TUGAS AKHIR – KI091391

TEMU KEMBALI CITRA BERBASIS ISI PADA CITRA KAIN BERDASARKAN FITUR WARNA, TEKSTUR, DAN BENTUK

ATIQOTUN NISWAH
NRP 5110 100 183

Dosen Pembimbing 1
Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

Dosen Pembimbing 2
Diana Purwitasari, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2014

UNDERGRADUATE THESES – KI091391

CONTENT BASED IMAGE RETRIEVAL ON FABRIC IMAGE USING COLOUR, TEXTURE, AND SHAPE FEATURES

ATIQOTUN NISWAH
NRP 5110 100 183

First Advisor
Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

Second Advisor
Diana Purwitasari, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2014

TEMU KEMBALI CITRA BERBASIS ISI PADA CITRA KAIN BERDASARKAN FITUR WARNA, TEKSTUR, DAN BENTUK

Nama Mahasiswa	**: ATIQOTUN NISWAH**
NRP	**: 5110 100 183**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Dr. Eng. Nanik Suciati, S.Kom., M.Kom.**
Dosen Pembimbing 2	**: Diana Purwitasari, S.Kom., M.Sc.**

Abstrak

Kain adalah bahan baku pembuatan berbagai produk, seperti seprai, taplak, dan baju. Kain memiliki berbagai jenis bahan, warna, dan motif. Terkadang, beberapa kain berbeda dipadukan untuk membuat suatu produk, misalnya baju. Tidak sembarang kain dipilih. Warna yang senada ataupun motif yang mirip biasanya menjadi syarat kain tersebut dipadukan dengan kain contoh. Untuk mempermudah pencarian kain yang mirip dengan kain contoh, content-based image retrieval (CBIR) dapat menjadi salah satu solusi. CBIR dapat menemukan kain yang mirip sehingga membantu pemilihan kain yang akan dipadukan dengan kain contoh.

Dalam Tugas Akhir ini, dibangun sistem temu kembali citra kain menggunakan metode dominant color descriptor *(DCD),* steerable filter, *dan* pseudo-Zernike moments. *DCD digunakan untuk mengekstraksi fitur warna dari citra yang warnanya telah dikuantisasi.* Steerable filter *digunakan untuk mengekstraksi fitur tekstur.* Pseudo-Zernike moment *digunakan untuk mengekstraksi fitur bentuk.*

Hasil pencarian citra mirip yang paling baik didapatkan dengan menggabungkan ketiga fitur, yaitu warna, tekstur, dan bentuk. Bobot dari tiap fitur berbeda, yaitu 0.6 untuk fitur tekstur, 0.25 untuk fitur warna, dan 0.15 untuk fitur bentuk. Apabila

hanya digunakan satu fitur, yang paling baik digunakan untuk pencarian adalah fitur warna.

Kata kunci : Temu Kembali Citra, Dominant Color Descriptor, Steerable Filter, Pseudo-Zernike Moment, Kain.

CONTENT BASED IMAGE RETRIEVAL ON FABRIC IMAGE USING COLOR, TEXTURE, AND SHAPE FEATURES

Student's Name	**: ATIQOTUN NISWAH**
Student's ID	**: 5110 100 183**
Department	**: Informatics, FTIF-ITS**
First Advisor	**: Dr. Eng. Nanik Suciati, S.Kom., M.Kom.**
Second Advisor	**: Diana Purwitasari, S.Kom., M.Sc.**

Abstract

Fabric is the raw material for making various products, such as bedsheet, tablecloth, and clothes. Fabrics have different types of materials, colors, and motif. Sometimes, more than one different fabrics combine to create a product, such as clothes. The selected fabrics are chosen by similarity. Matching in color or motif is usually a requirement to combined some fabrics. To facilitate the search of similar fabric, content-based image retrieval (CBIR) can be the solution. CBIR can find the similar fabric to help the selection of fabric that will be combine with another fabric.

In this Final Project, image retrieval system is constructed using dominant color descriptor (DCD), a steerable filter, and pseudo-Zernike moments. DCD is used to extract color features of quantized colored image. Steerable filter is used to extract texture features. Pseudo-Zernike moments are used to extract the shape feature.

The best result of image retrieval obtained by combining three features, color, texture, and shape. The weight of each features are different, 0.6 for texture features, 0.25 for color features, and 0.15 for shape features. Meanwhile, when only one feature is selected, color feature is the best used for the search.

Keywords: Image Retrieval, Dominant Color Descriptor, Steerable Filter, Pseudo-Zernike Moment, fabric.

LEMBAR PENGESAHAN

TEMU KEMBALI CITRA BERBASIS ISI PADA CITRA KAIN BERDASARKAN FITUR WARNA, TEKSTUR, DAN BENTUK

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visualisasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
ATIQOTUN NISWAH
NRP. 5110 100 183

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Dr. Eng. Nanik Suciati, S.Kom., M.Kom. .....
   NIP: 1971042 8199412 2 001 (Pembimbing 1)

2. Diana Purwitasari, S.Kom., M.Sc. .....
   NIP: 1977121 7200312 1 001 (Pembimbing 2)

SURABAYA
JULI, 2014

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah melimpahkan rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul :

"Temu Kembali Citra Berbasis Isi pada Citra Kain Berdasarkan Fitur Warna, Tekstur, dan Bentuk"

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat berharga bagi penulis. Dengan pengerjaan tugas akhir ini penulis bisa memperdalam, meningkatkan, serta menerapkan apa yang telah penulis dapatkan selama menempuh perkuliahan di Teknik Informatika ITS.

Terselesaikannya buku Tugas Akhir ini tak lepas dari dukungan dan bantuan semua pihak. Oleh karena itu, penulis ingin menyampaikan terima kasih kepada:

1. Allah SWT atas limpahan rahmat-Nya sehingga penulis dapat menyelesaikan tugas akhir ini dengan baik
2. Kedua orang tua penulis, yang telah memberikan dukungan jasmani dan rohani untuk penulis.
3. Ibu Dr. Eng. Nanik Suciati, S.Kom., M.Kom. dan Ibu Diana Purwitasari, S.Kom., M.Sc. selaku Dosen Pembimbing, yang telah memberikan kepercayaan, motivasi, bimbingan, dukungan, nasehat, perhatian, serta semua bantuan yang telah diberikan kepada penulis.
4. Teman-teman dari penulis yang memberikan semangat dalam menyelesaikan tugas akhir ini.
5. Serta semua pihak yang belum sempat disebutkan.

Kesempurnaan tentu masih jauh tercapai pada tugas akhir ini, maka penulis mengharapkan saran dan kritik yang membangun dari pembaca untuk perbaikan ke depan.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kain adalah bahan baku pembuatan berbagai produk, seperti seprai, taplak, dan baju. Kain memiliki berbagai jenis bahan, warna, dan motif. Terkadang, beberapa kain berbeda dipadukan untuk membuat suatu produk, misalnya baju. Tidak sembarang kain dipilih. Warna yang senada ataupun motif yang mirip biasanya menjadi syarat kain tersebut dipadukan dengan kain contoh. Untuk mempermudah pencarian kain yang mirip dengan kain contoh, *content-based image retrieval* (CBIR) dapat menjadi salah satu solusi. CBIR dapat menemukan kain yang mirip sehingga membantu pemilihan kain yang akan dipadukan dengan kain contoh.

Dalam era digital seperti saat ini, penyimpanan digital sudah biasa digunakan untuk mempermudah pekerjaan, termasuk data citra. Penyimpanan berupa data citra dapat diolah sehingga pengguna dapat mencari citra yang mirip dengan citra contoh. Teknik untuk memperoleh citra yang mirip dari *database* citra berdasarkan fitur-fiturnya disebut temu kembali citra berbasis isi atau *content-based image retrieval*.

Warna, tekstur, dan bentuk adalah fitur yang sering digunakan untuk mendiskripsikan suatu citra. Setiap fitur merepresentasikan hal yang berbeda dan membutuhkan penanganan yang berbeda pula. Fitur warna, misalnya, akan menunjukkan warna apa saja yang terdapat dalam suatu citra dan berapa persen suatu warna mendominasi citra. Bila suatu sistem temu kembali citra menggunakan fitur warna sebagai fitur pembanding, maka citra yang terambil adalah citra yang memiliki komposisi warna yang mirip dengan citra contoh. Satu fitur saja kurang handal bila digunakan sebagai pembanding antar citra, kecuali bila memang ingin membandingkan hanya terhadap satu fitur, misalnya hanya

melihat kesamaan warna. Gabungan beberapa fitur sebagai pembanding antar citra diharapkan dapat menemukan citra yang lebih mirip dengan citra contoh.

Penelitian tentang CBIR yang menggunakan gabungan beberapa fitur telah banyak dilakukan, seperti yang dilakukan oleh Xiang-Yang Wang et al [1]. Tiga fitur yaitu warna, tekstur, dan bentuk berhasil dikombinasikan untuk mendapat citra dari *database* yang mirip dengan citra contoh dengan lebih akurat dan efisien. Tiga fitur tersebut diekstrak secara terpisah. *Dominant color descriptor* (DCD) digunakan untuk mengekstraksi fitur warna dengan keluaran berupa suatu set warna dominan dan persentasenya dalam suatu citra. *Steerable filter* digunakan untuk mengekstraksi fitur tekstur dengan cara menggabungkan informasi yang didapat dari citra setelah disaring dengan *basis filter* dalam beberapa sudut berbeda. *Pseudo-Zernike moment* digunakan untuk mengekstraksi fitur bentuk berbasis *region* dengan cara merekonstruksi citra dengan suatu rumus sehingga mendekati bentuk objek pada citra asal. Setelah itu, hasil ekstraksi citra contoh akan dihitung kemiripannya dengan hasil ekstraksi dari *database* citra. Rumus perhitungan yang digunakan berbeda untuk tiap fitur. Hasil penelitian menunjukkan bahwa hasil CBIR dengan penggabungan tiga fitur tersebut lebih baik jika dibandingkan dengan hanya menggunakan fitur warna dan histogram dari subblok.

DCD adalah metode ekstraksi warna yang tidak bergantung pada *color space*. Manjunath et al [2] menyatakan bahwa DCD merepresentasikan fitur warna dari citra sesuai warna yang terdapat dalam citra tersebut, tidak terpaku pada *color space*. Representasi warna dengan DCD juga bagus untuk merepresentasikan suatu *region*. Dalam sistem temu kembali citra, representasi warna dengan DCD memiliki keunggulan yaitu jumlah warna yang dihasilkan cukup sedikit tetapi tetap mewakili warna keseluruhan, sehingga proses pencarian dapat dilakukan dengan cepat.

Steerable filter sebagai salah satu jenis *oriented filter* mengambil informasi tekstur dari sudut tertentu. Dengan mengambil dari beberapa sudut berbeda, informasi yang didapat lebih beragam. *Zernike moment* termasuk salah satu jenis momentum yang dapat digunakan untuk mengekstraksi fitur bentuk. Momentum ini mengambil informasi *region*, bagian tepi sekaligus interiornya, dari suatu citra. Ekstraksi fitur bentuk dengan *Zernike moment* akan menghasilkan nilai yang tidak terpengaruh pada orientasi citra. Sehingga untuk citra sama, dengan salah satu diputar beberapa derajat, akan menghasilkan nilai yang sama. Momentum ini juga lebih tahan terhadap *noise* sehingga representasi fitur bentuk yang didapat akan lebih baik.

Dalam Tugas Akhir ini, metode yang akan diimplementasikan untuk mengekstraksi fitur adalah *dominant color descriptor* (DCD), *steerable filter*, dan *pseudo-Zernike moments*. DCD digunakan untuk mengekstraksi fitur warna dari citra yang warnanya telah dikuantisasi. *Steerable filter* digunakan untuk mengekstraksi fitur tekstur. *Pseudo-Zernike moment* digunakan untuk mengekstraksi fitur bentuk. Ketiga fitur tersebut kemudian akan digunakan dalam proses pencarian citra sehingga akan didapat citra yang mirip dengan citra contoh.

1.2 **Perumusan Masalah**

Perumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana melakukan ekstraksi fitur dengan menggunakan *dominant color descriptor* untuk fitur warna, *steerable filter* untuk fitur tekstur, dan *pseudo-Zernike moment* untuk fitur bentuk.
2. Bagaimana melakukan pencarian citra dari *database* yang mirip dengan citra contoh berdasarkan salah satu dari fitur warna, tekstur, dan bentuk, atau gabungan beberapa fitur.

1.3 **Batasan Masalah**

Permasalahan dalam Tugas Akhir ini dibatasi ruang lingkup pembahasannya sebagai berikut:

1. Dataset yang digunakan adalah kumpulan citra kain yang dapat diunduh gratis dari website *www.photos-public-domain.com/tag/fabric*.
2. Implementasi dilakukan dengan menggunakan Matlab.
3. Besarnya ukuran citra 400x400 piksel dan hanya yang berekstensi .jpg (jpeg), .png, .bmp, .gif.

1.4 **Tujuan**

Tujuan dari pembuatan Tugas Akhir ini adalah membangun program temu kembali citra berbasis isi pada kain yang mirip dengan citra contoh berdasarkan kemiripan dalam salah satu fitur warna, tekstur, dan bentuk, maupun gabungan ketiganya.

1.5 **Metodologi**

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

A. Studi literatur

   Pada tahap ini dilakukan pencarian informasi dan studi literatur yang diperlukan untuk pengumpulan data dan desain sistem yang akan dibuat. Informasi didapatkan dari buku, internet, dan materi-materi lain yang berhubungan dengan *dominant color descriptor* (DCD), *steerable filter* dan *pseudo-Zernike moment* sebagai algoritma ektraksi fitur citra. Selain itu, dilakukan pencarian informasi dan materi perhitungan kemiripan antar dua citra berdasarkan hasil ekstraksi fiturnya.

B. Implementasi dan pembuatan perangkat lunak

   Pada tahap ini dilakukan implementasi proses ekstraksi fitur citra yang terdiri dari *dominant color descriptor (DCD), steerable filter dan pseudo-Zernike moment* serta perhitungan kemiripan dengan menggunakan Matlab 7.6.0.

C. Uji coba dan evaluasi
   Pada tahap ini dilakukan uji coba dengan menggunakan citra contoh untuk mencoba jalannya perangkat lunak telah sesuai dengan rancangan dan desain implementasi yang dibuat, juga untuk mencari kesalahan-kesalahan program yang mungkin terjadi untuk selanjutnya dapat dilakukan penyempurnaan.
D. Penyusunan laporan Tugas Akhir
   Pada tahap ini dilakukan penyusunan laporan yang berisi dasar teori, dokumentasi dari perangkat lunak, dan hasil-hasil yang diperoleh selama pengerjaan Tugas Akhir.

1.6 **Sistematika Penulisan**

Buku Tugas Akhir ini disusun dengan sistematika penulisan sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisi penjelasan mengenai latar belakang masalah dan tujuan dari pembuatan Tugas Akhir. Selain itu perumusan masalah, batasan masalah, metodologi, dan sistematika penulisan juga merupakan bagian dari bab ini.
2. Bab II. Tinjauan Pustaka
   Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang untuk mendukung pembuatan Tugas Akhir ini.
3. Bab III. Desain dan Perancangan
   Bab ini berisi penjelasan mengenai desain, perancangan, dan data yang digunakan untuk memenuhi Tugas Akhir, serta urutan pelaksanaan percobaan.
4. Bab IV. Implementasi
   Bab ini akan dilakukan pembuatan aplikasi yang dibangun dengan MATLAB 7.6.0 sesuai dengan permasalahan dan batasan yang telah dijabarkan pada Bab 1.
5. Bab V. Hasil Uji Coba dan Evaluasi
   Bab ini berisi penjelasan mengenai data hasil percobaan atau pengukuran, dan pembahasan mengenai hasil percobaan yang telah dilakukan.

6. Bab VI. Kesimpulan dan Saran
   Bab ini berupa hasil penelitian yang menjawab permasalahan atau yang berupa konsep, program, dan karya rancangan. Selain itu, pada bab ini diberikan saran-saran yang berisi hal-hal yang masih dapat dikerjakan dengan lebih baik dan dapat dikembangkan lebih lanjut, atau berisi masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir.

BAB II
DASAR TEORI

Bab ini berisi penjelasan teori-teori yang berkaitan dengan implementasi perangkat lunak. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap sistem yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Fitur pada Citra Digital

Citra digital dapat diartikan sebagai sebuah representasi diskrit dari informasi spasial dan intensitasnya. Informasi spasial disimpan dalam koordinat *Cartesian* sedangkan informasi intensitas bergantung pada jenis *color space* yang digunakan pada citra. Gambar 2.1 merupakan contoh dari citra digital dalam beberapa *color space*. Citra memiliki fitur-fitur yang dapat digunakan dalam pengolahan citra digital, beberapa di antaranya adalah fitur warna, tekstur, dan bentuk.

Gambar 2.1 Citra Digital dalam Beberapa *Color Space*; (a) RGB, (b) Abu-abu, (c) Hitam-putih.

Warna adalah komponen penting yang digunakan oleh mata manusia maupun komputer untuk mengenali suatu objek secara visual. Warna dalam suatu objek digital menjadi fitur yang sering digunakan dalam temu kembali citra berbasis isi. Warna yang biasa digunakan adalah warna pada *color space* RGB (red green blue). Secara teori, nilai *error* dari proses temu kembali citra berbasis isi akan minim jika warna diekstraksi langsung sesuai palet warna yang digunakan citra. Misalnya dalam citra RGB,

jumlah nilai warna yang ada adalah 256. Namun bila hal tersebut dilakukan, biaya komputasinya akan tinggi dan penyimpanan yang dibutuhkan untuk menyimpan informasi warna juga banyak. Karena itu, banyak penelitian telah dilakukan untuk mendapatkan informasi dari warna suatu citra secara efisien.

Tekstur citra merupakan salah satu fitur yang digunakan dalam beberapa pemrosesan citra. Tekstur menunjukkan sifat homogenitas dari suatu intensitas. Tekstur sebagai salah satu fitur citra telah lama dipelajari. Berbagai teknik telah dikembangkan untuk segmentasi tekstur, klasifikasi tekstur, mengambil bentuk dari tekstur, dan sintesis tekstur. Dua macam metode representasi tekstur adalah struktural dan statistikal. Metode struktural, termasuk operasi morfologi, menggambarkan tekstur dengan mengidentifikasi struktur primitif dan aturan penempatannya. Metode statistikal, salah satunya adalah *co-occurrence matrix*, menggambarkan tekstur dengan distribusi statistika dari intensitas citra.

Bentuk merupakan informasi geometri dari suatu objek yang tidak berubah meski objek tersebut dipindah, dirotasi, ataupun diubah ukurannya. Ekstraksi bentuk dapat berupa keseluruhan informasi dari *boundary* objek dan isinya atau hanya *boundary* saja. Pemilihan jenis ekstraksi ini tergantung pada bentuk apa yang ingin didapatkan [3]. Bentuk citra merupakan fitur visual yang penting dalam menggambarkan isi suatu citra. Namun, representasi dan deskripsi dari bentuk citra adalah pekerjaan yang sulit. Objek nyata 3D akan diproyeksikan dalam citra yang memiliki ruang 2D sehingga bentuk objek akan berubah tergantung sudut pengambilan citra.

Tugas Akhir ini akan menggunakan ketiga fitur citra tersebut, yaitu fitur warna, tekstur, dan bentuk. Masing-masing akan diekstraksi oleh sebuah metode sehingga akan diperoleh informasi dari sebuah citra. Pada Tugas Akhir ini, metode yang digunakan adalah *dominant color descriptor* (DCD), *steerable filter*, dan *pseudo-Zernike moments*.

2.1.1 *Dominan Color Descriptor* (DCD)

Dominant color descriptor (DCD) [4] adalah salah satu jenis deskriptor warna yang digunakan dalam temu kembali citra. DCD mendeskripsikan warna yang mewakili suatu citra berdasarkan warna dominan yang terdapat dalam suatu citra. DCD memiliki dua komponen utama yaitu warna yang mewakili citra dan persentase tiap warna tersebut di dalam citra. Namun, warna representatif yang dihasilkan terkadang tidak sesuai dengan yang diharapkan oleh manusia karena mata manusia sulit membedakan warna yang perbedaannya sangat kecil.

Untuk dapat mengambil warna dominan pada suatu citra, penyederhanaan warna perlu dilakukan. Hal ini dilakukan agar jumlah warna yang diambil tidak terlalu banyak namun tetap mewakili informasi warna keseluruhan dari citra. Ada beberapa algoritma yang bisa digunakan untuk proses ini. Salah satunya adalah *linear block algorithm* (LBA) [4].

