan variabel *pixels* yang digunakan untuk menyimpan nilai RGB setiap piksel dari citra
3. Baris 8 merupakan instansiasi objek *Histogram*
4. Baris 11 – 20 menjelaskan proses pencarian *stave* dengan cara menghitung nilai histogram tiap baris. Apabila nilai histogram tiap baris melebihi dari nilai *threshold stave* maka baris tersebut merupakan *stave*
5. Baris 20 menjelaskan tentang nilai kembalian dari proses yang berupa kumpulan dari *stave*

5.5.2 Implementasi Algoritma Penghapusan *Stave*

Proses penghapusan *stave* bertujuan untuk menghapus seluruh *stave* yang terdetekai pada citra. Proses ini dilakukan agar proses pengelompokan objek dapat berjalan dengan baik.

```
1.  public static void removing(BufferedImage image,
2.  ArrayList<StaveLine> ags, int staveHeight) {
3.          int width = image.getWidth();
4.          BufferedImage temp = new BufferedImage(width,
5.  image.getHeight(), image.getType());
6.          temp.setData(image.getData());
7.          int range = staveHeight + 3;
8.          int[] pixels = new int[width * range];
9.          int black;
10.         for (StaveLine ss : ags) {
11.             if(!ss.isLine()) continue;
12.             ImageUtils.getRGB(temp, 0, ss.getPosition() - rr,
13. width, range, pixels);
14.             for (int j = 0; j < width; j++) {
15.                 black = 0;
16.                 for (int i = 0; i < range; i++) {
17.                     if (pixels[j + i * width] ==
18. Color.BLACK.getRGB()) {
19.                         black++;
20.                     }
21.                 }
22.                 if (black <= staveHeight) {
23.                     for (int i = -rr; i <= rr; i++) {
24.                         image.setRGB(j, ss.getPosition() + i,
25. Color.WHITE.getRGB());
26.                     }
```

```
27.                     }
28.                 }
29.             }
30.         }
```

Gambar 5.2 Implementasi algoritma Penghapusan *Stave*

Sumber : [Implementasi]

Penjelasan implementasi algoritma Penghapusan *Stave* pada Gambar 5.2 yaitu:

1. Baris 3 – 9 menjelaskan tentang variabel yang dibutuhkan dalam proses ini
2. Baris 10 – 30 menjelaskan tentang proses penghapusan *stave*. Diawali dengan mengambil sejumlah piksel sebesar lebar *stave* + *3*. Apabila jumlah piksel hitam kurang dari lebar *stave*, maka piksel tersebut akan dihapus. Proses ini diulangi sebanyak jumlah *stave* yang terdeksi

5.5.3 Implementasi Algoritma *Region Growing*

Proses *region growing* bertujuan untuk mengelompokkan (segmentasi) seluruh objek yang terdeteksi dalam citra partitur. *Source code* proses *region growing* diambil dari buku *Digital Image Processing* [GON-07].

```
1.  public void segmentation() {
2.          regions = 0;
3.          for (int i = 0; i < height; i++) {
4.              for (int j = 0; j < width; j++) {
5.                  int rgb = isrc.getRGB(j, i);
6.                  if (isBlack(rgb) && label[j][i] <= 0) {
7.                      regions++;
8.                      sNode.push(new Node(j, i, true,
9.          regions));
10.                     label[j][i] = regions;
11.                 }
12.                 while (sNode.size() > 0) {
13.                     Node node = sNode.pop();
14.                     for (int th = -1; th <= 1; th++) {
15.                         for (int tw = -1; tw <= 1; tw++) {
16.                             int rx = node.getX() + tw;
17.                             int ry = node.getY() + th;
18.                             // Skip pixels outside of the
19.         image.
20.                             if ((rx < 0) || (ry < 0) || (ry
21.         >= height) || (rx >= width)) {
22.                                 continue;
23.                             }
24.                             if (label[rx][ry] <= 0) {
25.                                 if (isrc.getRGB(rx, ry) ==
26.         isrc.getRGB(node.getX(), node.getY())) {
27.                                     sNode.push(new Node(rx,
```

```
28.         ry, true, regions));
29.                                         label[rx][ry] = regions;
30.                                     }
31.                                 }
32.                             }
33.                         }
34.                     }
35.                 }
36.             }
37.             sNode.removeAllElements();
38.         }
```

Gambar 5.3 Implementasi algoritma *Region Growing*

Sumber : [Implementasi]

Penjelasan implementasi algoritma *Region Growing* pada Gambar 5.3 yaitu:

1. Baris 3 – 35 menjelaskan tentang proses segmentasi pada citra partitur
2. Baris 6 – 11 menjelaskan tentang proses deteksi piksel yang ditunjuk sudah terlabeli atau belum. Bila belum terlabeli, maka piksel akan diberi label sesuai region
3. Baris 12 – 35 menjelaskan tentang proses pengambilan 8 piksel tetangga dari piksel yang ditunjuk dan memberi label sesuai dengan region pada piksel yang ditunjuk
4. Apabila piksel telah terlabeli, maka *pointer* akan mencari piksel yang belum terlabeli sampai seluruh piksel terlabeli

