repository.ub.ac.id

PEROLEHAN INFORMASI RATING BUKU BERDASARKAN GAMBAR SAMPUL BUKU MENGGUNAKAN METODE *SCALE-INVARIANT FEATURE TRANSFORM*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Hamim Fathul Aziz
NIM: 145150200111066

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

UNIVERSITAS BRAWIJAYA

PENGESAHAN

PEROLEHAN INFORMASI RATING BUKU BERDASARKAN GAMBAR SAMPUL BUKU MENGGUNAKAN METODE *SCALE-INVARIANT FEATURE TRANSFORM*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Hamim Fathul Aziz
NIM: 145150200111066

Skripsi ini telah diuji dan dinyatakan lulus pada
2 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Yuita Arum Sari, S.Kom., M.Kom	Randy Cahya Wihandika, S.ST., M.Kom
NIK: 2016098807152001	NIK: 201405 8802061001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

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

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

Malang, 16 Juli 2018

Hamim Fathul Aziz

NIM: 145150200111066

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul "Perolehan Informasi Rating Buku Berdasarkan Gambar Sampul Buku Menggunakan Metode *Scale-Invariant Feature Transform*". Penulis menyampaikan terima kasih kepada pihak-pihak yang telah membantu dan membimbing penulis selama pengerjaan skripsi ini sedari awal hingga penulis dapat menyelesaikannya dengan baik, yaitu di antaranya:

1. Ibu Yuita Arum Sari, S.Kom., M.Kom., selaku dosen pembimbing I yang telah memberikan bimbingan, arahan, ilmu, dan saran dalam penyelesaian skripsi ini.
2. Bapak Randy Cahya Wihandika, S.ST., M.Kom., selaku dosen pembimbing 2 yang telah membimbing, memberikan ilmu, saran dan juga arahan selama penyusunan skripsi ini.
3. Bapak Wayan Firdaus Mahmudy, S.Si, M.Kom., Bapak Heru Nurwasito, Ir., M.Kom., Bapak Suprapto, S.T., M.T., Drs., M.T., Bapak Edy Santoso, S.Si, M.Kom., selaku Dekan, Wakil Dekan I, Wakil Dekan II, Wakil Dekan III Fakultas Ilmu Komputer Universitas Brawijaya Malang.
4. Dosen Fakultas Ilmu Komputer yang telah mendidik dan memberikan ilmu selama penulis menempuh pendidikan di Fakultas ini.
5. Orang tua dan saudara penulis yang selalu mendukung penulis baik secara moril dan materiil sedari awal menempuh pendidikan hingga penulis dapat menyelesaikan skripsi ini.
6. Seluruh keluarga besar kedua orang tua penulis yang selalu memberikan semangat dan memberikan bantuan selama ini.
7. Seluruh keluarga kontrakan penulis yang selalu mengingatkan, memberikan semangat dan membantu penulis dalam menyelesaikan skripsi ini.
8. Sahabat-sahabat saya Nurina Savanti Widya Gotami, Yane Marita Febrianti, San Sayyidul Akdam Augusta, Dwi Wahyu P.L, Alqis Rausanfita yang telah banyak membantu dan mendukung penulis dalam penyelesaian skripsi ini.
9. Sahabat-sahabat seperjuangan kepanitiaan Andre Rino Prasetyo, Fakhrudin Farid, Rizky Haris Risaldi, Robih Dini, Yunita Dwi yang telah memberikan banyak semangat, masukan, kritik, dan saran.
9. Serta semua pihak yang telah membantu dalam proses penyelesaian skripsi ini yang tak dapat penulis sebutkan satu persatu.

Dalam penulisan skripsi ini, penulis menyadari terdapat banyak kekurangan serta kesalahan, sehingga penulis mengharapkan adanya kritik dan saran yang

membangun dari berbagai pihak untuk memperbaiki diri. Penulis juga berharap agar skripsi ini dapat memberikan manfaat.

Malang, 16 Juli 2018

Hamim Fathul Aziz
hamimaziz115@gmail.com

repository.ub.ac.id

ABSTRAK

Hamim Fathul Aziz, Perolehan Informasi Rating Buku Berdasarkan Gambar Sampul Buku Menggunakan Metode *Scale-Invariant Feature Transform*

Pembimbing: Yuita Arum Sari, S.Kom., M.Kom. dan Randy Cahya Wihandika, S.ST., M.Kom.

Teknologi yang semakin berkembang diharapkan dapat lebih memudahkan untuk mengakses informasi-informasi yang terdapat pada media *online* tersebut. Seperti contoh mencari rating buku secara otomatis di media *online* dengan memanfaatkan sampul buku. Dengan adanya sebuah sistem yang dapat mencarikan rating dari buku dengan menggunakan gambar dari sampul yang diperoleh dari kamera ponsel, diharapkan akan lebih memudahkan dan mempercepat perolehan informasi rating dari buku, agar tidak salah dalam membeli sebuah buku. Berdasarkan penjelasan tersebut, penelitian ini dilakukan menggunakan *scale invariant feature transform* untuk mengenali objek buku pada sebuah gambar. Sebelum dilakukan pencarian gambar buku yang sesuai, gambar tersebut dilakukan *preprocessing* dengan mencari *scale space,* pencarian *letak keypoint* atau *keypoint localization,* perhitungan sudut piksel-pikselnya atau *orientation assignment,* dan terakhir pembentukan descriptor gambar atau *keypoint descriptor*. Pada penelitian ini gambar akan diuji pengaruhnya terhadap intensitas cahaya, rotasi gambar, dan *scaling*. Hasil pengujian pencocokan gambar sampul buku menggunakan metode *scale invariant feature transform* mempunyai akurasi yang tinggi pada kondisi dengan intensitas cahaya yang terang dan mempunyai akurasi yang rendah pada saat dilakukan rotasi gambar dan *scaling* gambar. Rata-rata akurasi yang didapatkan pada kondisi cahaya terang, rotasi, dan *scaling* masing-masing adalah 90%, 57,5% dan 46,6%.

Kata kunci: *Computer Vision,* Goodreads API, *CBIR*, *keypoint*, sampul buku

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

ABSTRACT

Hamim Fathul Aziz, *Retriecal of* Rating *Book's Information by Cover Book's Image Using Scale-Invariant Feature Transform Method*

***Advisor*: Yuita Arum Sari, S.Kom., M.Kom. *and* Randy Cahya Wihandika, S.ST., M.Kom.**

Along with the development of technology, almost kinds of all information are available on media online. The expanding technology are expected to make it easier to access all kinds of information in media online. As an example by searching book's rating automatically in media online by utilizing book cover. With the existence of a system that can find the book's rating by using the image on the cover getting from camera phone hopefully can make it easier and make it more fast to get rating information from the book, so it can make the customer do less mistaken when buying a book. Based on explanation above this research will uses scale-invariant feature transform to recognize the book's object in an image. Before find the appropriate image of the book; First, preprocessing will be doing on the image by searching for scale space; Then, find the key point or key point localization; Next, by calculation of pixel's angle or orientation assignment; Finally, transforming of the image descriptor or key point descriptor. On this research the image will be tested of this effect on light intensity, image rotation, and scaling. The result by matching test the image of book's cover using scale-invariant feature transform method has high accuracy in condition of bright light intensity and it has low accuracy when using image rotation and image scaling. The average accuracy can obtain in bright light conditions, rotation, and scaling are 90%, 57.5%, and 46.6% respectively.

Keywords: *Computer Vision,* Goodreads API, *CBIR*, *keypoint*, *Books Cover*

UNIVERSITAS BRAWIJAYA

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
ABSTRAK vi
DAFTAR ISI viii
DAFTAR GAMBAR xi
DAFTAR TABEL xii
DAFTAR LAMPIRAN xiii
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Tujuan 3
1.4 Manfaat 3
1.5 Batasan Masalah 3
1.6 Sistematika Pembahasan 3
BAB 2 LANDASAN KEPUSTAKAAN 5
2.1 Kajian Kepustakaan 5
2.2 Citra Digital 8
2.3 *Scale Invariant Feature Transform* 8
2.3.1 *Scale Space Extrema Detection* 9
2.3.2 *Keypoint localization* 13
2.3.3 *Orientation Assignment* 14
2.3.4 *Keypoint Descriptor* 15
BAB 3 METODOLOGI 16
3.1 Tipe Penelitian 16
3.2 Strategi Penelitian 16
3.3 Lokasi Penelitian 16
3.4 Pengumpulan Data 16
3.5 Perancangan Algoritme 17
3.6 Teknik Pengujian dan Analisis 17
3.7 Penarikan kesimpulan dan saran 17

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

BAB 4 PERANCANGAN 18
4.1 Perancangan Data 18
4.2 Perancangan Algoritme 18
4.2.1 Scale Space Extrema Detection 19
4.2.2 *Keypoint Localization* 24
4.2.3 *Orientation Assignment* 33
4.2.4 *Keypoint Descriptor* 35
4.2.5 Proses *Matching* 38
4.3 Perancangan Pengujian 39
4.3.1 Perancangan Pengujian Pengaruh Pencahayaan 39
4.3.2 Perancangan Pengujian Data Gambar dengan Rotasi 40
4.3.4 Perancangan Pengujian Data Gambar dengan Skala Berbeda-beda 41
BAB 5 IMPLEMENTASI 42
5.1 Deskripsi Lingkungan Implementasi 42
5.2 Implementasi Sistem 42
5.2.1 Implementasi Pengambilan Data *Input* 42
5.2.2 Implementasi Proses *Scale Space Extrema Detection* 43
5.2.3 Implementasi Proses *Keypoint Localization* 46
5.2.4 Implementasi Proses *Orientation Assignment* 54
5.2.5 Implementasi Proses *Keypoint Descriptor* 55
5.2.6 Implementasi Proses *Matching* 58
BAB 6 PENGUJIAN 60
6.1 Pengujian Pengaruh Pencahayaan 60
6.1.1 Pencahayaan Kurang 60
6.1.2 Pencahayaan Terang 62
6.2 Pengujian Data Gambar dengan Rotasi 63
6.2.1 Rotasi dengan Sudut Ekstrem 64
6.2.2 Rotasi dengan Selain Sudut Ekstrem 65
6.3 Pengujian Data Gambar dengan Perbedaan Skala 66
6.3.1 Jarak 15 Cm 66
6.3.2 Jarak 30 Cm 67
6.3.3 Jarak 10 Cm 68
BAB 7 PENUTUP 70

7.1 Kesimpulan 70
7.2 Saran 70
Daftar Pustaka 71
LAMPIRAN A DATA LATIH 72
LAMPIRAN B DATA UJI 74
B.1 Data uji pencahayaan kurang 74
B.2 Data uji pencahayaan terang 74
B.3 Data uji rotasi ekstrem 75
B.4 Data uji non ektrem 75
B.5 Data uji skala 8 cm 76
B.6 Data uji skala 15 cm 76
B.7 Data uji skala 30 cm 77

DAFTAR GAMBAR

Gambar 2.1 Contoh hasil blur dan Pengecilan skala 11
Gambar 2.2 Ilustrasi DoG 12
Gambar 2.3 Contoh hasil proses *DoG* 12
Gambar 2.4 Mencari *local extrema* (Lowe, 2004) 13
Gambar 2.5 Ilustrasi orientasi *gradient* (Lowe, 2004) 15
Gambar 4.1 Diagram alir tahapan proses algoritme 19
Gambar 4.2 Diagram alir tahapan proses *Scale Space Extrema Detection* 20
Gambar 4.3 Alur Proses *Resize Image* 20
Gambar 4.4 Contoh hasil *resize image* 21
Gambar 4.5 Alur Proses *Convert Grayscale* 22
Gambar 4.6 Contoh hasil proses *grayscale* 22
Gambar 4.7 Alur Proses *Gaussian Filter* 23
Gambar 4.8 Sebelum *Gaussian Filter* 24
Gambar 4.9 Sesudah *Gaussian Filter* 24
Gambar 4.10 Diagram alir tahapan proses *Keypoint Localization* 25
Gambar 4.11 Alur Proses *Difference of Gaussian* (DoG) 26
Gambar 4.12 Matriks nilai piksel gambar *Gaussian level 1* 26
Gambar 4.13 Matriks nilai piksel gambar *Gaussian level 2* 27
Gambar 4.14 Hasil *DoG* 27
Gambar 4.15 Contoh hasil proses DoG 27
Gambar 4.16 Alur Proses *Local Maxima* 28
Gambar 4.17 Nilai piksel tetangga *DoG1* 28
Gambar 4.18 Nilai piksel tetangga *DoG2* 29
Gambar 4.19 Nilai piksel tetangga *DoG3* 29
Gambar 4.20 Nilai piksel tetangga DoG3 29
Gambar 4.21 Nilai piksel tetangga DoG3 29
Gambar 4.22 Nilai piksel tetangga DoG3 30
Gambar 4.23 Alur Proses *Local Minima* 30
Gambar 4.24 Alur Proses *Find R* 31
Gambar 4.25 Nilai piksel turunan terhadap x 32
Gambar 4.26 Nilai piksel turunan terhadap x 32
Gambar 4.27 Hasil proses *find* r 33
Gambar 4.28 Alur Proses *Orientation Assignment* 34
Gambar 4.29 Alur Proses Pembentukan *Keypoint Descriptor* 35
Gambar 4.30 Alur Proses Pencarian *Index* x dan *Index* y 36
Gambar 4.31 Alur Proses Pencarian *Index Histogram* 37
Gambar 4.32 Alur Proses *Matching* 38
Gambar 6.1 Piksel Pembentukan *Histogram Keypoint* Data Latih 66
Gambar 6.2 Piksel Pembentukan *Histogram Keypoint* Data Test dengan Rotasi 66

DAFTAR TABEL

Tabel 4.1 Peracangan pengujian pengaruh pencahayaan dalam ruang 40
Tabel 4.2 Peracangan pengujian pengaruh rotasi gambar 40
Tabel 4.3 Peracangan pengujian pengaruh skala 41
Tabel 6.1 Pencahayaan Terang 60
Tabel 6.2 Pencahayaan kurang 61
Tabel 6.3 Pencahayaan Terang 62
Tabel 6.4 Pencahayaan Kurang 63
Tabel 6.5 Pengujian Pengaruh Rotasi Gambar dengan Sudut Ekstrem 64
Tabel 6.6 Pengujian Pengaruh Rotasi Gambar 65
Tabel 6.7 Pengujian Pengaruh Perbedaan Skala pada Jarak 15cm 66
Tabel 6.8 Pengujian Pengaruh Perbedaan Skala pada Jarak 30cm 67
Tabel 6.9 Pengujian Pengaruh Perbedaan Skala pada Jarak 30cm 68

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

DAFTAR LAMPIRAN

LAMPIRAN A DATA LATIH 72
LAMPIRAN B DATA UJI 74
B.1 Data uji pencahayaan kurang 74
B.2 Data uji pencahayaan terang 74
B.3 Data uji rotasi ekstrem 75
B.4 Data uji non ektrem 75
B.5 Data uji skala 8 cm 76
B.6 Data uji skala 15 cm 76
B.7 Data uji skala 30 cm 77

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Buku adalah jendela dari segala ilmu pengetahuan, melalui buku kita dapat mendapatkan ilmu pengetahuan, informasi, hiburan dan lain-lain. Pada jaman sekarang ini keberadaan buku sudah sangat melimpah, sehingga memaksa calon pembaca untuk dapat selektif memilih buku yang benar-benar berkualitas. Untuk bisa mendapatkan buku yang berkualitas tersebut bisa dilakukan dengan berbagai cara, seperti bertanya kepada teman maupun mencari informasi *review* dan pada internet. Namun semua itu akan memakan waktu dan tenaga apabila dilakukan secara manual dan akan sangat membantu apabila ada sistem yang membantu mencari informasi tersebut secara otomatis.

Sistem pencarian informasi secara otomatis atau dikenal dengan istilah *information retrieval* adalah sistem yang berguna untuk mencari informasi yang relevan sesuai dengan *query* yang dimasukkan. *Information retrieval* dapat memanfaatkan teks maupun gambar. Untuk mencari informasi dari sebuah buku dapat menggunakan judulnya maupun gambar sampul. Seiring dengan sangat banyaknya konten teks bahasa alami, termasuk halaman web, artikel berita, literatur ilmiah banyaknya buku yang bisa jadi memiliki judul yang hampir sama akan sedikit menyulitkan untuk mencari informasi dari buku yang tepat (Zhai, 2015). Namun apabila penarikan informasi di internet dilakukan dengan mencari berdasarkan sampul buku akan menjadi lebih akurat karena sampul buku akan berbeda meskipun dengan judul sama. *Website* maupun *blogspot* yang menyediakan informasi dari buku juga akan sangat banyak di internet yang dapat membingungkan saat memilihnya. Hal ini dapat disiasati dengan memfokuskan pada *website* tertentu yang memang fokus pada *review* buku, sebagai contoh Goodreads.

Dalam hal pencarian informasi berdasarkan media gambar lebih dikenal dengan istilah *Content Based Information Retrieval* (CBIR). CBIR menggunakan media gambar untuk mencari informasi yang relevan atau yang berhubungan dengan gambar tersebut. CBIR telah banyak diaplikasikan dalam kehidupan sehari-hari sebagai contoh identifikasi *finger print, biodiversity information systems,* perpustakaan digital, pencegahan tindakan kriminal, dalam bidang kesehatan, penelitian tentang sejarah, dan lain-lain (Tores & Falcao, 2006).

Sistem CBIR memanfatkan fitur-fitur seperti warna, tekstur, *keypoint* dan sebagainya untuk mencari gambar atau informasi yang relevan di *database*. Ada banyak sekali algoritme yang telah dikembangkan yang mempunyai tujuan untuk mencari fitur-fitur dari gambar, beberapa diantaranya adalah Harris *Corner Detection,*

Hessian, *affine-infariant, Laplacian of Gaussian (LoG)* dan *Difference of Gaussian (DoG),* SIFT, SURF, dan lain-lain

Penelitian tentang CBIR juga telah banyak dilakukan dan belakangan yang paling berkembang adalah menggunakan algoritme SIFT. Algoritme *Scale Invariant Feature Transform* (SIFT) sangat berguna untuk mendeskripsikan konten dan fitur dari sebuah gambar (Reghava & Muhammad, 2016). Dengan memanfaatkan algoritme SIFT akan didapatkan banyak sekali fitur dari gambar berupa *keypoint*. Banyak sistem CBIR yang menggunakan algoritme SIFT karena algoritme ini mempunyai kelebihan ketahan terhadap perubahan skala, rotasi gambar, dan juga perbedaan pencahayaan (Lowe, 2004).

Algoritme SIFT telah diterapkan dalam banyak hal seperti pengenalan objek, mengenali panorama, pembangunan model 3 dimensi, dan lain-lain (Almeida, et al., 2009). Algoritme SIFT akan mencari deskriptor dari sebuah gambar dan merepresentasikannya dalam vektor *keypoint*. *Keypoint* yang dihasilkan oleh algoritme SIFT akan dibagi menjadi 8 histogram arah yang diperoleh dari 16 x 16 piksel tetangga yang dibagi menjadi 4 x 4. Untuk pencocokan dengan deskriptor gambar lain dilakukan dengan cara menghitung kemiripan dari *keypoint-keypoint* yang dimilikinya (Reghava & Muhammad, 2016).

Penelitian tentang SIFT telah banyak dilakukan sebagai contoh penelitian *SIFT applied to CBIR* oleh Jurandy Almeida, Ricardo da S. Torres and Siome Goldenstein. Penelitian ini berfokus untuk mengambil gambar yang serupa pada *dataset*. Pada penelitian tersebut memperoleh hasil yang sangat baik apabila hanya terdapat satu objek pada gambar yang akan diambil.

Penelitian lain tentang CBIR menggunakan SIFT adalah penelitian dari *A Comparative Study of Sift and PCA for Content Based Image Retrieval* oleh Raghava Reddy K dan Dr. M. Narayana. Penelitian tersebut membandingkan kinerja dari algoritme SIFT dan PCA dalam hal CBIR. Hasil dari penelitian tersebut didapatkan bahwa SIFT bekerja lebih baik dalam CBIR dari pada PCA. Hal ini dikarenakan deskriptor yang dihasilkan oleh SIFT lebih tahan terhadan *scale*, *rotasi*, dan iluminasi. Sedangkan PCA akan lebih baik apabila bekerja dengan hal warna.