Tahapan yang dilakukan pada LBA akan diuraikan sebagai berikut:

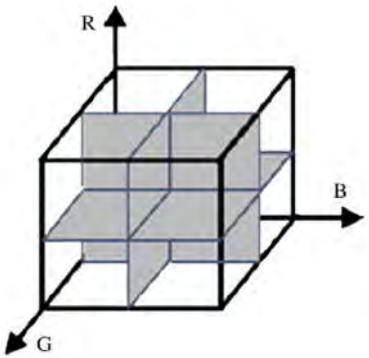
- Warna RGB dari suatu citra dibagi ke dalam beberapa partisi yang ukurannya sama, seperti pada Gambar 2.2. Warna yang terdapat dalam satu blok partisi dianggap mirip sehingga dapat direpresentasikan oleh suatu warna baru yang mewakili keseluruhan warna pada blok partisi tersebut.
- Nilai rata-rata tiap blok partisi dicari dan dijadikan sebagai warna kuantisasi atau warna yang merepresentasikan setiap warna pada blok partisi.

  $$X = (X^R, X^G, X^B) \tag{2.1}$$

  X merupakan warna dari tiap piksel yang terdiri atas komponen warna merah (R), hijau (G), dan biru (B). Warna kuantisasi dari partisi ke-i dilambangkan dengan

  $$C_i = \left(\bar{X}_i^R, \bar{X}_i^G, \bar{X}_i^B\right) (1 \leq i \leq 8) \tag{2.2}$$

dimana $\bar{X}_i$ merupakan nilai rata-rata warna pada partisi ke-*i*.



Gambar 2.2 Ilustrasi Pembagian Warna RGB dalam Delapan Partisi

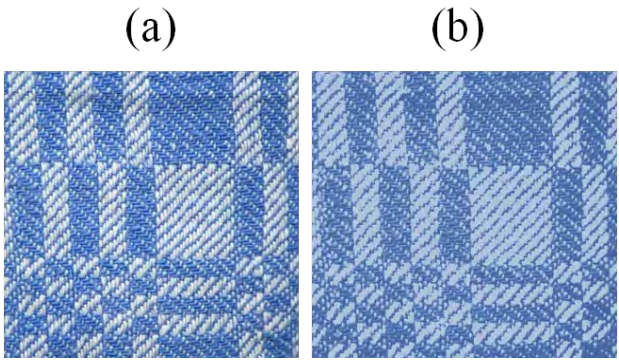
- Jarak *Euclidean* antara dua warna kuantisasi dihitung. Jika jaraknya kurang dari batas yang ditetapkan, maka dua warna kuantisasi tersebut dilebur dan ditentukan warna kuantisasi baru dari keduanya. Batas jarak antara dua warna dilambangkan dengan T_d, dan nilai yang disarankan adalah 10-15. Karena warna harus cukup signifikan, maka warna yang jaraknya dengan warna lain kurang dari T_d akan dilebur. Pada rumus berikut, P_R, P_G, dan P_B merupakan persentase dari komponen R, G, dan B suatu warna. Sedangkan X_1 dan X_2 merupakan nilai dua warna yang memiliki jarak kurang dari T_d.

$$
\begin{aligned}
X^R &= X_1^R \times \left(\frac{P_{R,1}}{P_{R,1}+P_{R,2}}\right) + X_2^R \times \left(\frac{P_{R,2}}{P_{R,1}+P_{R,2}}\right),\\
X^G &= X_1^G \times \left(\frac{P_{G,1}}{P_{G,1}+P_{G,2}}\right) + X_2^G \times \left(\frac{P_{G,2}}{P_{G,1}+P_{G,2}}\right),\\
X^B &= X_1^B \times \left(\frac{P_{B,1}}{P_{B,1}+P_{B,2}}\right) + X_2^B \times \left(\frac{P_{B,2}}{P_{B,1}+P_{B,2}}\right).
\end{aligned}
\qquad (2.3)
$$

  Langkah ini dilakukan hingga tidak ada jarak nilai dua warna yang kurang dari T_d.
- Persentase dari warna yang tersisa diperiksa. Jika ada warna yang persentasenya kurang dari batas T_m, maka warna

tersebut akan dilebur dengan tetangga terdekatnya. Pencarian tetangga terdekat dilakukan dengan menggunakan jarak *Euclidean*. Nilai T_m diset kecil, misalnya 6%. Hal ini dilakukan supaya warna yang tersisa cukup dominan.

- Warna terakhir yang tersisa adalah warna yang merepresentasikan suatu citra. Untuk satu citra, terdapat maksimal 8 warna yang memiliki 4 komponen, yaitu nilai RGB yang masing-masing memiliki rentang antara 0 hingga 255, dan persentase warna tersebut dalam citra yang telah dikuantisasi. Contohnya dapat dilihat pada Gambar 2.3. Terdapat tiga warna yang tersisa setelah warna pada citra asli dikuantisasi.



Gambar 2.3 Contoh Hasil DCD, dengan (a) Citra Asli dan (b) Citra Hasil Kuantisasi.

Tabel 2.1 Hasil Ekstraksi Fitur Warna dengan DCD

R	G	B	% warna
70	105	164	36 %
108	143	196	24 %
175	196	220	40 %

Persentase warna dihitung dari perbandingan jumlah piksel suatu warna dengan total piksel pada citra. Hasil ekstraksi fitur dari gambar tersebut dapat dilihat pada Tabel 2.1.

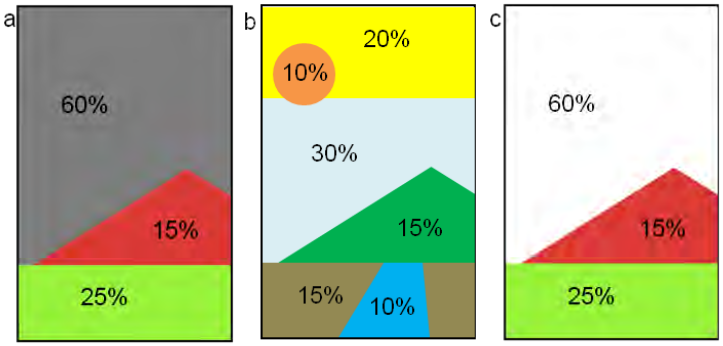
Perhitungan kemiripan dari DCD

Kemiripan dari warna-warna dominan antara dua citra dihitung berdasarkan kedekatan jarak dua warna dan persentase tiap warna tersebut. Dari dua warna dominan $F_1 = \{\{c_i, p_i\}, i = 1, \dots, N_1\}$ dan $F_2 = \{\{c_j, p_j\}, j = 1, \dots, N_2\}$, perhitungan kemiripan dilakukan dengan langkah sebagai berikut.

1. Menghitung nilai kemiripan dari dua warna dominan yang berbeda, yaitu citra dari *database* (t) dan citra contoh (q), dengan rumus berikut.

$$S_{i,j} = [1 - |p_q(i) - p_t(j)|] \times min(p_q(i), p_t(j)) \quad (2.4)$$

Dimana $p_q(i)$ adalah persentase warna dominan ke-i dari citra contoh dan $p_t(j)$ adalah persentase warna dominan ke-j dari citra dari *database*.



Gambar 2.4 Citra yang Warnanya telah Dikuantisasi Beserta Persentasenya, dimana (a) Citra Contoh Q, (b) Citra 1 dari *Database* F1, dan (c) Citra 2 dari *Database* F2.

Sebagai contoh, akan dihitung jarak antara dua citra dari Gambar 2.4 yaitu citra (a) dengan (b) dan citra (a) dengan (c). Fitur warna dari masing-masing citra dideskripsikan sebagai berikut:

$Q = \{(q_1, 0{,}6), (q_2, 0{,}25), (q_3, 0{,}15)\}$,
$F_1 = \{(t_{11}, 0{,}3), (t_{12}, 0{,}2), (t_{13}, 0{,}15)(t_{14}, 0{,}15), (t_{15}, 0{,}1)(t_{16}, 0{,}1)\}$,
$F_2 = \{(t_{21}, 0{,}6), (t_{22}, 0{,}25), (t_{23}, 0{,}15)\}$,

dimana komponen pertama dalam tanda kurung adalah warna dominan, dan komponen kedua adalah persentasenya. Nilai dari persamaan (2.4) untuk Gambar 2.4 (a) dan (b) dapat dilihat pada Tabel 2.2.

Tabel 2.2 Kemiripan Dua Warna Dominan

Warna	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}
q_1	0,210	0,120	0,083	0,083	0,050	0,050
q_2	0,238	0,190	0,135	0,135	0,085	0,085
q_3	0,128	0,143	0,150	0,150	0,095	0,095

2. Mencari koefisien kemiripan dilakukan dengan rumus

$$a_{i,j} = \begin{cases} 1 - d_{i,j}/d_{max} & d_{i,j} \le T \\ 0 & d_{i,j} > T \end{cases} \quad (2.5)$$

dimana $d_{i,j}$ adalah jarak *Euclidean* dari dua warna c_i dan b_j. T adalah jarak maksimum yang digunakan untuk menentukan apakah kedua warna dominan mirip, dan $d_{max} = \alpha T$. α merupakan koefisien yang jika nilainya tepat akan dapat membedakan dua warna terpisah jauh atau dekat. Pada Tugas Akhir ini, nilai α diset 2 dan T diset 25.
Asumsi warna pada citra contoh Q dan citra pada database F_1 pada Gambar 2.4 semuanya berbeda dan melewati batas T sehingga koefisien $a_{i,j}$ pada persamaan (2.5) bernilai nol. Sedangkan warna pada citra contoh F_2 memiliki dua warna yang sama dengan Q, yaitu $q_2 = t_{22}$ dan $q_3 = t_{23}$.
3. Mengukur kemiripan dua citra dengan rumus

$$SIM(F_1, F_2) = \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} a_{i,j}\, S_{i,j} \quad (2.6)$$

Nilai kemiripan dari Gambar 2.4 adalah sebagai berikut:

$$SIM(Q, F_1) = 0$$

$$SIM(Q, F_2) = \{(1 \times [1 - |0{,}25 - 0{,}25|] \times 0{,}25) + (1 \times [1 - |0{,}15 - 0{,}15|] \times 0{,}15)\} = 0{,}4$$

4. Jarak antara dua citra F_1 dan F_2 didapat dengan rumus

$$D^2(F_1, F_2) = 1 - SIM(F_1, F_2) \quad (2.7)$$

Semakin besar nilai D^2, artinya kedua citra semakin tidak mirip. Bila nilai D^2 adalah 1, maka berarti dua citra tersebut tidak mirip sama sekali.
Perhitungan jarak dari contoh pada Gambar 2.4 dengan menggunakan persamaan (2.7) akan menghasilkan

$$D^2(Q, F_1) = 1 - 0 = 1$$
$$D^2(Q, F_2) = 1 - 0{,}4 = 0{,}6$$

dimana hasilnya menunjukkan jarak atau perbedaan citra (a) dengan citra (b) lebih besar dibanding dengan jaraknya dengan citra (c). Hal ini sesuai pula dengan pengelihatan mata manusia yang menilai bahwa citra (a) lebih mirip dengan citra (c) dari segi komposisi warnanya.

2.1.2 *Steerable Filter*

Representasi fitur tekstur yang invarian terhadap rotasi dan skala dapat diperoleh dengan menggunakan dekomposisi dari *steerable filter* [5]. *Steerable filter* adalah suatu kelas filter yang menyaring citra secara sembarang putaran. *Steerable filter* termasuk salah satu jenis *oriented filter*. *Oriented filter* banyak digunakan dalam proses pengolahan citra, seperti analisis tekstur, deteksi tepi, kompresi data citra, analisis pergerakan, dan perbaikan citra.

Langkah yang dilakukan untuk mengekstraksi fitur tekstur dengan filter ini adalah sebagai berikut:

1. Memilih *basis filter* yang akan digunakan. Salah satu *basis filter* yang bisa digunakan adalah turunan pertama dari fungsi *Gaussian*. Fungsi *Gaussian* 2 dimensi yang digunakan untuk

pemrosesan citra jika ditulis dalam koordinat Cartesian x dan y adalah sebagai berikut

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{2.8}$$

dengan nilai σ menunjukkan seberapa lebar filter yang akan digunakan. Pada koordinat *x* dan *y*, turunan pertama fungsi *Gaussian* dapat didekati pada sumbu *x* yaitu $G_1^{0'}$ dan pada sumbu *y* yaitu $G_1^{90'}$.

$$G_1^{0^o} = -\frac{x}{\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \frac{1}{\sigma\sqrt{2\pi}} \tag{2.9}$$

$$G_1^{90^o} = -\frac{y}{\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \frac{1}{\sigma\sqrt{2\pi}} \tag{2.10}$$

Matriks yang berisi nilai dari fungsi pada persamaan (2.9) dan (2.10) memiliki ukuran yang bergantung pada besar σ. Untuk σ bernilai 3, ukuran matriks G_1^0 dan G_1^{90} adalah 7x7. Nilai matriks tersebut ditunjukkan oleh (2.11) dan (2.12).

$$G_1^{0^o} = \begin{bmatrix} 0{,}016 & 0{,}014 & 0{,}009 & 0{,}000 & -0{,}009 & -0{,}014 & -0{,}016 \\ 0{,}022 & 0{,}019 & 0{,}011 & 0{,}000 & -0{,}011 & -0{,}019 & -0{,}022 \\ 0{,}025 & 0{,}022 & 0{,}013 & 0{,}000 & -0{,}013 & -0{,}022 & -0{,}025 \\ 0{,}027 & 0{,}024 & 0{,}014 & 0{,}000 & -0{,}014 & -0{,}024 & -0{,}027 \\ 0{,}025 & 0{,}022 & 0{,}013 & 0{,}000 & -0{,}013 & -0{,}022 & -0{,}025 \\ 0{,}022 & 0{,}019 & 0{,}011 & 0{,}000 & -0{,}011 & -0{,}019 & -0{,}022 \\ 0{,}016 & 0{,}014 & 0{,}009 & 0{,}000 & -0{,}009 & -0{,}014 & -0{,}016 \end{bmatrix} \tag{2.11}$$

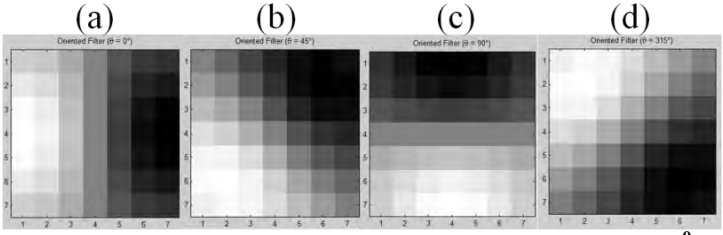
$$G_1^{90^o} = \begin{bmatrix} 0{,}016 & 0{,}022 & 0{,}025 & 0{,}027 & 0{,}025 & 0{,}022 & 0{,}016 \\ 0{,}014 & 0{,}019 & 0{,}022 & 0{,}024 & 0{,}022 & 0{,}019 & 0{,}014 \\ 0{,}009 & 0{,}011 & 0{,}013 & 0{,}014 & 0{,}013 & 0{,}011 & 0{,}009 \\ 0{,}000 & 0{,}000 & 0{,}000 & 0{,}000 & 0{,}000 & 0{,}000 & 0{,}000 \\ -0{,}009 & -0{,}011 & -0{,}013 & -0{,}014 & -0{,}013 & -0{,}011 & -0{,}009 \\ -0{,}014 & -0{,}019 & -0{,}022 & -0{,}024 & -0{,}022 & -0{,}019 & -0{,}014 \\ -0{,}016 & -0{,}022 & -0{,}025 & -0{,}027 & -0{,}025 & -0{,}022 & -0{,}016 \end{bmatrix} \tag{2.12}$$

Dalam memilih *basis filter* yang akan digunakan, perlu diperhatikan apakah fungsi tersebut dapat ditulis dalam bentuk polinomial x dan polinomial y. Tidak semua fungsi dapat digunakan sebagai *basis filter* dalam *steerable filter*. Apabila suatu fungsi telah memenuhi syarat, berarti fungsi tersebut bersifat *separable* atau dapat diuraikan.

2. Menentukan fungsi interpolar untuk mengatur orientasi dari *basis filter* yang dipilih. Untuk *basis filter* turunan pertama fungsi *Gaussian*, fungsi interpolarnya adalah seperti pada persamaan (2.11).

$$G_1^{\theta} = \cos\theta\, G_1^{0^o} + \sin\theta\, G_1^{90^o} \quad (2.13)$$

3. Menentukan sudut yang ingin digunakan pada *basis filter* untuk menyaring citra. Karena pola yang dihasilkan akan berulang setelah sudut 180^o, maka batas sudut yang dipilih adalah antara 0^o hingga 180^o. Pada penelitian yang dilakukan oleh X. -Y. Wang et al [1], sudut yang digunakan adalah kelipatan 45^o yaitu 0^0, 45^0, 90^0, dan -45^o. Sudut -45^o atau 315^o akan memiliki nilai yang sama dengan nilai pada sudut 135^o. Masing-masing sudut akan menghasilkan citra terfilter dengan sumbu filter mengikuti sudut yang digunakan. Visualisasi dari fungsi turunan pertama *Gaussian* dengan masing-masing sudut tersebut dapat dilihat pada Gambar 2.6.



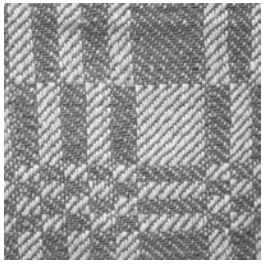
Gambar 2.5 Turunan Pertama Fungsi *Gaussian* G_1^{θ} dengan Sudut (a) 0^o, (b) 45^o, (c) 90^o, dan (d) 315^o.

4. Hasil konvolusi pada citra I oleh *basis filter* berupa turunan pertama fungsi *Gaussian* didapat dari rumus berikut:

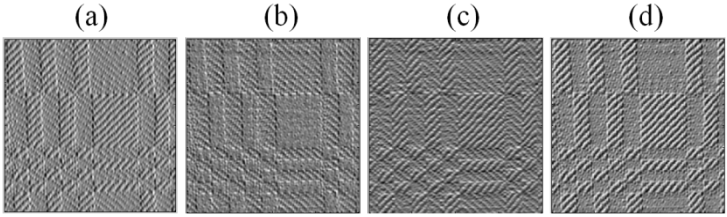
$$S = G_1^{\theta} * I = \cos\theta\,(G_1^{0^o} * I) + \sin\theta\,(G_1^{90^o} * I) \quad (2.14)$$

$$I * h[x,y] = \sum_{k,l} h[x-k, y-l] I[k,l] \quad (2.15)$$

Dengan persamaan (2.12), citra asli seperti pada Gambar 2.7 akan terfilter sesuai dengan sudut yang digunakan. Rumus konvolusi dari citra I oleh kernel h dituliskan dalam persamaan (2.13). Gambar 2.8 menunjukkan citra yang telah terfilter dengan sudut tertentu.



Gambar 2.6 Citra Asli



Gambar 2.7 Citra Terfilter dengan Sudut (a) 0^o, (b) 45^o, (c) 90^o, dan (d) -45^o atau 315^o.

5. Untuk setiap sudut *i* yang digunakan, dilakukan proses perhitungan energi *E* dari citra. dimana S_i merupakan hasil konvolusi pada sudut ke-*i*.

$$E_i = \sum_x \sum_y |S_i(x, y)| \tag{2.16}$$

6. Fitur tekstur dari citra yang berukuran MxN dapat direpresentasikan dengan nilai statistika, misalnya nilai rata-rata (μ) dan standar deviasi (σ).

$$\mu_i = \frac{1}{MN} E_i(x, y) \tag{2.17}$$

$$\sigma_i = \sqrt{\frac{1}{MN} \sum_x \sum_y (S_i(x, y) - \mu_i)^2} \tag{2.18}$$

7. Jumlah elemen vektor fitur yang dihasilkan adalah dua kali jumlah sudut yang digunakan. Jika sudut yang dikenakan pada *basis filter* ada empat, maka fitur tekstur yang dihasilkan dapat didefinisikan sebagai vektor fitur berikut:

$$F_T = (\mu_1, \sigma_1, \mu_2, \sigma_2, \mu_3, \sigma_3, \mu_4, \sigma_4) \quad (2.19)$$

dimana μ_1 dan σ_1 merupakan hasil dari sudut pertama, misalnya sudut 45^o, μ_2 dan σ_2 merupakan hasil dari sudut kedua, dan seterusnya.

2.1.3 *Pseudo-Zernike Moment*

Zernike moment adalah salah satu teori tentang momentum yang digunakan untuk merepresentasikan bentuk objek dari citra [6]. Momentum dan fungsi momentum dipakai untuk merepresentasikan fitur pola karena tidak perlu *closed boundary* dan dapat merekam informasi global tentang citra. Momentum dapat mengambarkan suatu objek dalam hal area, posisi, orientasi, dan parameter terdefinisi lainnya. Momentum dan fungsi momentum biasa digunakan untuk merepresentasikan bentuk objek berdasarkan region, yang berarti baik pinggiran maupun interior dari objek diperlakukan sebagai satu objek. Representasi berdasarkan region menutupi kekurangan representasi berdasarkan kontur yaitu mengabaikan informasi yang mungkin penting dalam interior suatu objek.