5.5.4 Implementasi Algoritma Pengenalan Simbol Musik

Proses pengenalan simbol musik bertujuan untuk mengenali objek yang terdetksi dengan proses *region growing* merupakan simbol musik atau bukan. Proses pengenalan menggunakan algoritma *Coefficient Correlation and Covariance* yang diambil dari salah satu *plugin ImageJ* [GON-07][BUR-07].

```
1.      public void matching(Node node) {
2.          BufferedImage img = node.getImage();
3.          BufferedImage resize = ImageResize.resize(img, 100,
4.      100);
5.          ImagePlus im = new ImagePlus(null, resize);
6.
7.          for (int i = 0; i < images.length; i++) {
8.              matcher = new CorrCoeffMatcher((FloatProcessor)
9.      im.getProcessor().convertToFloat(), (FloatProcessor)
10.     images[i].getProcessor().convertToFloat());
11.             FloatProcessor computeMatch =
12.     matcher.computeMatch();
13.             similarity[i] = computeMatch.getMax();
```

```
14.         }
15.         int idx = getIndexMaxSimilarity(similarity);
16.         if (similarity[idx] > 0.7) {
17.             node.setName(files[idx].getName());
18.             if (node.getName().contains("note") ||
19.       node.getName().contains("clef")) {
20.                 tNodes.add(node);
21.             }
22.         }
23.     }
```

Gambar 5.4 Implementasi algoritma Pengenalan Simbol Musik

Sumber : [Implementasi]

Penjelasan implementasi algoritma Pengenalan Simbol Musik pada Gambar 5.4 yaitu:

1. Baris 2 – 5 menjelaskan tentang proses *resize* citra dari setiap objek yang telah terdeteksi
2. Baris 7 – 15 menjelaskan tentang proses perhitungan *similarity* dari setiap objek dengan *template* dari simbol musik
3. Baris 16 – 21 menjelaskan tentang proses pengenalan simbol musik. Proses pengenalan didasari dengan nilai *similarity* tiap objek. Apabila nilai *similarity* lebih dari 0.7 maka objek termasuk simbol musik

5.5.5 Implementasi Algoritma Perhitungan Titik Tengah Simbol Musik

Proses perhitungan titik tengah bertujuan untuk memberi posisi titik tengah pada setiap simbol musik. Titik tengah setiap simbol musik inilah yang digunakan untuk mengetahui nada dari setiap simbol musik.

```
1.  public void run(ArrayList<Node> tones) {
2.          int rowCenter = 0;
3.          int colCenter = 0;
4.          int pixelCounter = 0;
5.
6.          for (Node node : tones) {
7.              int awl = node.getY();
8.              int akh = node.getY() + node.getHeight();
9.
10.             if (node.getName().contains("eight_up")) {
11.                 awl += node.getHeight() / 2;
12.             } else if (node.getName().contains("eight_down"))
13.       {
14.                 akh -= node.getHeight() / 2;
15.             }
16.             for (int k = awl; k < akh; k++) {
17.                 for (int j = node.getX(); j < node.getX() +
18.       node.getWidth(); j++) {
19.                     if (image.getRGB(j, k) ==
```

```
20.         Color.BLACK.getRGB()) {
21.                         rowCenter += k;
22.                         colCenter += j;
23.                         pixelCounter++;
24.                     }
25.                 }
26.             }
27.             node.setCtr_x(Math.round(colCenter /
28.         pixelCounter));
29.             node.setCtr_y(Math.round(rowCenter /
30.         pixelCounter));
31.             pixelCounter = 0;
32.             rowCenter = 0;
33.             colCenter = 0;
34.         }
35.     }
```

Gambar 5.5 Implementasi algoritma Perhitungan Titik Tengah Simbol Musik

Sumber : [Implementasi]

Penjelasan implementasi algoritma Perhitungan Titik Tengah Simbol Musik pada Gambar 5.5 yaitu:

1. Baris 2 – 4 menjelaskan tentang variabel yang dibutuhkan dalam proses ini
2. Baris 6 – 15 menjelaskan tentang inisialisasi nilai variable *awal* dan *akhir* yang berfungsi sebagai batasan penelusuran pada citra
3. Baris 16 – 30 menjelaskan tentang proses perhitungan titik tengah

5.5.6 Implementasi Algoritma Konversi Notasi

Proses konversi bertujuan untuk memberi nilai notasi angka pada setiap simbol musik yang ada. Proses ini merupakan tujuan utama dari aplikasi. Acuan pengonversian notasi ditaruh dalam file xml.

```
1.  public static void konversi(String xmlPath, ArrayList<Node>
2.         tones, String nadaDasar) {
3.           try {
4.               SAXReader reader = new SAXReader();
5.               Document document = reader.read(xmlPath);
6.
7.               Element root = document.getRootElement();
8.               for (Iterator it = root.elementIterator("clef");
9.         it.hasNext();) {
10.                  Element clef = (Element) it.next();
11.                  String attr = clef.attributeValue("name");
12.                  if (attr.equals(nadaDasar)) {
13.                      for (Iterator cit =
14.         clef.elementIterator("nada"); cit.hasNext();) {
15.                          Element nada = (Element) cit.next();
16.                          String n = (String)
17.         nada.element("tone").getData();
```

```
18.                         String a = (String)
19.      nada.element("angka").getData();
20.                         for (Node node : tones) {
21.                             if (node.getTones().contains(n))
22.      {
23.                                 node.setAngka(a);
24.                             }
25.                         }
26.                     }
27.                 }
28.             }
29.           } catch (DocumentException ex) {
30.
31.      Logger.getLogger(NotesConversion.class.getName()).log(
32.      Level.SEVERE, null, ex);
33.           }
34.     }
```