Berdasarkan permasalahan dan alasan yang telah dipaparkan diatas, dan didukung oleh beberapa penelitian yang sudah dilakukan, maka pada penelitian ini dibangun sebuah sistem CBIR untuk mendapatkan rating dari buku yang ada pada goodreads berdasarkan dari sampul buku tersebut. Dengan demikian sistem yang akan dibangun ini diharapkan dapat membantu seoptimal mungkin terhadap permasalahan pencarian rating buku yang berkualitas.

1.2 Rumusan Masalah

Berdasarkan dari latar belakang yang telah dipaparkan sebelumnya, maka dapat dibuat rumusan masalah yang mendasari penelitian ini. Adapun rumusan masalah tersebut ialah:

1. Bagaimana menerapkan metode SIFT untuk memperoleh informasi rating buku.
2. Bagaimana keakuratan metode SIFT untuk mengenali objek buku dan mencari buku yang sesuai di data latih.

1.3 Tujuan

Tujuan dari penelitian Sistem Pencarian Rating Buku Berdasarkan Sampul Buku menggunakan metode SIFT ini adalah sebagai berikut:

1. Menerapkan algortime SIFT untuk mencari sampul buku yang sesuai.
2. Menguji tingkat akurasi metode SIFT dalam mengenali objek gambar.

1.4 Manfaat

Manfaat dari penelitian ini adalah sebagai berikut:

1. Manfaat Teoritis

Bagi penulis, peneliti lain, dan masyarakat pada umumnya, penelitian ini diharapkan mampu memperkaya wawasan dan pengetahuan mengenai pengenalan objek menggunakan metode *Scale-Invariant Feature Transform* (SIFT).

2. Manfaat Praktis

Penelitian ini diharapkan dapat dikembangkan lebih lanjut dan dapat digunakan untuk membantu melakukan pencarian rating buku agar dapat menyeleksi buku yang berkualitas dengan lebih cepat dan efisien.

1.5 Batasan Masalah

Dalam penelitian Sistem Pencarian Rating Buku Berdasarkan Sampul Buku menggunakan Metode SIFT ini batasan penelitian yang digunakan adalah:

1. Input berupa file gambar sampul buku.
2. Gambar input hanya berisi objek sampul buku dan tidak boleh berisi objek selain sampul buku.

1.6 Sistematika Pembahasan

Untuk mencapai tujuan yang telah dikemukakan di atas maka, maka sitematika penulisan laporan penelitian Sistem Pencarian Rating Buku Berdasarkan Sampul Buku menggunakan Metode SIFT adalah sebagai berikut:

BAB 1 Pendahuluan

Bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian dan sistematika penelitian.

BAB 2 Landasan Kepustakaan

Landasan kepustakaan menjelaskan tentang kajian pustaka terkait dengan penelitian sistem pencarian rating buku berdasarkan sampul buku menggunakan metode SIFT.

BAB 3 Metode Penelitian

Metodologi menjelaskan tentang metode yang digunakan untuk mendeteksi objek buku.

BAB 4 Perancangan Sistem

Perancangan menjelaskan tentang bagaimana rancangan sistem pencarian rating buku berdasarkan sampul buku menggunakan metode SIFT.

BAB 5 Implementasi Sistem

Implementasi menjelaskan tentang bagaimana implementasi penelitian sistem pencarian rating buku berdasarkan sampul buku menggunakan metode SIFT.

BAB 6 Pengujian dan Analisis

Pengujian dan analisis menjelaskan tentang akurasi hasil pengujian pada sistem pencarian rating buku berdasarkan sampul buku menggunakan metode SIFT.

BAB 7 Penutup

Penutup berisi kesimpulan yang diperoleh dari perancangan, implementasi, dan pengujian sistem pencarian rating buku berdasarkan sampul buku menggunakan metode SIFT.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini membahas mengenai dasar teori dan algoritme yang dipakai. Pada bab ini dijelaskan mengenai beberapa teori dan perhitungan mengenai algoritme yang dipakai.

2.1 Kajian Kepustakaan

Penelitian ini didasarkan pada studi *literature* buku dan beberapa penelitian sebelumnya yang pernah dilakukan terkait dengan algoritme *Scale-Invariant Feature Transform* (SIFT). Publikasi penelitian terdahulu dalam bentuk *paper*, jurnal, skripsi digunakan sebagai sumber kajian pustaka dalam penelitian ini. Studi *literature* ini dilakukan dengan tujuan untuk memperuat pemahaman domain permasalahan dan juga untuk mencari cara penyelesaian masalah terbaik. Beberapa penelitian terdahulu yang digunakan sebagai dasar dalam penelitian ini disajikan dalam Tabel 2.1.

Penelitian implementasi metode *Scale-Invariant Feature Transform* (SIFT) dan metode *Contiously Adaptive Mean-Shift* (*CamShift*) pada pejejakan objek bergerak (Agustina et al., 2012) melakukan penelitian dengan menerapkan SIFT dan *CamShift* untuk mendeteksi objek bergerak. SIFT melakukan pendeteksian berdasarkan *keypoint* yang diperoleh dari fitur-fitur lokal. Sedangkan *CamShift* mendeteksi objek dengan melakukan adapatasi atau penyesuaian distribusi probabilitas warna yang selalu berubah-ubah tiap pergatian *frame* dari video. Dari uji coba yang dilakukan, perangkat lunak penjejakan objek bergerak menggunakan metode SIFT dan *CamShift* masing-masing mempunyai kelebihan dan kekurangan masing-masing.

Penelitian lain dengan metode SIFT oleh Setiyawan dan Basuki (2012) dengan judul Pencocokan Citra Berbasis *Scale-Invariant Feature Transform* (SIFT) menggunakan *Arc Consinus*. Tujuan dari penelitian tersebut adalah mengekstrak fitur untuk keperluan *image matching*. Dalam peelitian tersebut digunakan algoritme SIFT untuk medeteksi *keypoint* dan *Arc Consinus* yang merupakan *inverse trigonometry* untuk mencocokkan citra dengan berbagai parameter seperti sudut pandang, skala, dan rotasi. Pengujian menggunakan 4 jenis ekstensi gambar yang berbeda yaitu PNG, BMP, JPEG, dan PGM.

Hasil pengujian dari 100 citra training dan 1500 citra testing didapatkan hasil rata-rata cocok pada perubahan sudut pandang sebesar 33.49%, perubahan skala 60.23%, dan perubahan rotasi 82.77%. Dari hasil yang didapat bisa diambil kesimpulan pencocokan citra dengan SIFT dan *Arc Consine* mempunyai keunggulan dalam mencocokkan perubahan rotasi pada citra.

Tabel 2.1 Kajian Kepustakaan

No.	Penelitian	Objek	Metode (proses)	Hasil
1.	(Agustina & Mukhlash, 2012)	Penelitian ini bertujuan untuk mendeteksi objek bergerak menggunakan metode SIFT dan Camshift.	Metode SIFT Langkah-langkah : 1. Pembentukan *Gussian Scale-Space* dan *Difference of Gaussian.* 2. Deteksi extremum (maksimum dan minimum) pada *Difference of Gaussian.* 3. Penetapan orientasi keypoint *(keypoint assignment)* 4. Penetapan *keypoint descriptor* Metode Camshift Langkah-langkah : 1. Tentukan ukuran awal search window 2. Tentukan lokasi awal search window 3. Tentukan daerah kalkulasi pada bagian tengah *search window* dengan ukuran lebih besar dari *search window* 4. Konversi citra dalam HSV, kemudian buat histogram untuk mengetahui distribusi probabilitas warna 5. Lakukan algoritme mean-shift 6. Set nilai x, y, z yang diperoleh dari langkah 5 7. Nilai x dan y dipakai untuk menentukan titik tengah *search window*. 8. Ulangi mulai langkah 3 untuk setiap pergantian frame video.	Penelitian menghasilkan dari hasil uji coba menunjukkan bahwa waktu komputasi dari algoritme SIFT lebih efisien dibandingkan dengan algoritme Camshift. Akan tetapi, hasil penjejakan yang dilakukan oleh algoritme Camshift lebih baik dibandingkan dengan algoritme SIFT.
2.	(Setiyawan & Basuki, 2013)	Tujuan dari penelitian ini adalah megekstrak fitur untuk keperluan *image matching.*	Metode fuzzy SIFT Langkah-langkah : 1. Pembentukan *Gussian Scale-Space* dan *Difference of Gaussian.* 2. Deteksi extremum (maksimum dan minimum) pada *Difference of Gaussian.* 3. Penetapan orientasi keypoint *(keypoint assignment)* 4. Penetapan *keypoint descriptor*	Penelitian menghasilkan rata-rata kecocokan citra berdasarkan parameter Perubahan Sudut pengambilan Citra 33,498 %, Perubahan Ukuran (*Scala*) Citra60,237 %, Perubahan Rotasi Citra 82,78 %. Dengan data hasil pengujian tersebut *Arc Cosinus* dengan tingkat baik dalam mencocokkan

				citra yang terkena perubahan rotasi, sedangkan untuk Perubahan Ukuran (*Scala*) dan Sudut Pengambilan Citra masih dalam tingkat kurang baik.
3.	(Almeida, et al., 2009)	Tujuan dari penelitian ini adalah evaluasi hasil dari algoritme SIFT yang diterpakan untuk CBIR	Metode SIFT Langkah-langkah : 1. *Scale-space extrema detection.* 2. *Keypoint localization* 3. *Orientation assignment* 4. *keypoint description*	Hasil penelitian menunjukkan bahwa SIFT mempunyai sifat *invariant* pada warna. Jadi perbedaan warna akan mengakibatkan pendeteksian yang berbeda. Namun kelebihan dari SIFT adalah pada ukuran vektor fitur dan kualitas deskriptor yang dapat menghasilkan akurasi dengan baik.

2.2 Citra Digital

Menurut Kamus Besar Bahasa Indonesia citra mempunyai arti rupa, gambar, atau gambaran, dan digital diartikan berhubungan dengan angka-angka untuk sistem perhitungan tertentu. Sedangkan citra digital biasa diartikan sebagai gambar dua dimensi yang bisa ditampilkan sebagai pada layar komputer sebagai himpunan/diskrit nilai digital yang disebut piksel.

Citra digital adalah gambar yang elemen penyusunnya berupa piksel. Pada umunya piksel-piksel yang menyusun gambar tersebut berbentuk sebuah *array*. Ukuran dimensi dari sebuah citra digital ditentukan berdasarkan pada dimensi dari array pikselnya. Lebar dari sebuah citra digital adalah sebanyak kolom dari *array*, dan tingginya sebanyak dari baris *array*. Untuk menunjuk sebuah piksel tertentu pada umunya digunakan koordinat *x* dan *y*.

Untuk dapat merepresentasikan gambar yang nyata sebuah citra digital membutuhkan satu unsur lagi yang biasa disebut resolusi. Resolusi adalah skala spasial dari piksel citra. Sebagai contoh, sebuah citra digital 3300x2550 piksel dengan resolusi 300 piksel per inci (ppi) akan menjadi gambar dengan ukuran 11x8.5.

Ukuran dari *array* piksel hanya dapat didefinisikan MxN, array tersebut hanya dapat terbentuk dengan bentuk persegi. Untuk mendefinisikan sebuah gambar menjadi citra digital dibutuhkan parameter berupa intensitas. Maksud dari intensitas adalah setiap piksel mempunyai nilai kecerahan masing-masing. Apabila intensitas dari setiap piksel pada *array* tersebut sama maka gambar tersebut akan sepenuhnya hitam, putih, abu-abu dan sebagainya tergantung pada intensitasya. Gambar hitam putih hanya memiliki intensitas keabu-abuan yang paling gelap (hitam) sampai intensitas keabu-abuan paling terang (putih). Di sisi lain gambar berwarna mempunyai kombinasi intensitas yang paling gelap sampai yang paling terang dari 3 macam warna yang berbeda yaitu *red* (merah), *green* (hijau), dan *blue* (biru). Kombinasi dari 3 macam warna ini dikenal sebagai RGB.

Nilai dari intensitas RGB dalam citra digital didefinisikan dalam satuan bit. Bit mempunyai 2 macam tipe yaitu 1-bit atau *binary* dan 8-bit. Tipe *binary* hanya mempunyai 2^1 = 2 kemungkinan nilai warna, yaitu 0 dan 1. Sedangkan tipe 8-bit mempunyai nilai dengan *range* 2^8 = 256 yang memungkinkan. RGB menggunakan tipe 8-bit, maka gambar dengan tipe RGB akan mempunyai 3x8-bit intensitas warna biasanya RGB juga biasa dikenal dengan citra digital tipe 24-bit.

2.3 *Scale Invariant Feature Transform*

Computer vision atau visi komputer menyiratkan bahwa sebuah komputer dapat melihat dan mengerti yang dilihatnya. Sebuah komputer yang dapat menampilkan atau mengambil gambar belum dapat diartikan sebagai *computer*

vision. *Vision* atau visi adalah proses untuk memahami lingkungan sekitar melalui *spectrum* yang terlihat. Hasil dari pemahaman tersebut bisa berupa objek yang menarik dengan berbagai macam ciri-cirinya yang dapat dideteksi. Terlepas dari kesan yang tampak sederhana tersebut, *computer vision* merupakan suatu bahasan yang sangat menantang dalam dunia ilmu komputer.

Untuk dapat mengenali sebuah objek pertama objek tersebut harus disegmentasi atau dipisahkan dari *background*-nya. Kemudian fitur-fitur dari objek tersebut harus diidentifikasi. Agar program dapat berjalan dengan handal, fitur-fitur tersebut harus dapat dideteksi dengan tanpa memperhitungkan orientasi dari objek. Karena citra digital hanya proyeksi dua dimensi dari objek 3 dimensi di dunia nyata, maka langkah ini akan menimbulkan rintangan yang signifikan. Sebagai perbandingan, penglihatan manusia dapat dengan mudah untuk mengenali wajah seseorang pada hampir semua sudut pandang. Secara umum, teknisi pada bidang *computer vision* mampu untuk memahami dan menafsirkan dunia nyata dengan 3 dimensi ke dalam 2 dimensi.

Mencocokkan fitur antara objek yang sama pada gambar yang berbeda adalah masalah yang umum pada Visi Komputer. Apabila kedua gambar sama baik dari segi ukuran, orientasi, sudut padang dan lain-lain, maka algoritme deteksi sudut sederhana akan bisa bekerja untuk pendeteksian objek, tetapi ketika gambar tersebut berbeda ukuran, orientasi, sudut pandang maka *Scale Invariant Feature Transform* (SIFT) akan dapat bekerja lebih baik.

SIFT mendeskripsikan fitur sebuah gambar dengan banyak properti agar dapat dicocokkan dengan objek atau pemandangan pada gambar yang berbeda. Fitur-fitur yang dihasilkan oleh algoritme SIFT bersifat *invariant* terhadap ukuran dan rotasi gambar. Fitur tersebut juga dapat berfungsi dengan baik pada domain spasial dan domain frekuensi. Akan ada banyak fitur yang dapat diambil dari sebuah gambar dengan algoritme yang efisien. Selain itu, fiturnya pun sangat khas, yang memungkinkan satu fitur dicocokkan dengan probabilitas yang tinggi terhadap database fitur yang besar. Hal ini memungkinkan memberikan dasar pengenalan objek dan pegenalan adegan (Lowe, 2004). Algortime SIFT mempunyai 4 tahapan utama pada komputasi yang digunakan untuk mendeskripsikan satu set fitur sebuah gambar (Lowe, 2004).

2.3.1 *Scale Space Extrema Detection*

Sebuah objek yang sama kemungkianan bisa terlihat mempunyai skala ukuran yang berbeda. Perbedaan skala ukuran sebuah objek pada gambar bisa disebabkan oleh perbedaan sudut pandang, perbedaan jarak objek, dan lain-lain. Perbedaan skala secara alami seperti ini sangat sering terjadi, oleh karena itu *scale space* atau ruang skala akan mengadopsi konsep ini.

Untuk dapat mendeteksi objek sama dengan ukuran yang berbeda karena perbedaan jarak padang dapat dilakukan dengan mencari fitur yang stabil disemua skala ukuran yang mungkin, menggunakan sebuah fungsi skala kontinyu yang dikenal dengan sebutan *scale space* atau ruang skala (Witkins, 1983).

Langkah pertama untuk membuat *scale space* adalah dengan secara bertahap membuat *blur* gambar. Kemudian *resize* gambar tersebut menjadi setengahnya, kemudian lakukan *blur* lagi pada gambar yang telah diperkecil ukurannya. Ulangi proses tersebut sampai 4 kali. Berikut adalah hasil dari langkah pertama.

Jumlah dari *octave* dan skala bergantung kepada ukuran dari gambar. Ketika membuat program dengan algortime SIFT kita sendiri yang akan menentukan berapa banyak jumlah *octave* dan skala, namun penemu algoritme SIFT menyarankan untuk menggunakan 4 *octave* dan 5 skala pengecilan. Untuk proses *blur* menggunakan konvolusi kernel *Gaussian Filter*.

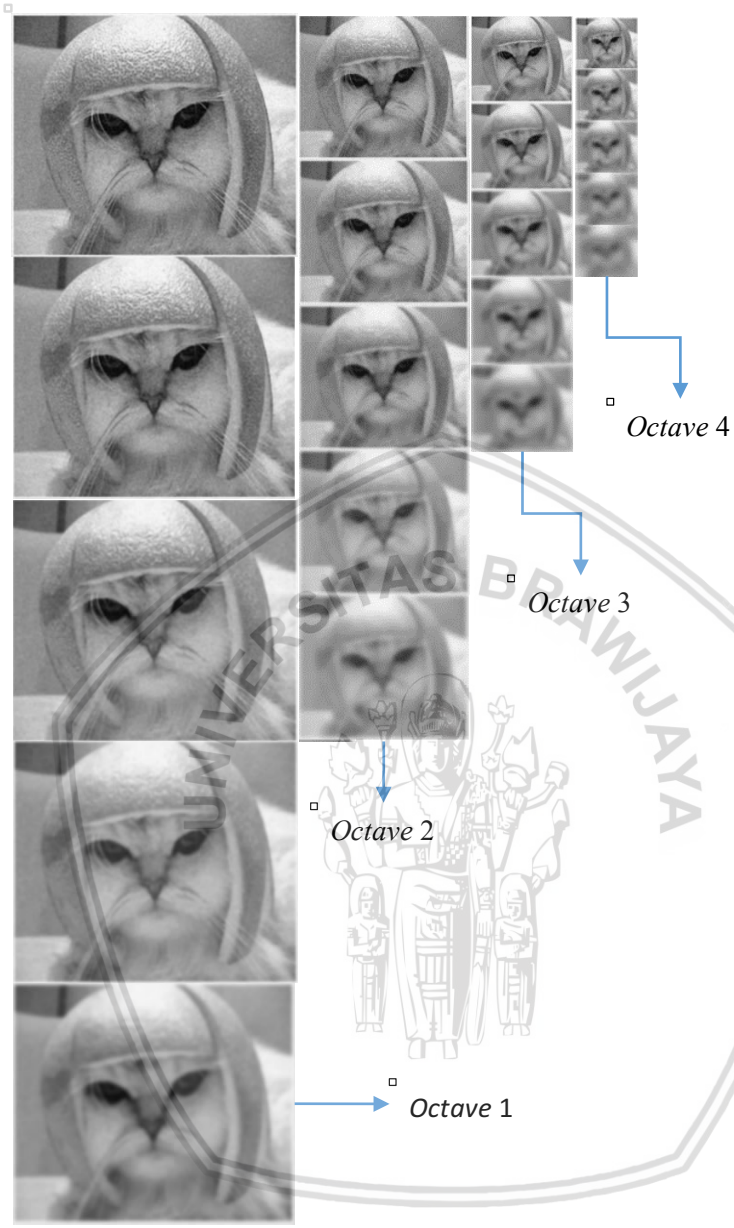
Fungsi konvolusi *Gaussian Filter*:

$$L\,(x, y, \sigma) = G\,(x, y, \sigma) * I\,(x, y) \tag{2.1}$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{2.2}$$

Keterangan Persamaan 2.1 dan Persamaan 2.2.