Langkah yang dilakukan untuk mengekstraksi fitur bentuk dengan *pseudo-Zernike moment* adalah sebagai berikut:

1. Melakukan perhitungan untuk mendapatkan suatu set nilai polinomial *Zernike*. Pada sepasang koordinat *x* dan *y* sehingga $x^2+y^2 \leq 1$, digunakan rumus berikut:

   $$V_{nm}(x, y) = V_{nm}(\rho, \theta) = R_{nm}(\rho)\exp(jm\theta) \quad (2.20)$$

   dimana $\rho = \sqrt{x^2 + y^2}$ menunjukkan panjang vektor dari titik pusat ke koordinat (x, y), $\theta = \tan^{-1}\frac{y}{x}$ menunjukkan besar sudut antara vektor ρ dan sumbu x dengan arah *counterclockwise*, *j* merupakan bilangan imajiner yaitu bernilai $\sqrt{-1}$, *n* dan *m* adalah bilangan positif, dan *m* bernilai antara -*n* hingga *n*. Banyaknya nilai ρ dan θ mengikuti ukuran citra yang akan direpresentasikan. Untuk citra berukuran 9x9 piksel, nilai ρ dan θ awalnya berjumlah 81. Namun nilai ρ dan θ yang

digunakan hanya yang berada dalam lingkaran, yaitu yang nilai ρ-nya kurang dari 1. Polinomial radial *Zernike* atau *R* dicari dengan menggunakan rumus berikut:

$$R_{nm}(\rho) = \sum_{s=0}^{n-|m|} \frac{(-1)^s (2n+1-s)! \rho^{n-s}}{s!(n-|m|+1-s)!(n-|m|-s)!} \quad (2.21)$$

Jumlah nilai yang dihasilkan oleh persamaan (2.19) mengikuti jumlah nilai dari ρ. Misalkan citra yang akan diproses memiliki ukuran 9x9 piksel. Nilai ρ dapat dilihat pada (2.22).

1,41	1,25	1,12	1,03	1,00	1,03	1,12	1,25	1,41
1,25	1,06	0,90	0,79	0,75	0,79	0,90	1,06	1,25
1,12	0,90	0,71	0,56	0,50	0,56	0,71	0,90	1,12
1,03	0,79	0,56	0,35	0,25	0,35	0,56	0,79	1,03
1,00	0,75	0,50	0,25	0,00	0,25	0,50	0,75	1,00
1,03	0,79	0,56	0,35	0,25	0,35	0,56	0,79	1,03
1,12	0,90	0,71	0,56	0,50	0,56	0,71	0,90	1,12
1,25	1,06	0,90	0,79	0,75	0,79	0,90	1,06	1,25
1,41	1,25	1,12	1,03	1,00	1,03	1,12	1,25	1,41

(2.22)

−2,36	−2,21	−2,03	−1,82	−1,57	−1,33	−1,11	−0,93	−0,79
−2,50	−2,36	−2,16	−1,89	−1,57	−1,25	−0,98	−0,79	−0,64
−2,68	−2,55	−2,36	−2,03	−1,57	−1,11	−0,79	−0,59	−0,46
−2,90	−2,82	−2,68	−2,36	−1,57	−0,79	−0,46	−0,32	−0,25
3,14	3,14	3,14	3,14	0,00	0,00	0,00	0,00	0,00
2,90	2,82	2,68	2,36	1,57	0,79	0,46	0,32	0,25
2,68	2,55	2,36	2,03	1,57	1,11	0,79	0,59	0,46
2,50	2,36	2,16	1,89	1,57	1,25	0,98	0,79	0,64
2,36	2,21	2,03	1,82	1,57	1,33	1,11	0,93	0,79

(2.23)

Nilai yang diberi warna adalah nilai yang akan digunakan dalam persamaan (2.18), sehingga jumlah ρ yang akan digunakan adalah 49. Nilai θ yang digunakan mengikuti koordinat nilai ρ yang digunakan. Nilai θ bila citra yang akan diproses memiliki ukuran 9x9 piksel dapat dilihat pada (2.23).

2. Melakukan proses rekonstruksi citra dengan orde tertentu. Semakin besar nilai orde, maka semakin mirip citra dengan aslinya, namun akan semakin besar juga biaya komputasinya. Selain itu, semakin besar nilai orde, tingkat sensitifitas terhadap *noise* juga semakin besar. Gambar 2.8 menunjukkan semakin besar orde, citra rekonstruksi semakin mirip dengan aslinya.

(a) (b) (c) (d) (e) (f)

Gambar 2.8 Citra Segitiga dan Citra Hasil Rekonstruksinya. (a) Citra hitam putih; (b) Citra rekonstruksi orde 1; (c) Citra rekonstruksi orde 5; (d) Citra rekonstruksi orde 10; (e) Citra rekonstruksi orde 15; (f) Citra rekonstruksi orde 20.

Dengan orde sebesar *n* dan perulangan sebanyak *m*, maka hasil rekonstruksi dari fungsi citra *f* dapat dicari dengan menggunakan rumus berikut:

$$A_{nm} = \frac{n+1}{\pi} \sum_x \sum_y V_{nm}(x,y) f(x,y) \qquad (2.24)$$

Nilai *n* dan *m* yang digunakan pada persamaan (2.24) adalah sama dengan yang digunakan pada persamaan (2.20). Setiap pasangan nilai *n* dan *m* akan memiliki satu nilai A. Fungsi citra *f* yang digunakan adalah citra yang telah berwarna hitam-putih.

3. Hasil akhir dari ekstraksi fitur dengan *Zernike moment* ini adalah sebuah vektor fitur A dengan banyak elemen sesuai banyak orde dan perulangan yang dipilih.

2.2 Sistem Temu Kembali Citra Berbasis Isi

Sistem temu kembali informasi adalah sistem yang berfungsi menemukan informasi yang relevan dengan kebutuhan pengguna. Sistem temu kembali informasi pada data citra biasa disebut *image retrieval*. Ada dua cara yang dapat dilakukan dalam temu kembali suatu citra atau *image retrieval*, yaitu:

a. *Context-based* adalah pengambilan data dengan merujuk pada kandungan semantik berkaitan dengan citra, biasanya berhubungan dengan deskripsi citra.
b. *Content-based* adalah pengambilan data dengan merujuk pada fitur citra seperti warna, tekstur, bentuk. Jenis pencarian ini

biasa desebut dengan *content based image retrieval* (CBIR). [7]

Secara umum, sistem CBIR memiliki 4 proses utama. Proses pertama adalah ekstraksi fitur dari semua *database* citra. Proses kedua adalah ekstraksi fitur dari citra contoh. Proses ketiga adalah membandingkan kemiripan antara fitur dari citra contoh dengan fitur dari *database* citra. Proses keempat adalah mengembalikan citra yang mirip dengan citra contoh sesuai dengan parameter-parameter yang ditentukan oleh pengguna. Dalam menentukan kemiripan, nilai yang biasanya dicari adalah jarak suatu fitur dari dua buah citra. Pada Tugas Akhir ini, perhitungan jarak yang digunakan adalah *Euclidean distance*.

2.2.1 Perhitungan Jarak dengan *Euclidean Distance*

Perhitungan jarak antara dua titik dapat dilakukan dengan berbagai macam rumus, salah satunya adalah dengan rumus *Euclidean distance*. *Euclidean distance* menerapkan teorema *Pythagoras* dalam menghitung jarak antar titik. Rumus umum *Euclidean distance* untuk dua vektor, *x* dan *y*, yang berdimensi *J* dapat dilihat pada persamaan (2.21).

$$d_{x,y} = \sqrt{\sum_{j=1}^{J}(x_j - y_j)^2} \tag{2.25}$$

Pada Tugas Akhir ini, perhitungan kemiripan fitur tekstur dilakukan dengan mencari jarak *Euclidean* antara fitur tekstur citra contoh dan citra dari *database*. Sebanyak *n* sudut yang digunakan dalam ekstraksi fitur tekstur, selisih kuadrat dari nilai rata-rata citra contoh (μ^Q) dan citra dalam *database* (μ^I) ditambahkan dengan selisih kuadrat dari nilai standar deviasi citra contoh (σ^Q) dan citra dalam *database* (σ^I). Nilai *n* menunjukkan banyak sudut yang digunakan dalam proses ekstraksi fitur tekstur. Hasilnya kemudian diakarkuadratkan untuk mendapat nilai kemiripan antara dua citra yaitu S_T.

$$S_T(Q,I) = \sqrt{\sum_{i=1}^{n}\left[\left(\mu_i^Q - \mu_i^I\right)^2 + \left(\sigma_i^Q - \sigma_i^I\right)^2\right]} \quad (2.26)$$

Perhitungan kemiripan fitur bentuk juga menggunakan jarak *Euclidean*. Selisih nilai vektor fitur bentuk *A* dari citra contoh (*Q*) dan citra dalam *database* (*I*) pada indeks yang bersesuaian akan dikuadratkan. Nilai *n* menunjukkan banyak fitur yang didapat pada proses ekstraksi fitur bentuk. Setelah dijumlahkan semuanya lalu diakar, skor kemiripan bentuk citra S_B didapatkan.

$$S_B(Q,I) = \sqrt{\sum_{i=0}^{n}\sum_{j}^{i}\left(\left|A_{ij}^Q\right| - \left|A_{ij}^I\right|\right)^2} \quad (2.27)$$

Skor total kemiripan *database* citra dapat dihitung dengan menjumlahkan skor tiap fitur yang dipilih. Tiap fitur akan memiliki bobot tersendiri yang menunjukkan seberapa besar pengaruh fitur tersebut terhadap skor total.

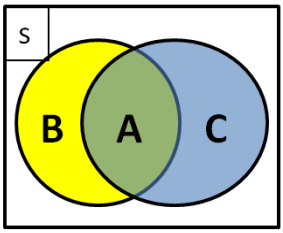
2.2.2 Perhitungan Performa Pencarian

Precision dan *recall* adalah fungsi yang biasa digunakan dalam evaluasi temu kembali informasi. *Precision* dapat dilihat sebagai ukuran ketepatan, sedangkan recall dapat dilihat sebagai ukuran kelengkapan. *Precision* dan recall dapat dihitung menggunakan persamaan (2.28) dan (2.29).

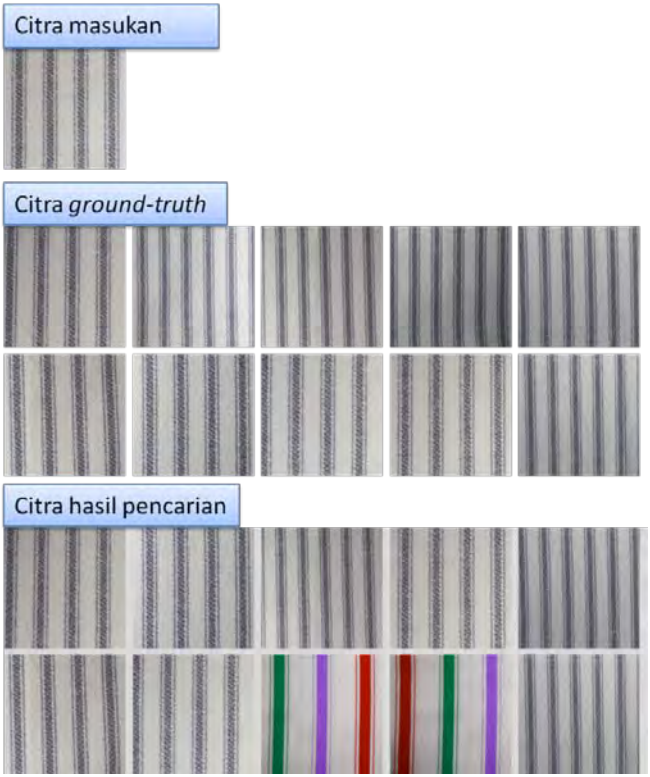
$$precision = \frac{correct}{correct + false\ positive} \quad (2.28)$$

$$recall = \frac{correct}{corect + false\ negative} \quad (2.29)$$

False positive merupakan data yang tidak sesuai dengan *query* namun ikut terambil, sedangkan *false negative* merupakan data yang sesuai namun tidak ikut terambil. Gambar 2.9 merupakan penggambaran sederhana dari *precision* dan *recall* dimana lingkaran kuning mewakili data yang sesuai dengan *query* dan lingkaran biru mewakili data yang berhasil terambil dari seluruh data pada himpunan S. Kedua ukuran ini biasanya disajikan dalam bentuk persentase, yaitu antara 0 hingga 100.



Gambar 2.9 Ilustrasi *Precision* dan *Recall*



Gambar 2.10 Contoh CBIR Berdasarkan Fitur Warna

Pada Gambar 2.10 terdapat citra yang seharusnya tampil pada hasil pencarian namun ternyata tidak tampil, citra ini termasuk *false negative*. Citra yang tidak seharusnya tampil pada hasil

pencarian karena kurang mirip dengan citra contoh namun ternyata tampil disebut *false positive*. Citra yang mirip dengan citra contoh dan ikut tampil dalam hasil pencarian disebut *correct*. Perhitungan performa pencarian dari contoh Gambar 2.10 adalah sebagai berikut:

$$precision = \frac{8}{8+2} * 100\% = 80\%$$

$$recall = \frac{8}{8+2} * 100\% = 80\%$$

BAB III
DESAIN DAN PERANCANGAN

Pada bab ini akan diuraikan mengenai desain dan perancangan sistem perangkat lunak agar dapat mencapai tujuan dari Tugas Akhir ini. Perangkat lunak yang dibuat pada Tugas Akhir ini berguna untuk melakukan temu kembali citra kain. Perangkat lunak ini menggunakan kombinasi fitur warna, tekstur, dan bentuk. Perangkat lunak ini memiliki dua bagian utama yaitu proses ekstraksi fitur dan proses perhitungan kemiripan. Proses perancangan aplikasi meliputi perancangan data, perancangan proses, perancangan tabel, arsitektur sistem dan diagram alir dari masing-masing proses.

3.1 Lingkungan Perancangan Perangkat Lunak

Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam perancangan perangkat lunak untuk tugas akhir ini ditampilkan pada Tabel 3.1.

Tabel 3.1 Lingkungan Perancangan Perangkat Lunak

Perangkat	Spesifikasi
Perangkat Keras	Prosesor : Intel(R) Core(TM)i3 CPU M350 @2.27 GHz Memori : 2 GB
Perangkat Lunak	Sistem Operasi : Windows 7 Ultimate 32-bit Perangkat Pengembang : Ms. Visio 2013 Ms. Word 2007 Matlab 7.6.0

3.2 Perancangan Data

Pada subbab ini akan dijelaskan mengenai perancangan data yang dibutuhkan untuk membangun aplikasi Temu Kembali Citra Berbasis Isi pada Citra Kain Berdasarkan Fitur Warna, Tekstur,

dan Bentuk. Perancangan data meliputi data masukan, data proses berupa data-data yang dibutuhkan dan dihasilkan selama menjalankan proses eksekusi perangkat lunak, serta data keluaran yang berupa hasil kemiripan *database* citra dengan citra contoh.

3.2.1 Data Masukan

Data masukan adalah data awal yang akan diproses. Terdapat dua jenis data masukan. Data pertama berupa 280 citra berukuran 400x400 piksel dimana setiap 10 citra saling mirip karena diambil dari kain yang sama. Data kedua berupa 172 citra berukuran seperti data pertama, namun setiap citra berbeda dari yang lain. *Dataset* citra diunduh secara gratis dari website *www.photos-public-domain.com/tag/fabric* dan telah dilakukan *preprocessing* berupa penyamaan ukuran.

3.2.2 Data Proses

Data proses adalah data-data yang digunakan dalam proses ekstraksi fitur dan proses perhitungan kemiripan. Dalam sistem ini, terdapat data proses yang berdasarkan subproses yang ada. Pembagian subproses yang ada dapat dilihat pada Tabel 3.2.

Pada subproses *dcd.m* variabel yang digunakan tertera dalam Tabel 3.3. Data masukan berupa citra yang masuk ke dalam proses ini akan diolah hingga didapat fitur warna darinya. Keluaran dari proses ini akan menjadi masukan pada subproses *color_sim.m*.

Tabel 3.2 Data Sub Proses

No.	Nama Data	Keterangan
1.	*color_sim.m*	Kelas untuk menghitung kemiripan fitur warna.
2.	*dcd.m*	Kelas untuk mendapatkan hasil ekstraksi fitur warna.
3.	*pzernfun.m*	Kelas untuk mendapatkan

		hasil ekstraksi fitur bentuk.
4.	*shape_sim.m*	Kelas untuk menghitung kemiripan fitur bentuk.
5.	*steerGaussFilterOrder 1.m*	Kelas untuk mendapatkan hasil ekstraksi fitur tekstur.
6.	*texture_sim.m*	Kelas untuk menghitung kemiripan fitur tekstur.

Tabel 3.3 Data Proses Ekstraksi Fitur Warna

No.	Nama Data	Tipe Data	Keterangan
1.	I	uint8	Citra contoh sebelum diproses.
2.	tempArr	double	*Array* untuk menyimpan hasil ekstraksi fitur per citra.
3.	outArr	double	*Array* yang menyimpan hasil ekstraksi fitur *database* citra.
4.	inputImg	double	*Array* yang menyimpan warna dari citra.
5.	Td	int	Batas minimum jarak antar warna.
6.	Tm	int	Batas minimum persentase warna.
7.	sumGrup	double	*Array* yang menyimpan data kuantisasi warna.
8.	dist	double	Hasil perhitungan jarak *Euclidean* antar dua titik.
9.	distSort	double	Hasil pengurutan dari *dist* secara *ascending*.
10.	minIndex	int	Indeks warna yang terdekat.

Tabel 3.4 Data Proses Ekstraksi Fitur Tekstur

No.	Nama Data	Tipe Data	Keterangan
1.	sigma	int	Variabel penentu besarnya filter.
2.	G1_0	double	Nilai filter turunan pertama *Gaussian* pada sumbu x.

3.	G1_90	double	Nilai filter turunan pertama *Gaussian* pada sumbu y.
4.	Ix	double	Citra yang telah dikonvolusi oleh filter G1_0.
5.	Iy	double	Citra yang telah dikonvolusi oleh filter G1_90.
6.	S	double	Citra yang telah terfilter oleh steerable filter.
7.	E	double	Nilai energi dari citra yang terfilter.
8.	meanT	double	Nilai rata-rata dari citra terfilter.
9.	sdtT	double	Nilai standar deviasi dari citra terfilter.

Tabel 3.4 berisikan variabel yang digunakan selama proses ekstraksi fitur tekstur yaitu subproses *steerGaussFilterOrder1.m*. Data masukan pada proses ini adalah citra dan hasilnya adalah *array* yang berisi fitur tekstur dari citra. Masukan lainnya pada proses ekstraksi fitur tekstur adalah beberapa sudut. Keluaran proses ini akan menjadi masukan pada subproses *texture_sim.m*.

Tabel 3.5 Data Proses Ekstraksi Fitur Bentuk

No.	Nama Data	Tipe Data	Keterangan
1.	bw	logical	Citra hitam putih.
2.	theta	double	Besar sudut antara suatu titik dalam citra dan sumbu x.
3.	r	double	Jarak antara suatu titik dalam citra ke titik pusat citra.
4.	imgSize	int	Ukuran dari citra hitam putih.
5.	is_in_circle	logical	Array biner yang menunjukkan apakah suatu titik dalam citra berada dalam jangkauan lingkaran.
6.	F	logical	Citra hitam putih yang akan dihitung oleh *moment*.
7.	P	double	Nilai fungsi *Zernike*.
8.	A	double	Nilai estimasi dari *Zernike moment*.

Ekstraksi fitur yang ketiga adalah fitur bentuk yang dilakukan oleh subproses *pzernfun.m*. Data masukan pada proses ini adalah citra dan hasilnya berupa *array* fitur bentuk. Data masukan lainnya adalah nilai orde. Variabel yang digunakan selama proses ini dapat dilihat pada Tabel 3.5.

Tabel 3.6 Data Proses Pencarian Citra Mirip

No.	Nama Data	Tipe Data	Keterangan
1.	dbColor	double	*Array* yang menyimpan fitur warna dari *database* citra.
2.	dbImg	double	*Array* yang menyimpan nilai kemiripan dari *database* citra terhadap citra contoh.
3.	inColor	double	*Array* yang berisi fitur warna dari citra contoh.
4.	d	double	Jarak *Euclidean* antara dua warna dari citra yang berbeda.
5.	s	double	Nilai kemiripan dua warna dari citra yang berbeda.
6.	sim	double	Nilai kemiripan antara dua citra.
7.	distance	double	Jarak perbedaan antara dua citra.
8.	dbTexture	double	*Array* yang menyimpan fitur tekstur dari *database* citra.
9.	inTexture	double	*Array* yang berisi fitur tekstur dari citra contoh.
10.	dbShape	double	*Array* yang menyimpan fitur bentuk dari *database* citra.
11.	inShape	double	*Array* yang berisi fitur bentuk dari citra contoh.

Setelah data fitur citra didapatkan, proses pencarian citra mirip dapat dilakukan. Proses ini dilakukan untuk tiap fitur yang dipilih. Misalnya pengguna hanya ingin mencari citra mirip berdasarkan

warnanya saja, maka hanya akan dijalankan proses perhitungan kemiripan berdasarkan warna. Proses ini terdapat pada *color_sim.m*, *texture_sim.m*, dan *shape_sim.m*. Variabel yang digunakan dalam proses ini terdapat pada Tabel 3.6. Keluaran dari proses ini adalah beberapa citra mirip dan nilai kemiripannya terhadap citra contoh.