Gambar 5.6 Implementasi algoritma Konversi Notasi

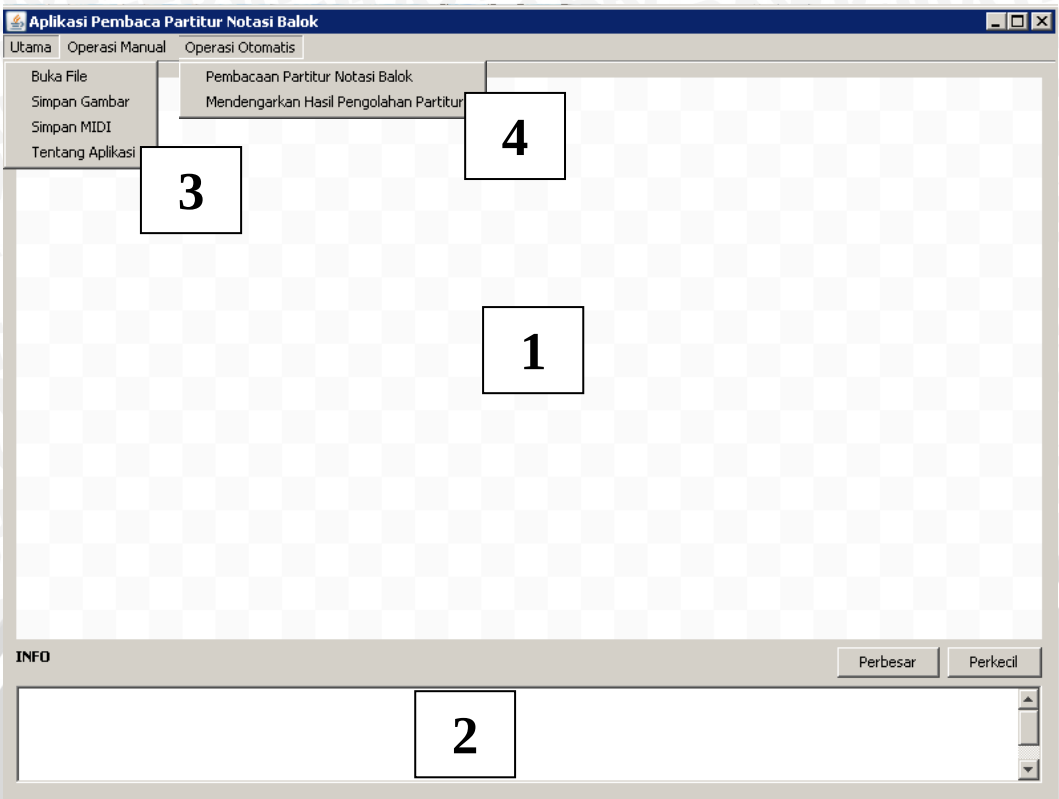
Sumber : **[Implementasi]**

Penjelasan implementasi algoritma Konversi Notasi pada Gambar 5.6 yaitu:

1. Baris 4 – 5 menjelaskan tentang proses pembacaan file xml
2. Baris 7 – 19 menjelaskan tentang proses *parsing* data dari file xml
3. Baris 20 – 24 menjelaskan tentang proses pemberian nilai notasi angka untuk setiap simbol musik

5.6 Implementasi Antarmuka

Implementasi antarmuka *Aplikasi Pembaca Partitur Notasi Balok* digunakan oleh pengguna untuk berinteraksi dengan sistem perangkat lunak. Implementasi dilakukan dengan komponen *graphical user interface* Java Swing dan Java SwingX.



Gambar 5.7 Implementasi antarmuka *Aplikasi Pembaca Partitur Notasi Balok*

Sumber : [Implementasi]

Penjelasan implementasi antarmuka *Aplikasi Pembaca Partitur Notasi Balok* pada Gambar 5.7 yaitu:

1. *Image Container*, komponen Java SwingX yang berfungsi sebagai *container* dari citra partitur yang dimasukkan dalam aplikasi. Komponen ini dilengkapi dengan fitur perbesar atau perkecil citra yang terdapat di dalamnya
2. Kotak info, komponen yang berfungsi untuk mencatat segala aktivtas yang dilakukan pengguna
3. Menu utama, komponen yang berisi fitur buka file citra, simpan citra dan simpan audio midi hasil pemrosesan citra
4. Menu pembacaan otomatis, komponen yang berisi fitur pemrosesan citra partitur secara otomatis dan fitur mendengarkan audio midi hasil pemrosesan citra

BAB VI
PENGUJIAN DAN PEMBAHASAN

Bab ini membahas mengenai tahapan pengujian dan pembahasan perangkat lunak *Aplikasi Pembaca Partitur Notasi Balok* yang telah dikembangkan. Proses pengujian dilakukan melalui tiga tahapan (strategi) yaitu pengujian fungsional, pengujian akurasi dan pengujian performa. Pada pengujian fungsional akan digunakan teknik pengujian *blackbox.* Pada pengujian akurasi dilakukan dengan menghitung prosentase kebenaran aplikasi dalam membaca citra partitur. Pada pengujian performa akan dilakukan dengan cara menghitung waktu eksekusi yang dibutuhkan aplikasi untuk memproses citra partitur.

6.1 Pengujian Fungsional

Pengujian fungsional digunakan untuk mengetahui apakah sistem yang dibangun sudah benar sesuai dengan yang dibutuhkan. *Item – item* yang telah dirumuskan dalam daftar kebutuhan dan merupakan hasil analisis kebutuhan akan menjadi acuan untuk melakukan pengujian fungsional. Pengujian fungsional menggunakan metode pengujian *black box*, karena tidak diperlukan konsentrasi terhadap alur jalannya algoritma program dan lebih ditekankan untuk menemukan konformitas antara kinerja sistem dengan daftar kebutuhan.