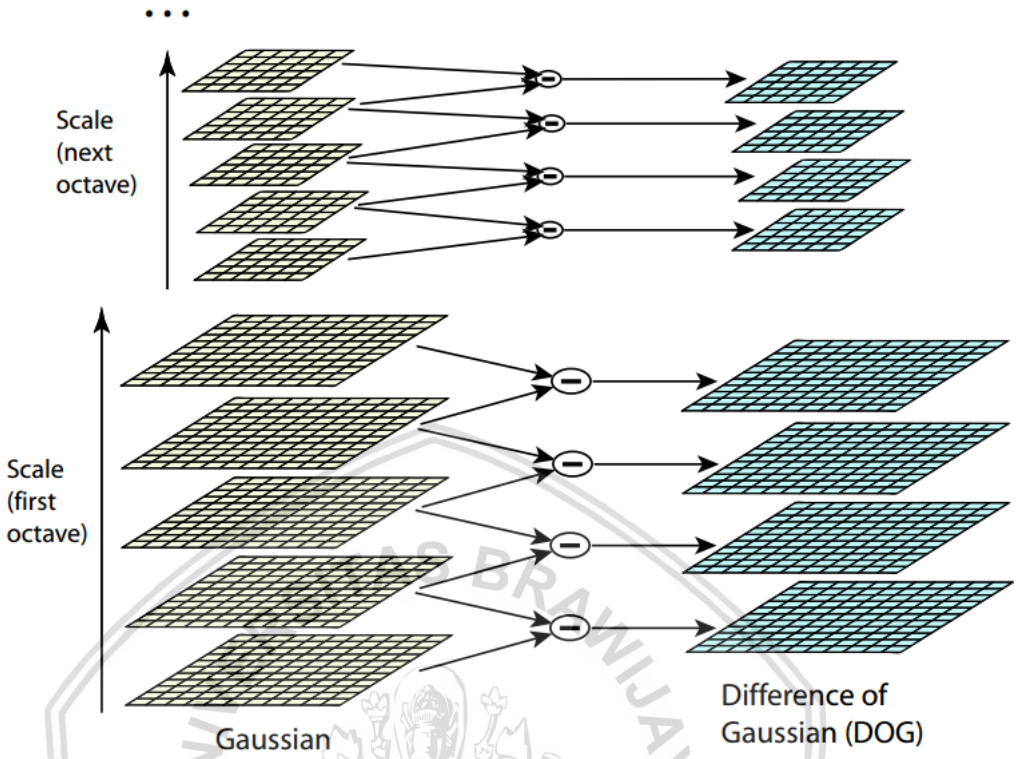
- L = Gambar yang telah diblur
- G = Operator *Gaussian kernel*
- I = Gambar untuk *blur*
- x, y = Koordinat lokasi piksel
- σ = Parameter skala. Semakin besar nilainya, semakin kuat *blur*nya.



Gambar 2.1 Contoh hasil blur dan Pengecilan skala

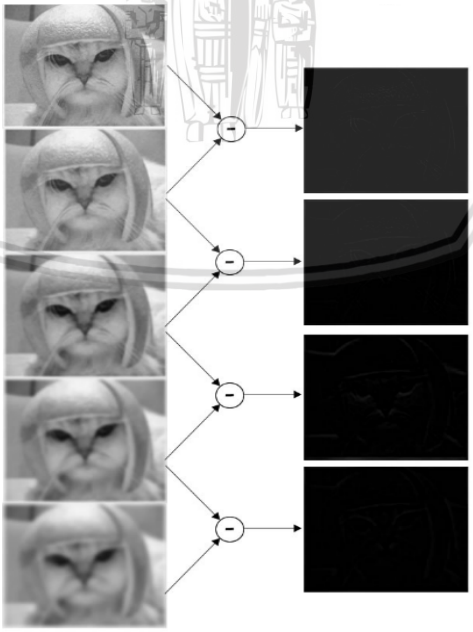
Agar dapat mendeteksi lokasi *keypoint* yang stabil secara efisien, maka akan digunakan *scale-space extrema* pada fungsi *different-of-Gaussian*(DoG) yang dikonvolusikan pada gambar. DoG (D)didapatkan dengan cara mengurangkan dua skala terdekat yang dipisahkan dengan k (Lowe, 2004).

$$D(x,y,\sigma) = \big(G(x,y,k\sigma) - G(x,y,\sigma)\big) * I(x,y) \\ = L(x,y,k\sigma) - L(x,y,\sigma) \quad (2.3)$$



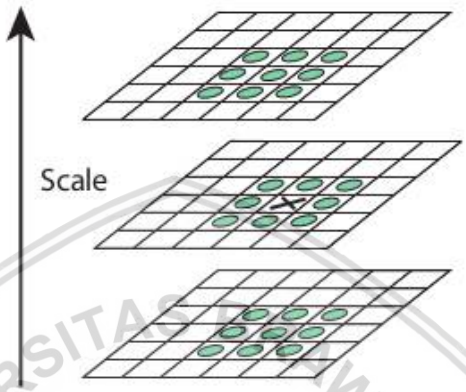
Gambar 2.2 Ilustrasi DoG

Contoh hasil dari gambar yang dilakukan proses *Different-of-Gaussian* (DoG) ditunjukkan pada Gambar 2.3.



Gambar 2.3 Contoh hasil proses *DoG*

Setelah proses DoG selesai, kemudian dicari *local extrema* dengan membandingkan gambar antar skala. Setiap piksel pada gambar DoG akan dibandingkan dengan 3 x 3 piksel tetangganya dengan total 26 piksel tetangga pada skala sebelumnya dan skala selanjutnya. Jika nilai tersebut adalah nilai terbesar dari piksel tetangganya maka dia adalah *local extrema*. Jika piksel adalah *local extrema* maka piksel tersebut berpotensi sebagai *keypoint*.



Gambar 2.4 Mencari *local extrema* (Lowe, 2004)

2.3.2 *Keypoint localization*

Setelah kandidat *keypoint* ditemukan langkah selanjutnya adalah melakukan seleksi secara lebih detail berdasarkan data lokasi, skala, ratio, dan lengkungan terdekat. Proses ini memungkinkan untuk mengeliminasi *keypoint* yang memiliki kontras rendah (untuk mengantisipasi *noise*). Fungsi ini menerapkan *Taylor Expansion* pada *scale-space extrema*, $D\ (x, y, \sigma)$. Hasil penerapan *Taylor Expansion* pada fungsi *scale-space extrema* (Lowe, 2004):

$$D(x) = D + \frac{\partial D^T}{\partial x}\ x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x \qquad (2.4)$$

Karena D dan turunannya dievaluasi dari sample *point* dan $\hat{X} = (x, y, \sigma)^T$ adalah nilai *offset* dari titik tersebut. Lokasi dari *extremum* $\hat{X}$ didapatkan menurunkan fungsi berikut dengan nilai x dan set nilai menjadi 0 (Lowe, 2004).

$$\hat{x} = -\frac{\partial^2 D}{\partial x^2}^{-1} \frac{\partial D}{\partial x} \qquad (2.5)$$

Untuk stabilitas, tidak cukup hanya dengan menghilangkan *keypoint* dengan *contrast* rendah. Fungsi DoG akan mempunyai respon yang kuat terhadap garis atau tepi, meskipun jika lokasi *keypoint* berada di sepanjang garis tepi penentuannya bisa buruk dan rawan terhadap *noise* dalam jumlah kecil (Lowe, 2004).

Sebuah puncak yang terdefinisi dengan buruk pada fungsi *DoG* akan mempunyai lengkungan utama yang besar terhadap tepian objek tapi mempunyai

nilai yang kecil terhadap arah tegak lurus. Lengkungan utama dapat dihitung menggunakan 2 x 2 *Hessian* matrik yang ditunjukkan pada Persamaan 2.6.

$$\boldsymbol{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad (2.6)$$

Nilai turunannya dapat dicari dengan mengambil perbedaan titik sampel tetangga. *Eigen value* dari matrik **H** sesuai dengan lengkungan utama pada D. Dengan menggunakan algoritme *Harris Corner Detection*, apabila α adalah *eigen value* dan β adalah yang terkecil, maka kita bisa mencari hasil penjumlahan *eigen value* dari *trace* pada H menggunakan Persamaan 2.7 dan hasil perkalian dari *Determinant* menggunakan Persamaan 2.8 (Lowe, 2004).

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta \quad (2.7)$$

$$Det(H) = D_{xx}D_{yy} - \left(D_{xy}\right)^2 = \alpha\beta \quad (2.8)$$

Pada suatu keadaan yang tidak diinginkan jika determinan bernilai negatif, lengkungannya mempunyai tanda yang berbeda, jadi titik tersebut tidak akan disimpan sebagai sebuah *extremum*. Apabila r adalah ratio antara nilai *Eigen value magnitude* terbesar dan yang terkecil, maka $\alpha = r\beta$ (Lowe, 2004).

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \quad (2.9)$$

Rumus tersebut lebih bergantung pada rasio dari *eigen value* dari dari pada nilai individualnya. Nilai dari $(r+1)^2/r$ berada pada nilai minimum ketika 2 *eigen value* bernilai sama da meningkat dengan r. Untuk memeriksa apakah rasio dari lengkungan dibawah *threshold* maka perlu dilakukan pemeriksaan dengan Persamaan 2.10.

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \quad (2.10)$$

Hal ini sangat efektif untuk dihitung, dengan kurang dari 20 *floating point operator* yang dibutuhkan untuk memeriksa setiap *keypoint*. Eksperimen dari *paper* Lowe, David (2004) meggunakan nilai r = 10.

2.3.3 *Orientation Assignment*

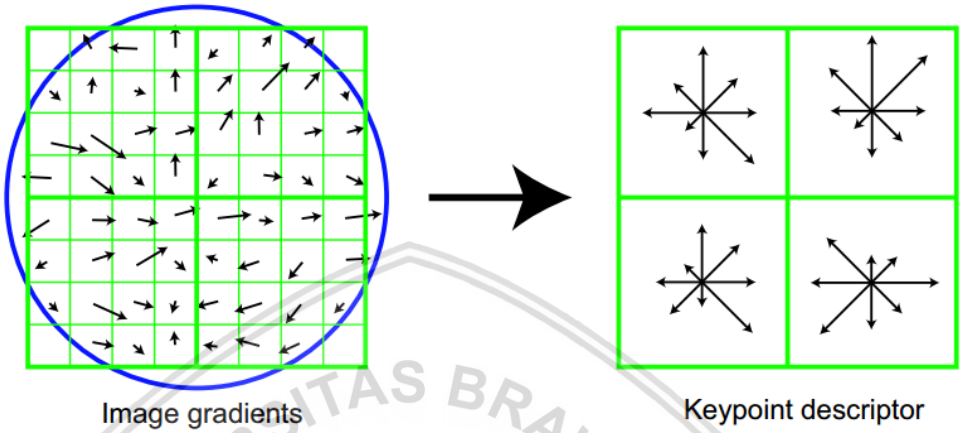
Dengan menempatkan orientasi yang konsisten pada setiap *keypoint* berdesarkan pada *property* dari gambar, *descriptor* dari *keypoint* dapat direpresetasikan pada orientasi yang lebih relatif dan akan dapat tetap bekerja pada rotasi yang bermacam-macam. Untuk dapat memberikan orientasi yang konsisten adalah dengan mencari arah *gradient* dan *magnitude* disekitar setiap *keypoint* (Lowe, 2004). *Gradient* dan *magnitude* dapat dicari menggunakan Persamaan 2.11 dan Persamaan 2.12.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y) + L(x,y+1) - L(x,y-1))^2} \quad (2.11)$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))) \quad (2.12)$$

2.3.4 *Keypoint Descriptor*

Magnitude dan *gradient* dihitung pada semua piksel di sekitar *keypoint*. Kemudian dibuat sebuah *histogram* untuk *magnitude* dan *gradient* tersebut. Untuk orientasi akan dibagi menjadi 8 arah.



Gambar 2.5 Ilustrasi orientasi *gradient* (Lowe, 2004)

Histogram dari *descriptor* akan dibentuk menjadi sebuah vektor yang mengandung nilai-nilai dari semua masukan *histogram*. Gambar 2.5 menunjukkan sebuah *array* 2 x 2 dari orietasi *histogram*, sedangkan percobaan menunjukkan bahwa hasil terbaik didapatkan dengan menggunakan array 4 x 4 dengan *histogram* arah *gradient* adalah 8. Jadi akan ada 4x4x8 = 128 elemen fitur pada setiap *keypoint* (Lowe, 2004).

BAB 3 METODOLOGI

Pada bab ini menjelaskan langkah-langkah dalam menyusun dan melakukan penelitian. Tahapan-tahapan yang dilakukan diantaranya studi pustaka, pengumpulan data, perancangan algoritme, serta pengujian dan analisis, yang terakhir adalah kesimpulan dan saran.

3.1 Tipe Penelitian

Pada penelitian ini, dilakukan penelitian dengan tipe *non-implementatif*. Penelitian tipe *non-implementatif* berfokus kepada pengkajian terhadap sebuah kasus atau kondisi tertentu, yang pada akhirnya menghasilkan analisis ilmiah. Metode yang dapat digunakan untuk menghasilkan analisis ilmiah dapat berupa eksperimen, kuesioner, observasi, survei, studi kasus, dan metode-metode yang lainnya. Jika dilihat dari kegiatan yang dilakukan saat penelitian, maka dapat dikatakan bahwa pendekatan yang dilakukan pada penelitian ini adalah penelitian *non-implementatif analitik*. Penelitian dengan pendekatan tersebut memiliki tujuan untuk menjelaskan relasi atau hubungan antara komponen dalam penelitian dengan kondisi tertentu yang sedang diteliti.

3.2 Strategi Penelitian

Pada penelitian ini, strategi yang digunakan adalah eksperimen. Eksperimen merupakan jenis penelitian yang bertujuan untuk mencari hubungan sebab akibat. Eksperimen juga sepenuhnya dikontrol oleh peneliti serta pengembang yang bertanggung jawab, biasanya eksperimen dilakukan di dalam laboratorium. Metode eksperimen yang digunakan dalam penelitian ini adalah *Scale Invariant Feature Transform*

3.3 Lokasi Penelitian

Penelitian ini dilakukan di Laboratorium Komputasi Cerdas yang bertempat di Fakultas Ilmu Komputer (Filkom) Universitas Brawijaya. Lokasi dipilih dikarenakan di dalamnya dapat dilakukan eksperimen terkait dengan pencarian informasi rating buku menggunakan metode *scale invariant feature transform*.

3.4 Pengumpulan Data

Metode pengumpulan data yang digunakan pada penelitian ini adalah dengan pengumpulan data primer. *Dataset* yang digunakan dalam penelitian ini adalah gambar sampul buku yang didapatkan pada halaman *website* Goodreads.com.

Gambar sampul buku yang didapatkan dari halaman *website* Goodreads kemudian dicetak. Hasil cetakan gambar sampul buku tersebut kemudian diambil gambarnya menggunakan kamera. Gambar-gambar hasil pengambilan dari kamera inilah yang menjadi *dataset*.

3.5 Perancangan Algoritme

Pada tahap ini merupakan tahap perancangan algoritme yang digunakan dalam penelitian ini. Selain itu pada tahap perancangan algoritme menggambarkan jalannya algoritme yang digunakan.

3.6 Teknik Pengujian dan Analisis

Pada tahap ini adalah menguji algoritme yang telah dirancang untuk penelitian ini. Algortime klasifikasi diterapkan ke dalam data uji yang telah ada sebelumnya. Tujuan dari pengujian adalah menemukan kesalahan yang mungkin terjadi dalam proses klasifikasi. Setelah didapatkan hasil pengujian kemudian dilanjutkan dengan analisis terhadapa hasil yang didapat, bagaimana sebuah kondisi terjadi dan bagaimana cara mengatasi kesalahan yang muncul. Tujuan analisis juga melihat apakah penelitian sudah mencapai tujuan yang diinginkan.

Pengujian yang dilakukan pada penelitian ini ada tiga jenis pegujian:

1. Pengujian pengaruh pencahayaan.
2. Pengujian pengaruh rotasi.
3. Pengujian pengaruh jarak pengambilan gambar.

3.7 Penarikan kesimpulan dan saran

Kesimpulan dari penelitian dapat diambil setelah semua tahap dan langkah penelitian terselesaikan, langkah tersebut adalah perancangan algoritme, pengujian dan analisis. Kesimpulan berisi hasil pengujian dan analisis hasil. Selanjutnya adalah saran, saran bertujuan untuk memberikan masukan kepada peneliti lain yang ingin melakukan penelitian dengan tema yang sama.

BAB 4 PERANCANGAN

Pada bab ini akan membahas tentang perancangan sistem yang menggambarkan apa yang akan dibangun pada sistem perolehan rating buku berdasarkan dari teori serta penelitian yang telah dibahas sebelumnya. Pada bab perancangan ini secara detil meliputi: Perancangan data, perancangan algoritme serta perancangan pengujian.

4.1 Perancangan Data

Pada subbab perancangan data ini dibahas penentuan dari bentuk data yang akan digunakan pada penelitian. Data yang akan digunakan pada penelitian ini merupakan data yang berbentuk gambar diantaranya:

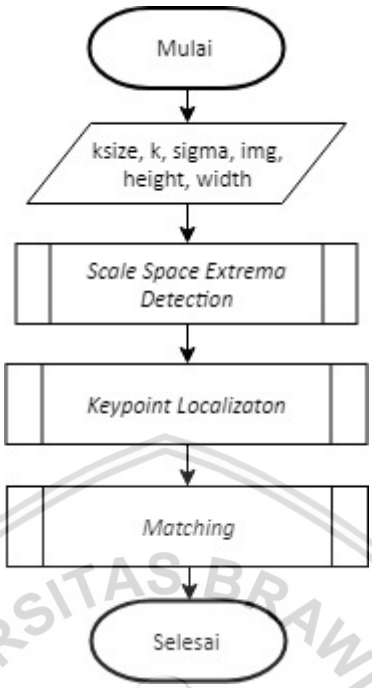
1. Data gambar sampul buku untuk data latih didapatkan dari *goodreads.com*, sedangkan data gambar sampul buku untuk data latih didapatkan dari hasil foto.
2. Ukuran gambar yang terlalu besar akan diperkecil untuk mengurangi waktu komputasi.

Data hasil dari latih yang berisi *descriptor* dari sebuah gambar akan disimpan dalam *file store* yang berekstensi *pickle*.

4.2 Perancangan Algoritme

Subbab perancangan algoritme membahas proses perancangan sistem yang akan dibangun secara keseluruhan. Bab ini akan menjabarkan tahapan-tahapan dari algoritme yang akan diimplementasikan dalam penelitian. Tahapan proses algoritme akan dibagi dibagi menjadi empat tahapan yakni diantaranya *scale space extrema detection, keypoint localization, orientation assignement,* dan terakhir *keypoint descriptor*.

Scale space extrema detection akan memproses data masukan gambar menjadi gambar yang lebih kecil dan difilter. Setelah proses filter selesai maka akan dicari piksel-piksel yang memenuhi syarat untuk menjadi *keypoint*, proses ini dinamakan *keypoint localization*. Setelah didapatkan piksel-pikse yang memenuhi syarat maka piksel itu akan diproses pada tahapan *orientation assignment* untuk mendapatkan arah dari piksel tersebut. Pada tahapan terakhir *keypoint descriptor* akan dicari *descriptor* atau hal-hal yang bias menjelaskan *keypoint* tersebut, proses ini dilakukan dengan mencari arah dari 16x16 piksel tetangganya dan menyimpannya dalam *histogram*. Untuk lebih jelasnya tahapan proses algoritme dalam penelitian ini dapat dilihat pada Gambar 4.1.