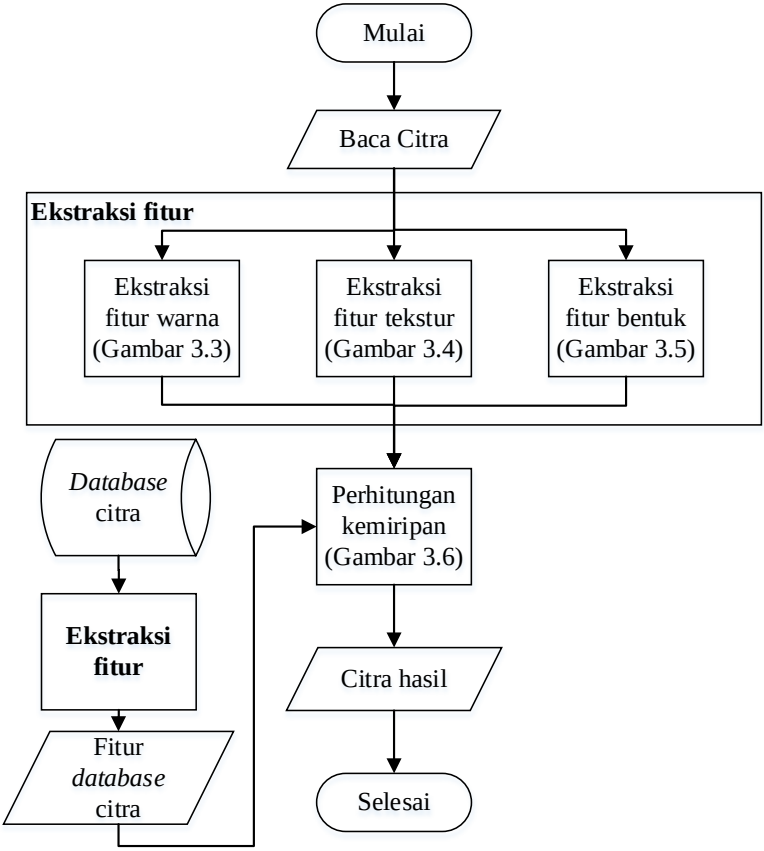
3.2.3 Data Keluaran

Data keluaran adalah data yang dihasilkan oleh proses yang berjalan pada aplikasi. Pada aplikasi ini, data yang dihasilkan adalah rekomendasi citra yang mirip dengan citra contoh. Tampilan keluaran aplikasi tergambar pada Gambar 3.1.

Gambar 3.1 Tampilan Keluaran Aplikasi

3.3 Perancangan Proses

Perancangan proses dilakukan untuk mengetahui alur penerapan algoritma yang nantinya akan dipakai dalam tahap implementasi. Alur tersebut akan ditampilkan dalam diagram alir dari masing-masing proses. Secara garis besar, sistem temu kembali citra berbasis isi terdiri atas tiga proses, yaitu ekstraksi fitur, perhitungan kemiripan, dan pengembalian citra mirip. Diagram alir sistem secara umum dapat dilihat pada Gambar 3.2.



Gambar 3.2 Diagram alir sistem secara umum.

3.3.1 Tahap Ekstraksi Fitur

Terdapat tiga fitur yang akan diekstraksi yaitu warna, tekstur, dan bentuk. Masing-masing fitur diekstraksi oleh metode tersendiri. Dalam Tugas Akhir ini akan digunakan metode *dominan color descriptor* untuk mengekstraksi fitur warna, *steerable filter* untuk mengekstraksi fitur tekstur, dan *pseudo-Zernike moment* untuk mengekstraksi fitur bentuk.

3.3.1.1 *Dominan Color Descriptor (DCD)*

DCD adalah salah satu metode untuk mengekstraksi warna berdasarkan warna dominan yang terdapat dalam suatu citra. Sebelum itu, warna akan disederhanakan dengan membaginya ke dalam beberapa grup partisi. Dalam Tugas Akhir ini, warna dikelompokkan ke dalam delapan grup sesuai nilai RGB. Jumlah grup akan berkurang jika ada dua warna yang mirip, dilihat dari jarak kedua warna tersebut. Jika sudah tidak ada warna yang mirip, warna yang hanya memiliki sedikit anggota piksel akan digabung ke warna terdekatnya. Diagram alir metode DCD dapat dilihat pada Gambar 3.3.

3.3.1.2 *Steerable Filter*

Steerable filter adalah salah satu metode untuk mengekstraksi tekstur berdasarkan sudut-sudut tertentu. Sebelum itu, akan dibuat sebuah filter dengan menggunakan fungsi yang dapat dipisahkan (memiliki sifat *separable*). Dalam Tugas Akhir ini, fungsi yang digunakan adalah turunan pertama fungsi *Gaussian*. Diagram alir metode ini dapat dilihat pada Gambar 3.4.

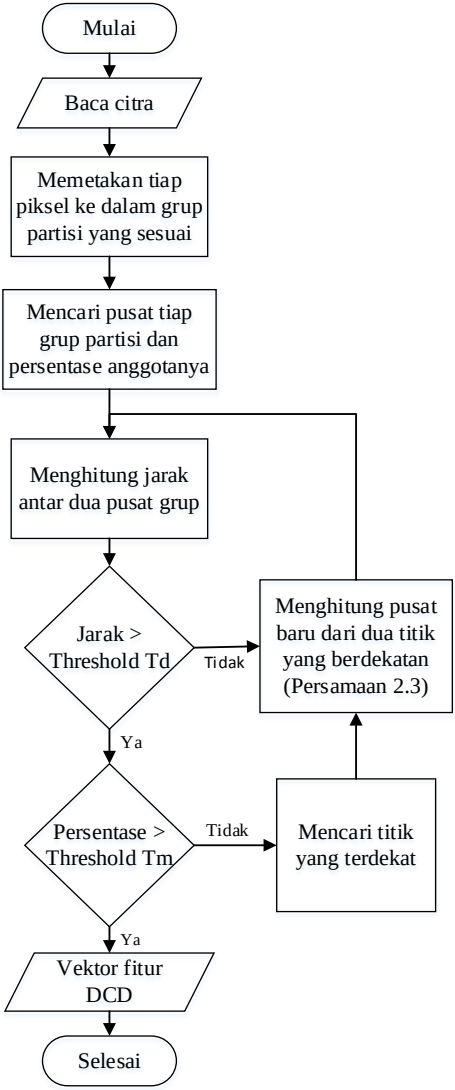
3.3.1.3 *Pseudo-Zernike Moment*

Proses ekstraksi fitur bentuk dalam Tugas Akhir ini akan menggunakan salah satu jenis momentum, yaitu *Zernike moment*. Untuk memperjelas bentuk objek dalam citra, maka citra berwarna terlebih dahulu diubah menjadi citra hitam putih. Setelah itu, polinomial *Zernike* akan dihitung. Fitur bentuk yang dihasilkan adalah nilai dari *Zernike moment* pada orde tertentu. Diagram alir metode ini dapat dilihat pada Gambar 3.5.

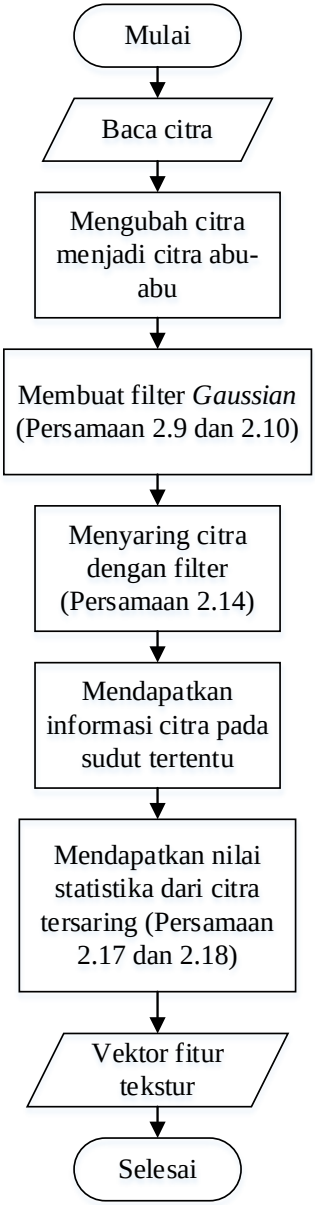
3.3.2 Tahap Pencarian Citra Mirip

Pada tahap pencarian citra mirip, masing-masing fitur akan dihitung kemiripannya. Setiap fitur juga dapat ditentukan

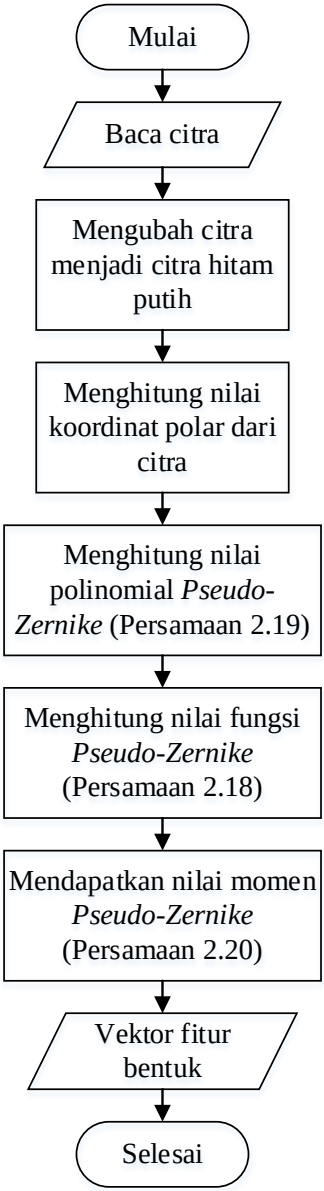
bobotnya. Semakin besar bobotnya, maka semakin berpengaruh nilai fitur tersebut terhadap hasil akhir. Diagram alir metode ini dapat dilihat pada Gambar 3.6.



Gambar 3.3 Diagram Alir DCD



Gambar 3.4 Diagram Alir dari *Steerable Filter*



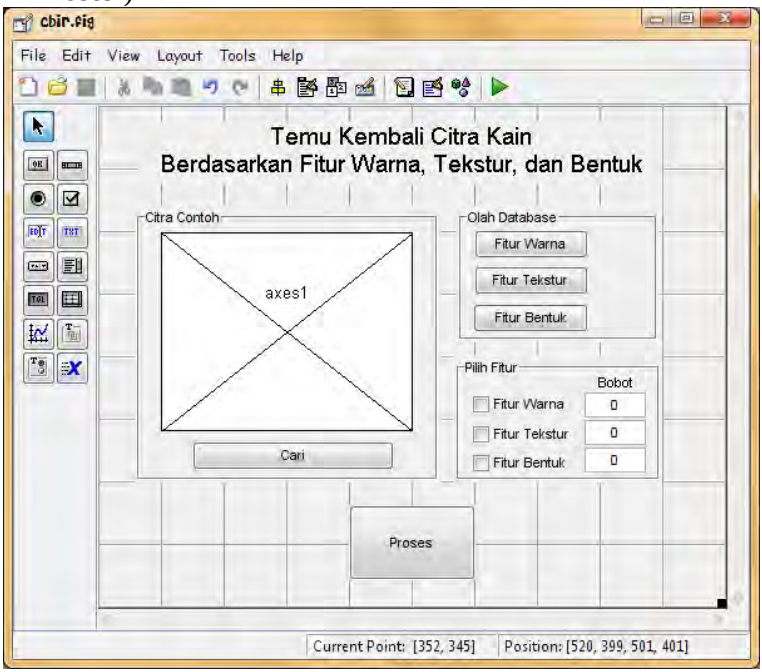
Gambar 3.5 Diagram Alir dari *Pseudo-Zernike Moment*

Gambar 3.6 Diagram Alir Pencarian Citra Mirip

3.4 Perancangan Antarmuka

Gambar 3.7 memperlihatkan rancangan antarmuka perangkat lunak temu kembali citra. Pengguna dapat melakukan hal-hal berikut pada aplikasi:

- Mengekstraksi fitur warna dari *database* citra (klik tombol 'Fitur Warna')
- Mengekstraksi fitur tekstur dari *database* citra (klik tombol 'Fitur Tekstur')
- Mengekstraksi fitur bentuk dari *database* citra (klik tombol 'Fitur Bentuk)
- Memilih citra contoh (klik tombol 'Cari')
- Memilih fitur pembanding kemiripan (centang *checkbox*)
- Menetapkan bobot fitur (isi *textbox* di sebelah nama fitur)
- Mencari citra yang mirip dengan citra contoh (klik tombol 'Proses')



Gambar 3.7 Rancangan Antarmuka Aplikasi

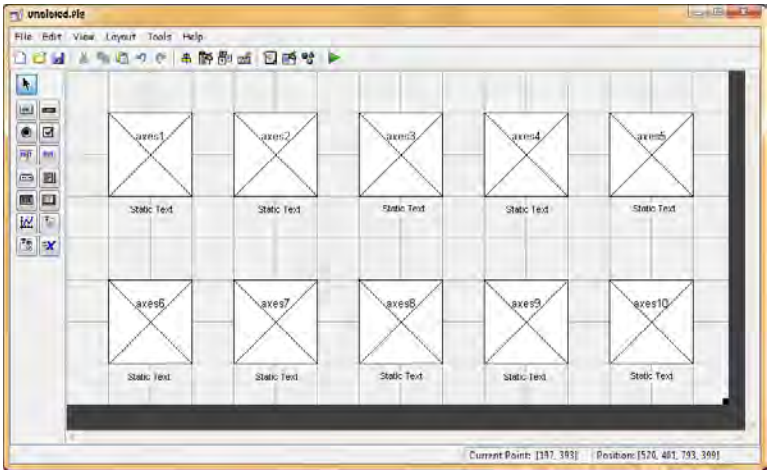
Tabel 3.7 Keterangan Komponen Rancangan Antarmuka Aplikasi

No	Nama Komponen	Jenis	Keterangan
1.	axes1	*axes*	Menampilkan citra contoh yang dipilih pengguna.
2.	btnCari	*Button*	Tombol untuk mencari citra yang menjadi citra contoh.
3.	btnFitur Warna	*Button*	Tombol untuk mengekstraksi fitur warna dari *database* citra.
4.	btnFitur Tekstur	*Button*	Tombol untuk mengekstraksi fitur tekstur dari *database* citra.
5.	btnFitur Bentuk	*Button*	Tombol untuk mengekstraksi fitur bentuk dari *database* citra.
6.	checkFitur Warna	*Checkbox*	Digunakan untuk memilih fitur warna dalam cek kemiripan.
7.	checkFitur Tekstur	*Checkbox*	Digunakan untuk memilih fitur tekstur dalam cek kemiripan.
8.	checkFitur Bentuk	*Checkbox*	Digunakan untuk memilih fitur bentuk dalam cek kemiripan.
9.	txtFitur Warna	*EditText*	Digunakan untuk mengisi nilai bobot dari fitur warna.
10.	txtFitur Tekstur	*EditText*	Digunakan untuk mengisi nilai bobot dari fitur tekstur.
11.	txtFitur Bentuk	*EditText*	Digunakan untuk mengisi nilai bobot dari fitur bentuk.
12.	btnProses	*Button*	Tombol yang digunakan untuk menghitung kemiripan antara citra contoh dan citra dalam *database.*

Hal yang pertama kali perlu dilakukan saat menjalankan perangkat lunak ini adalah mengekstraksi fitur citra dalam *database* dengan menekan tombol-tombol pada *button group*, yaitu tombol ‘Fitur Warna’, ‘Fitur Tekstur’, dan ‘Fitur Bentuk’. Setelah ditekan, citra dari *database* akan diekstraksi sesuai fitur yang dipilih. Setelah proses ekstraksi fitur selesai, akan muncul tulisan ‘OK’ disamping tombol fitur yang dipilih yang menandakan bahwa proses ekstraksi fitur telah selesai.

Setelah proses ekstraksi fitur selesai, citra contoh dipilih. Citra inilah yang akan dijadikan patokan pencarian citra mirip. Fitur yang ingin dijadikan pengukur kemiripan dipilih. Bobot dari fitur terpilih diisi dengan bilangan antara 0 hingga 1. Setelah itu tombol ‘Proses’ ditekan. Proses pencarian citra mirip pun akan dijalankan.

Setelah proses pencarian selesai, aplikasi akan menampilkan sepuluh citra yang mirip dengan citra contoh. Citra hasil tersebut adalah yang menurut sistem memiliki kemiripan tinggi dengan citra contoh. Keterangan mengenai komponen yang terdapat pada Gambar 3.7 dapat dilihat pada tabel Tabel 3.7.



Gambar 3.8 Rancangan Antarmuka Hasil

Beberapa saat setelah tombol ‘Proses’ ditekan, akan muncul jendela baru yang menampilkan beberapa citra yang mirip dengan citra contoh. Gambar 3.8 merupakan rancangan tampilan dari keluaran aplikasi. Hanya terdapat dua jenis komponen, yaitu *axes* dan *text*. Komponen *axes* akan menampilkan citra sedangkan *text* akan menampilkan nama file dari citra tersebut dan kemiripannya dengan citra contoh.

BAB IV
IMPLEMENTASI

Pada bab ini akan diuraikan mengenai implementasi perangkat lunak yang meliputi algoritma dan kode program yang terdapat dalam perangkat lunak. Pada tahap implementasi dari tiap fungsi, akan dijelaskan mengenai parameter masukan, keluaran, dan beberapa keterangan yang berhubungan dengan program dan teori.

4.1 Lingkungan Implementasi

Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam implementasi perangkat lunak untuk tugas akhir ini ditampilkan pada Tabel 4.1.

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak

Perangkat	Spesifikasi
Perangkat Keras	Prosesor : Intel(R) Core(TM)i3 CPU M350 @2.27 GHz Memori : 2 GB
Perangkat Lunak	Sistem Operasi : Windows 7 Ultimate 32-bit Perangkat Pengembang : Ms. Visio 2013 Ms. Word 2007 Matlab 7.6.0

4.2 Implementasi Algoritma pada Aplikasi

Pada bagian ini akan dijelaskan mengenai implementasi kode program yang terdapat dalam perangkat lunak. Pada tahap implementasi akan dijelaskan mengenai parameter masukan, keluaran dan beberapa keterangan yang berhubungan dengan program dan teori. Impelentasi yang dilakukan terdiri atas dua bagian besar, yaitu implementasi ekstraksi fitur dan perhitungan kemiripan.

4.2.1 Fungsi Utama

Fungsi utama perangkat lunak yang dibuat ini terdapat pada *file* cbir.m. *File* ini berfungsi untuk memanggil fungsi ekstraksi fitur dan perhitungan kemiripan.

Masukan : Citra contoh Keluaran : -	
1.	`function btnCari_Callback(hObject, eventdata, handles)`
2.	`[filename, pathname] = uigetfile('*.jpg;*.bmp;*.gif;*.png','Pilih Gambar : ');`
3.	`S = imread([pathname,filename]);`
4.	`axes(handles.axes1);`
5.	`imshow(S);`
6.	`handles.imgInput = S;`
7.	`guidata(hObject, handles);`

Gambar 4.1 Mengambil Citra dari *Dataset*

Masukan : - Keluaran : Fitur warna	
8.	`function btnWarna_Callback(hObject, eventdata, handles)`
9.	`tic()`
10.	`warna = callFitur(1);`
11.	`toc()`
12.	`handles.warna = warna;`
13.	`guidata(hObject, handles);`
14.	`set(handles.btnWarna, 'Enable', 'off');`
15.	`set(handles.statusWarna, 'String', 'OK')`

Gambar 4.2 Memanggil Fungsi Ekstraksi Fitur Warna

Gambar 4.1 menunjukkan pengambilan *file* citra dari *dataset* citra kain. Citra ini yang akan diolah dan ditemukan citra yang mirip dengannya. Bagian ini adalah implementasi dari diagram alir pada Gambar 3.2 pada bagian ‘Baca Citra’. Gambar 4.2 berfungsi untuk memanggil fungsi ekstraksi fitur warna. Citra yang

diekstraksi adalah semua citra dalam *database*. Ekstraksi fitur warna dilakukan dengan menggunakan metode *Dominant Color Descriptor* (DCD). Untuk setiap citra akan didapatkan *array* berdimensi 8x4 dimana setiap baris mewakili satu warna dan persentase warna tersebut. Bagian ini adalah implementasi dari Gambar 3.2 pada bagian 'Ekstraksi fitur warna'.

```
Masukan : -
Keluaran : Fitur tekstur
16.  function btnTekstur_Callback(hObject,
     eventdata, handles)
17.  tic()
18.  tekstur = callFitur(2);
19.  toc()
20.  handles.tekstur = tekstur;
21.  guidata(hObject, handles);
22.  set(handles. btnTekstur, 'Enable', 'off');
23.  set(handles.statusTekstur, 'String', 'OK')
```

Gambar 4.3 Memanggil Fungsi Ekstraksi Fitur Tekstur

Gambar 4.3 berfungsi untuk memanggil fungsi ekstraksi fitur tekstur. Citra yang diekstraksi adalah semua citra dalam *database*. Ekstraksi fitur tekstur akan dilakukan dengan menggunakan metode *Steerable filter*. Bagian ini adalah implementasi dari Gambar 3.2 pada bagian 'Ekstraksi fitur tekstur'.

```
Masukan : -
Keluaran : Fitur bentuk
24.  function btnBentuk_Callback(hObject,
     eventdata, handles)
25.  tic()
26.  bentuk = callFitur(3);
27.  toc()
28.  handles.bentuk = bentuk;
29.  guidata(hObject, handles);
30.  set(handles. btnBentuk, 'Enable', 'off');
31.  set(handles.statusBentuk, 'String', 'OK')
```

Gambar 4.4 Memanggil Fungsi Ekstraksi Fitur Bentuk

Gambar 4.4 berfungsi untuk mengekstraksi fitur bentuk dari *database* citra. Ekstraksi fitur bentuk akan dilakukan dengan menggunakan *Zernike moment*. Bagian ini adalah implementasi dari Gambar 3.2 pada bagian 'Ekstraksi fitur bentuk'.