6.1.1 Kasus Uji Fungsional

Tabel 6.1 Kasus uji untuk pengujian fungsional memasukkan citra partitur

Nama Kasus Uji	Kasus Uji Memasukkan Citra Partitur
Objek Uji	SRS-001-01
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional bagi pengguna untuk memasukkan citra partitur
Prosedur Uji	1. Pengguna membuka aplikasi 2. Tampilan utama keluar 3. Pengguna memilih menu [Utama] 4. Pengguna memilih menu item [Buka File] 5. Tampilan *browse file* muncul

	6. Pengguna memilih citra partitur
Hasil yang diharapkan	Aplikasi dapat menampilkan citra partitur yang telah dipilih oleh pengguna

Sumber : [Pengujian dan Pembahasan]

Tabel 6.2 Kasus uji untuk pengujian fungsional memproses citra partitur

Nama Kasus Uji	Kasus Uji Memproses Citra Partitur
Objek Uji	SRS-001-02
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional bagi pengguna untuk memproses citra partitur
Prosedur Uji	1. Aplikasi telah berjalan dan pengguna telah memasukkan citra partitur 2. Pengguna memilih menu [Operasi Ototmatis] 3. Pengguna memilih menu item [Pembacaan Partitur Notasi Balok]
Hasil yang diharapkan	Aplikasi dapat menampilkan citra partitur yang telah selesai diproses

Sumber : [Pengujian dan Pembahasan]

Tabel 6.3 Kasus uji untuk pengujian fungsional melihat hasil pemrosesan citra

Nama Kasus Uji	Kasus Uji Melihat Hasil Pemrosesan Citra
Objek Uji	SRS-001-03
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional bagi pengguna untuk melihat hasil pemrosesan citra
Prosedur Uji	1. Aplikasi telah berjalan dan pengguna telah memasukkan citra partitur serta memproses citra 2. Pengguna menunggu hasil pemrosesan citra
Hasil yang diharapkan	Aplikasi dapat menampilkan citra partitur yang telah selesai diproses

Sumber : [Pengujian dan Pembahasan]

Tabel 6.4 Kasus uji untuk pengujian fungsional mendengarkan audio hasil pemrosesan citra

Nama Kasus Uji	Kasus Uji Mendengarkan Audio Hasil Pemrosesan Citra
Objek Uji	SRS-001-04
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional bagi pengguna untuk mendengarkan audio hasil pemrosesan citra
Prosedur Uji	1. Aplikasi telah berjalan dan pengguna telah memasukkan citra partitur serta memproses citra 2. Pengguna memilih menu [Operasi Ototmatis] 3. Pengguna memilih menu item [Mendengarkan Hasil Pengolahan Citra]
Hasil yang diharapkan	Aplikasi dapat memutar audio midi dari notasi yang tertera pada citra partitur

Sumber : [Pengujian dan Pembahasan]

Tabel 6.5 Kasus uji untuk pengujian fungsional menyimpan citra

Nama Kasus Uji	Kasus Uji Menyimpan Citra
Objek Uji	SRS-001-05
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional bagi pengguna untuk menyimpan citra
Prosedur Uji	1. Aplikasi telah berjalan dan pengguna telah memasukkan citra partitur serta memproses citra 2. Pengguna memilih menu [Utama] 3. Pengguna memilih menu item [Simpan Gambar] 4. Tampilan *browse file* muncul 5. Pengguna memilih *directory* sebagai tempat menyimpan citra
Hasil yang diharapkan	Aplikasi dapat menyimpan citra sesuai *directory* yang dipilih oleh pengguna

Sumber : [Pengujian dan Pembahasan]

Tabel 6.6 Kasus uji untuk pengujian fungsional menyimpan MIDI

Nama Kasus Uji	Kasus Uji Menyimpan MIDI
Objek Uji	SRS-001-06

Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional bagi pengguna untuk menyimpan file audio MIDI
Prosedur Uji	1. Aplikasi telah berjalan dan pengguna telah memasukkan citra partitur serta memproses citra 2. Pengguna memilih menu [Utama] 3. Pengguna memilih menu item [Simpan MIDI] 4. Tampilan *browse file* muncul 5. Pengguna memilih *directory* sebagai tempat menyimpan file audio MIDI
Hasil yang diharapkan	Aplikasi dapat menyimpan file audio MIDI sesuai *directory* yang dipilih oleh pengguna

Sumber : [Pengujian dan Pembahasan]

6.1.2 Hasil Pengujian Fungsional

Tabel 6.7 Hasil Pengujian Fungsional

No	Nama Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan	Status Validitas
1	Kasus Uji Memasukkan Citra Partitur	Aplikasi dapat menampilkan citra partitur yang telah dipilih oleh pengguna	Aplikasi dapat menampilkan citra partitur yang telah dipilih oleh pengguna	Valid
2	Kasus Uji Memproses Citra Partitur	Aplikasi dapat menampilkan citra partitur yang telah selesai diproses	Aplikasi dapat menampilkan citra partitur yang telah selesai diproses	Valid
3	Kasus Uji Melihat Hasil Pemrosesan Citra	Aplikasi dapat menampilkan citra partitur yang telah selesai diproses	Aplikasi dapat menampilkan citra partitur yang telah selesai diproses	Valid
4	Kasus Uji Mendengarkan Audio Hasil	Aplikasi dapat memutar audio midi dari notasi yang	Aplikasi dapat memutar audio midi dari notasi yang	Valid