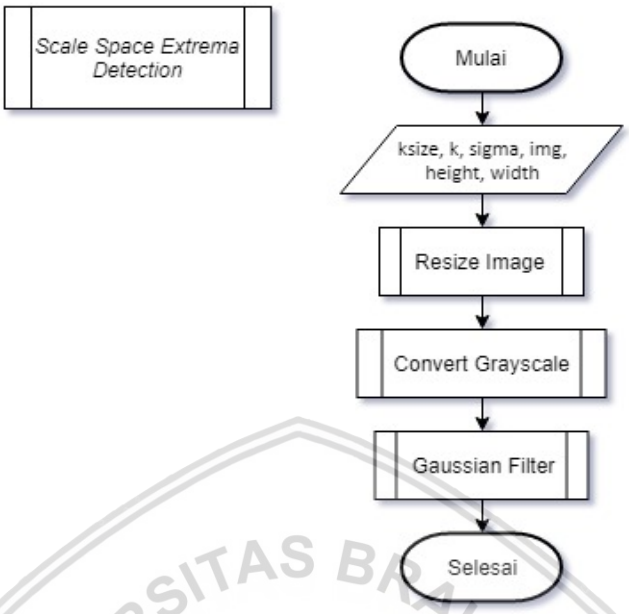


Gambar 4.1 Diagram alir tahapan proses algoritme

Diagram alir pada Gambar 4.1 dijelaskan bahwa di dalam diagram alir sistem, sebelumnya ada beberapa *predefined process* dimana di dalam proses yang ada tersebut masih terdapat subproses-subproses yang lain. Beberapa subproses tersebut contohnya adalah subproses *scale space extrema detection* yang digunakan untuk tahapan *resize image*, *convert grayscale*, dan *Gaussian filter* dan juga subproses dari tahap proses perhitungan *keypoint localization, orientation assignment, keypoint descriptor* dan *matching*.

4.2.1 Scale Space Extrema Detection

Seperti yang telah dijelaskan tahapan pada Gambar 4.1 yang mengacu pada *scale space extrema detection*, dalam tahapan ini akan dilakukan proses yang bertujuan untuk mendapatkan gambar yang bermacam-macam ukuran. Hal ini dilakukan untuk membuat gambar tetap bisa dicocokkan dengan data latih walaupun gambar data latih berbeda dalam hal ukuran. Proses-proses pada tahapan ini dapat dilihat pada Gambar 4.2.

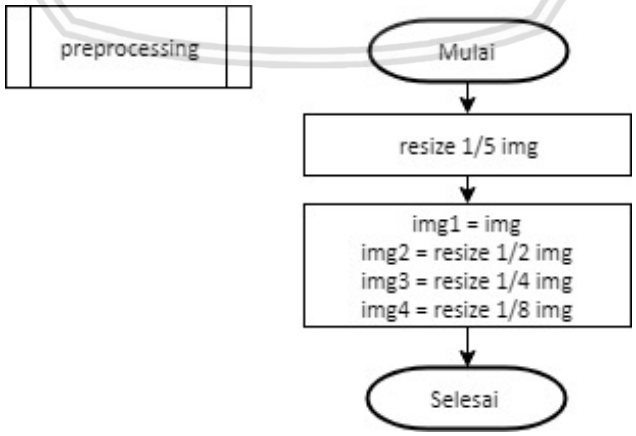


Gambar 4.2 Diagram alir tahapan proses *Scale Space Extrema Detection*

Pada Gambar 4.2 menjelaskan gambaran umum dari proses *scale space extrema detection* yang akan dilakukan. Sebagaimana yang telah dijelaskan sebelumnya, akan ada tiga tahapan *predefined process* yang kemudian akan dibahas lebih dalam pada sebuah proses-proses yang lebih detail pada sub-bab berikutnya. Gambar yang pada dasarnya adalah kumpulan angka-angka yang membentuk *array* akan dilakukan tiga tahapan *scale space extrema detection* yakni sebagai berikut.

1. *Resize Image*

   Resize image merupakan proses untuk mengubah ukuran gambar menjadi 1/2, 1/4, dan 1/8 dari ukuran aslinya. Hal ini bertujuan agar didapatkan skala gambar yang berbeda-beda untuk pencarian *descriptor* gambar. Alur tahapan *resize image* ditunjukkan pada Gambar 4.3.



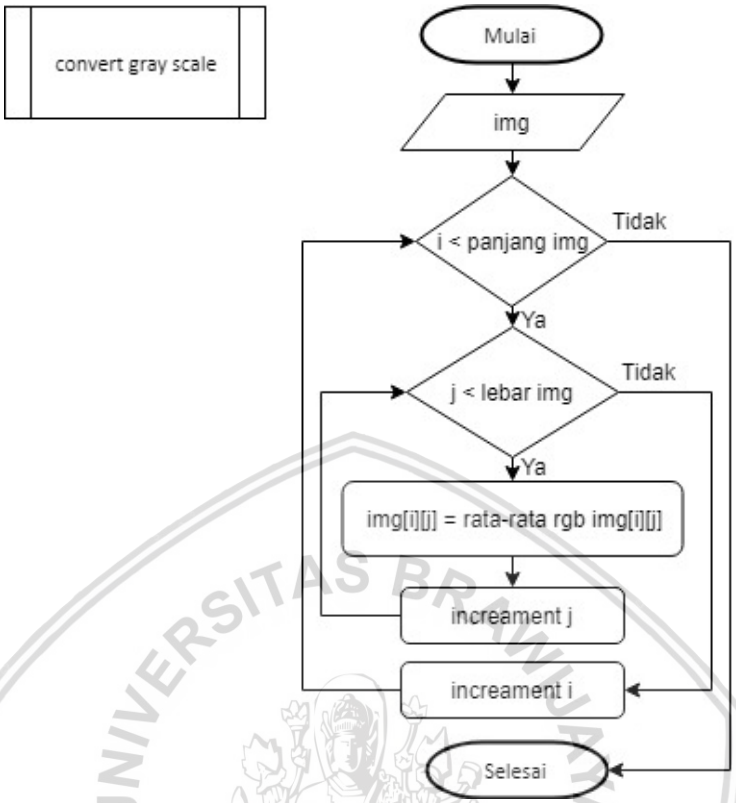
Gambar 4.3 Alur Proses *Resize Image*

Pada Gambar 4.3 dijelaskan mengenai alur proses dalam *resize* gambar. Proses pertama adalah mengubah ukuran gambar menjadi lima kali lebih kecil dari ukuran sebenarnya, hal ini bertujuan agar proses komputasi tidak terlampau besar. Hasil dari pengecilan gambar asli akan disimpan dalam variable *img1*. Kemudian hasil dari *img1* akan diperkecil setengahnya dan disimpan dalam variable *img2*, *img2* akan diperkecil setengahnya dan disimpan dalam *variable img3*, begitupun *img3* dan *img4*. Hal ini dilakukan karena SIFT membutuhkan gambar dengan skala ukuran awal 1 sampai gambar yang diperkecil hingga 1/8. Untuk proses *resize image* sendiri menggunakan *library* OpenCV yang telah ada di Python. Contoh hasil pengecilan gambar dapat dilihat pada Gambar 4.4.

Gambar 4.4 Contoh hasil *resize image*

2. *Convert Gray Scale*

Convert grayscale merupakan proses mengubah gambar yang awalnya mempunyai format RGB atau BGR pada Python menjadi *grayscale*. Gambar perlu diubah menjadi *grayscale* karena akan memakan komputasi yang juga sangat besar apabila diproses dalam keadaan masih berwarna. Alur tahapan *convert grayscale* ditunjukkan pada Gambar 4.5.



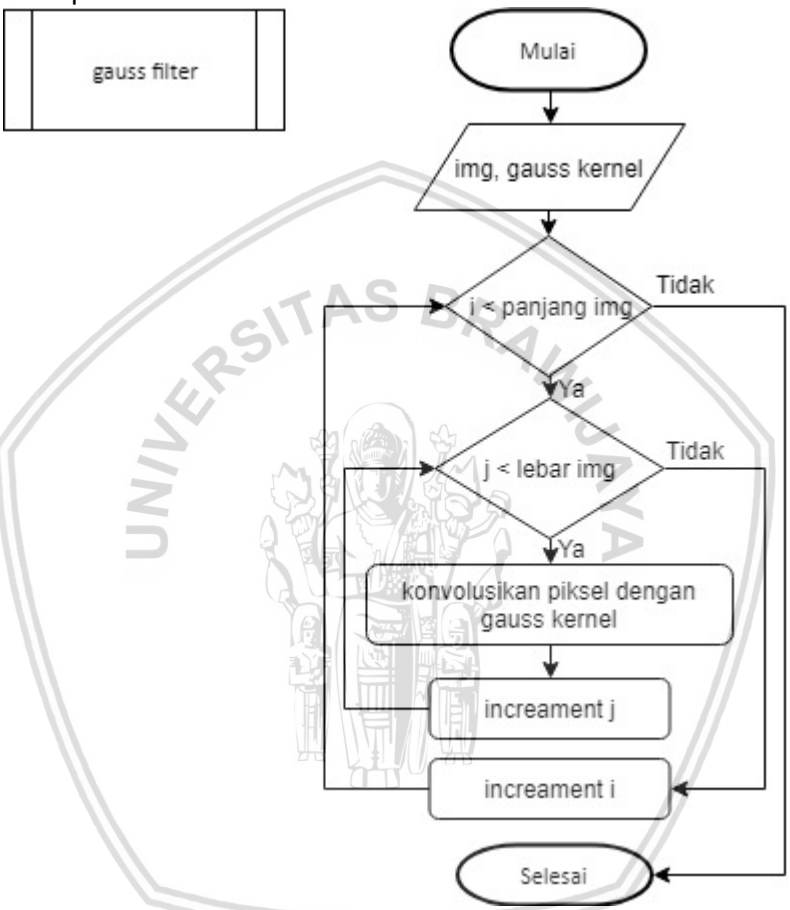
Gambar 4.5 Alur Proses *Convert Grayscale*

Gambar 4.5 menunjukkan alur dari proses mengubah gambar menjadi *grayscale*. Semua gambar mulai dari gambar pertama sampai gambar yang telah diperkecil sampai 8 kali akan disimpan dalam objek gambar baru yang diberi nama *level_1* sampai *level_4*. Proses pengubahan ke dalam bentuk *grayscale* memanfaatkan *library* yang telah tersedia dalam OpenCV. Setelah gambar diubah dalam bentuk *grayscale* nilainya yang masih mempunyai *range* antara 0-255 dinormalisasi agar menjadi nilai yang mempunyai *range* 0-1. Hal ini diperlukan untuk mengurangi komputasi angka yang terlalu besar nantinya. Untuk menormalisasi nilai dari piksel digunakan fungsi yang telah tersedia dalam *library* OpenCV. Contoh hasil *grayscale* dapat dilihat pada Gambar 4.6.

Gambar 4.6 Contoh hasil proses *grayscale*

3. Gaussian Filter

Tahapan terakhir dari proses *scale space extrema detection* adalah proses *Gaussian filter*. Pada tahapan ini setiap gambar dari semua *level* ukuran difilter sampai 5 kali tingkat *filtering*. Pada 1 *level* ukuran, akan dihasilkan 5 gambar baru yang semuanya mempunyai tingkat *blur* yang berbeda-beda. Tingkat *blur* akan terus meningkat untuk tiap 5 gambar tadi. Alur tahapan *Gaussian filter* ditunjukkan pada Gambar 4.7.



Gambar 4.7 Alur Proses *Gaussian Filter*

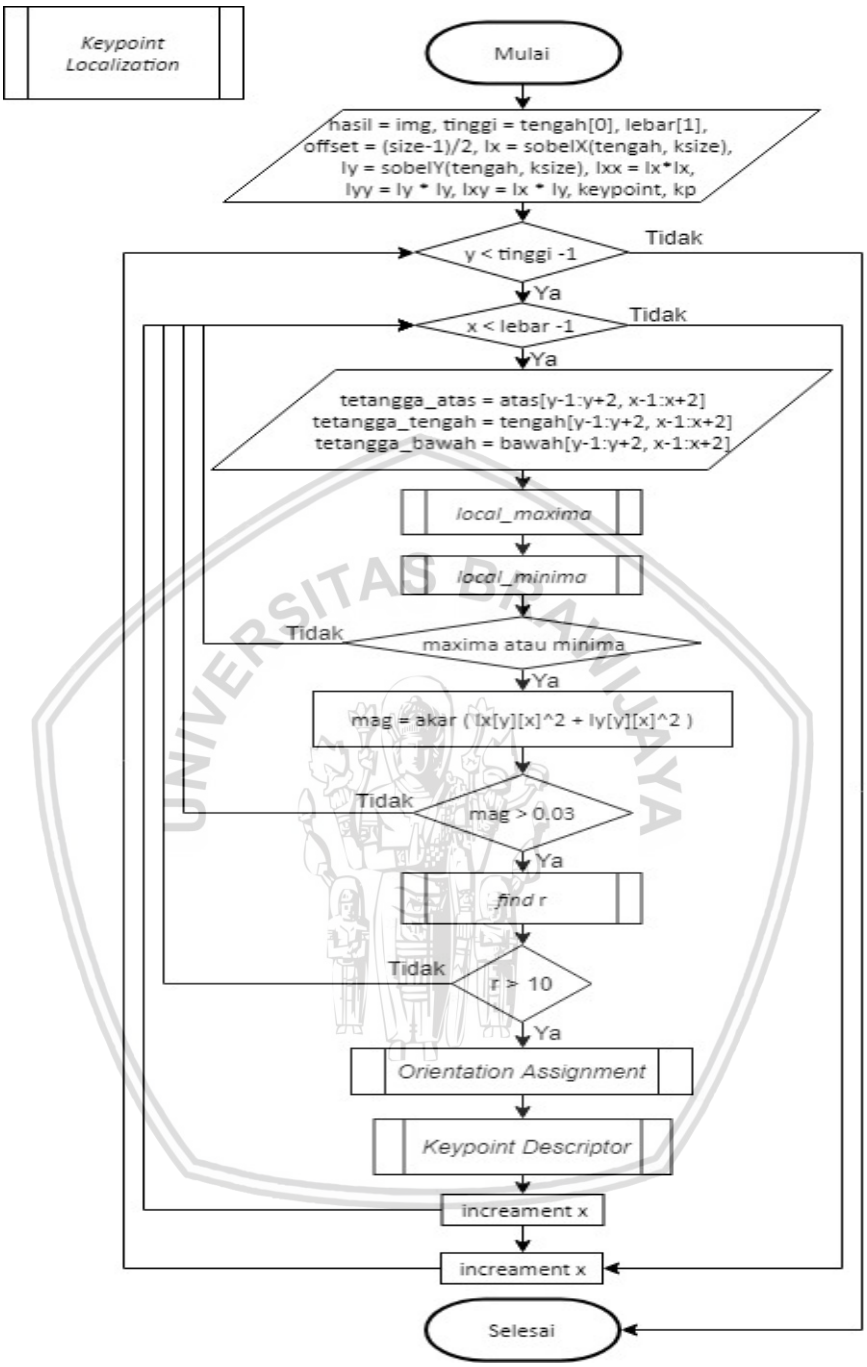
Gambar 4.7 menjelaskan alur proses dari *Gaussian filter*. Proses Gaussian filter sendiri disini menggunakan fungsi yang telah tersedia dalam *library* OpenCV. Saat konvolusi *Gaussian* memanfaatkan nilai *sigma* untuk menentukan kekuatan *blur* gambar. Semakin besar nilai *sigma* yang digunakan sebagai *kernel* maka akan semakin besar pula *blur* yang dihasilkan. Contoh gambar sebelum dan sesudah *Gaussian filter* dapat dilihat pada Gambar 4.8 dan Gambar 4.9.

Gambar 4.8 Sebelum *Gaussian Filter*

Gambar 4.9 Sesudah *Gaussian Filter*

4.2.2 *Keypoint Localization*

Pada tahapan *keypoint localization* ini akan diseleksi piksel-piksel yang memungkinkan untuk menjadi *keypoint*. Dalam tahapan ini akan dibagi menjadi beberapa subproses. Tahapan proses dari proses dapat dilihat pada Gambar 4.10.

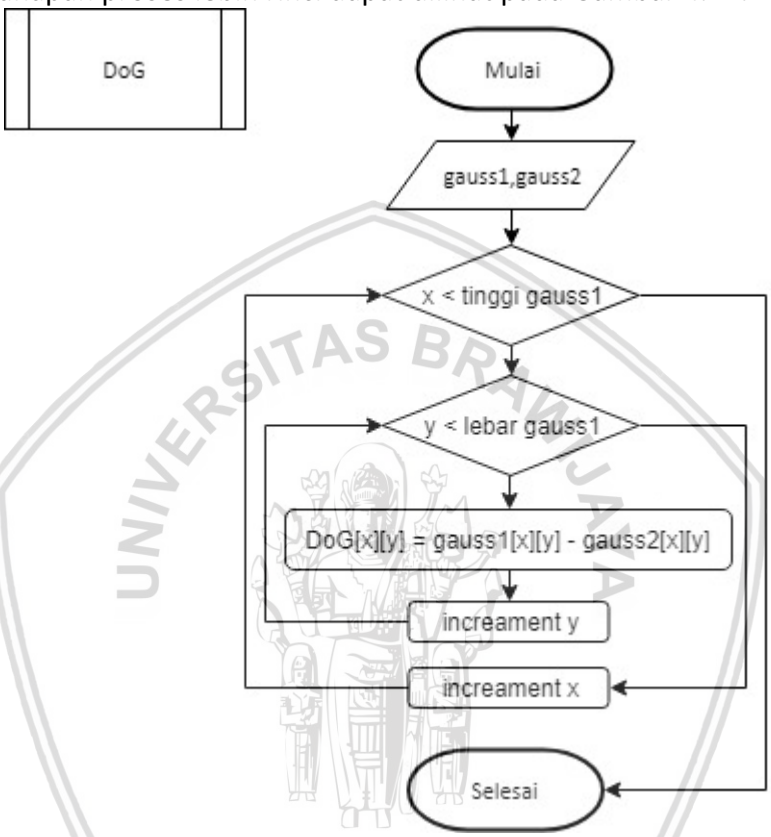


Gambar 4.10 Diagram alir tahapan proses *Keypoint Localization*

Untuk mencari *keypoint* tersebut ada beberapa tahapan proses antara lain proses *difference of Gaussian (DoG), local maxima*, *local minima*, dan terakhir *find r* untuk menentukan *corner*.

1. *Difference of Gaussian* (DoG)

Tahapan pertama untuk mencari *keypoint* adalah mencari *difference of Gaussian* dari masing-masing gambar. Hal ini bertujuan untuk mencari piksel-piksel yang mempunyai nilai lebih besar dari piksel tetangganya. *DoG* akan menghasilkan gambar yang hanya berisi garis-garis halus yang ada pada *edge*. Untuk tahapan proses lebih rinci dapat dilihat pada Gambar 4.11.



Gambar 4.11 Alur Proses *Difference of Gaussian* (DoG)

Pada proses DoG akan digunakan gambar-gambar yang telah difilter pada proses *Gaussian filter*. Nilai piksel dari gambar dalam 1 skala ukuran akan dikurangi dengan nilai piksel dari gambar yang dengan filter yang lebih kuat dalam satu skala ukuran yang sama. Contoh perhitungan manualisasi proses *Difference of Gaussian* dijelaskan pada Gambar 4.12, Gambar 4.13, Gambar 4.14.