Gambar 4.5 berfungsi untuk memilih fitur apa saja yang akan diperhatikan dalam proses pencarian citra mirip. Terdapat tiga fitur yang dapat dipilih, yaitu warna, tekstur, dan bentuk. Pengguna dapat memilih satu hingga tiga fitur.

Masukan : -
Keluaran : Fitur terpilih

```
32. function checkWarna_Callback(hObject,
    eventdata, handles)
33. useWarna = get(hObject, 'Value');
34. guidata(hObject,handles); % to save the
    change of the handles structure
35.
36. function checkTekstur_Callback(hObject,
    eventdata, handles)
37. useTekstur = get(hObject, 'Value');
38. handles.useTekstur = useTekstur;
39. guidata(hObject,handles);
40.
41. function checkBentuk_Callback(hObject,
    eventdata, handles)
42. useBentuk = get(hObject, 'Value');
43. handles.useBentuk = useBentuk;
44. guidata(hObject,handles);
```

Gambar 4.5 Memilih Fitur

Gambar 4.6 berfungsi untuk mengeset bobot fitur yang dipilih. Total bobot dari semua fitur yang dipilih harus 1. Apabila hanya satu fitur yang dipilih, maka bobot fitur tersebut harus diisi 1. Apabila dipilih lebih dari satu fitur, maka bobot masing-masing fitur dapat bervariasi. Namun total bobot dari semua fitur yang dipilih tersebut tetap harus 1. Semakin besar bobot suatu fitur, maka akan semakin diperhitungkan kemiripan pada fitur tersebut dalam proses pencarian citra mirip.

Gambar 4.7 menunjukkan proses pencarian citra mirip. Citra contoh yang telah dipilih pada Gambar 4.2 akan diekstraksi fitur-fiturnya. Bagian ini adalah implementasi dari Gambar 3.2 pada bagian 'Perhitungan kemiripan'. Sesuai dengan fitur yang dipilih oleh pengguna, kemiripan citra contoh dengan *database* citra akan dihitung. Sepuluh citra dari *database* yang paling mirip dengan citra contoh akan ditampilkan.

Masukan : - Keluaran : Bobot fitur	
45.	`function wColor_Callback(hObject, eventdata, handles)`
46.	`bobotWarna = str2double(get(hObject, 'String'));`
47.	`handles.bobotWarna = bobotWarna;`
48.	`guidata(hObject,handles);`
49.	
50.	`function wTexture_Callback(hObject, eventdata, handles)`
51.	`bobotTekstur = str2double(get(hObject, 'String'));`
52.	`handles.bobotTekstur = bobotTekstur;`
53.	`guidata(hObject,handles);`
54.	
55.	`function wShape_Callback(hObject, eventdata, handles)`
56.	`bobotBentuk = str2double(get(hObject, 'String'));`
57.	`handles.bobotBentuk = bobotBentuk;`
58.	`guidata(hObject,handles);`

Gambar 4.6 Mengeset Bobot Tiap Fitur

Masukan : Citra contoh Keluaran : Citra mirip	
59.	`function btnProses_Callback(hObject, eventdata, handles)`
60.	`if ((handles.useWarna*handles.bobotWarna)+(han`

```
    dles.useTekstur*handles.bobotTekstur)+(hand
    les.useBentuk*handles.bobotBentuk))~=1
61.     error('Total bobot harus 1');
62. end
63. contents = dir('dataset kain/*.jpg');
64. dbImgCount = numel(contents);
65. colorSim=zeros(dbImgCount,1);
66. textureSim=zeros(dbImgCount,1);
67. shapeSim=zeros(dbImgCount,1);
68. if handles.useWarna==1
69.     imgColor = dcd(handles.imgInput);
70.     temp =
    horzcat(imgColor(:,1:3),imgColor(:,5));
71.     inColor=[];
72.     for i=1:8
73.       if(imgColor(i,5)~=0)
74.           inColor=vertcat(inColor,temp);
75.       end
76.     end
77.     colorSim = color_sim(inColor,
    handles.warna);
78.     nmax=max(colorSim);
79.     nmin=min(colorSim);
80.     colorSim=(colorSim-nmin)/(nmax-nmin);
81. end
82. if handles.useTekstur==1
83.     theta = [0, 45, 90, 315];
84.     I = rgb2gray(handles.imgInput);
85.     tempOut =
    steerGaussFilterOrder1(I,theta,3);
86.     textureSim = texture_sim(tempOut,
    handles.tekstur);
87.     nmax=max(textureSim);
88.     nmin=min(textureSim);
89.     textureSim=(textureSim-nmin)/(nmax-
    nmin);
90. end
91. if handles.useBentuk==1
92.     inShape=pzernfun(handles.imgInput);
93.     shapeSim =
```

```
    shape_sim(inShape,handles.bentuk);
94.     nmax=max(shapeSim);
95.     nmin=min(shapeSim);
96.     shapeSim=(shapeSim-nmin)/(nmax-nmin);
97. end
98. allSim =
    (colorSim*handles.bobotWarna)+(textureSim*h
    andles.bobotTekstur)+(shapeSim*handles.bobo
    tBentuk);
99. [outSim,index] = sort(allSim);
100 % show 10th similar images
101 imgdir = 'dataset kain/';
102 figure
103 for i=1:10
104     if outSim(i)<1
105         img =
    imread([imgdir,contents(index(i)).name]);
106         subplot(2,5,i);
107         imshow(img);
108 title({[contents(index(i)).name(1:20),'...'
    ];['Sim: #',num2str(outSim(i),'%.3f')]});
109     end
110 end
```

Gambar 4.7 Mencari Citra Mirip

4.2.2 Implementasi Proses Ekstraksi Fitur

Pada bagian ini, akan dijelaskan mengenai implementasi proses ekstraksi fitur yang telah dijelaskan pada subbab 2.1. Gambar 4.8 adalah fungsi yang akan memanggil metode ekstraksi fitur sesuai dengan fitur yang akan diekstraksi. Fungsi ini terdapat dalam *file* callFitur.m dan digunakan untuk mengekstraksi *database* citra.

```
Masukan : Id fitur
Keluaran : Hasil ekstraksi fitur
111. function outArr = callFitur(idFitur)
112. outArr = [];
113. contents = dir('dataset kain/*.jpg');
114. for z = 1:numel(contents)
```

```
115.     filename = contents(z).name;
116.     %read image
117.     I = imread(['dataset
kain/',filename]);
118.     switch idFitur
119.         case 1 % warna
120.             sumGroup = dcd(I);
121.             tempArr =
horzcat(sumGroup(:,1:3),sumGroup(:,5));
122.         case 2 % tekstur
123.             theta = [0, 45, 90, 315];
124.             dim = ndims(I);
125.             if(dim == 3)
126.                 I = rgb2gray(I);
127.             end
128.             tempArr =
steerGaussFilterOrder1(I,theta,3);
129.         case 3 % bentuk
130.             tempArr=pzernfun(I);
131.     end
132.     outArr = vertcat(outArr,tempArr);
133. end
134. % write feature file
135. switch idFitur
136. case 1
137.     xlswrite('db_color.xlsx',outArr);
138. case 2
139.     xlswrite('db_texture.xlsx',outArr);
140. case 3
141.     xlswrite('db_shape.xlsx',outArr);
142. end
143. end
```

Gambar 4.8 Fungsi Pemanggil Metode Ekstraksi Fitur

4.2.2.1 *Dominant Color Descriptor*

Bagian ini adalah implementasi dari diagram alir pada Gambar 3.3. Implementasi dari proses ekstraksi fitur warna oleh metode *dominant color descriptor* dimulai dengan membagi

warna yang terdapat dalam citra ke dalam grup partisi seperti yang ditampilkan pada Gambar 4.9.

Masukan : Citra
Keluaran : Nilai total warna tiap grup

```
1.  function sumGroup = dcd(I)
2.      s_img = size(I);
3.      r = I(:,:,1);
4.      g = I(:,:,2);
5.      b = I(:,:,3);
6.      inputImg = zeros((s_img(1) * s_img(2)),
    4); %4-th column for num of partition
7.      inputImg(:,1) = r(:);
8.      inputImg(:,2) = g(:);
9.      inputImg(:,3) = b(:);
10.     %grouping
11.     sumGroup = zeros(8,5); %R G B
    numOfPixel percen, row i means group i
12.     for i=1:(size(I,1) * size(I,2))
13.         if((inputImg(i,1)<128) &&
    (inputImg(i,2)<128) && (inputImg(i,3)<128))
14.             inputImg(i,4)=1; %group 1
15.             sumGroup(1,1:3) =
    sumGroup(1,1:3) + inputImg(i,1:3);
16.             sumGroup(1,4) = sumGroup(1,4)+1;
17.         elseif((inputImg(i,1)>=128) &&
    (inputImg(i,2)<128) && (inputImg(i,3)<128))
18.             inputImg(i,4)=2; %group 2
19.             sumGroup(2,1:3) =
    sumGroup(2,1:3) + inputImg(i,1:3);
20.             sumGroup(2,4) = sumGroup(2,4)+1;
21.         elseif((inputImg(i,1)<128) &&
    (inputImg(i,2)<128) &&
    (inputImg(i,3)>=128))
22.             inputImg(i,4) = 3; %group 3
23.             sumGroup(3,1:3) =
    sumGroup(3,1:3) + inputImg(i,1:3);
24.             sumGroup(3,4) = sumGroup(3,4)+1;
25.         elseif((inputImg(i,1)>=128) &&
```

```
    (inputImg(i,2)<128) &&
    (inputImg(i,3)>=128))
26.             inputImg(i,4) = 4; %group 4
27.             sumGroup(4,1:3) =
    sumGroup(4,1:3) + inputImg(i,1:3);
28.             sumGroup(4,4) = sumGroup(4,4)+1;
29.         elseif((inputImg(i,1)<128) &&
    (inputImg(i,2)>=128) &&
    (inputImg(i,3)<128))
30.             inputImg(i,4) = 5; %group 5
31.             sumGroup(5,1:3) =
    sumGroup(5,1:3) + inputImg(i,1:3);
32.             sumGroup(5,4) = sumGroup(5,4)+1;
33.         elseif((inputImg(i,1)>=128)         &&
    (inputImg(i,2)>=128)                        &&
    (inputImg(i,3)<128))
34.             inputImg(i,4) = 6; %group 6
35.             sumGroup(6,1:3)                  =
    sumGroup(6,1:3) + inputImg(i,1:3);
36.             sumGroup(6,4) = sumGroup(6,4)+1;
37.         elseif((inputImg(i,1)<128)          &&
    (inputImg(i,2)>=128)                        &&
    (inputImg(i,3)>=128))
38.             inputImg(i,4) = 7; %group 7
39.             sumGroup(7,1:3)                  =
    sumGroup(7,1:3) + inputImg(i,1:3);
40.             sumGroup(7,4) = sumGroup(7,4)+1;
41.         elseif((inputImg(i,1)>=128)         &&
    (inputImg(i,2)>=128)                        &&
    (inputImg(i,3)>=128))
42.             inputImg(i,4) = 8; %group 8
43.             sumGroup(8,1:3)                  =
    sumGroup(8,1:3) + inputImg(i,1:3);
44.             sumGroup(8,4) = sumGroup(8,4)+1;
45.         end
46.     end
```

Gambar 4.9 Membagi Warna RGB ke dalam 8 Grup Partisi

Setelah setiap warna dalam citra terkelompokkan dalam masing-masing grup, akan dihitung rata-rata per grup. Rata-rata tersebut

dianggap sebagai warna yang merepresentasikan keseluruhan warna pada grup tersebut. Gambar 4.10 menunjukkan proses mencari warna rata-rata per grup.

Masukan : Nilai total warna tiap grup
Keluaran : Nilai rata-rata warna dan persentasenya

```
47.     for i=1:8
48.         if(sumGroup(i,4)>0)
49.             sumGroup(i,1:3) =
    sumGroup(i,1:3) / sumGroup(i,4);
50.             sumGroup(i,5) = sumGroup(i,4) *
    100  / (size(I,1) * size(I,2));
51.         end
52.     end
```

Gambar 4.10 Menghitung Rata-Rata Warna per Grup

Setelah warna disederhanakan menjadi delapan, jarak antar warna akan dihitung. Gambar 4.11 menunjukkan perhitungan jarak antar warna. Jarak ini akan menjadi penentu apakah dua warna perlu dilebur atau tidak. Dua warna yang jaraknya kurang dari batas T_d akan dilebur dengan cara seperti dalam Gambar 4.12. Hal ini dilakukan supaya warna yang tersisa hanyalah warna yang saling beda namun tetap merepresentasikan isi citra.

Masukan : Nilai rata-rata warna dan persentasenya
Keluaran : Jarak antar warna

```
53.     dist = zeros(8,8);
54.     % checking threshold
55.     Tm = 6; % persentage threshold
56.     Td = 10; % distance threshold
57.
58.     % calculate the distance between Ci&Cj
59.     for i=1:8
60.        if(sumGroup(i,5)>0)
61.            for j=i+1:8
62.                if(sumGroup(j,5)>0)
63.                    dist(i,j) =
    sqrt((sumGroup(i,1) - sumGroup(j,1))^2 ...
64.                        +(sumGroup(i,2)-
```

```
    sumGroup(j,2))^2+(sumGroup(i,3)-
    sumGroup(j,3))^2);
65.                     dist(j,i) = dist(i,j);
66.                 end
67.             end
68.         end
69.     end
```

Gambar 4.11 Menghitung Jarak Antar Warna

Masukan : Jarak antar warna
Keluaran : Nilai warna baru

```
70.     for j=1:8
71.         anyMerge = 0; % 1 if some color is
    merged
72.         for i=1:8
73.             [distSort,indexSort] =
    sort(dist(i,:)); %sort per baris
74.             temp =
    find(distSort>0,1,'first');
75.             minIndex =
    indexSort(temp); %get index of minDist
76.             if(isempty(temp)==0 &&
    sumGroup(minIndex,5)>0 &&
    dist(i,minIndex)<Td)
77.                 % calculate new centroid
78.     sumGroup(i,1)=(sumGroup(i,1)*
    sumGroup(i,5)/(sumGroup(i,5)+
    sumGroup(minIndex,5))) ...
79.                     +(sumGroup(minIndex,1)*
    sumGroup(minIndex,5)/(sumGroup(i,5)+
    sumGroup(minIndex,5))); % R
80.                 sumGroup(i,2)=
    (sumGroup(i,2)*sumGroup(i,5)/
    (sumGroup(i,5)+sumGroup(minIndex,5))) ...
81.                     +(sumGroup(minIndex,2)*
    sumGroup(minIndex,5)/(sumGroup(i,5)+
    sumGroup(minIndex,5))); % G
82.                 sumGroup(i,3)=
    (sumGroup(i,3)*sumGroup(i,5)/
    (sumGroup(i,5)+sumGroup(minIndex,5))) ...
83.                     +(sumGroup(minIndex,3)*
```

```
        sumGroup(minIndex,5)/(sumGroup(i,5)+
        sumGroup(minIndex,5))); % B
84.                     sumGroup(minIndex,4:5)=
        sumGroup(i,4:5)+ sumGroup(minIndex,4:5);
85.                     % minIndex is index of
        similar color
86.     sumGroup(minIndex,1:3)=sumGroup(i,1:3);
87.                     sumGroup(i,4:5)=0;
88.                     anyMerge=1;
89.                     dist(i,minIndex) = 0;
90.                     dist(:,i) = 0;
91.                 end
92.             end
93.             if(anyMerge==0)
94.                     break;
95.             end
96.         end
```

Gambar 4.12 Proses *Merge* Antar Warna yang Berdekatan

Setelah tidak ada warna yang berdekatan, yaitu jarak antar warna tidak kurang dari batas T_d, pengecekan selanjutnya dilakukan. Karena warna yang ingin diambil adalah warna dominan, maka warna yang memiliki persentase kecil akan dilebur dengan warna yang terdekat dengannya. Hal ini dilakukan supaya warna yang terambil adalah warna yang cukup dominan. Gambar 4.13 menunjukkan proses *merge* pada warna yang kurang dominan.

```
Masukan : Persentase tiap warna
Keluaran : Fitur warna
97.     for i=1:8
98.         if(sumGroup(i,5)<Tm &&
        sumGroup(i,5)>0)
99.             [distSort,indexSort] =
        sort(dist(i,:)); %sort per baris
100             temp =
        find(distSort>0,1,'first');
101             minIndex =
        indexSort(temp); %get index of minDist
102             if(isempty(temp)==0 &&
```

```
    sumGroup(minIndex,5)>0)
103                 % calculate new centroid
104
    sumGroup(i,1)=(sumGroup(i,1)
    *sumGroup(i,5)/(sumGroup(i,5)+
    sumGroup(minIndex,5))) ...
105                     +(sumGroup(minIndex,1)*
    sumGroup(minIndex,5)/(sumGroup(i,5)+
    sumGroup(minIndex,5))); % R
106
    sumGroup(i,2)=(sumGroup(i,2)*
    sumGroup(i,5)/(sumGroup(i,5)+
    sumGroup(minIndex,5))) ...
107                     +(sumGroup(minIndex,2)*
    sumGroup(minIndex,5)/(sumGroup(i,5)+
    sumGroup(minIndex,5))); % G
108
    sumGroup(i,3)=(sumGroup(i,3)*
    sumGroup(i,5)/(sumGroup(i,5)+
    sumGroup(minIndex,5))) ...
109                     +(sumGroup(minIndex,3)*
    sumGroup(minIndex,5)/(sumGroup(i,5)+
    sumGroup(minIndex,5))); % B
110                 sumGroup(minIndex,4:5)=
    sumGroup(i,4:5)+ sumGroup(minIndex,4:5);
111                 % minIndex is index of
    similar color
112                 sumGroup(minIndex,1:3)=
    sumGroup(i,1:3);
113                 sumGroup(i,4:5)=0;
114                 dist(i,minIndex) = 0;
115                 dist(:,i) = 0;
116             end
117         end
118     end
119     sumGroup=round(sumGroup);
120
121 end
```

Gambar 4.13 Proses *Merge* Warna yang Persentasenya Kecil

Ekstraksi fitur warna dengan metode ini akan menghasilkan delapan warna sesuai dengan jumlah grup partisi. Setiap warna terdiri atas elemen warna merah, hijau, dan biru, serta persentase dari warna tersebut. Untuk satu citra akan dihasilkan 32 nilai.

4.2.2.2 *Steerable Filter*

Masukan : Citra
Keluaran : Citra hasil konvolusi

```
1.  function J =
    steerGaussFilterOrder1(I,theta,sigma)
2.  I = mean(double(I),3);
3.  % Determine necessary filter support (for
    Gaussian).
4.  x = [-sigma:sigma];
5.  [xx,yy] = meshgrid(x,x);
6.  G1_0 = -(xx/sigma^2).* exp(-
    (xx.^2+yy.^2)/(2*sigma^2))/
    (sigma*sqrt(2*pi));
7.  G1_90 = -(yy/sigma^2).* exp(-
    (xx.^2+yy.^2)/(2*sigma^2))/
    (sigma*sqrt(2*pi));
8.  % Calculate image gradients (using
    separability).
9.  Ix = imfilter(I,G1_0,'same','replicate');
10. Iy = imfilter(I,G1_90,'same','replicate');
11. tempOut = [];
```

Gambar 4.14 Menyaring Citra dengan Filter *Gaussian*

Metode ekstraksi fitur tekstur yang digunakan adalah *steerable filter*. Metode ini terdapat pada file s*teerGaussFilterOrder1.m*. Bagian ini adalah implementasi dari diagram alir pada Gambar 3.4. Gambar 4.14 menunjukkan proses pembuatan filter *Gaussian* dan konvolusi citra dengan filter tersebut.