	Pemrosesan Citra	tertera pada citra partitur	tertera pada citra partitur	
5	Kasus Uji Menyimpan Citra	Aplikasi dapat menyimpan citra sesuai *directory* yang dipilih oleh pengguna	Aplikasi dapat menyimpan citra sesuai *directory* yang dipilih oleh pengguna	Valid
6	Kasus Uji Menyimpan MIDI	Aplikasi dapat menyimpan file audio MIDI sesuai *directory* yang dipilih oleh pengguna	Aplikasi dapat menyimpan file audio MIDI sesuai *directory* yang dipilih oleh pengguna	Valid

Sumber : [Pengujian dan Pembahasan]

6.1.3 Pembahasan Hasil Pengujian Fungsional

Proses pembahasan terhadap hasil pengujian fungsional dilakukan dengan melihat konformitas antara hasil kinerja sistem dengan daftar kebutuhan. Berdasarkan hasil pengujian fungsional dapat disimpulkan bahwa implementasi dan fungsionalitas *Aplikasi Pembaca Partitur Notasi Balok* telah memenuhi kebutuhan yang telah dijabarkan pada tahap analisis kebutuhan.

6.2 Pengujian Akurasi

Pengujian akurasi dilakukan untuk mengetahui tingkat ketepatan aplikasi dalam menyelesaikan suatu masalah. Pengujian akurasi pada skripsi ditujukan untuk menghitung ketepatan aplikasi mengubah notasi balok menjadi notasi angka pada citra partitur.

6.2.1 Hasil Pengujian Akurasi

Tabel 6.8 Hasil Pengujian Akurasi

No	Nama Citra Partitur	Asal Citra	Akurasi	Tingkat Kesalahan
1	Silent Night	*Sibelius*	100 %	0 %

2	Amazing Grace	*Sibelius*	100 %	0 %
3	American Pie	PDF	98 %	2 %
4	Fly Me To The Moon	PDF	97 %	3 %
5	Can You Feel The Love Tonight	PDF	99 %	1 %
6	As Fair as Morn	PDF	100 %	0 %
7	Cantigas de Santa Maria	PDF	100 %	0 %
8	Lazy Song	PDF	99 %	1 %
9	Wake Me Up When September Ends	PDF	100 %	0 %
10	Viva la Vida	PDF	98 %	2 %
11	Home	PDF	100 %	0 %
12	Everything	PDF	99 %	1 %
13	American Pie	*Scanning*	77 %	23 %
14	As Fair as Morn	*Scanning*	93 %	7 %
15	Can You Feel The Love Tonight	*Scanning*	50 %	50 %
16	Cantigas de Santa Maria	*Scanning*	98 %	2 %
17	Home	*Scanning*	94 %	6 %
18	Viva la Vida	*Scanning*	91 %	9 %
19	Lazy Song	*Scanning*	95 %	4 %
20	Ave Maria (posisi citra miring)	*Scanning*	0 %	100 %
21	Spotlight (tulisan tangan)	*Scanning*	50 %	50 %

Sumber : [Pengujian dan Pembahasan]

6.2.2 Pembahasan Hasil Pengujian Akurasi

Proses pembahasan terhadap hasil pengujian akurasi dilakukan dengan melihat tingkat akurasi dan tingkat kesalahan aplikasi dalam mengubah notasi balok pada citra partitur. Berdasarkan hasil pengujian akurasi dapat disimpulkan bahwa *Aplikasi Pembaca Partitur Notasi* memiliki tingkat akurasi yang cukup tinggi sebesar 99% untuk citra partitur yang berasal dari *Sibelius* maupun bertipe data PDF, namun mengalami penurunan tingkat akurasi untuk citra partitur hasil *scanning* menjadi sebesar 85%. Hal ini kemungkinan disebabkan oleh kualitas hasil *scanning* yang menghasilkan *noise* pada citra, alat yang digunakan untuk melakukan *scanning.* Untuk mengetahui penyebab sebenarnya diperlukan penelitian lebih lanjut.

6.3 Pengujian Performa

Pengujian performa dilakukan untuk mengetahui waktu yang dibutukan aplikasi untuk menyelesaikan suatu masalah. Pengujian performa pada skripsi ditujukan untuk menghitung waktu eksekusi aplikasi untuk memproses citra partitur.

6.3.1 Hasil Pengujian Performa

Tabel 6.9 Hasil Pengujian Performa

No	Nama Citra Partitur	Jumlah Notasi	Waktu Eksekusi Aplikasi (detik)	Waktu Eksekusi Manusia (detik)	Tingkat Kecepatan (cepat / lambat)
1	Amazing Grace	42	24	45	Cepat
2	American Pie	131	106	133	Cepat
3	Silent Night	47	18	66	Cepat
4	Fly Me To The Moon	112	65	136	Cepat
5	Can You Feel The Love Tonight	155	149	193	Cepat

Sumber : [Pengujian dan Pembahasan]

6.3.2 Pembahasan Hasil Pengujian Performa

Proses pembahasan terhadap hasil pengujian akurasi dilakukan dengan mendapatkan waktu eksekusi dari aplikasi dalam memproses citra dan membandingkan waktu eksekusi yang diperlukan manusia untuk menyelesaikan masalah. Berdasarkan hasil pengujian akurasi dapat disimpulkan bahwa *Aplikasi Pembaca Partitur Notasi Balok* memiliki waktu eksekusi yang lebih cepat daripada waktu eksekusi yang diperlukan oleh manusia dengan besaran 0,6 detik/notasi.