0,5930027	0,292599	0,334318	0,539358	0,718099
0,4382748	0,196344	0,549042	0,743061	0,834443
0,3021682	0,244133	0,746204	0,825011	0,865886
0,2720072	0,273102	0,788568	0,837409	0,867182
0,3799703	0,233025	0,658857	0,79479	0,846047

Gambar 4.12 Matriks nilai piksel gambar *Gaussian level 1*

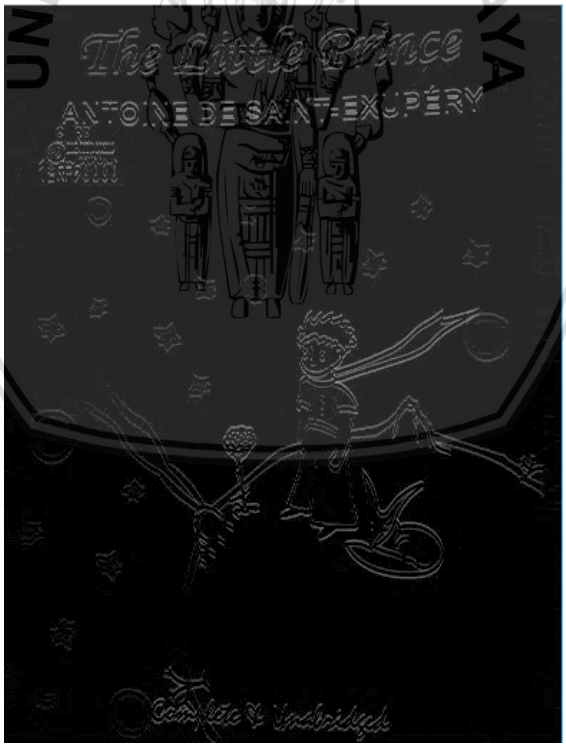
0,579426	0,323815	0,398574	0,527765	0,68166
0,441348	0,240385	0,536565	0,706137	0,81641
0,327873	0,233995	0,705782	0,812411	0,861844
0,310202	0,244219	0,74909	0,831557	0,863519
0,390716	0,250684	0,623872	0,769374	0,83785

Gambar 4.13 Matriks nilai piksel gambar *Gaussian level 2*

0,0135764	-0,03122	-0,06426	0,011593	0,036439
-0,003073	-0,04404	0,012478	0,036924	0,018032
-0,025705	0,010138	0,040422	0,012599	0,004043
-0,038195	0,028884	0,039478	0,005852	0,003663
-0,010746	-0,01766	0,034985	0,025416	0,008197

Gambar 4.14 Hasil *DoG*

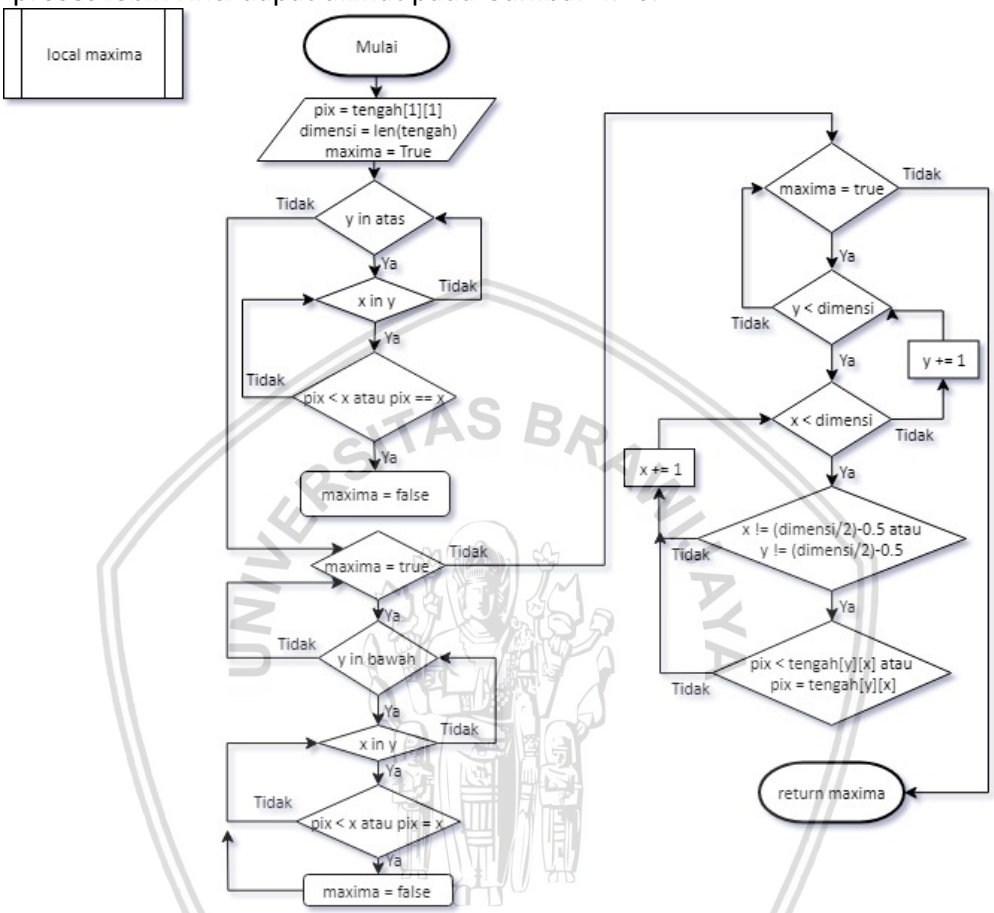
Sebagai contoh gambar yang telah difilter *Gaussian* level 1 pada skala ukuran 1 akan dikurangi gambar yang telah difilter *Gaussian* level 2 pada skala ukuran 1. Apabila nilai piksel dari kedua gambar tersebut dikurangkan maka akan membentuk sebuah gambar *DoG*. Begitu seterusnya sampai semua pada semua *level* filter dan semua skala ukuran. Contoh hasil dari proses *DoG* dapat dilihat pada Gambar 4.15.



Gambar 4.15 Contoh hasil proses DoG

2. *Local Maxima*

Pencarian *local maxima* bertujuan untuk mengetahui apakah nilai suatu piksel adalah nilai piksel terbesar di antara 26 piksel tetangganya. Untuk tahapan proses lebih rinci dapat dilihat pada Gambar 4.16.



Gambar 4.16 Alur Proses *Local Maxima*

Untuk menentukan apakah suatu piksel merupakan nilai maksima, maka digunakan 3 gambar *DoG*. Misal ada gambar *DoG1*, *DoG2*, dan *DoG3* maka yang akan dibandingkan adalah piksel x, y pada *DoG2*. Contoh perhitungan manual dapat dilihat pada Gambar 4.17, Gambar4.18, Gambar 4.19.

0,19634	0,54904	0,74306
0,24413	0,7462	0,82501
0,2731	0,78857	0,83741

Gambar 4.17 Nilai piksel tetangga *DoG1*

0,240385	0,536565	0,706137
0,233995	0,705782	0,812411
0,244219	0,74909	0,831557

Gambar 4.18 Nilai piksel tetangga *DoG2*

-0,04404	0,012478	0,036924
0,010138	0,040422	0,012599
0,028884	0,039478	0,005852

Gambar 4.19 Nilai piksel tetangga *DoG3*

Piksel yang berada pada tengah *DoG2* tersebut akan dibandingkan dengan 8 piksel tetangga di gambar *DoG2*, dan akan dibandingkan dengan 9 piksel tetangga dari *DoG1* dan *DoG3*. Kalau piksel tersebut lebih besar nilainya dari semua piksel tetangganya, maka piksel itu adalah nilai *local maxima*.

3. *Local Minima*

Pencarian local minima bertujuan untuk mengetahui apakah nilai suatu piksel adalah nilai piksel terkecil diantara 26 piksel tetangganya. Untuk proses perhitungannya hampir sama dengan proses menghitung *local maxima*. Apabila *local maxima* mencari nilai tertinggi diantara piksel-piksel maka *local minima* akan mencari nilai terkecil di antara piksel-piksel. Contoh perhitungan dapat dilihat pada Gambar 4.20, Gambar 4.21, dan Gambar 4.22.

0,19634	0,54904	0,74306
0,24413	0,7462	0,82501
0,2731	0,78857	0,83741

Gambar 4.20 Nilai piksel tetangga DoG3

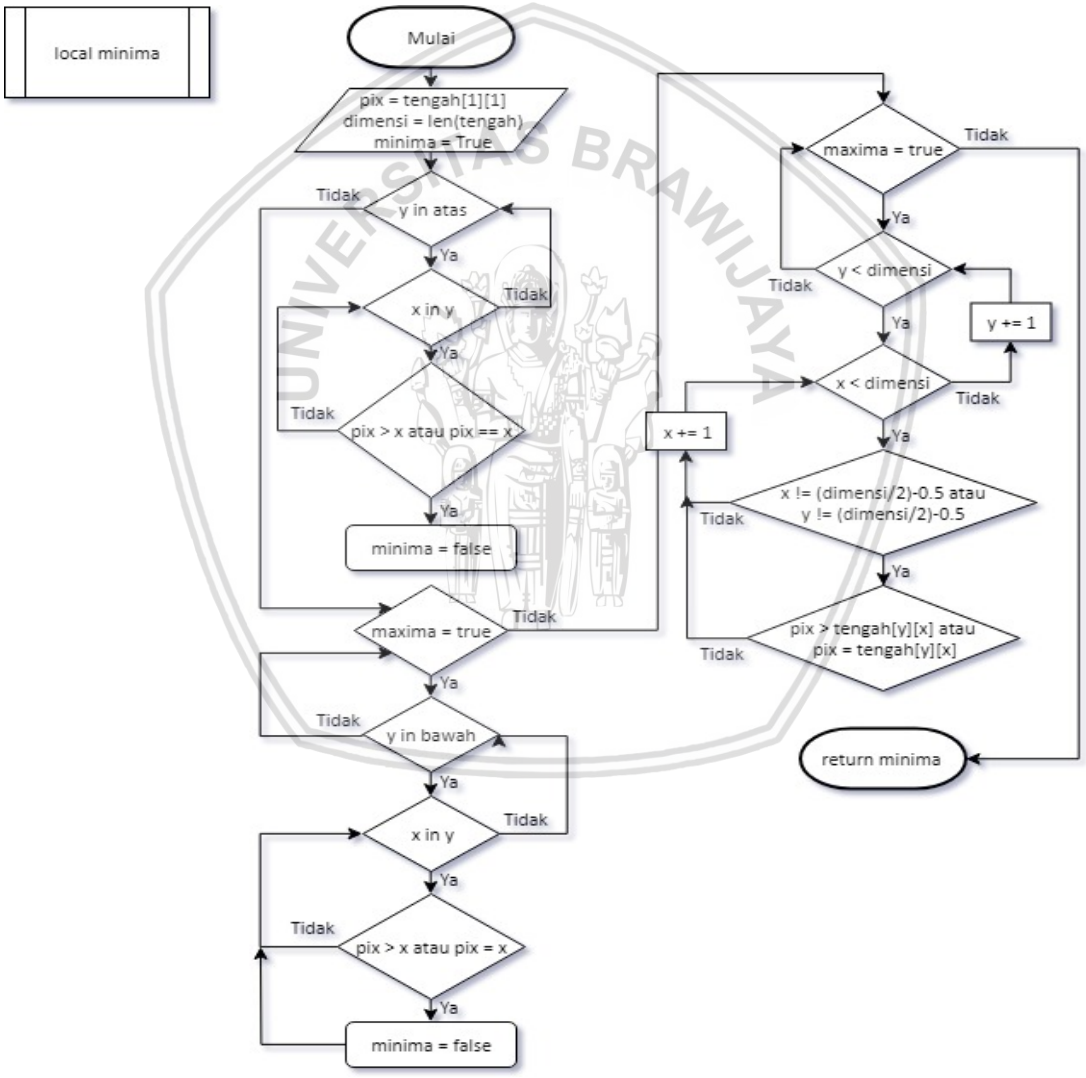
0,240385	0,536565	0,706137
0,233995	0,705782	0,812411
0,244219	0,74909	0,831557

Gambar 4.21 Nilai piksel tetangga DoG3

-0,04404	0,012478	0,036924
0,010138	0,040422	0,012599
0,028884	0,039478	0,005852

Gambar 4.22 Nilai piksel tetangga DoG3

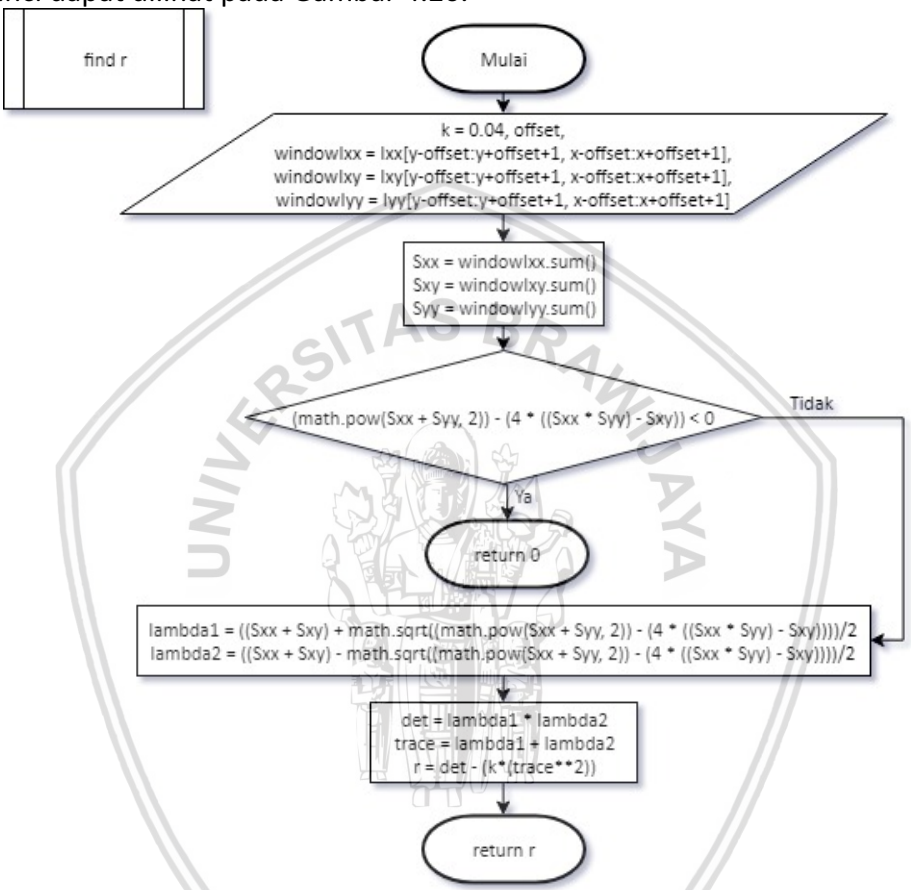
Apabila piksel yang berada pada tengah *DoG2* tersebut lebih kecil dibandingkan dengan 8 piksel tetangganya di *DoG2*, dan lebih kecil dari 9 piksel tetangganya pada *DoG1* dan *Dog3* maka piksel tersebut adalah *local minima*. Untuk tahapan proses lebih rinci dapat dilihat pada Gambar 4.23.



Gambar 4.23 Alur Proses *Local Minima*

4. *Find r*

Pada proses *find r* ini bertujuan untuk mengetahui apakah sebuah piksel terletak pada *edge* atau garis. Untuk membuat *keypoint* dibutuhkan titik piksel yang merupakan sebuah *corner*. Oleh karena itu dibutuhkan proses untuk seleksi calon-calon *keypoint* yang masih berada pada garis dan buka *corner*. Untuk tahapan proses lebih rinci dapat dilihat pada Gambar 4.20.



Gambar 4.24 Alur Proses *Find R*

Pada proses *Find* R digunakan konsep dari algoritme *Harris Corner Detection*. Pada proses ini dicari nilai R yang apabila nilai R tersebut melebihi *threshold* tertentu maka dapat disimpulkan bahwa piksel tersebut adalah sebuah *corner*. Sebelum dilakukan pengecekan apakah dia *corner* pada proses ini, terlebih dahulu diperiksa apakah piksel tersebut adalah piksel yang kontras dengan sekitarnya. Untuk menentukan apakah piksel tersebut kontras dengan sekitar digunakan nilai *gradient magnitude*. Dalam mencari *gradient magnitude* dibutuhkan gambar dengan diturunkan terhadap *x* dan diturunkan terhadap *y*. Contoh perhitungan *gradient* dapat dilihat pada Gambar 4.25 dan 4.26.

-0,02715	0,031216	0	0,011593	0,072878
0,006146	0,044041	0	0,036924	0,036064
0,102821	-0,02028	0	0,025199	0,016171
0,076391	-0,02888	0	0,005852	0,007326
0,021492	0,017659	0	0,025416	0,016394

Gambar 4.25 Nilai piksel turunan terhadap x

0,027153	-0,06243	-0,25702	0,023186	0,072878
-0,00307	-0,04404	0,024955	0,036924	0,018032
0	0	0	0	0
0,038195	-0,02888	-0,07896	-0,00585	-0,00366
0,021492	0,035318	-0,13994	-0,05083	-0,01639

Gambar 4.26 Nilai piksel turunan terhadap x

Setelah didapatkan turunan *x* dan turunan *y*, maka jumlahkan semua piksel tetangganya dalam *windows* 5 x 5. Sebagai contoh nilai jumlah turunan *x* adalah ∑turunan *x* = 0,47726844 dan ∑turunan *y* = -0,39295626, maka *magnitude* dapat dihitung dengan rumus sebagai berikut:

$$magnitude = \sqrt{\frac{\sum turunan\ x^2}{\sum turunan\ y^2}} = \sqrt{\frac{0{,}47726844^2}{-0{,}39295626^2}} = 0{,}618223088$$

Apabila *gradient magnitude* melebihi *threshold* yang ditentukan maka dia mempunyai kontras yang cukup besar dan memenuhi syarat untuk diperiksa apakah *corner* atau bukan.

Setelah *gradient magnitude* memenuhi syarat maka akan diperiksa apakah *corner* atau bukan dengan cara dicari nilai dari turunan *x* kuadrat yang disimpan dalam variable *windowIxx*, turunan *y* kuadrat yang disimpan dalam variabel *windowIyy*, dan turunan *x* dikalikan turunan *y* yang disimpan dalam variabel *windowIxy*. Kemudian turunan-turunan tersebut dicari jumlah keseluruhan pikselnya dengan ukuran *window* 5x5 yang masing-masing disimpan dalam *variable*. Sebagai contoh menggunakan nilai piksel pada Gambar 4.24, Gambar 4.25, dan Gambar 4.26 maka akan didapatkan nilai *windowIxx*, *windowIxy*, dan *windowIxy* masing-masing dengan nilai 0.414729, 0.376031, dan 0.177983. Setelah itu diperiksa nilai *determinant* $(Sxx + Syy)^2 - (4\ (Sxx * Syy) - Sxy)$ apakah nilai *determinant* lebih dari 0. Sebagai contoh *determinant* dari Gambar 4.24, Gambar 4.25, dan Gambar 4.26 adalah:

$$determinant = (Sxx + Syy)^2 - (4\,(Sxx * Syy) - Sxy)$$

$$determinant = (0.414729 + 0.376031)^2 - (4\,(0.414729 * 0.376031) - 0.177983) = 1{,}23658$$

Apabila nilai *determinant* kurang dari sama dengan 0 maka *return* 0, apabila nilai *determinant* lebih dari 0 maka dicari nilai *lambda* satu dan *lambda* duanya dengan rumus sebagai berikut:

$$Lambda\ 1 = \frac{(Sxx+Syy)+\sqrt{(Sxx+Syy)^2-4*((Sxx*Syy)-Sxy)}}{2}$$

$$= \frac{(0.414729+0.376031)+\sqrt{(0.414729+0.376031)^2-4*((0.414729*0.376031)-0.177983)}}{2}$$

$$= 0.468825644$$

$$Lambda\ 2 = \frac{(Sxx+Syy)-\sqrt{(Sxx+Syy)^2-4*((Sxx*Syy)-Sxy)}}{2}$$

$$= \frac{(0.414729+0.376031)-\sqrt{(0.414729+0.376031)^2-4*((0.414729*0.376031)-0.177983)}}{2}$$

$$= 0.123886366$$

Setelah didapatkan nilai *lambda* dicari *determinant* dan *trace* untuk menghitung nilai dari *r* .