Setelah citra disaring, nilai statistika dari citra yang telah terfilter dihitung. Nilai statistika yang digunakan adalah rata-rata dan

standar deviasi. Gambar 4.15 menunjukkan proses penghitungan nilai statistika yang menjadi hasil ekstraksi fitur tekstur.

Masukan : Citra hasil konvolusi Keluaran : Fitur tekstur	
12.	`for i=1:length(theta)`
13.	`    % Evaluate oriented filter response.`
14.	`    S = cos(theta(i))*Ix+sin(theta(i))*Iy;`
15.	`    E = 0;`
16.	`    for x = 1:size(I,1)`
17.	`        for y = 1:size(I,2) % calculate the energy`
18.	`            E = E+abs(S(x,y));`
19.	`        end`
20.	`    end`
21.	`    meanT = E/(size(I,1)*size(I,2)); % mean`
22.	`    tempOut = horzcat(tempOut,meanT);`
23.	`    temp = 0;`
24.	`    for x = 1:size(I,1)`
25.	`        for y = 1:size(I,2) % standard deviation`
26.	`            temp = temp+((S(x,y)-meanT)^2);`
27.	`        end`
28.	`    end`
29.	`    stdT = sqrt(temp/(size(I,1)*size(I,2)));`
30.	`    tempOut = horzcat(tempOut,stdT);`
31.	`end`
32.	`J = tempOut;`
33.	`end`

Gambar 4.15 Mendapatkan Nilai Statistika dari Citra Terfilter

4.2.2.3 *Pseudo-Zernike Moment*

Bagian ini adalah implementasi dari diagram alir pada Gambar 3.5. *Pseudo-Zernike moment* adalah metode yang digunakan untuk mengekstraksi fitur bentuk. Citra yang diekstraksi dengan metode ini adalah citra hitam putih.

Gambar 4.16 menunjukkan proses ekstraksi fitur bentuk pada Tugas Akhir ini.

Masukan : Citra Keluaran : Fitur bentuk	
1.	`function A = pzernfun(I)`
2.	
3.	`bw=im2bw(I);`
4.	`imgSize=size(bw);`
5.	
6.	`% Build a grid, based on image size`
7.	`N = ceil(min(imgSize(1),imgSize(2))/2)-1;`
8.	`x = (-N:N)/N;`
9.	`[X,Y] = meshgrid(x,x);`
10.	`[theta,r] = cart2pol(X,Y);`
11.	`is_in_circle = r <= 1;`
12.	`r = r(is_in_circle);`
13.	`theta = theta(is_in_circle);`
14.	
15.	`% Create some data`
16.	`F=bw(1:size(X,2),1:size(X,1));`
17.	
18.	`% Compute a (finite) basis of pseudo-Zernike functions`
19.	`n_max = 10; % >> more similar with the original image`
20.	`n = zeros(1,(n_max+1)*(n_max+2)/2);`
21.	`m = zeros(1,(n_max+1)*(n_max+2)/2);`
22.	`for k = 0:n_max`
23.	`n((k+1)*k/2+1:(k+1)*(k+2)/2) = repmat(k,1,k+1);`
24.	`m((k+1)*k/2+1:(k+1)*(k+2)/2) = 0:k;`
25.	`end`
26.	
27.	`% Compute the values of the Zernike polynomials:`
28.	`P = zeros(length(r),length(n));`
29.	`for j = 1:length(n)`
30.	`s = 0:(n(j)-m(j));`

```
31.     for k = length(s):-1:1
32.         c = (1-2*mod(s(k),2))*          ...
33.                 prod(2:2*n(j)+1-s(k))/  ...
34.                 prod(2:s(k))/           ...
35.                 prod(2:(n(j)+m(j)+1-
    s(k)))/ ...
36.                 prod(2:(n(j)-m(j)  -s(k)));
37.         P(:,j)  =  P(:,j)  +  (c*(r.^(n(j)-
    s(k))));
38.     end
39.     P(:,j) = P(:,j)*((n(j)+1)/pi);
40. end
41.
42. % Compute the Zernike functions
43. P = P.*exp(i*theta*m);
44.
45. %  Estimate  the  pseudo-Zernike  moments,
    using simple summation to approxmiate the
    integrals
46. A = zeros(1,(n_max+1)*(n_max+2)/2);
47. for k = 1:((n_max+1)*(n_max+2)/2)
48.    A(k) = sum(F(is_in_circle)'*P(:,k));
49. end
50. end
```

Gambar 4.16 Ekstraksi Fitur Bentuk

4.2.3 Implementasi Perhitungan Kemiripan Fitur

Pada bagian ini, akan dijelaskan implementasi dari proses perhitungan kemiripan dan pencarian citra mirip. Tiap fitur memiliki rumus perhitungan kemiripan tersendiri. Data yang digunakan dalam perhitungan kemiripan adalah fitur *database* citra dan fitur citra contoh. Bagian ini adalah implementasi dari diagram alir pada Gambar 3.6.

Perhitungan kemiripan fitur warna dapat dilihat pada Gambar 4.17. Fitur warna dari citra contoh yang berupa *array* berukuran 8x4 akan dihitung kemiripannya dengan fitur warna *database* citra.

Masukan : Fitur warna citra contoh dan *database* citra
Keluaran : Jarak berdasarkan warna

```
1.  function dbImg = color_sim(inColor, dbColor)
2.      dbImg = []; %arr for sim of i-th img
3.      for z=1:8:size(dbColor,1) % 1 9 17 ...
4.          dbTest = [];
5.          for i=1:8
6.              if(dbColor(z+i-1,4)>0)
7.                  dbTest = vertcat(dbTest, dbColor(z+i-1,:)); % skip zero value
8.              end
9.          end
10.         sim = 0;
11.         for i=1:size(inColor,1)
12.             for j=1:size(dbTest,1)
13.                 d = sqrt((inColor(i,1)-dbTest(j,1))^2 + (inColor(i,2)-dbTest(j,2))^2 + (inColor(i,3)-dbTest(j,3))^2);
14.                 s = (1-abs((inColor(i,4)-dbTest(j,4))/100))* min(inColor(i,4)/100,dbTest(j,4)/100);
15.                 if(d<=25) % 25=Td
16.                     a = 1-(d/50); % 50=d_max=alpha*Td >> alpha=2
17.                 else
18.                     a = 0;
19.                 end
20.                 sim = sim+(a*s);
21.             end
22.         end
23.         distance = 1-sim;
24.         dbImg=vertcat(dbImg,distance);
25.     end
26. end
```

Gambar 4.17 Perhitungan Kemiripan Fitur Warna

Gambar 4.18 menunjukkan perhitungan kemiripan fitur tekstur dan Gambar 4.19 untuk fitur bentuk. Keduanya menggunakan rumus *Euclidean* untuk mencari jarak antar citra.

Masukan : Fitur tekstur citra contoh dan *database* citra Keluaran : Jarak berdasarkan tekstur	
1.	`function dbImg = texture_sim(inTexture,dbTexture)`
2.	`  dbImg = []; %arr for sim of i-th img`
3.	`  for z = 1:size(dbTexture,1)`
4.	`    dist = 0;`
5.	`    for i=1:2:8`
6.	`      dist = dist+((inTexture(i)-dbTexture(z,i))^2+(inTexture(i+1)-dbTexture(z,i+1))^2);`
7.	`    end`
8.	`    sim = sqrt(dist);`
9.	`    dbImg=vertcat(dbImg,sim);`
10.	`  end`
11.	`end`

Gambar 4.18 Perhitungan Kemiripan Fitur Tekstur

Masukan : Fitur bentuk citra contoh dan *database* citra Keluaran : Jarak berdasarkan bentuk	
1.	`function dbImg = shape_sim(inShape, dbShape)`
2.	`  dbImg = []; %arr for similarity of i-th img`
3.	`  for i = 1:size(dbShape,1)`
4.	`    dist = 0;`
5.	`    for j=1:size(dbShape,2)`
6.	`      dist = dist+((abs(inShape(j))-abs(dbShape(i,j)))^2);`
7.	`    end`
8.	`    sim = sqrt(dist);`
9.	`    dbImg=vertcat(dbImg,sim);`
10.	`  end`
11.	`end`

Gambar 4.19 Perhitungan Kemiripan Fitur Bentuk

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan mengenai rangkaian uji coba dan evaluasi yang dilakukan. Pembahasan yang dikemukakan meliputi data uji coba, hasil uji coba, dan evaluasi.

5.1 Lingkungan Uji Coba

Lingkungan uji coba yang digunakan dalam pembuatan Tugas Akhir ini meliputi perangkat lunak dan perangkat keras yang digunakan untuk Temu Kembali Citra Berbasis Isi pada Citra Kain Berdasarkan Fitur Warna, Tekstur, dan Bentuk. Lingkungan uji coba merupakan komputer tempat uji coba perangkat lunak. Berikut adalah lingkungan uji coba yang digunakan pada Tugas Akhir ini.

Tabel 5.1 Lingkungan Uji Coba

Perangkat	Spesifikasi
Perangkat Keras	Prosesor : Intel(R) Core(TM)i3 CPU M350 @2.27 GHz Memori : 4 GB
Perangkat Lunak	Sistem Operasi : Windows 7 Ultimate 32-bit Perangkat Pengembang : Ms. Visio 2013 Ms. Word 2007 Matlab 7.6

5.2 Data Uji Coba

Terdapat dua jenis *dataset* yang digunakan dalam uji coba. *Dataset* yang pertama berisi 280 citra yang memiliki ukuran sama, yaitu 400x400 piksel dan setiap 10 citra diambil dari satu citra kain yang sama sehingga memiliki warna, tekstur, maupun bentuk yang mirip. Citra kain sebelum dipotong memiliki ukuran beragam, salah satunya ukurannya adalah 3000x2000 piksel. Contoh citra kain dapat dilihat pada Gambar 5.1. Gambar 5.2

adalah 10 citra untuk *dataset* pertama yang diambil dari Gambar 5.1 pada daerah yang berbeda-beda dan telah diperkecil ukurannya. *Dataset* kedua berisi 172 citra yang berbeda antara satu dengan yang lain namun memiliki ukuran sama, yaitu 400x400 piksel. Pada *dataset* kedua, setiap citra diambil dari kain yang berbeda meskipun ada beberapa kain yang memiliki kemiripan tekstur maupun warna. Beberapa citra pada *dataset* kedua dapat dilihat pada Tabel A.2.

Gambar 5.1 Citra Kain dengan Ukuran Besar

Gambar 5.2 10 Citra dari Kain yang sama

5.3 Skenario Uji Coba

Pada bagian ini dijelaskan mengenai skenario uji coba yang telah dilakukan. Terdapat beberapa jenis uji coba yang dilakukan, yaitu:

1. Uji coba 1 yaitu uji coba ekstraksi fitur tekstur dengan kombinasi sudut yang berbeda.
2. Uji coba 2 yaitu uji coba ekstraksi fitur bentuk dengan nilai orde berbeda.
3. Uji coba 3 yaitu uji performa temu kembali citra kain berdasarkan variasi fitur pada *dataset* pertama.
4. Uji coba 4 yaitu uji performa temu kembali citra kain dengan variasi bobot pada kombinasi fitur terbaik.
5. Uji coba pada *dataset* kedua.

5.3.1 Uji Coba 1

Pada uji coba pertama, proses ekstraksi fitur tekstur akan diuji performanya dalam merepresentasikan citra. Nilai performa diambil dari nilai *precision* temu kembali citra berdasarkan fitur tekstur. Pada proses ekstraksi fitur tekstur, *steerable filter* sebagai metode ekstraksi fitur tekstur memerlukan masukan berupa beberapa sudut. Banyak sudut dan kombinasi nilainya akan diuji mana yang dapat menghasilkan performa terbaik. Kombinasi nilai sudut yang akan diuji dapat dilihat pada Tabel 5.2.

Tabel 5.2 Kombinasi Sudut pada Ekstraksi Fitur Tekstur

Nama Kombinasi	Nilai kombinasi sudut	Waktu ekstraksi
Kombinasi 1	[0; 45; 90; 135]	14,84 s
Kombinasi 2	[0; 30; 60; 90; 120; 150]	16,15 s
Kombinasi 3	[0; 22,5; 45; 67,5; 90; 112,5; 135; 157,5]	17,23 s

Citra contoh yang digunakan adalah beberapa dari citra yang ada dalam *database* pertama dan dapat dilihat pada Gambar 5.3 dengan urutan dari kiri atas ke kanan. Setelah dihitung kemiripan

dengan masing-masing citra dalam *database*, akan dimunculkan 10 citra yang mirip berdasarkan jarak antara dua citra. Karena pada *database* terdapat 10 citra yang saling mirip, maka diharapkan semua citra yang terambil adalah ke-10 citra tersebut. Citra yang seharusnya mirip dengan citra contoh dapat dilihat pada Tabel A.1. Tampilan hasil uji coba dapat dilihat pada lampiran.

Gambar 5.3 Citra Contoh untuk Uji Coba

Tabel 5.3 Hasil Uji Coba Ekstraksi Fitur Tekstur

Nama Citra	*Precision*		
	Kombinasi sudut 1	Kombinasi sudut 2	Kombinasi sudut 3
Citra 1	0,50	0,50	0,50
Citra 2	0,90	0,90	0,90
Citra 3	0,90	0,90	0,90
Citra 4	0,50	0,50	0,50
Citra 5	1,00	1,00	1,00
Citra 6	0,40	0,40	0,40
Citra 7	0,80	0,80	0,80
Citra 8	0,70	0,70	0,70
Citra 9	0,70	0,70	0,70
Citra 10	0,70	0,70	0,70
Rata-rata	0,71	0,71	0,71

Hasil uji coba ekstraksi fitur dengan beberapa macam kombinasi sudut dapat dilihat pada Tabel 5.3. Nilai *precision* terbaik bernilai 1 yang memiliki arti semua citra dari kain yang sama terambil.

Hasil yang didapat dari masing-masing kombinasi sudut ternyata bernilai sama. Rata-rata performa temu kembali citra dari setiap kombinasi sudut adalah 0,71. Karena itu, dapat dipilih salah satu dari kombinasi yang telah diuji. Namun, karena pada saat ekstraksi fitur kombinasi sudut pertama membutuhkan waktu paling sedikit, maka kombinasi tersebut akan digunakan. Pada uji coba selanjutnya, ekstraksi fitur tekstur akan menggunakan kombinasi sudut pertama, yaitu 0, 45, 90, dan 135.

5.3.2 Uji Coba 2

Pada uji coba kedua, proses ekstraksi fitur bentuk akan diuji performanya dalam merepresentasikan citra. *Pseudo-Zernike moment* sebagai metode ekstraksi fitur bentuk memerlukan masukan berupa besar orde yang digunakan. Semakin besar orde, informasi citra semakin mendekati citra asli, namun akan semakin banyak *noise* yang ikut terambil. Pada uji coba ini, akan digunakan beberapa nilai orde. Setelah itu, performa berdasarkan fitur bentuk dengan nilai orde tertentu akan dihitung.

Gambar 5.4 Waktu Ekstraksi Fitur Bentuk dengan Orde Tertentu

Citra yang digunakan sebagai citra contoh diambil dari *database* pertama dan dapat dilihat pada Gambar 5.3. Setelah dihitung kemiripan dengan masing-masing citra dalam *database*, akan dimunculkan 10 citra yang mirip berdasarkan jarak antara dua citra. Karena pada *database* pertama terdapat 10 citra yang saling

mirip, maka diharapkan semua citra yang terambil adalah ke-10 citra tersebut. Tampilan hasil uji coba dapat dilihat pada lampiran.

Tabel 5.4 Hasil Uji Coba Ekstraksi Fitur Bentuk

Nama Citra	*Precision*			
	Orde = 6	Orde = 7	Orde = 8	Orde = 9
Citra 1	0,60	0,60	0,60	0,50
Citra 2	0,30	0,40	0,40	0,20
Citra 3	0,80	0,80	0,80	0,80
Citra 4	0,30	0,30	0,20	0,20
Citra 5	0,30	0,40	0,40	0,40
Citra 6	0,20	0,20	0,20	0,20
Citra 7	0,60	0,50	0,50	0,50
Citra 8	0,40	0,40	0,50	0,40
Citra 9	0,30	0,30	0,30	0,30
Citra 10	0,20	0,20	0,20	0,30
Rata-rata	0,40	0,41	0,41	0,38

Hasil uji coba ekstraksi fitur bentuk dengan beberapa nilai orde dapat dilihat pada Tabel 5.4. Dari tiga nilai orde yang diuji, nilai rata-rata *precision* yang paling tinggi diperoleh pada orde 7 dan 8 dengan nilai 0,41. Karena ingin diambil bentuk sedetail mungkin namun performa yang dihasilkan juga baik, maka dipilih orde 8, dimana hasil ekstraksi yang diperoleh lebih detail dari penggunaan orde 7. Pada uji coba selanjutnya, ekstraksi fitur bentuk akan menggunakan orde 8.

5.3.3 Uji Coba 3

Pada uji coba yang ketiga, akan diuji performa dari pencarian citra mirip dengan variasi fitur yang digunakan. Citra contoh yang digunakan adalah beberapa dari citra yang ada dalam *database* pertama dan dapat dilihat pada Gambar 5.3. Citra yang terambil ada yang memang mirip dengan citra contoh, namun tidak berasal dari kain yang sama. Meskipun demikian, citra yang dihitung benar adalah citra yang berasal dari kain yang sama.

Variasi fitur yang akan diuji yaitu:
1. Fitur warna
2. Fitur tekstur
3. Fitur bentuk
4. Fitur warna dan tekstur dengan bobot sama
5. Fitur warna dan bentuk dengan bobot sama
6. Fitur tekstur dan bentuk dengan bobot sama
7. Ketiga fitur dengan bobot sama untuk tiap fitur

Tabel 5.5 Hasil Pengujian Satu Fitur

Nama Citra	*Precision*		
	Fitur warna	Fitur tekstur	Fitur bentuk
Citra 1	1,00	0,50	0,60
Citra 2	1,00	0,90	0,40
Citra 3	0,50	0,90	0,80
Citra 4	1,00	0,50	0,20
Citra 5	0,80	1,00	0,40
Citra 6	1,00	0,40	0,20
Citra 7	1,00	0,80	0,50
Citra 8	1,00	0,70	0,50
Citra 9	0,60	0,70	0,30
Citra 10	1,00	0,70	0,20
Rata-rata	0,89	0,71	0,41

Pada kombinasi beberapa fitur, perhitungan kemiripan dilakukan dengan mengalikan bobot dari fitur dengan fiturnya. Semua fitur terpilih kemudian dijumlah dan akan didapatkan nilai kemiripannya. Dari hasil tersebut, akan diambil 10 nilai paling tinggi sebagai citra hasil. Tampilan hasil uji coba dapat dilihat pada lampiran.

Performa dari pengujian ini dapat dilihat pada Tabel 5.5 dan Tabel 5.6. Performa terbaik bernilai adalah 1 yang diperoleh jika

citra dari kain yang sama terambil semua. Rata-rata performa terbaik dari uji coba ini diperoleh dari kombinasi fitur warna, tekstur, dan bentuk dengan nilai 0,93. Sedangkan pada penggunaan satu fitur, temu kembali citra memiliki performa terbaik jika menggunakan fitur warna dalam mencari citra mirip.

Tabel 5.6 Hasil Pengujian Kombinasi Beberapa Fitur

Nama Citra	*Precision*			
	Warna + Tekstur	Warna + Bentuk	Tekstur + Bentuk	Ketiga Fitur
Citra 1	1,00	1,00	0,90	1,00
Citra 2	1,00	0,90	0,80	1,00
Citra 3	0,50	0,50	0,90	0,80
Citra 4	1,00	1,00	0,50	1,00
Citra 5	0,80	0,80	0,60	1,00
Citra 6	1,00	0,80	0,40	0,90
Citra 7	1,00	1,00	0,70	1,00
Citra 8	1,00	1,00	0,80	1,00
Citra 9	0,80	0,60	0,50	0,70
Citra 10	1,00	1,00	0,50	1,00
Rata-rata	0,91	0,86	0,66	0,94

5.3.4 Uji Coba 4

Pada uji coba keempat akan dicari kombinasi bobot yang dapat menghasilkan performa terbaik. Nilai performa diambil dari nilai *precision* temu kembali citra. Fitur yang digunakan adalah kombinasi tiga fitur, yaitu warna, tekstur, dan bentuk karena memiliki performa terbaik pada Uji Coba 3. Citra contoh yang digunakan adalah beberapa dari citra yang ada dalam *database* dan dapat dilihat pada Gambar 5.3. Citra yang terambil ada yang memang mirip dengan citra contoh, namun tidak berasal dari kain yang sama. Meskipun demikian, citra yang dihitung benar adalah citra yang berasal dari kain yang sama. Variasi bobot tiap fitur yang akan diuji dapat dilihat pada Tabel 5.7. Performa dari pengujian ini dapat dilihat pada Tabel 5.8.