6.4 Pembahasan *Reuse-Oriented Model*

Aplikasi telah berhasil dirancang dan diimplementasikan menggunakan *Reuse-Oriented Model*. Analisis tahapan dalam model ini yang diimplementasikan, yaitu :

1. Proses analisis komponen telah dilakukan dan menghasilkan JHLabs dan JFugue sebagai komponen yang digunakan dalam aplikasi
2. Proses modifikasi kebutuhan tidak perlu dilakukan karena komponen-komponen yang ditemukan telah memenuhi kebutuhan yang telah dijabarkan dalam spesifikasi kebutuhan
3. Proses mendesain sistem menggunakan *reuse* telah dilakukan dengan cakupan *program libray* dan pola-pola perancangan (*abstract factory pattern, template method pattern, strategy pattern*). Desain *resue* difokuskan pada rencana pengembangan algortima segmentasi dan pengenalan citra, pengaturan komponen dan aksi-aksi antarmuka, serta pengaturan instansiasi objek.

Proses pengembangan semakin efektif dan efisien dengan menggunakan model ini. Hal ini dikarenakan dengan menggunakan model ini, aplikasi dapat digunakan kembali pada proses pengembangan selanjutnya, baik komponen yang digunakan maupun secara keseluruhan. Dengan demikian *Reuse-Oriented Model* dapat dikatakan sebagai model yang sesuai dalam proses pengambangan aplikasi mengingat terdapat beberapa pilihan metode / algoritma yang dapat digunakan untuk menyelesaikan masalah dalam skripsi ini sehingga pengembang selanjutnya tidak perlu memulai dari awal dalam pembuatan aplikasi.

BAB VII
PENUTUP

7.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut :

1. *Aplikasi Pembaca Partitur Notasi Balok* telah dirancang dan diimplementasikan dengan menggunakan model *Reuse-Oriented* dengan tujuan membuat suatu aplikasi yang berasal dari komponen-komponen yang telah ada dan menjadikan aplikasi ini dapat dengan mudah dikembangkan di kemudian hari
2. *Aplikasi Pembaca Partitur Notasi Balok* telah memenuhi kebutuhan fungsional yang telah dijabarakan dalam spesifikasi kebutuhan
3. *Aplikasi Pembaca Partitur Notasi Balok* memiliki akurasi sebesar 99% untuk citra partitur yang berasal dari *Sibelius* maupun yang bertipe data PDF, namun aplikasi mengalami penurunan akurasi untuk citra partitur berasal dari hasil *scanning* menjadi sebesar 85%
4. *Aplikasi Pembaca Partitur Notasi Balok* mengalami kesalahan pada salah satu kasus uji akurasi dalam mengenali notasi yang memiliki bentuk mirip dengan data *training.* Hal ini dikarenakan tingkat kemiripan notasi tidak memenuhi syarat ambang kemiripan dengan data *training*
5. *Aplikasi Pembaca Partitur Notasi Balok* memiliki waktu eksekusi yang lebih cepat daripada waktu eksekusi yang diperlukan oleh manusia dengan besaran 0,6 detik/notasi.

7.2 Saran

Saran yang dapat diberikan untuk pengembangan perangkat lunak ini antara lain :

1. Untuk pengembangan lebih lanjut aplikasi ini dapat dilakukan penambahan langkah dalam proses *preprocessing* citra, seperti perbaikan orientasi atau penyimpangan citra hasil *scanning* apabila

posisi citra kurang tepat, penghapusan noise pada citra hasil scanning, sehingga dapat mengurangi tingkat kesalahan pada aplikasi

2. Untuk pengembangan lebih lanjut aplikasi ini dapat ditambahkan nada dasar dari citra partitur yang dapat dikenali
3. Untuk pengembangan lebih lanjut aplikasi ini dapat dioptimalkan dalam proses segmentasi dan pengenalan simbol musik sehingga waktu eksekusi semakin cepat serta tingkat akurasi yang semakin tinggi
4. Untuk pengembangan lebih lanjut aplikasi ini dapat ditambahkan pengenalan simbol musik yang lain, seperti tanda mol, tanda kruis, tanda pugar, sehingga lebih banyak simbol musik yang dapat dikenali

DAFTAR PUSTAKA

[ACH-05] Achmad, B dan Firdausy, K.2005.*Teknik Pengolahan Citra Digital Menggunakan Delphi*.Ardhi Publishing.Yogyakarta

[BEL-01] Bellini, P.2001.*Optical Music Sheet Segmentation.*IEEE.Italy

[BOL-12] Su,Bolan.2012.*An Effective Staff Detection and Removal Technique for Musical Documents*. IEEE.

[BUR-07] Burger,Wilhelm.2007.*Digital Image Processing.*Springer

[CIU-10] Cui,Jiali.2010.*An Adaptive Staff Line Removal in Music Score Images.*IEEE

[DEI-06] Deitel. 2006. *Java™ How to Program, Seventh Edition.*Prentice Hall.New Jersey

[FEL-07] Feldman, D.2007.*Apa Gajah Bisa Lompat dan Apa-Mengapa Lainnya*.PT Gramedia Pustaka Utama.Jakarta

[GAM-01] Gamma,Eric.1995.*Design Pattern Elements of Reusable Object-Oriented Software*.Addison-Wesley Pub Co.New York

[GON-07] Gonzalez,Rafael.2007.*Digital Image Processing.*Prentice Hall.New Jersey

[HAR-01] Harnum, Jonathan.2001.*Basic Music Theory*.Sol-Ut Press.Chicago

[HUX-12] Huxtable,Jerry.*JHLabs Java Stuff*. http://www.jhlabs.com/ip/filters/index.html diakses 10 September 2012.