Determinan = 0. 468825644 * 0.123886366 = 0.058081105

Trace = 0. 468825644 * 0.123886366 = 0.59271201

r = 0.058081105 – (0.04 * 0.59271201^2) = 0,04402

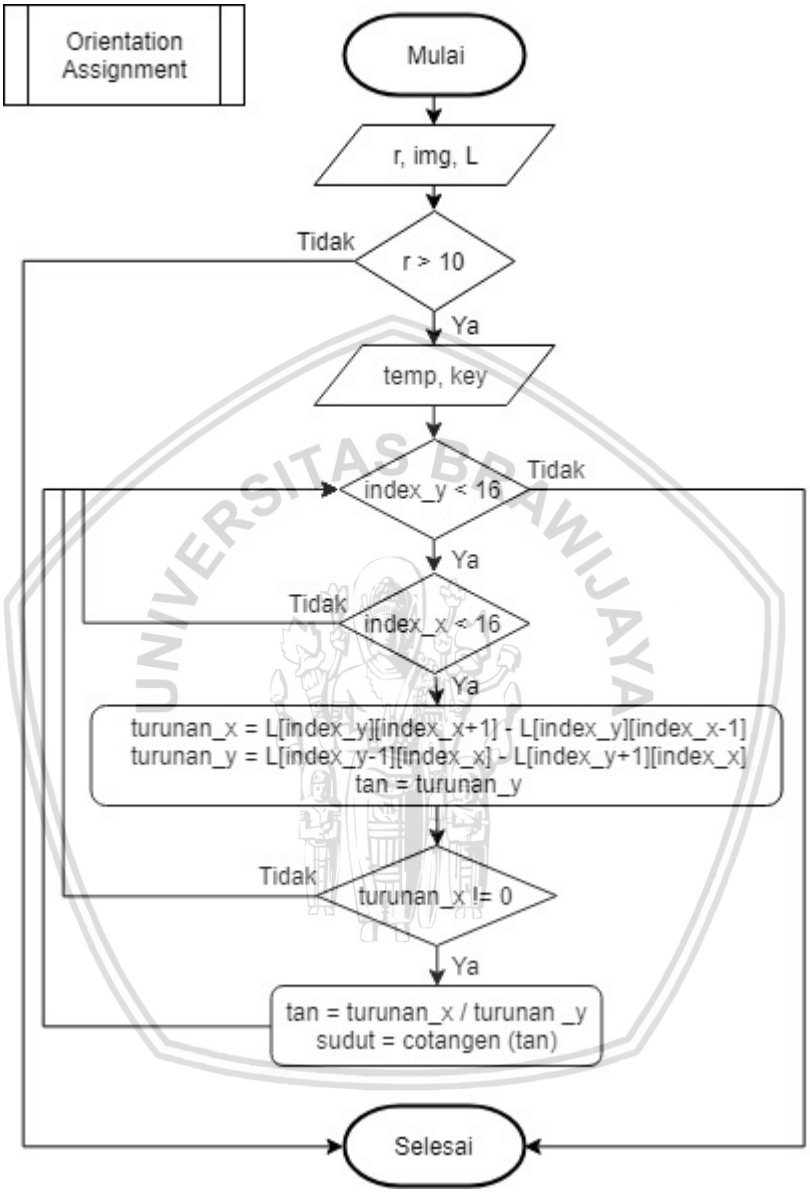
Contoh hasil dari proses *find* r dapat dilihat pada Gambar 4.27.

Gambar 4.27 Hasil proses *find* r

4.2.3 *Orientation Assignment*

Dalam tahapan *orientation assignment* ini akan dilakukan proses yang bertujuan untuk mencari nilai sudut dari masing-masing piksel disekitar *keypoint*.

Sudut dari piksel-piksel tersebut akan berfungsi sebagai bahan untuk membentuk descriptor dari gambar. Proses-proses pada tahapan ini dapat dilihat pada Gambar 4.28.



Gambar 4.28 Alur Proses *Orientation Assignment*

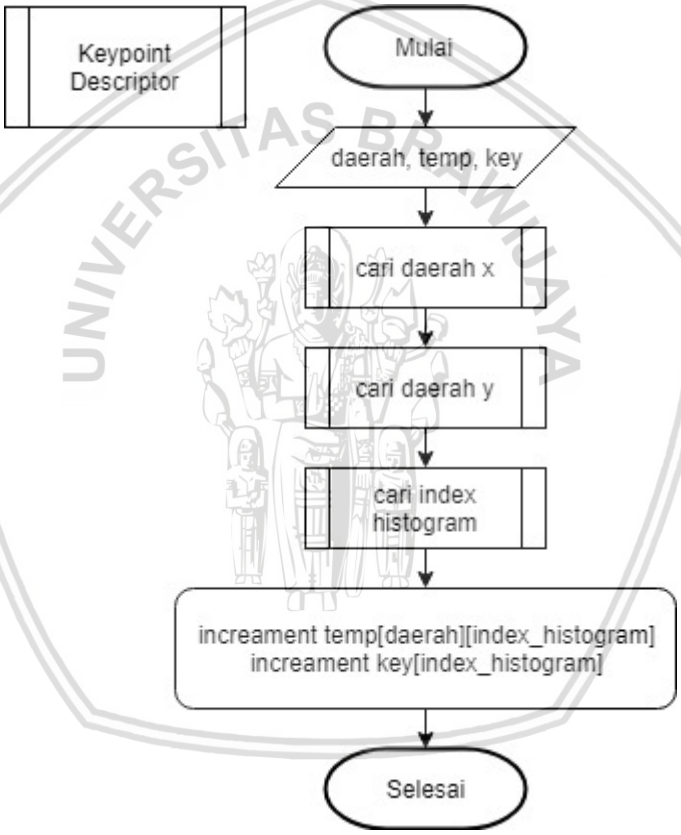
Sebelum mencari nilai sudut dari piksel-piksel tetangga *keypoint*, maka akan terlebih dahulu diperiksa apakah nilai *r* dari piksel tersebut lebih dari 10. Apabila nilai *r* lebih dari 10 maka piksel tersebut berada pada *corner*. Pencarian dari sudut akan dilakukan pada 16 x 16 piksel tetangga *keypoint*.

Untuk mencari sudut pertama cari nilai tan yang dilakukan dengan membagi turunan *y* dengan turunan *x*. Nilai dari turunan *y* dan turunan *x* didapatkan dari gambar yang telah difilter menggunakan *Gaussian* yang dimisalkan dengan L. Setelah

didapatkan nilai tan maka untuk mendapatkan nilai sudut dengan mencari nilai *cotangen* dari tan tersebut.

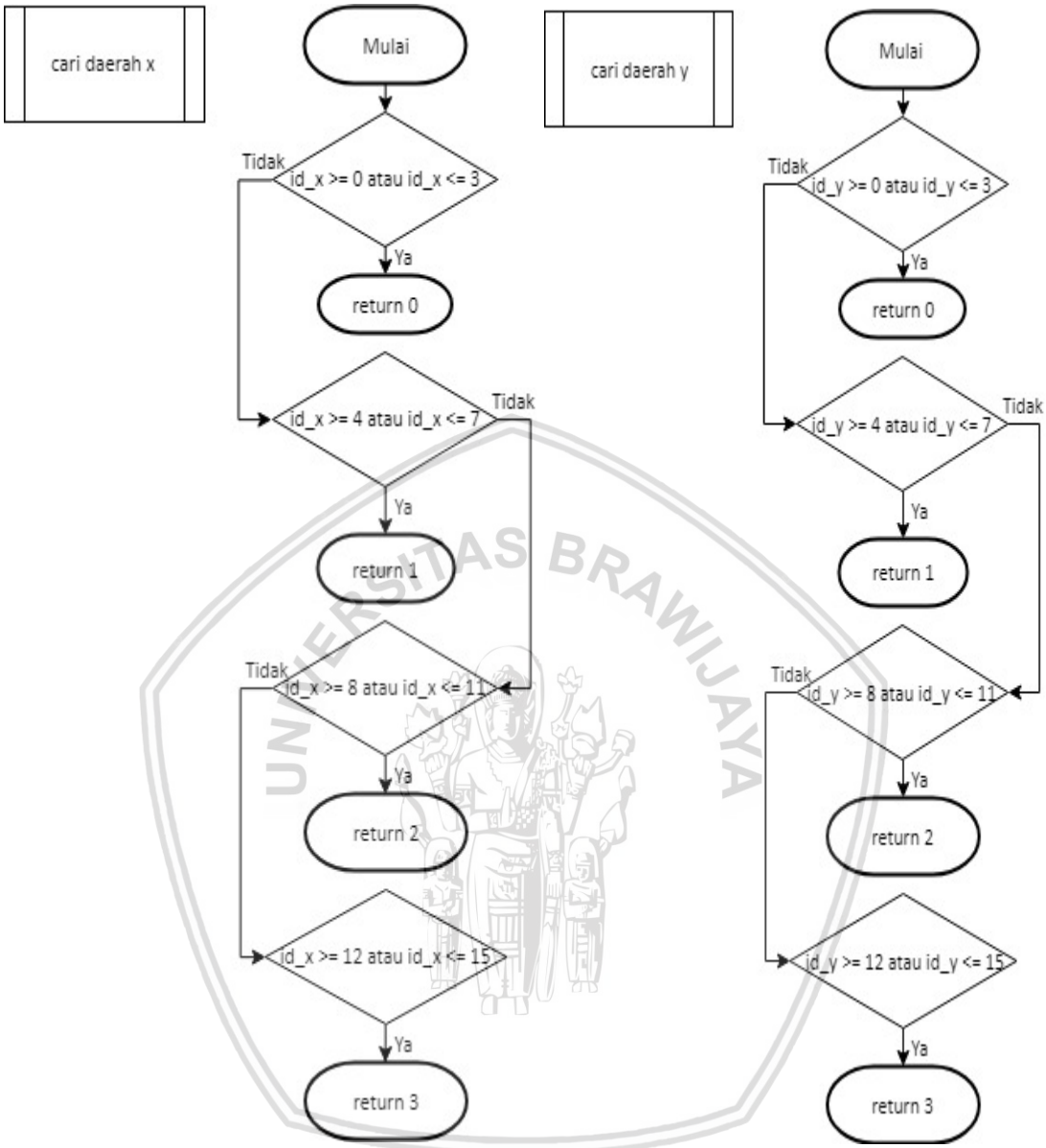
4.2.4 *Keypoint Descriptor*

Pada tahapan ini akan dibentuk *keypoint descriptor*. *Keypoint descriptor* berfungsi untuk mendeskripsikan dari *keypoint*. *Keypoint* yang telah terpilih melalui proses-proses sebelumnya dan akan dibuatkan *histogram* dari sudut-sudut piksel-piksel tetangganya. *Keypoint-keypoint descriptor* inilah yang nantinya digunakan untuk membuat data latih dan *matching* gambar. Tahapan proses untuk pembentukan *descriptor* dari *keypoint* dapat dilihat pada Gambar 4.29.



Gambar 4.29 Alur Proses Pembentukan *Keypoint Descriptor*

Descriptor dari *keypoint* yang dibentuk dengan menggunakan nilai sudut dari 16 x 16 piksel tetangga akan dibagi menjadi 16 bagian, itu berarti masing-masing akan mempunyai cakupan 4 x 4. Setiap 4 x 4 piksel tersebut sudut-sudutnya akan dibentuk menjadi sebuah *histogram* sepanjang 8 bit. Isi dari *histogram* tersebut merepresentasikan nilai sudutnya. Agar dapat menambahkan pada indeks yang benar pada *histogram* maka dibutuhkan proses pencarian *index* x dan *index* y. Proses pencarian *index* x dan *index* y ditunjukkan pada Gambar 4.30.

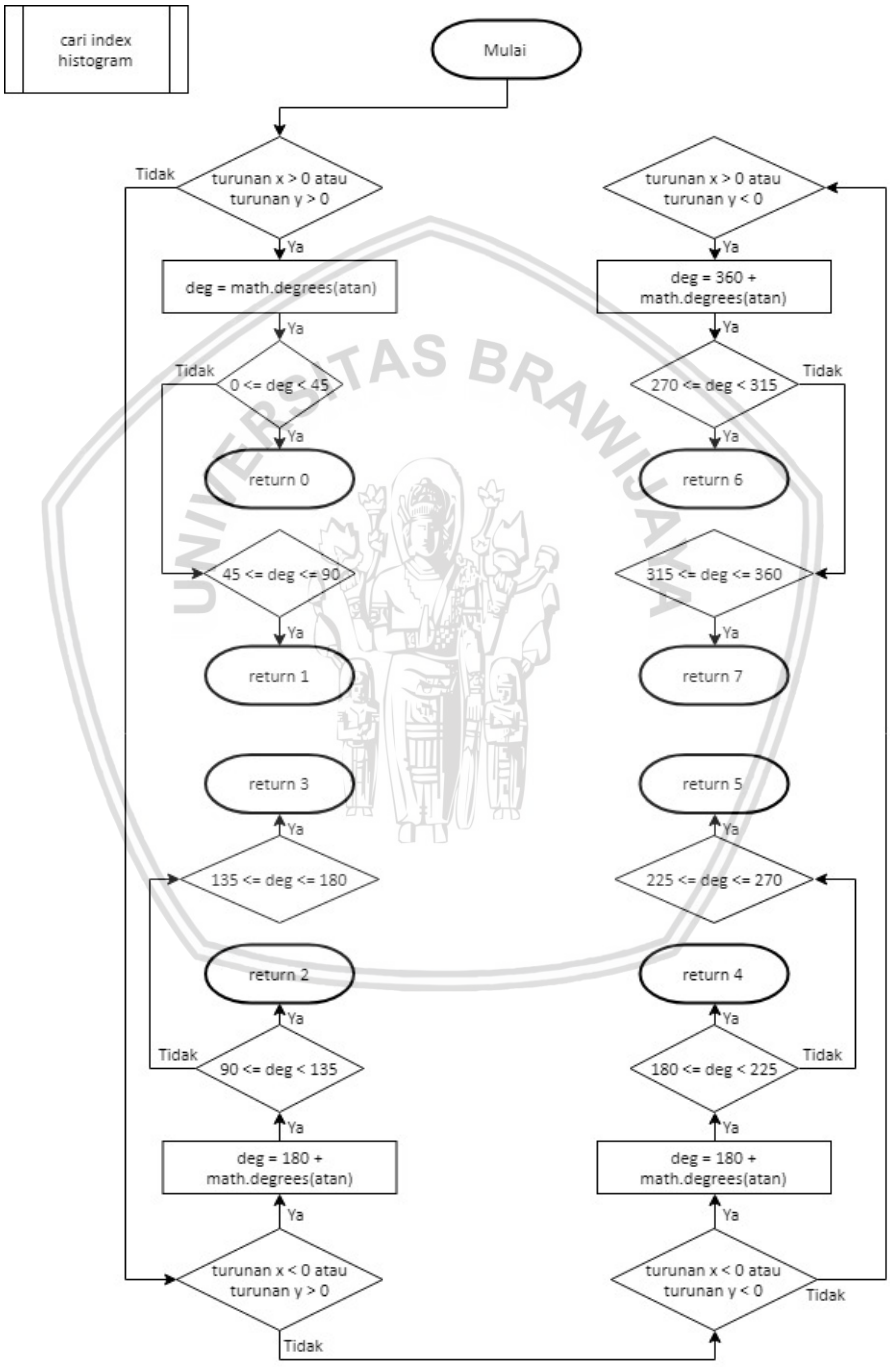


Gambar 4.30 Alur Proses Pencarian *Index* x dan *Index* y

Histogram index pertama berisi jumlah sudut yang bernilai antara 0 – 45, *index* kedua sudut dengan nilai 46 – 90, dan seterusnya sampai index terakhir berisi jumlah piksel dengan nilai sudut 316 – 360. Pembentukan histogram tersebut berlakukan pada semua daerah 4 x 4 tadi.

Pada akhirnya akan semua *histogram* 8 bit dari 16 bagian tadi akan digabung menjadi 1. Gabungan dari semua *histogram* tadi akan berperan sebagai *keypoint descriptor*. Karena akan ada 16 gabungan *histogram* 8 bit, maka panjang total dari *keypoint descriptor* adalah 128 bit.

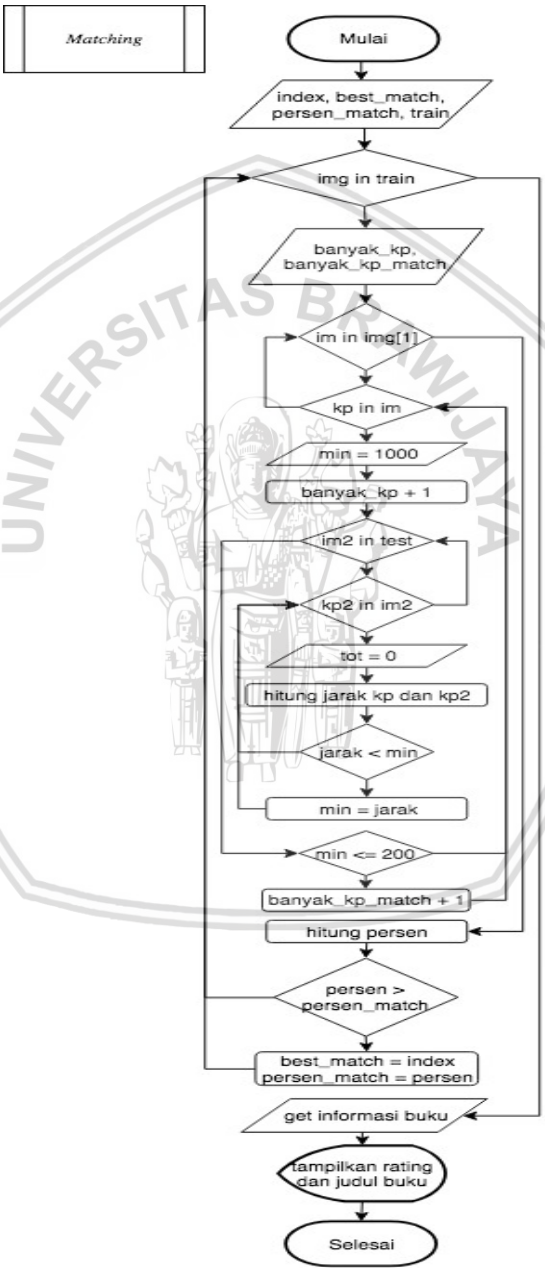
Selain pembentukan *histogram* dengan panjang 128 bit, akan dibentuk juga histogram dengan panjang 8 bit yang berfungsi untuk menyimpan nilai histogram dari keseluruhan orientasi 16x16 piksel tetangga. Agar dapat memasukkan pada *index* histogram yang sesuai maka dibutuhkan proses untuk mencari *index histogram*. Proses untuk mencari nilai *histogram* ditunjukkan pada Gambar 4.31.



Gambar 4.31 Alur Proses Pencarian *Index Histogram*

4.2.5 Proses *Matching*

Proses terakhir adalah proses *matching.* Proses ini bertujuan untuk mencocokkan gambar yang telah dicari keypointnya dengan gambar yang telah dimasukkan data latih sebelumnya. *Output* dari proses ini adalah berupa *index* dari buku tersebut pada halaman *website Goodreads*. Tahapan proses untuk pembentukan *descriptor* dari *keypoint* dapat dilihat pada Gambar 4.32.



Gambar 4.32 Alur Proses *Matching*

Pada proses matching digunakan algoritma *brute force* dengan memeriksa keseluruhan jarak antara *keypoint* dari data latih dengan data uji. Setiap gambar dari data latih akan diambil *descriptor*-nya, kemudian *keypoint* dari *descriptor* tersebut akan dicocokkan dengan *keypoint* yang ada pada data uji.

Proses pencocokan antar *keypoint* menggunakan jarak *eucledian*. Kemudian dicari jarak terkecil untuk menentukan pasangan *keypoint* data latih dan data uji. Setelah itu ditotal keseluruhan *keypoint* yang ada untuk kemudian dibagi dengan jumlah *keypoint* yang mempunyai kecocokan untuk menentukan besar kemiripan antara gambar dari data latih dan gambar dari data uji.

Setelah didapatkan ID dari gambar dengan nilai kemiripan terbesar, kemudian ID tersebut digunakan untuk melakukan pengambilan informasi rating buku dari halaman *website* Goodreads dengan memanfaatkan API dari Goodreads.

4.3 Perancangan Pengujian

Seperti sistem pada umumnya, setelah melakukan proses perancangan sistem, pembuatan sistem, maka tahapan selajutnya adalah melakukan pengujian pada sistem. Tahap pengujian bertujuan untuk mengetahui apakah sistem yang dibuat memiliki fungsi yang sesuai atau belum. Selain itu pengujian juga digunakan untuk menguji besarnya nilai akurasi yang dihasilkan oleh sistem. Pengujian yang akan dilakukan terdiri atas pengujian pengaruh pencahayaan, pengujian pengaruh rotasi pada gambar, dan percobaan perbedaan skala objek pada gambar.

Untuk masing-masing pengujian akan digunakan 20 gambar sebagai data uji. Pengambilan gambar baik data uji dan data latih akan diambil menggunakan kamera dari ponsel IPhone 6. Data latih dan data latih dicantumkan pada lampiran. Dari ketiga pengujian tersebut masing-masing akan menguji akurasi dari judul gambar yang diperoleh.