Dari hasil pengujian, didapatkan nilai *precision* terbaik pada variasi bobot ke-3. Fitur tekstur diberi bobot paling besar dan fitur warna diberi bobot lebih besar dari fitur bentuk. Bobot tekstur sesuai hasil uji coba adalah 0,60, bobot warna 0,25, dan bobot bentuk 0,15. Dengan bobot seperti itu, nilai *precision* yang dihasilkan cukup tinggi, yaitu 0,96 dari skala 0 hingga 1. Tampilan hasil uji coba dapat dilihat pada lampiran.

Tabel 5.7 Variasi Bobot pada Kombinasi Tiga Fitur

Nomor Kombinasi	Bobot		
	Warna	Tekstur	Bentuk
1	0,33	0,33	0,34
2	0,60	0,25	0,15
3	0,25	0,60	0,15
4	0,15	0,60	0,25
5	0,50	0,25	0,25
6	0,25	0,50	0,25
7	0,25	0,25	0,50

Tabel 5.8 Hasil Pengujian Variasi Bobot pada Tiga Fitur

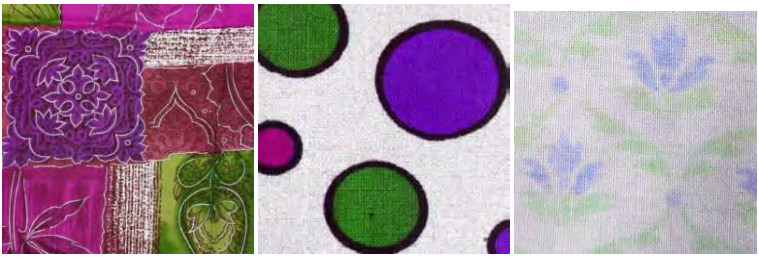
Nama Citra	*Precision* pada variasi bobot ke-						
	1	**2**	**3**	**4**	**5**	**6**	**7**
Citra 1	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Citra 2	1,00	1,00	1,00	0,90	1,00	0,90	0,80
Citra 3	0,80	0,50	0,90	0,90	0,50	0,90	0,90
Citra 4	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Citra 5	1,00	0,80	1,00	1,00	0,80	1,00	1,00
Citra 6	0,90	0,90	0,90	0,80	0,90	0,90	0,60
Citra 7	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Citra 8	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Citra 9	0,70	0,70	0,80	0,70	0,70	0,70	0,60
Citra 10	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Rata-rata	0,94	0,89	0,96	0,93	0,89	0,94	0,89

5.3.5 Uji Coba 5

Uji coba ini bertujuan untuk melihat hasil rekomendasi sistem tanpa dihitung *precision* dan *recall*-nya. *Dataset* yang digunakan pada uji coba ini adalah *dataset* kedua. Pada *dataset* kedua, tidak ada citra yang diambil dari kain yang sama sehingga perhitungan *precision* dan *recall* pun tidak dapat dilakukan.

Pada uji coba ini, akan ditampilkan bagaimana rekomendasi citra mirip dari sistem terhadap suatu citra contoh. Citra yang menjadi citra contoh dapat dilihat pada Gambar 5.5 dengan urutan dari kiri ke kanan. Fitur yang akan diuji adalah sebagai berikut:

1. Fitur warna
2. Fitur tekstur
3. Fitur bentuk
4. Kombinasi ketiga fitur

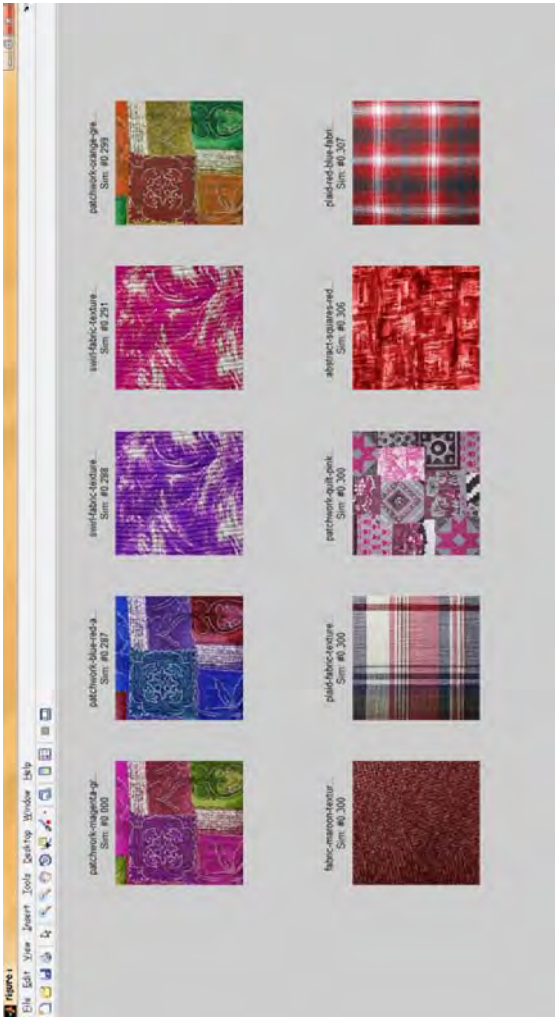


Gambar 5.5 Citra Contoh untuk Uji Coba *Database* Kedua

Dari uji coba fitur warna, citra yang terambil adalah citra yang memiliki kemiripan warna menurut sistem. Karena hasil ekstraksi warna tidak selalu sama dengan pengelihatan manusia, warna citra yang mirip menurut sistem belum tentu sesuai dengan warna yang terlihat oleh mata. Hasil uji coba dapat dilihat pada lampiran.

Pada uji coba fitur tekstur, citra yang terambil adalah yang menurut perhitungan sistem memiliki perulangan pola yang mirip dengan citra contoh. Pada uji coba fitur bentuk, citra yang terambil adalah yang menurut sistem memiliki bentuk objek mirip citra contoh. Proses ekstraksi bentuk dilakukan setelah citra

diubah menjadi citra hitam putih. Karena itu, bentuk yang diambil bisa jadi berbeda dari yang terlihat pada citra berwarna.



Gambar 5.6 Hasil uji coba kombinasi tiga fitur pada *dataset* kedua dengan citra contoh pertama.

Uji coba untuk kombinasi tiga fitur menggunakan bobot terbaik sesuai Uji Coba 4. Bobot fitur warna adalah 0,25, bobot fitur tekstur adalah 0,6, dan bobot fitur bentuk 0,15. Hasil temu kembali citra berdasarkan ketiga fitur ini dapat dilihat pada Gambar 5.6.

BAB VI
PENUTUP

Pada bab ini akan dibahas mengenai kesimpulan yang dapat diambil dari tujuan pembuatan perangkat lunak, serta hasil uji coba yang telah dilakukan. Selain itu terdapat beberapa saran untuk pengembangan lebih lanjut.

6.1 **Kesimpulan**

Dari hasil pengamatan selama perancangan, implementasi, dan uji coba perangkat lunak, dapat diambil kesimpulan sebagai berikut:

1. Pada ekstraksi fitur tekstur dengan *steerable filter* diperlukan beberapa nilai sudut untuk merepresentasikan fitur tekstur. Namun banyak sudut dan nilainya tidak mempengaruhi hasil ekstraksi fitur.
2. Pada ekstraksi fitur bentuk dengan *Zernike moment*, diperlukan nilai orde. Nilai orde yang menghasilkan performa terbaik untuk *dataset* yang berisi 280 citra berukuran 400x400 piksel adalah 7 atau 8.
3. Hasil pencarian citra mirip yang paling baik didapatkan dengan menggabungkan ketiga fitur. Sedangkan bila hanya digunakan satu fitur, yang paling baik digunakan untuk pencarian adalah fitur warna.
4. Besar bobot untuk kombinasi tiga fitur citra adalah 0,25 untuk fitur warna, 0,6 untuk fitur tekstur, dan 0,15 untuk fitur bentuk.

6.2 **Saran**

Berikut merupakan beberapa saran oleh penulis untuk pengembangan aplikasi di masa yang akan datang, berdasarkan hasil perancangan, implementasi, dan uji coba yang telah dilakukan:

1. Perangkat lunak ini dapat dikembangkan untuk memberi rekomendasi citra kain yang mirip berdasarkan fitur warna, tekstur, dan bentuk dengan *dataset* yang lebih banyak.
2. Kemampuan perangkat lunak ini masih kurang baik. Hal ini mungkin disebabkan oleh metode ekstraksi fitur yang kurang tepat, pemilihan *dataset* yang kurang baik, dan perhitungan jarak yang kurang tepat. Diharapkan pada pengembangan selanjutnya dapat dibangun perangkat lunak yang memberikan hasil lebih baik.

DAFTAR PUSTAKA

[1] X. Y. Wang, Y. J. Yu dan H. Y. Yang, "An effective image retrieval scheme using color, texture, and shape features," *Computer Standards and Interfaces,* vol. 33, pp. 59-68, 2011.

[2] B. S. Manjunath, V. V. Vasudevan dan A. Yamada, "Color and Texture Descriptors," *IEEE Transactions on Circuits and Systems for Video Tecnology,* vol. 11, no. 6, pp. 703-714, 2001.

[3] C. Solomon dan T. Breckon, Fundamentals of Digital Image Processing, Oxford: Wiley-Blaackwell, 2011.

[4] N. C. Yang, W. H. Chang, C. M. Kuo dan T. H. Li, "A fast MPEG-7 dominant color extraction with new similarity measure for image retrieval," *Journal of Visual Communication and Image Representation,* vol. 19, pp. 92-105, 2007.

[5] W. T. Freeman dan E. H. Adelson, "The Design and Use of Steerable Filters," *Transactional on Pattern Analysis and Machine Intelligence,* vol. 13, pp. 891-906, 1991.

[6] A. Khotanzad dan Y. H. Hong, "Invariant Image Recognition by Zernike Moments," *Transactional on Pattern Analysis and Machine Intelligence,* vol. 12, pp. 489-497, 1990.

[7] M. Oussalah, "Content Based Image Retrieval: Review of State of Art and Future Directions," dalam *IEEE First Workshops on Image Processing Theory, Tools and Applications*, Sousse, Tunisia, 2008.

[8] J. Pang, *steerable filters,* Tufts University, 2013.

[9] P. Fricker, "Pseudo-Zernike functions," MathWorks, Inc., 2011.

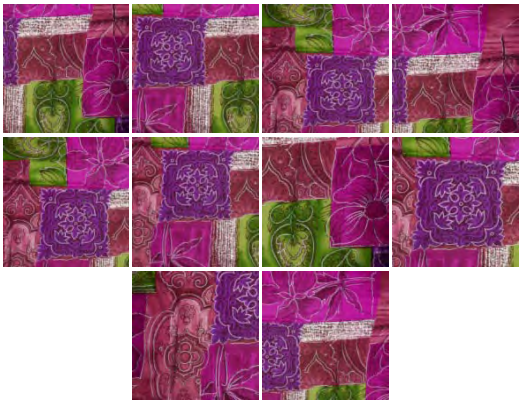
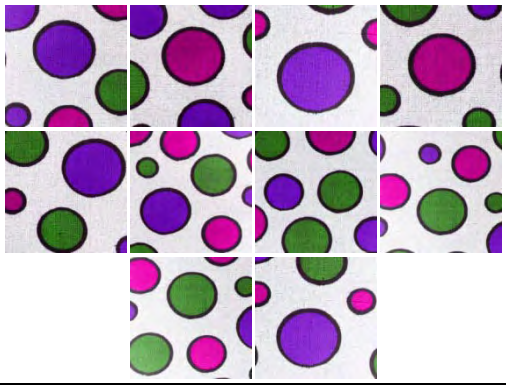
BIODATA PENULIS

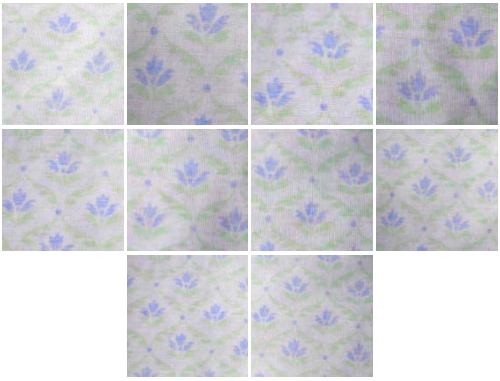
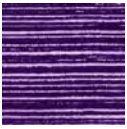
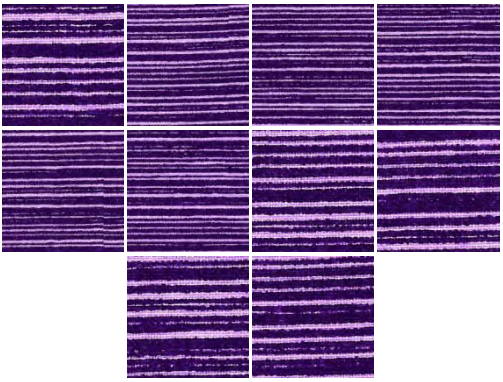
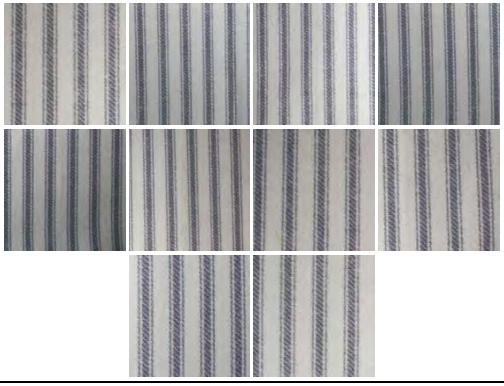
ATIQOTUN NISWAH, lahir di Gresik pada tanggal 17 Mei 1992 dan dibesarkan di Surabaya. Penulis merupakan anak pertama dari 5 bersaudara. Penulis menempuh pendidikan SD hingga SMP di Khadijah Surabaya (1998-2007), SMAN 5 Surabaya (2007-2010), dan terakhir sebagai mahasiswa Teknik Informatika ITS (2010-2014). Bidang Studi yang diambil oleh penulis pada saat kuliah di Teknik Informatika ITS adalah Komputasi Cerdas dan Visualisasi. Selama menempuh kuliah, penulis aktif sebagai staf dan pengurus harian Himpunan Mahasiswa Teknik Computer (HMTC), staf Keluarga Muslim Informatika (KMI), dan staf Jamaah Masjid Manarul Ilmi (JMMI) ITS. Penulis dapat dihubungi melalui e-mail: atiqotun@gmail.com.

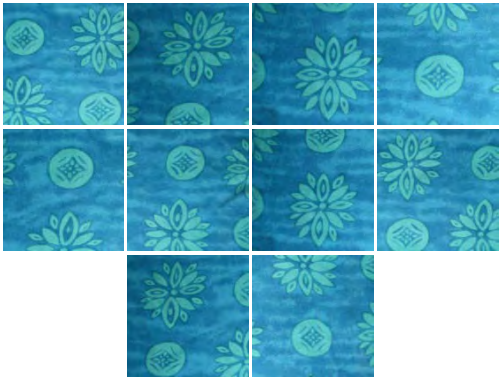
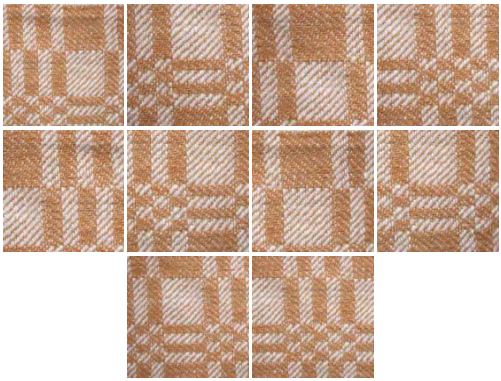
LAMPIRAN

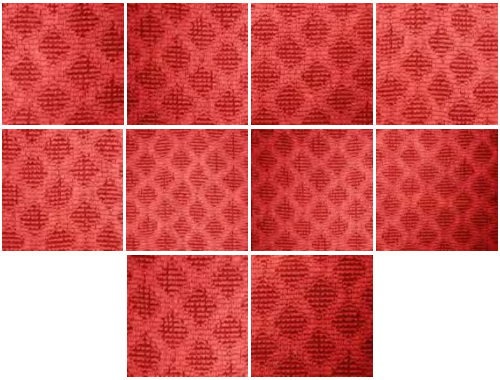
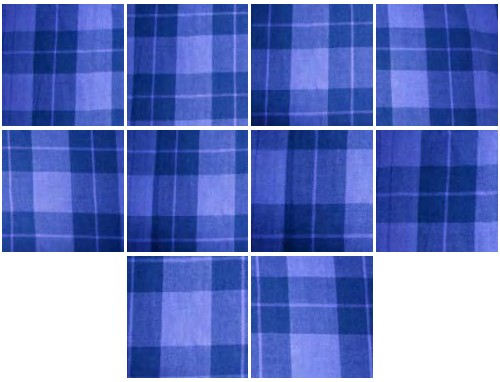
A. Gambar *Dataset* Kain

Tabel A.1 Citra *Ground Truth* untuk Uji Coba *Dataset* Pertama

No.	Citra Contoh	Citra *Ground Truth*
1.		
2.		

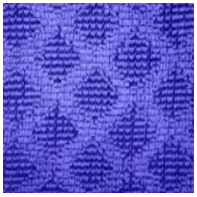
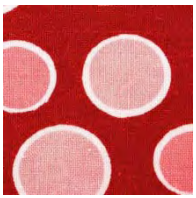
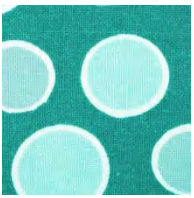
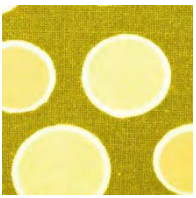
No.	Citra Contoh	Citra *Ground Truth*
3.		
4.		
5.		

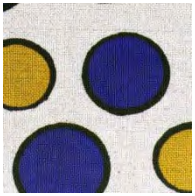
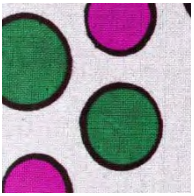
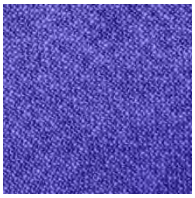
No.	Citra Contoh	Citra *Ground Truth*
6.		
7.		
8.		

No.	Citra Contoh	Citra *Ground Truth*
9.		
10.		

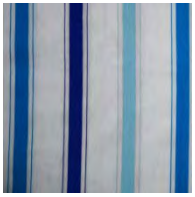
Tabel A.2 Beberapa Citra Kain pada *Dataset* Kedua

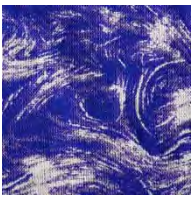
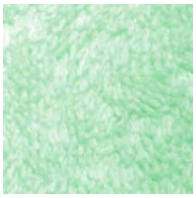
No.	Variasi dalam Satu Jenis Kain
1.	

No.	Variasi dalam Satu Jenis Kain		
2.			
3.			
4.			
5.			
6.			

No.	Variasi dalam Satu Jenis Kain		
7.			
8.			
9.			
10.			
11.			

No.	Variasi dalam Satu Jenis Kain		
12.			
13.			
14.			
15.			
16.			

No.	Variasi dalam Satu Jenis Kain		
17.			
18.			
19.			
20.			
21.			

No.	Variasi dalam Satu Jenis Kain		
22.			
23.			
24.			
25.			