[JFU-13] *Java API for Music Programming*.http://www.jfugue.org/ diakses pada 01 Februari 2013.

[KAM-08] Tim Penyusun.2008.*Kamus Bahasa Indonesia.*Kamus Pusat Bahasa.Jakarta

[NAI-08] Naik ,Kshirasagar.Tripathy,Priyadarshi.2008.*Software Testing and Quality Assurance*.John Wiley & Sons, Inc.

[PHI-07] Philhofer,Michael.2007.*Music Theory For Dummies.*Wiley Publishing.Indiana

[PRE-10] Pressman, Roger. 2010. *Software Engineering: A Practioner's Approach, 7th Edition*. Mc Graw-Hill.

[ROS-11] A.S.,Rosa.2011.*Modul Pembelajaran Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek)*.Modula.Bandung

[SAR-09] Saripudin, Aip dkk.2009.*Praktis Belajar Fisika*.Pusat Perbukuan.Departemen Pendidikan Nasional

[SOM-11] Sommerville, Ian. 2011. *Software Engineering / Ninth Edition*. New York : Addison-Wesley.

[TAN-06] Tan, Pang-Ning., Steinbach, M., dan Kumar, Vipin. 2006. *Data Mining*. Pearson Education, Inc.

LAMPIRAN

Lampiran 1 Hasil Pengujian Performa

No	Nama Responden	Nama Citra Partitur	Waktu (detik)
1	Betha	Amazing Grace	34
2	Rangga		53
3	Mia		30
4	Herry		53
5	Oki		92
6	Agustinus Wahyu		39
7	Sharlene		48
8	Alfredo		40
9	Lala		65
10	Nania		32
Rata-rata			**45**
11	Betha	American Pie	119
12	Pemadi		315
13	Getha		116
14	Rangga		120
15	Mia		80
16	Herry		156
17	Oki		144
18	Sharlene		124
19	Agustinus Wahyu		75
20	Nania		78
Rata-rata			**133**
21	Randy	Silent Night	90
22	Faisal		72
23	Betha		44
24	Permadi		78
25	Lala		88
26	Mia		42
27	Herry		64

28	Agustinus Wahyu		46
29	Oki		68
30	Sharlene		68
Rata-rata			**66**
31	Firman	Fly Me To The Moon	302
32	Rangga		164
33	Getha		109
34	Nania		73
35	Mia		98
36	Herry		166
37	Agustinus Wahyu		79
38	Sharlenen		130
39	Oki		119
40	Alfredo		120
Rata-rata			**136**
41	Oki	Can You Feel The Love Tonight	170
42	Agustinus Wahyu		126
43	Faisal		266
44	Herry		156
45	Sharlene		151
46	Alfredo		230
47	Arum Ajeng		250
Rata-rata			**193**

Sumber : [Pengujian dan Pembahasan]

Lampiran 2 Hasil Pengujian Akurasi

No	Nama Citra Partitur	Musik Simbol	Jumlah Total (a)	Jumlah Terdeteksi Benar (b)	Jumlah Terdeteksi Salah (c)
1	Silent Night	*Whole note*	0	0	0
		Half note	15	15	0
		Quarter note	24	24	0
		Eight note	8	8	0
		Rest note	0	0	0
	Jumlah total		**47**	**47**	**0**
2	Amazing Grace	*Whole note*	0	0	0
		Half note	13	13	0
		Quarter note	15	15	0
		Eight note	12	12	0
		Rest note	2	2	0
	Jumlah total		**42**	**42**	**0**
3	American Pie	*Whole note*	0	0	0
		Half note	4	1	3
		Quarter note	11	11	0
		Eight note	104	104	0
		Rest note	12	12	0
	Jumlah total		**131**	**128**	**3**
4	Fly Me To The Moon	*Whole note*	8	5	3
		Half note	17	17	0
		Quarter note	48	48	0
		Eight note	39	39	0
		Rest note	0	0	0
	Jumlah total		**112**	**109**	**3**
5	Can You Feel The Love Tonight	*Whole note*	0	0	0
		Half note	4	3	1
		Quarter note	23	23	0
		Eight note	113	113	0
		Rest note	15	15	0
	Jumlah total		**155**	**154**	**1**
6	As Fair as Morn	*Whole note*	2	2	0
		Half note	17	17	0
		Quarter note	95	95	0
		Eight note	51	51	0
		Rest note	10	10	0
	Jumlah total		**175**	**175**	**0**
7	Cantigas de Santa Maria	*Whole note*	0	0	0
		Half note	1	1	0
		Quarter note	36	36	0
		Eight note	21	21	0
		Rest note	0	0	0