4.3.1 Perancangan Pengujian Pengaruh Pencahayaan

Pada pengujian ini dilakukan untuk mengetahui apakah nilai pencahayaan dalam ruangan akan memengaruhi hasil akurasi dari sistem. Pengujian ini perlu dilakukan karena pencahayaan sangat mempengaruhi hasil dari gambar yang diperoleh. Pada pengujian ini akan dilakukan dalam dua tahapan. Tahapan pertama akan menggunakan data latih yang didapatkan dari pencahayaan kurang dan mengujinya pada gambar uji yang didapatkan dari pencahayaan cukup dan kurang. Tahap kedua menggunakan data latih yang didapatkan pada tempat dengan pencahayaan yang cukup dan akan diuji menggunakan data uji yang didapatkan pada pencahayaan yang kurang dan cukup. Adapun rancangan detail untuk perancangan pengujian pengaruh pencahayaan dalam ruangan ditunjukkan pada Tabel 4.1 .

Tabel 4.1 Peracangan pengujian pengaruh pencahayaan dalam ruang

Gambar ke-	Judul asli	Judul Terdeteksi	Hasil
1			
2			
3			
...			
20			
Total			
Akurasi			(total/jumlah)*100

4.3.2 Perancangan Pengujian Data Gambar dengan Rotasi

Pada pengujian ini dilakukan untuk mengetahui apakah rotasi pada gambar akan mempengaruhi akurasi dari sistem. Pengujian ini perlu dilakukan karena perbedaan rotasi dan sudut pandang akan mempengaruhi nilai piksel dari gambar-gambar tersebut. Pengujian rotasi akan dilakukan dengan dua macam, yaitu rotasi dengan sudut-sudut ekstrim seperti 90, 180, dan 270, kemudian rotasi dengan sudut-sudut yang lain selain sudut tersebut. Dalam pengujian rotasi ini gambar data uji dan data latih didapatkan diambil pada tempat dengan pencahayaan yang cukup. Adapun rancangan detail untuk perancangan pengujian pengaruh rotasi ditunjukkan pada Tabel 4.3.

Tabel 4.2 Peracangan pengujian pengaruh rotasi gambar

Gambar ke-	Judul asli	Judul Terdeteksi	Sudut	Hasil
1				
2				
3				
...				
20				
Total				
Akurasi				(total/jumlah)*100

4.3.4 Perancangan Pengujian Data Gambar dengan Skala Berbeda-beda

Pada pengujian ini dilakukan untuk mengetahui apakah perbedaan ukuran objek akan memengaruhi akurasi dari sistem. Pengujian ini perlu dilakukan karena perbedaan jarak pengambilan gambar akan memengaruhi ukuran objek yang terlihat pada gambar, hal ini otomatis juga akan memengaruhi nilai-nilai dari piksel. Pengujian skala akan dilakukan dengan mengambil gambar dari jarak yang berbeda-beda, yaitu dari jarak 10 cm, 15 cm, dan 30 cm. Gambar yang digunakan pada pengujian diambil pada tempat dengan pencahayaan yang cukup dan tanpa adanya rotasi. Adapun rancangan detail untuk perancangan pengujian pengaruh skala ditunjukkan pada Tabel 4.4.

Tabel 4.3 Perancangan pengujian pengaruh skala

Gambar ke-	Judul asli	Judul Terdeteksi	Hasil
1			
2			
3			
...			
20			
Total			
Akurasi			(total/jumlah)*100

BAB 5 IMPLEMENTASI

5.1 Deskripsi Lingkungan Implementasi

Pada deskripsi lingkungan implementasi ini bertujuan untuk merancang sistem dalam ruang lingkup yang jelas sesuai dengan tujuan sistem, serta perancangan yang sudah dibuat pada bab sebelumnya. Berikut merupakan deksripsi lingkungan implementasi yang ada pada penelitian yang akan dibangun ini.

1. Penggunaan perangkat keras
   - Laptop dengan processor Intel® Core i5 CPU @2.6GHz
   - RAM 4.00 GB
   - *Memory* 128 GB
2. Penggunaan perangkat lunak
   - Sistem Operasi *Windows* 8 64-bit
   - Mac Os Sierra High
   - *Anaconda-Navigator*
   - *Microsoft Word 2010*
   - *Microsoft Excel 2010*

5.2 Implementasi Sistem

Pada implementasi algoritme akan dibahas mengenai implementasi algoritme yang telah dilakukan. Dalam pengerjaan implementasi algoritme pada bab ini, penulis merujuk pada perancangan algoritme yang telah dijabarkan serta telah dibuat sebelumnya pada subbab 4.2.

5.2.1 Implementasi Pengambilan Data *Input*

Yang pertama dari proses implementasi ini adalah pengambilan data *input* gambar dan pengambilan data store.pckl yang berfungsi untuk menyimpan data *keypoint* yang telah disimpan. Selain itu didefinisikan juga variabel-variabel yang dibutuhkan seperti *ksize*, *k*, dan *sigma*. Setelah berhasil mengambil data gambar dicari juga nilai panjang dan lebarnya. Implementasi sistem untuk proses pengambilan data ditunjukan dengan potongan kode program pada Kode Program 5.1.

Kode Program 5.1 Implementasi Proses *Prepocessing* Data *Input*

Baris Ke-	Kode Program
1	`f = open('store.pckl', 'rb')`
2	`train = pickle.load(f)`
3	`f.close()`
4	`ksize = 5`
5	`k = math.sqrt(2)`

```
6   sigma = k/2
7   img = cv2.imread('img.jpeg)
8   height, width = img.shape[:2]img = cv2.resize(img,
    (int(width/5), int (height/5)), interpolation =
    cv2.INTER_CUBIC)
```

Keterangan Kode Program 5.1:

Baris 1 - 3 : Membuka *file store* yang berfungsi untuk menyimpan data *keypoint* dari data latih.
Mengubah ukuran gambar yang telah dimasukkan menjadi 1/5.

Baris 4 : Pendefinisian *variable ksize* yang berfungsi sebagai ukuran *window*.

Baris 5 : Pendefinisian konstanta *k*.

Baris 6 : Pendefinisian *variable sigma* yang berguna untuk mendapatkan *Gaussian kernel*.

Baris 7 : Mengambil *file* gambar pada *defice*.

Baris 8 : Mengambil nilai lebar dan tinggi dari gambar.

5.2.2 Implementasi Proses *Scale Space Extrema Detection*

Setelah semua data yang dibutuhkan sudah didapatkan maka langkah pertama adalah merubah ukuran gambar sesuai dan sebanyak yang dibutuhkan. Untuk memenuhi *Scale Space* ini dibutuhkan 4 macam ukuran gambar seperti yang telah dijelaskan pada Gambar 4.3 sebelumnya. Implementasi sistem untuk *resize image* ditunjukan dengan potongan kode program pada Kode Program 5.2.

Kode Program 5.2 Implementasi Proses *Resize Image*

Baris Ke-	Kode Program
1	`img = cv2.resize(img, (int(width/5), int (height/5)), interpolation = cv2.INTER_CUBIC)`
2	`img1 = img.copy()`
3	`img2 = cv2.resize(img1, None, fx=0.5, fy=0.5, interpolation = cv2.INTER_CUBIC)`
4	`img3 = cv2.resize(img2, None, fx=0.5, fy=0.5, interpolation = cv2.INTER_CUBIC)`
5	`img4 = cv2.resize(img3, None, fx=0.5, fy=0.5, interpolation = cv2.INTER_CUBIC)`

Keterangan Kode Program 5.2:

Baris 1 : Mengubah ukuran gambar yang telah dimasukkan menjadi 1/5.

Baris 2 : Membuat salinan *img* dan menyimpannya pada *img1*.

Baris 3 : Memperkecil ukuran *img* menjadi setengahnya dan menyimpan pada *variable img2*.

Baris 4 : Memperkecil ukuran *img* menjadi seperempat dan menyimpannya pada *variable img3*.

Baris 5 : Memperkecil ukuran *img* menjadi seperempat dan menyimpannya pada *variable img4*.

Setelah didapatkan ukuran gambar mulai dari yang terbesar sampai yang terkecil satu per delapannya, maka gambar-gambar tersebut akan diubah ke dalam bentuk *grayscale* seperti yang telah dijelaskan pada Gambar 4.2 dan 4.5. Gambar yang pada awalnya mempunyai format RGB diubah menjadi *grayscale* dengan tujuan memperkecil komputasi. Implementasi sistem untuk mengubah gambar menjadi *grayscale* ditunjukan dengan potongan kode program pada Kode Program 5.3.

Kode Program 5.3 Implementasi Proses *Convert Gray Scale*

Baris Ke-	Kode Program
1	`level_1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)`
2	`level_2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)`
3	`level_3 = cv2.cvtColor(img3, cv2.COLOR_BGR2GRAY)`
4	`level_4 = cv2.cvtColor(img4, cv2.COLOR_BGR2GRAY)`
5	`level_1 = cv2.normalize(level_1, level_1.shape, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)`
6	`level_2 = cv2.normalize(level_2, level_2.shape, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)`
7	`level_3 = cv2.normalize(level_3, level_3.shape, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)`
8	`level_4 = cv2.normalize(level_4, level_4.shape, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)`

Keterangan Kode Program 5.3:

Baris 1 – 4 : Mengubah warna dari gambar yang awalnya berformat BGR menjadi *grayscale*.

Baris 5 – 8 : Menormalisasi nilai *grayscale* menjadi 0 sampai 1 agar tidak terlalu besar saat proses komputasi.

Proses selanjutnya adalah melakukan proses *Gaussian filter*. *Gaussian filter* akan menyebabkan gambar menjadi lebih halus seperti yang telah dijelaskan pada Gambar 4.7. Semakin besar nilai *sigma* yang dipakai dalam Gaussian filter ini gambar akan semakin halus dan semakin kabur (*blur*). Berikut proses implementasi *Gaussian filter*. Implementasi sistem untuk proses *Gaussian filter* ditunjukan dengan potongan kode program pada Kode Program 5.4.

Kode Program 5.4 Implementasi Proses *Gaussian Filter*

Baris Ke-	Kode Program
1	`GaussianKernel0 = cv2.getGaussianKernel(ksize, sigma, cv2.CV_32F)`
2	`GaussianKernel1 = cv2.getGaussianKernel(ksize, sigma*math.pow(k, 1), cv2.CV_32F)`
3	`GaussianKernel2 = cv2.getGaussianKernel(ksize, sigma*math.pow(k, 2), cv2.CV_32F)`
4	`GaussianKernel3 = cv2.getGaussianKernel(ksize, sigma*math.pow(k, 3), cv2.CV_32F)`

```
        GaussianKernel4 = cv2.getGaussianKernel(ksize,
5       sigma*math.pow(k, 4), cv2.CV_32F)
        GaussianKernel5 = cv2.getGaussianKernel(ksize,
6       sigma*math.pow(k, 5), cv2.CV_32F)
        GaussianKernel6 = cv2.getGaussianKernel(ksize,
7       sigma*math.pow(k, 6), cv2.CV_32F)
        GaussianKernel7 = cv2.getGaussianKernel(ksize,
8       sigma*math.pow(k, 7), cv2.CV_32F)
        GaussianKernel8 = cv2.getGaussianKernel(ksize,
9       sigma*math.pow(k, 8), cv2.CV_32F)
        GaussianKernel9 = cv2.getGaussianKernel(ksize,
10      sigma*math.pow(k, 9), cv2.CV_32F)
        GaussianKernel10 = cv2.getGaussianKernel(ksize,
11      sigma*math.pow(k, 10), cv2.CV_32F)
        GaussianKernel = cv2.getGaussianKernel(ksize, sigma*1.5,
12      cv2.CV_32F)
        gauss1 = cv2.filter2D(level_1, -1, GaussianKernel)
13      gauss2 = cv2.filter2D(level_2, -1, GaussianKernel)
14      gauss3 = cv2.filter2D(level_3, -1, GaussianKernel)
15      gauss4 = cv2.filter2D(level_4, -1, GaussianKernel)
16      gauss1_level1 = cv2.filter2D(level_1, -1,
17      GaussianKernel0)
        gauss2_level1 = cv2.filter2D(level_1, -1,
18      GaussianKernel1)
        gauss3_level1 = cv2.filter2D(level_1, -1,
19      GaussianKernel2)
        gauss4_level1 = cv2.filter2D(level_1, -1,
20      GaussianKernel3)
        gauss5_level1 = cv2.filter2D(level_1, -1,
21      GaussianKernel4)
        gauss1_level2 = cv2.filter2D(level_2, -1,
        GaussianKernel2)
22      gauss2_level2 = cv2.filter2D(level_2, -1,
        GaussianKernel3)
23      gauss3_level2 = cv2.filter2D(level_2, -1,
        GaussianKernel4)
24      gauss4_level2 = cv2.filter2D(level_2, -1,
        GaussianKernel5)
25      gauss5_level2 = cv2.filter2D(level_2, -1,
        GaussianKernel6)
26      gauss1_level3 = cv2.filter2D(level_3, -1,
        GaussianKernel4)
27      gauss2_level3 = cv2.filter2D(level_3, -1,
        GaussianKernel5)
28      gauss3_level3 = cv2.filter2D(level_3, -1,
        GaussianKernel6)
29      gauss4_level3 = cv2.filter2D(level_3, -1,
        GaussianKernel7)
30      gauss5_level3 = cv2.filter2D(level_3, -1,
        GaussianKernel8)
31      gauss1_level4 = cv2.filter2D(level_4, -1,
        GaussianKernel6)
32      gauss2_level4 = cv2.filter2D(level_4, -1,
        GaussianKernel7)
```

```
33  gauss3_level4 = cv2.filter2D(level_4, -1,
    GaussianKernel8)
34  gauss4_level4 = cv2.filter2D(level_4, -1,
    GaussianKernel9)
35  gauss5_level4 = cv2.filter2D(level_4, -1,
    GaussianKernel10)
36
```

Keterangan Gambar 5.4:

Baris 1 – 12 : Pendefinisian Gaussian kernel untuk proses *filtering*. Dibutuhkan bermacam *Gaussian filter* untuk kepentingan proses *DoG*. Untuk mendapatkan *kernel Gaussian* digunakan fungsi dari OpenCV yang membutuhkan parameter *window size* (*ksize*), dan *level* dari *blurring* yang menggunakan *variable sigma*.

Baris 13 – 16 : *Filtering image* level 1 sampai 4 menggunakan *Gaussian filter* untuk digunakan saat pencarian *sobel*, yang nantinya berguna untuk mencari nilai *gradient*.

Baris 17 – 21 : *Filtering image level* 1 dengan *kernel Gaussian* 0 sampai 4. Hasil dari *filtering* ini nantinya akan digunakan pada proses *DoG level 1*.

Baris 22 – 26 : *Filtering image level* 2 dengan *kernel Gaussian* 2 sampai 6. Hasil dari *filtering* ini nantinya akan digunakan pada proses *DoG level 2*.

Baris 27 – 31 : *Filtering image level* 3 dengan *kernel Gaussian* 4 sampai 8. Hasil dari *filtering* ini nantinya akan digunakan pada proses *DoG level 3*.

Baris 32 – 36 : *Filtering image level* 4 dengan *kernel Gaussian* 6 sampai 10. Hasil dari *filtering* ini nantinya akan digunakan pada proses *DoG level 4*.

5.2.3 Implementasi Proses *Keypoint Localization*

Setelah melakukan proses pembentukan *scale space* berhasil maka proses selanjutnya adalah melakukan *keypoint localization*. Proses ini bertujuan untuk menyeleksi piksel-piksel yang memenuhi syarat untuk menjadi *keypoint* seperti yang telah dijelaskan pada Gambar 4.10. Implementasi dari sistem untuk proses *keypoint localization* ditunjukan dengan potongan kode program pada Kode Program 5.5.

Kode Program 5.5 Implementasi Proses *Keypoint Localization*

Baris Ke-	Kode Program
1	`hasil = img.copy()`
2	`tinggi = tengah.shape[0]`
3	`lebar = tengah.shape[1]`
4	`offset = (size-1)/2`
5	`Ix = cv2.Sobel(tengah, cv2.CV_64F, 1, 0, ksize=size)`
6	`Iy = cv2.Sobel(tengah, cv2.CV_64F, 0, 1, ksize=size)`
7	`Ixx = Ix**2`
8	`Ixy = Ix*Iy`
9	`Iyy = Iy**2`

```
10  keypoint = []
11  kp = []
12  for y in range(1, tinggi-1):
13    for x in range(1, lebar-1):
14      tetangga_atas = atas[y-1:y+2, x-1:x+2]
15      tetangga_tengah = tengah[y-1:y+2, x-1:x+2]
16      tetangga_bawah = bawah[y-1:y+2, x-1:x+2]
17      maxima = local_maxima(tetangga_atas, tetangga_tengah,
        tetangga_bawah)
18      minima = local_minima(tetangga_atas, tetangga_tengah,
        tetangga_bawah)
19      if (maxima == True) or (minima == True):
20        mag = math.sqrt(math.pow(Ix[y][x], 2) +
        math.pow(Iy[y][x], 2))
21        if mag > 0.03:
22          r = find_r(Ixx, Ixy, Iyy, y, x, offset)
23          if (r > 10):
24            temp = []
25            key = [0, 0, 0, 0, 0, 0, 0, 0]
26            for i in range(16):
27              temp.append([0, 0, 0, 0, 0, 0, 0, 0])
28              cv2.circle(hasil, (x, y), 4, (0, 255, 0), 2)
29              id_y = 0
30              for index_y in range(y-8, y+8, 1):
31                id_x = 0
32                for index_x in range(x-8, x+8, 1):
33                  daerah = cari_daerah_x(id_x)
34                  daerah += cari_daerah_y(id_y)
35                  turunan_x = L[index_y][index_x+1] -
        L[index_y][index_x-1]
36                  turunan_y = L[index_y-1][index_x] -
        L[index_y+1][index_x]
37                  tan = turunan_y
38                  if turunan_x != 0:
39                    tan = turunan_y / turunan_x
40                  atan = math.atan(tan)
41                  index_histogram =
        cari_index_histogram(turunan_x, turunan_y, atan);
42                  temp[daerah][index_histogram] += 1
43                  key[index_histogram] += 1
44                  id_x += 1
45                id_y += 1
46              keypoint.append(temp)
47              kp.append(key)
48  return hasil, keypoint, kp
```

Keterangan Kode Program 5.5:

Baris 1 : Membuat salinan dari gambar asli untuk ditandai bagian yang terdeteksi sebagai *keypoint*.

Baris 2 – 3 : Mengambil nilai tinggi dan lebar dari gambar.

Baris 4 : Pendefinisian *variable offset* untuk digunakan pada proses pemeriksaan *corner*.

Baris 5 – 6 : Mencari turunan dari gambar terhadap x dan y.
Baris 7 – 9 : Mancari nilai kuadrat dari turunan x dan turunan y dan mencari nilai turunan x yang dikalikan dengan turunan y.
Baris 10 – 11 : Pendeklarasian *variable* untuk menyimpan hasil *keypoint*.
Baris 12 : Perulangan untuk memperoleh *index* baris dari gambar.
Baris 13 : Perulangan untuk memperoleh *index* kolom dari gambar.
Baris 14 – 16 : Mengambil 9 piksel tetangga dari DoG yang di atasnya, di bawahnya dan juga 8 piksel tetangga dari piksel di DoG yang sama.
Baris 17 – 18 : Pencarian *local maxima* dan *local minima*.
Baris 19 : Kondisi jika piksel tersebut adalah *local maxima* atau *local minima.*
Baris 20 : Maka hitung nilai *magnitude*.
Baris 21 : Jika nilai *magnitude* lebih dari *threshold* yang sebesar 0.03.
Baris 22 : Maka cari nilai *r*.
Baris 23 : Jika nilai *r* lebih dari 10 maka piksel tersebut adalah sebuah *corner*.
Baris 24 – 25 : Apabila kondisi *corner* terpenuhi maka buat *variable temp* dan *histogram keypoint* secara umum.
Baris 26 : Perulangan sebanyak 16 kali karena *keypoint* akan dipecah menjadi 16 buah *histogram*.
Baris 27 : Pendefinisian awal *histogram*.
Baris 28 : Menandai piksel yang telah dianggap sebagai *keypoint* pada gambar dengan sebuah lingkaran.
Baris 29 : Pendeklarasian *index* y atau baris dari tetangga piksel *keypoint*.
Baris 30 : Perulangan untuk mendapatkan baris dari 32 x 32 piksel tetangga.
Baris 31 : Pendeklarasian *index* x atau kolom dari tetangga piksel *keypoint*.
Baris 32 : Perulangan untuk mendapatkan kolom dari 32 x 32 piksel tetangga.
Baris 33 – 34 : Mencari nilai daerah. Nilai daerah untuk mengetahui *histogram* ke-berapa yang akan diisi.
Baris 35- 36 : Mencari nilai dari turunan x dan turunan y untuk digunakan sebagai bahan perhitungan orientasi.
Baris 37 : Perhitungan nilai *tangen*.
Baris 38 : Kondisi jika *turunan_x* !=0.
Baris 39 : Maka nilai tangen adalah hasil dari pembagian turunan x dibagi dengan turunan y.
Baris 40 : Perhitungan nilai *arctangent*.
Baris 41 : Mencari nilai *index* dari *histogram* yang akan diisi berdasarkan hasil orientasi dari *arctangent*.
Baris 42 - 43 : Menyimpan nilai dari *histogram*.
Baris 44 : *Increament* nilai *index* x.
Baris 45 : Increament nilai *index* y.