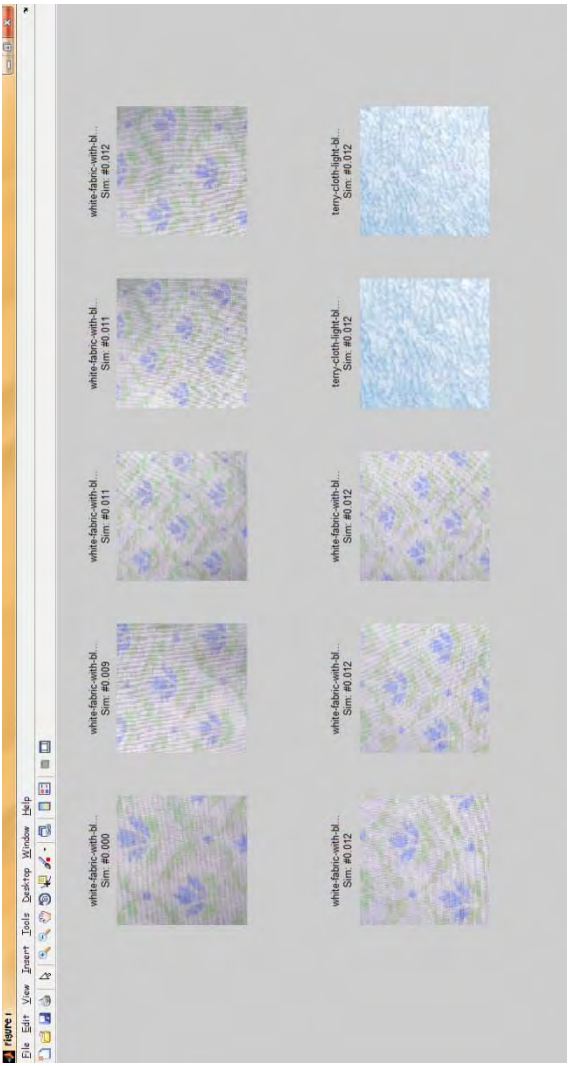
B. Hasil Uji Coba

Hasil uji coba 1

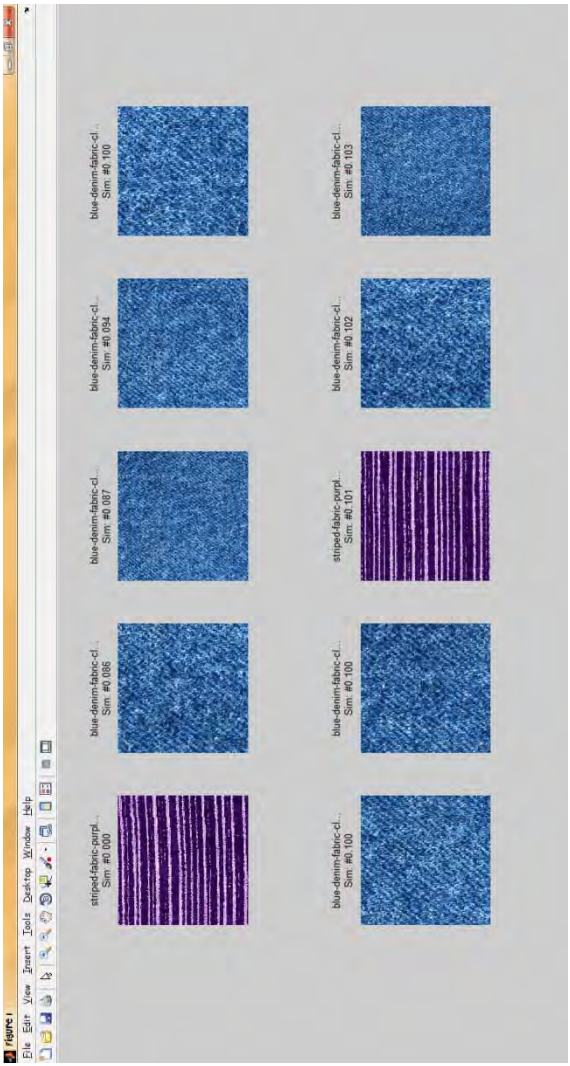
Gambar B.1 Tampilan Keluaran dari Citra Contoh ke-5 dan Kombinasi Sudut 1

Gambar B.2 Tampilan Keluaran dari Citra Contoh ke-6 dan Kombinasi Sudut 1

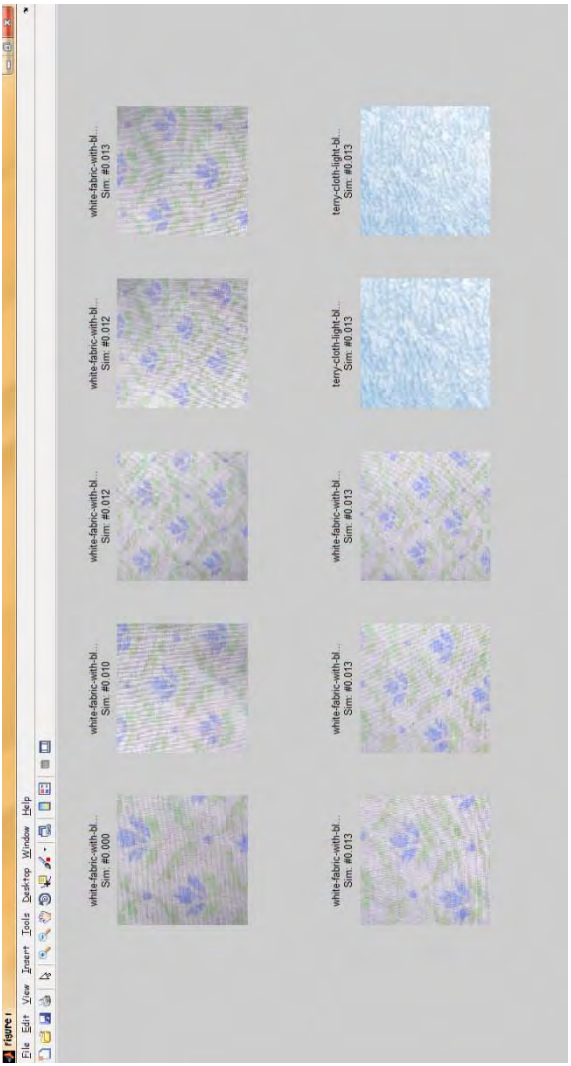
Hasil uji coba 2



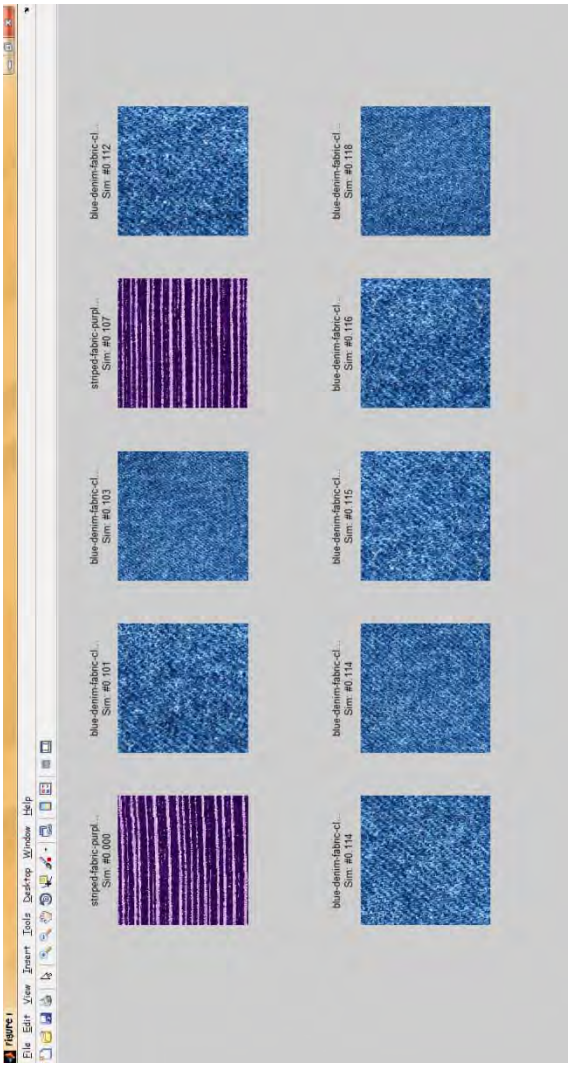
Gambar B.3 Tampilan Keluaran dari Citra Contoh ke-3 dan Orde *Zernike Moment* 8



Gambar B.4 Tampilan Keluaran dari Citra Contoh ke-4 dan Orde *Zernike Moment* 8



Gambar B.5 Tampilan Keluaran dari Citra Contoh ke-3 dan Orde *Zernike Moment* 9



Gambar B.6 Tampilan Keluaran dari Citra Contoh ke-4 dan Orde *Zernike Moment* 9

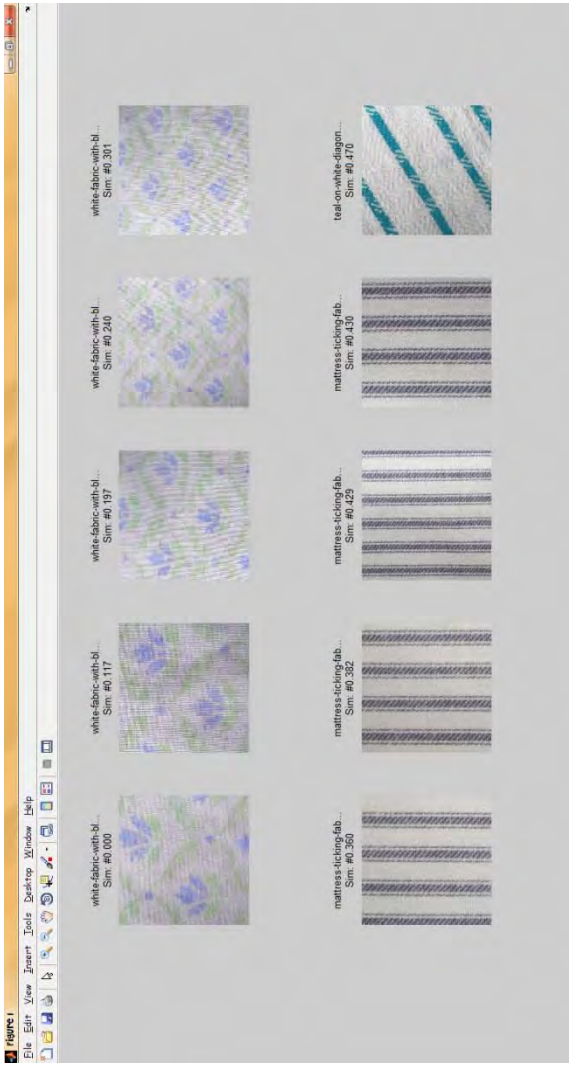
Hasil uji coba 3

Gambar B.7 Tampilan Keluaran dari Citra Contoh ke-4 Berdasarkan Kombinasi Tiga Fitur

Gambar B.8 Tampilan Keluaran dari Citra Contoh ke-9 Berdasarkan Kombinasi Tiga Fitur

Hasil uji coba 4

Gambar B.9 Tampilan Keluaran dari Citra Contoh ke-2 dengan Variasi Bobot ke-2

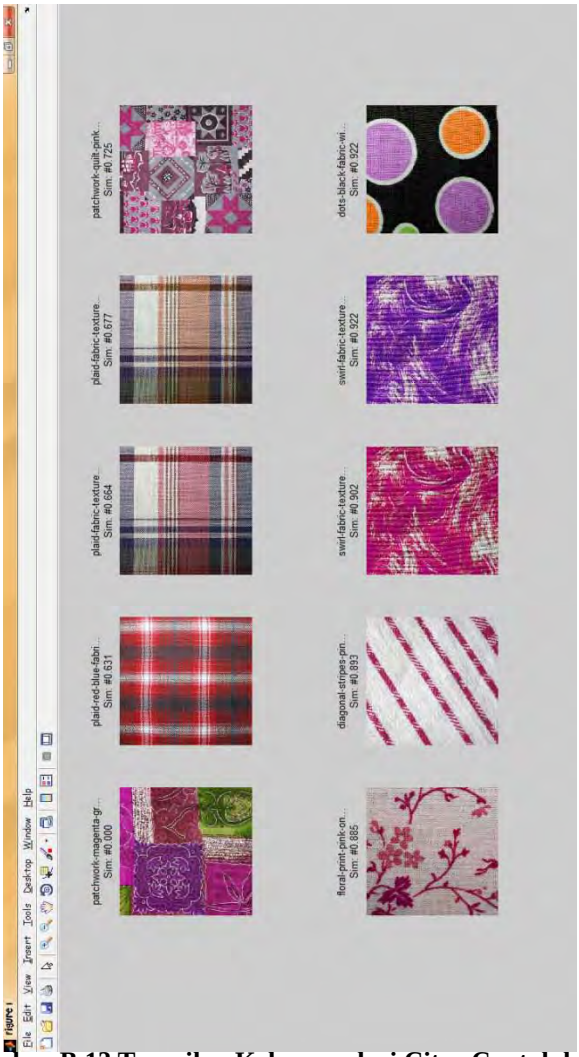


Gambar B.10 Tampilan Keluaran dari Citra Contoh ke-3 dengan Variasi Bobot ke-2

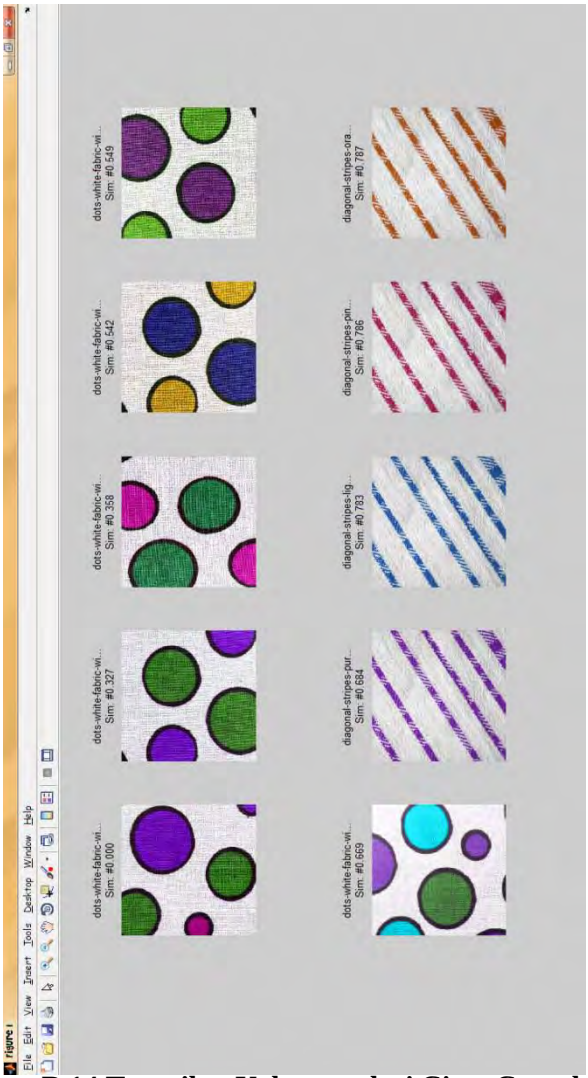
Gambar B.11 Tampilan Keluaran dari Citra Contoh ke-2 dengan Variasi Bobot ke-3

Gambar B.12 Tampilan Keluaran dari Citra Contoh ke-9 dengan Variasi Bobot ke-3

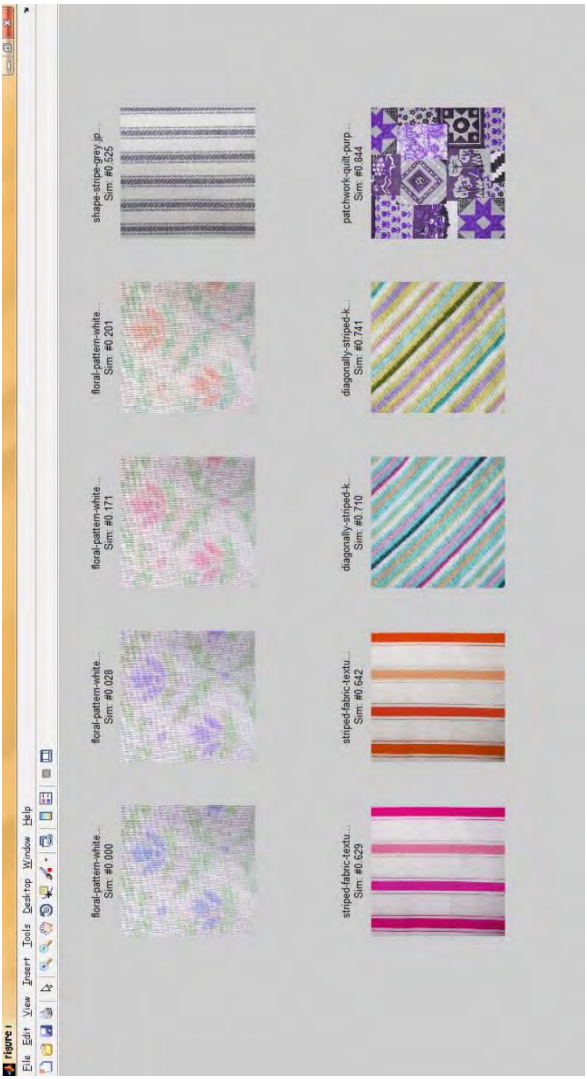
Hasil uji coba 5



Gambar B.13 Tampilan Keluaran dari Citra Contoh ke-1 Berdasarkan Fitur Warna



Gambar B.14 Tampilan Keluaran dari Citra Contoh ke-2 Berdasarkan Fitur Warna



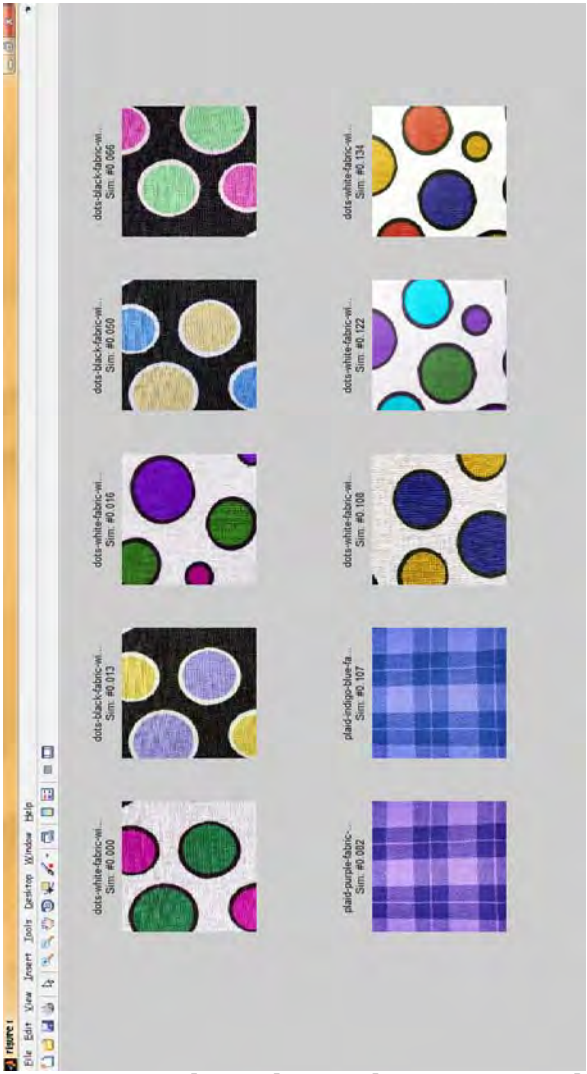
Gambar B.15 Tampilan Keluaran dari Citra Contoh ke-3 Berdasarkan Fitur Warna

Gambar B.16 Tampilan Keluaran dari Citra Contoh ke-1 Berdasarkan Fitur Tekstur

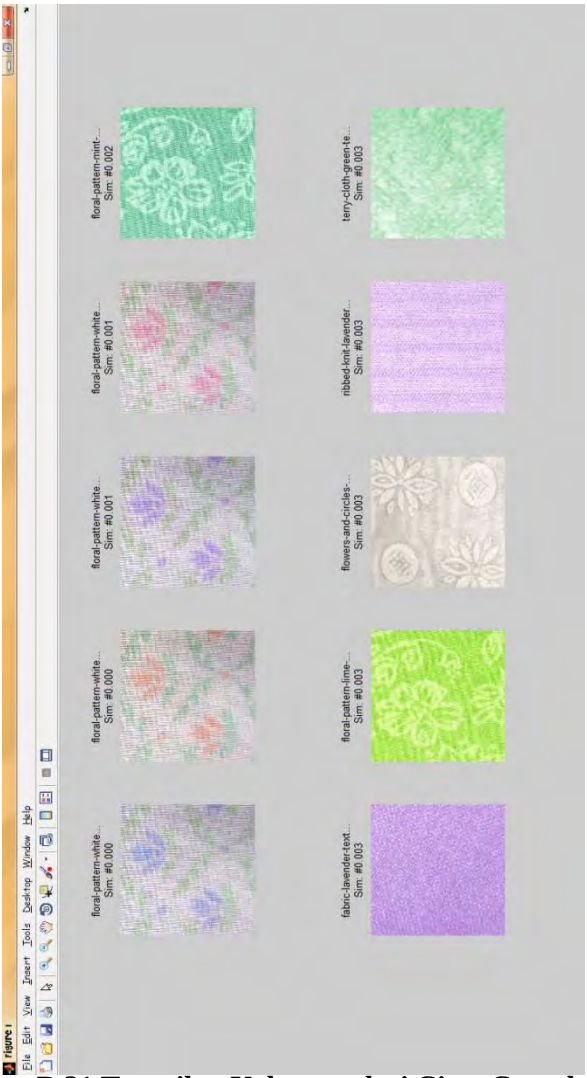
Gambar B.17 Tampilan Keluaran dari Citra Contoh ke-2 Berdasarkan Fitur Tekstur

Gambar B.18 Tampilan Keluaran dari Citra Contoh ke-3 Berdasarkan Fitur Tekstur

Gambar B.19 Tampilan Keluaran dari Citra Contoh ke-1 Berdasarkan Fitur Bentuk



Gambar B.20 Tampilan Keluaran dari Citra Contoh ke-2 Berdasarkan Fitur Bentuk



Gambar B.21 Tampilan Keluaran dari Citra Contoh ke-3 Berdasarkan Fitur Bentuk

Gambar B.22 Tampilan Keluaran dari Citra Contoh ke-2 Berdasarkan Kombinasi Tiga Fitur

Gambar B.23 Tampilan Keluaran dari Citra Contoh ke-3 Berdasarkan Kombinasi Tiga Fitur