	Jumlah total		**58**	**58**	**0**
8	Lazy Song	*Whole note*	0	0	0
		Half note	4	3	1
		Quarter note	59	59	0
		Eight note	67	67	0
		Rest note	4	4	0
	Jumlah total		**134**	**133**	**1**
9	Wake Me Up When September Ends	*Whole note*	0	0	0
		Half note	0	0	0
		Quarter note	97	97	0
		Eight note	116	116	0
		Rest note	25	25	0
	Jumlah total		**238**	**238**	**0**
10	Viva la Vida	*Whole note*	0	0	0
		Half note	12	10	2
		Quarter note	35	35	0
		Eight note	71	71	0
		Rest note	11	11	0
	Jumlah total		**129**	**127**	**2**
11	Home	*Whole note*	0	0	0
		Half note	1	1	0
		Quarter note	5	5	0
		Eight note	121	121	0
		Rest note	27	27	0
	Jumlah total		**154**	**154**	**0**
12	Everything	*Whole note*	0	0	0
		Half note	2	2	0
		Quarter note	43	43	0
		Eight note	154	152	2
		Rest note	34	34	0
	Jumlah total		**233**	**231**	**2**
13	American Pie (*Scanning*)	*Whole note*	0	0	0
		Half note	4	1	3
		Quarter note	11	11	0
		Eight note	104	78	26
		Rest note	12	11	1
	Jumlah total		**131**	**101**	**30**
6	As Fair as Morn (*Scanning*)	*Whole note*	2	1	1
		Half note	17	16	1
		Quarter note	95	95	0
		Eight note	51	43	8
		Rest note	10	8	2
	Jumlah total		**175**	**163**	**12**
16	Can You Feel The Love Tonight	*Whole note*	0	0	0
		Half note	4	3	1
		Quarter note	23	19	4

	(*Scanning*)	*Eight note*	113	41	72
		Rest note	15	14	1
	Jumlah total		**155**	**77**	**78**
17	Cantigas de Santa Maria (*Scanning*)	*Whole note*	0	0	0
		Half note	1	1	0
		Quarter note	36	36	0
		Eight note	21	20	1
		Rest note	0	0	0
	Jumlah total		**58**	**57**	**1**
18	Home (*Scanning*)	*Whole note*	0	0	0
		Half note	1	1	0
		Quarter note	5	5	0
		Eight note	121	113	8
		Rest note	27	25	2
	Jumlah total		**154**	**144**	**10**
19	Viva la Vida (*Scanning*)	*Whole note*	0	0	0
		Half note	12	6	6
		Quarter note	35	35	0
		Eight note	71	65	6
		Rest note	11	11	0
	Jumlah total		**129**	**117**	**12**

Sumber : [Pengujian dan Pembahasan]

Lampiran 3 Analisis Pengenalan Objek

Analisis Perhitungan Manual :

Data Uji (I)	*Mean*	$\sum I^2$
1	227,001	578843968

$$Similarity = \frac{\sum IR - \sum piksel * Mean(I) * Mean(R)}{\left(\sqrt{\sum I^2 - \sum piksel * Mean(I) * Mean(I)}\right) * Variance(R)}$$

$$Similarity = \frac{578843968 - 10000 * 227,001 * 229,0665}{\left(\sqrt{578843968 - 10000 * 227,001 * 227,001}\right) * 7706,8990}$$

$$Similarity = 0,8692$$

Nama *Template* (R)	*Mean*	*Variance*	$\sum IR$	*Similarity*
Data 1	229,0665	7706,8990	573381952	0,8692
Data 2	227,5620	7901,2570	544056128	0,4364
Data 3	84,9150	12017,6790	208207120	0,1613
Data 4	91,3410	12226,3830	223487760	0,1656
Data 5	57,8340	10678,3420	145393936	0,1658
Data 6	173,6550	11884,9970	419990304	0,2722
Data 7	81,6000	11895,0190	203395344	0,1915
Data 8	90,3975	12198,0850	225893648	0,2128
Data 9	58,3695	10713,0870	148840208	0,1913
Data 10	69,1050	11334,0320	173549328	0,1846
Data 11	225,5475	8149,8994	520452384	0,1302
Data 12	214,3020	9338,5520	499449632	0,1744
Data 13	222,7935	8470,2750	517981472	0,1812
Data 14	218,5095	8928,9940	509983520	0,1962
Data 15	225,3180	8177,4320	520842528	0,1437
Data 16	244,0860	5160,4526	556280640	0,0536
Data 17	241,7655	5655,7285	551859008	0,0676
Data 18	241,7655	5655,7285	550363456	0,0345
Data 19	230,7750	7476,3970	529685792	0,0977
Data 20	230,0865	7570,5938	526759712	0,0739
Data 21	229,7040	7622,1553	525069088	0,0599
Data 22	238,8330	6213,1370	546982208	0,0975
Data 23	243,6780	5251,6694	555435328	0,0546
Data 24	239,7000	6055,1650	547372352	0,0673
Data 25	230,8005	7472,8716	532546848	0,1448
Data 26	241,7145	5666,0225	551728960	0,0672

Data 27	232,0245	7300,6826	534497568	0,1340
Data 28	223,2780	8415,4500	516030752	0,1370
Data 29	230,5965	7500,9930	534172448	0,1792
Data 30	230,7240	7483,4365	537098560	0,2238
Data 31	232,1775	7278,7295	537748800	0,1845
Data 32	220,4475	8727,0740	505366816	0,0711
Data 33	206,8305	9981,0510	482608416	0,1647
Data 34	217,1325	9067,2180	512389408	0,2697
Data 35	223,4055	8400,9160	532156704	0,3737
Data 36	225,8535	8112,9430	535993120	0,3603
Data 37	218,3820	8941,9730	519477024	0,3331
Data 38	175,9500	11793,2970	389038880	0,1103
Data 39	205,5810	10079,1045	457249056	0,1173
Data 40	203,5920	10230,0970	474415392	0,1503
Data 41	186,5325	11300,7570	413422880	0,1111

Sumber : [Pengujian dan Analisis]

Dari data diatas dapat disimpulkan bahwa data uji 1 dikenali sebagai data *template* 1, karena memiliki tingkat kemiripan sebesar 0,8692 dan telah melebihi ambang batas kemiripan sebesar 0,7