Baris 46 – 47 : Menyimpan nilai histogram dari *keypoint*.

Baris 48 : Mengembalikan hasil dari pendeteksian *keypoint*.

Proses pertama dari *keypoint localization* adalah mencari gambar *DoG*, yang berfungsi untuk mengubah gambar menjadi garis-garis seperti yang telah dijelaskan pada 4.11 sampai dengan Gambar 4.14. Implementasi sistem untuk mencari gambar *DoG* ditunjukan dengan potongan kode program pada Kode Program 5.6.

Kode Program 5.6 Implementasi Proses *DoG*

Baris Ke-	Kode Program
1	`# DoG Level 1`
2	`dog1_level1 = level_1.copy()`
3	`dog2_level1 = level_1.copy()`
4	`dog3_level1 = level_1.copy()`
5	`dog4_level1 = level_1.copy()`
6	`dog5_level1 = level_1.copy()`
7	`for y in range(dog1_level1.shape[0]):`
8	`  for x in range(dog1_level1.shape[1]):`
9	`  dog1_level1[y][x] = gauss1_level1[y][x] - gauss2_level1[y][x]`
10	`  dog2_level1[y][x] = gauss2_level1[y][x] - gauss3_level1[y][x]`
11	`  dog3_level1[y][x] = gauss3_level1[y][x] - gauss4_level1[y][x]`
12	`  dog4_level1[y][x] = gauss4_level1[y][x] - gauss5_level1[y][x]`
13	
14	`# DoG Level 2`
15	`dog1_level2 = level_2.copy()`
16	`dog2_level2 = level_2.copy()`
17	`dog3_level2 = level_2.copy()`
18	`dog4_level2 = level_2.copy()`
19	`dog5_level2 = level_2.copy()`
20	`for y in range(dog1_level2.shape[0]):`
21	`  for x in range(dog1_level2.shape[1]):`
22	`  dog1_level2[y][x] = gauss1_level2[y][x] - gauss2_level2[y][x]`
23	`  dog2_level2[y][x] = gauss2_level2[y][x] - gauss3_level2[y][x]`
24	`  dog3_level2[y][x] = gauss3_level2[y][x] - gauss4_level2[y][x]`
25	`  dog4_level2[y][x] = gauss4_level2[y][x] - gauss5_level2[y][x]`
26	
27	`# DoG Level 3`
28	`dog1_level3 = level_3.copy()`
29	`dog2_level3 = level_3.copy()`
30	`dog3_level3 = level_3.copy()`
31	`dog4_level3 = level_3.copy()`
32	`dog5_level3 = level_3.copy()`

```
33  for y in range(dog1_level3.shape[0]):
34       for x in range(dog1_level3.shape[1]):
35            dog1_level3[y][x] = gauss1_level3[y][x] -
    gauss2_level3[y][x]
36            dog2_level3[y][x] = gauss2_level3[y][x] -
    gauss3_level3[y][x]
37            dog3_level3[y][x] = gauss3_level3[y][x] -
    gauss4_level3[y][x]
38            dog4_level3[y][x] = gauss4_level3[y][x] -
    gauss5_level3[y][x]
39
40  # DoG Level 4
41  dog1_level4 = level_4.copy()
42  dog2_level4 = level_4.copy()
43  dog3_level4 = level_4.copy()
44  dog4_level4 = level_4.copy()
45  dog5_level4 = level_4.copy()
46  for y in range(dog1_level4.shape[0]):
47       for x in range(dog1_level4.shape[1]):
48            dog1_level4[y][x] = gauss1_level4[y][x] -
    gauss2_level4[y][x]
             dog2_level4[y][x] = gauss2_level4[y][x] -
49  gauss3_level4[y][x]
             dog3_level4[y][x] = gauss3_level4[y][x] -
50  gauss4_level4[y][x]
             dog4_level4[y][x] = gauss4_level4[y][x] -
51  gauss5_level4[y][x]
```

Keterangan Kode Program 5.6:

Baris 2 – 6	:	Membuat salinan dari *image* level_4 untuk digunakan sebagai wadah default DoG level 4.
Baris 7	:	Perulangan untuk setiap *index* y dari piksel pada *image*.
Baris 8	:	Perulangan untuk setiap *index* x dari piksel pada *image* baris y.
Baris 9 – 12	:	Mengurangi nilai dari *gauss1* dengan *gauss2* untuk disimpan pada *dog1*, *gauss2* dengan *gauss3* untuk disimpan pada *dog2* dan seterusnya. *Gaussian* yang dipakai adalah dari *image level 4*.
Baris 15 – 19	:	Membuat salinan dari *image level_2* untuk digunakan sebagai wadah *default DoG level 2.*
Baris 20	:	Perulangan untuk setiap *index* y dari piksel pada *image*.
Baris 21	:	Perulangan untuk setiap *index* x dari piksel pada *image* baris y.
Baris 22 – 25	:	Mengurangi nilai dari *gauss1* dengan *Gaussian* 2 untuk disimpan pada *dog1*, *gauss2* dengan *gauss3* untuk disimpan pada *dog2* dan seterusnya. *Gaussian* yang dipakai adalah dari *image level 2*.

Baris 28 – 32 : Membuat salinan dari *image level_3* untuk digunakan sebagai wadah default *DoG level 3*.

Baris 33 : Perulangan untuk setiap *index y* dari piksel pada *image*.

Baris 34 : Perulangan untuk setiap *index x* dari piksel pada *image baris y.*

Baris 35 – 38 : Mengurangi nilai dari *gauss1* dengan *gauss2* untuk disimpan pada *dog1*, *gauss2* dengan *gauss3* untuk disimpan pada *dog2* dan seterusnya. *Gaussian* yang dipakai adalah dari *image level3*.

Baris 41 – 45 : Membuat salinan dari *image level_4* untuk digunakan sebagai wadah *default DoG level* 4.

Baris 46 : Perulangan untuk setiap *index y* dari piksel pada *image*.

Baris 47 : Perulangan untuk setiap *index x* dari piksel pada *image* baris *y*.

Baris 48 – 51 : Mengurangi nilai dari *gauss1* dengan *gauss2* untuk disimpan pada *dog1*, *gauss2* dengan *gauss3* untuk disimpan pada *dog2* dan seterusnya. *Gaussian* yang dipakai adalah dari *image level 4*.

Setelah didapatkan gambar *DoG* yang dibutuhkan dari masing-masing *level* proses selanjutnya adalah memeriksa apakah piksel tersebut adalah *local extrema* dengan cara memeriksa *local maxima* dan *local minima* seperti yang telah dijelaskan pada Gambar 4.16 dan Gambar 4.23. Apabila piksel tersebut adalah *local extrema* maka akan diproses ke proses selanjutnya. Implementasi sistem untuk memeriksa *local extrema* ditunjukan dengan potongan kode program pada Kode Program 5.7.

Kode Program 5.7 Implementasi Proses *Local Maxima* dan *Local Minima*

Baris ke-	Kode Program
1	`def local_maxima(atas, tengah, bawah):`
2	`    pix = tengah[1][1]`
3	`    dimensi = len(tengah)`
4	`    maxima = True`
5	`    for y in atas:`
6	`        for x in y:`
7	`            if (pix < x) or (pix == x):`
8	`                maxima = False`
9	`    if maxima == True:`
10	`        for y in bawah:`
11	`            for x in y:`
12	`                if (pix < x) or (pix == x):`
13	`                    maxima = False`
14	`    if maxima == True:`
15	`        for y in range(dimensi):`
16	`            for x in range(dimensi):`
17	`                if(x!=((dimensi/2)-0.5))or(y!=((dimensi/2)-0.5)):`
18	`                    if(pix<tengah[y][x])or(pix== tengah[y]`

```
                    [x]):
19                  maxima = False
20                  return maxima

21  def local_minima(atas, tengah, bawah):
22    pix = tengah[1][1]
23    dimensi = len(tengah)
24    minima = True
25    for y in atas:
26      for x in y:
27        if (pix > x) or (pix == x):
28          minima = False
29    if minima == True:
30      for y in bawah:
31        for x in y:
32          if (pix < x) or (pix == x):
33             minima = False
34    if minima == True:
35     for y in range(dimensi):
36       for x in range(dimensi):
37        if(x!=((dimensi/2)-0.5))or(y!=((dimensi/2)-0.5)):
38          if(pix>tengah[y][x])or(pix==tengah[y][x]):
39              maxima = False
40    return minima
```

Keterangan Kode Program 5.7:

Baris 1	:	Pendefinisian *method* `local_maxima`
Baris 2	:	Mengambil nilai piksel yang akan dibandingkan dengan tetangganya
Baris 3	:	Mengambil nilai panjang dan lebar dari tetangga
Baris 5 – 8	:	Perulangan untuk membandingkan dengan tetangga dengan *level DoG* di atasnya
Baris 9 – 13	:	Jika masih terdeteksi sebagai maxima maka buat perulangan untuk membandingkan dengan tetangga *level* dog di bawahnya
Baris 14 – 20	:	Jika masih terdeteksi sebagai *maxima* maka bandingkan lagi dengan tetangganya pada *level dog* yang sama.
Baris 21	:	Pendefinisian *method* `local_maxima`
Baris 22	:	Mengambil nilai piksel yang akan dibandingkan dengan tetangganya
Baris 23	:	Mengambil nilai panjang dan lebar dari tetangga
Baris 25 – 28	:	Perulangan untuk membandingkan dengan tetangga dengan *level DoG* di atasnya
Baris 29 – 33	:	Jika masih terdeteksi sebagai maxima maka buat perulangan untuk membandingkan dengan tetangga *level* dog di bawahnya
Baris 34 – 40	:	Jika masih terdeteksi sebagai *maxima* maka bandingkan lagi dengan tetangganya pada *level dog* yang sama

Apabila piksel yang diperiksa masih *local extrema* maka apakah piksel tersebut adalah corner. Pemeriksaan *corner* dilakukan dengan mencari nilai *r* dari piksel tersebut seperti yang telah dijelaskan pada Gambar 4.24. Implementasi sistem untuk *resize image* ditunjukan dengan potongan kode program pada Kode Program 5.8.

Kode Program 5.8 Implementasi Proses *Find R*

Baris ke-	Kode Program
1	`def find_r(Ixx, Ixy, Iyy, y, x, off):`
2	`    k = 0.04`
3	`    offset = int (off)`
4	`    windowIxx = Ixx[y-offset:y+offset+1, x-offset:x+offset+1]`
5	`    windowIxy = Ixy[y-offset:y+offset+1, x-offset:x+offset+1]`
6	`    windowIyy = Iyy[y-offset:y+offset+1, x-offset:x+offset+1]`
7	`    Sxx = windowIxx.sum()`
8	`    Sxy = windowIxy.sum()`
9	`    Syy = windowIyy.sum()`
10	`    if ((math.pow(Sxx + Syy, 2)) - (4 * ((Sxx * Syy) - Sxy)) < 0):`
11	`        return 0`
12	`    else:`
13	`    lambda1 = ((Sxx + Sxy) + math.sqrt((math.pow(Sxx + Syy, 2)) - (4 * ((Sxx * Syy) - Sxy))))/2`
14	`        lambda2 = ((Sxx + Sxy) - math.sqrt((math.pow(Sxx + Syy, 2)) - (4 * ((Sxx * Syy) - Sxy))))/2`
16	`        det = lambda1 * lambda2`
17	`        trace = lambda1 + lambda2`
18	`        r = det - (k*(trace**2))`
19	`        return r`

Keterangan Kode Program 5.8:

Baris 1 : Pendefinisian *method* `find_r` untuk mencari nilai *r*

Baris 2 : Pendefinisian konstanta *k*

Baris 3 : Pendefinisian nilai *offset*

Baris 4 – 6 : Mengambil gambar *turunan x*, *turunan y*, dan perkalian *turunan x* kalikan dengan *turunan y* sebanyak nilai *windows*.

Baris 7 – 9 : Mengambil nilai penjumlahan dari piksel-piksel dalam setiap *windows*

Baris 10 – 11 : Memeriksa nilai *determinan* matrix, apabila nilainya kurang dari 0 maka *return* 0

Baris 12 – 19 : Jika tidak maka hitung nilai *r*

Baris 13 – 14 : Mencari nilai *lambda 1* dan *lambda 2*

Baris 16 : Menghitung nilai *determinan* dengan mengalikan *lambda 1* dan *lambda 2*

Baris 17 : Menghitung nilai *trace* dengan menambahkan nilai dari *lambda 1* dan *lambda 2*
Baris 18 : Menghitung nilai *r*
Baris 19 : Mengembalikan nilai *r*

Setelah didapatkan nilai *r* dari piksel maka akan diperiksa apakah nilai *r* lebih dari sama dengan 10. Apabila nilai lebih dari 10 maka piksel tersebut adalah *keypoint*, apabila kurang maka piksel tersebut bukan *keypoint* karena masih berada pada garis dan bukan *corner*.

5.2.4 Implementasi Proses *Orientation Assignment*

Setelah lolos melalui tahap pemeriksaan *keypoint*, maka piksel tersebut akan dicari nilai orientasinya. Orientasi dari piksel ini akan digunakan sebagai bahan untuk membentuk *descriptor* nantinya seperti yang telah dijelaskan pada Gambar 4.28. Implementasi sistem untuk memeriksa *local extrema* ditunjukan dengan potongan kode program pada Kode Program 5.9.

Kode Program 5.9 Implementasi Proses *Orientation Assignment*

Baris ke-	Kode Program
1	`if (r > 10):`
2	`  temp = []`
3	`  key = [0, 0, 0, 0, 0, 0, 0, 0]`
4	`  for i in range(16):`
5	`  temp.append([0, 0, 0, 0, 0, 0, 0, 0])`
6	`  cv2.circle(hasil, (x, y), 4, (0, 255, 0), 2)`
7	`  id_y = 0`
8	`  for index_y in range(y-8, y+8, 1):`
9	`  id_x = 0`
10	`  for index_x in range(x-8, x+8, 1):`
11	`  turunan_x=L[index_y][index_x+1]-L[index_y][index_x-1]`
12	`  turunan_y = L[index_y-1][index_x] - L[index_y+1][index_x]`
13	`  tan = turunan_y`
14	`  if turunan_x != 0:`
15	`  tan = turunan_y / turunan_x`
16	`  sudut = math.atan(tan)`

Keterangan Kode Program 5.9:

Baris 1 : Seleksi kondisi untuk memeriksa nilai *r* apakah lebih dari 10
Baris 2 : Membentuk *array* temporer dengan nama *temp* untuk digunakan membentuk *descriptor* nantinya
Baris 3 : Membuan *array key* yang digunakan untuk mencatat *histogram* dari orientasi
Baris 4 : Perulangan untuk membuat *histogram* fitur
Baris 5 : Inisialisasi *histogram temp*
Baris 6 : Melingkari piksel *keypoint*
Baris 7 – 16 : Perulangan untuk mencari nilai orientasi piksel tetangga

Baris 11 : Menghitung nilai *turunan* terhadap x
Baris 12 : Menghitung nilai *turunan* terhadap y
Baris 13 : Inisiaisasi nilai *default tan* adalah *turunan* y
Baris 14 - 15 : Seleksi kondisi jika turunan x tidak sama dengan 0 maka hitung nilai *tan* dengan membagi nilai *turunan* y dengan nilai *turunan* x
Baris 16 : Menghitung nilai sudut berdasarkan nilai *tan*

5.2.5 Implementasi Proses *Keypoint Descriptor*

Setelah didapatkan nilai sudut dari piksel maka proses selanjutnya adalah memasukkannya pada *histogram*. Akan ada 2 *histogram* yang dibentuk seperti yang telah dijelaskan pada Gambar 4.30 dan Gambar 4.31. Implementasi sistem untuk proses *keypoint descriptor* ditunjukan dengan potongan kode program pada Kode Program 5.10.

Kode Program 5.10 Implementasi Proses *Keypoint Descriptor*

Baris ke-	Kode Program
1	`daerah = cari_daerah_x(id_x)`
2	`daerah += cari_daerah_y(id_y)`
3	`index_histogram=cari_index_histogram(turunan_x,turunan_y, sudut)`
	`temp[daerah][index_histogram] += 1`
4	`key[index_histogram] += 1`
5	

Keterangan Kode Program 5.10:

Baris 1 : Mencari *index histogram* berdasarkan dari *index_x*
Baris 2 : Mencari *index histogram* berdasarkan dari index_y
Baris 3 : Mencari *index* dari *histogram* keseluruhan berdasarkan dari *turunan x*, turunan y dan sudutnya
Baris 4 – 5 : *Increament* untuk *update* jumlah yang ada dalam *histogram*

Untuk dapat memasukkan ke dalam *index* yang benar maka diperlukan proses pencarian *index* dari *histogram* berdasarkan dari posisi tetangga yang dicari orientasinya. Ketika hanya didapatkan sebuah nilai *cottangen* maka akan menemukan kesulitan untuk mencari nilai sudut yang sebenarnya. Oleh karena itu, dibutuhkan nilai *turunan_x* dan *turunan_y* untuk menentukan nilai sudutnya. Implementasi sistem untuk mencari nilai index *histogram* ditunjukan dengan potongan kode program pada Kode Program 5.11.

Kode Program 5.11 Implementasi Proses Cari Index *Histogram*

Baris ke-	Kode Program
1	`def cari_daerah_x(id_x):`
2	`    if (id_x >= 0) and (id_x <= 3):`
3	`        return 0`

```
4       elif (id_x >= 4) and (id_x <= 7):
5           return 1
6       elif (id_x >= 8) and (id_x <= 11):
7           return 2
8       elif (id_x >= 12) and (id_x <= 15):
9           return 3

10  def cari_daerah_y(id_y):
11      if (id_y >= 0) and (id_y <= 3):
12          return 0
13      elif (id_y >= 4) and (id_y <= 7):
14          return 4
15      elif (id_y >= 8) and (id_y <= 11):
16          return 8
17      elif (id_y >= 12) and (id_y <= 15):
18          return 12

19  def cari_index_histogram(turunan_x, turunan_y, atan):
20      if (turunan_x > 0) and (turunan_y > 0):
21          deg = math.degrees(atan)
22          if (deg >= 0) and (deg < 45):
23              return 0
24          elif (deg >= 45) and (deg <= 90):
25              return 1
26      elif (turunan_x < 0) and (turunan_y > 0):
27          deg = 180 + math.degrees(atan)
28          if (deg >= 90) and (deg < 135):
29              return 2
30          elif (deg >= 135) and (deg <= 180):
31              return 3
32      elif (turunan_x < 0) and (turunan_y < 0):
33          deg = 180 + math.degrees(atan)
34          if (deg >= 180) and (deg < 225):
35              return 4
36          elif (deg >= 225) and (deg <= 270):
37              return 5
38      elif (turunan_x > 0) and (turunan_y < 0):
39          deg = 360 + math.degrees(atan)
40          if (deg >= 270) and (deg < 315):
41              return 6
42          elif (deg >= 315) and (deg <= 360):
43              return 7
44      else:
45          return 0
```

Keterangan Kode Program 5.11:

Baris 1 : Pendefinisian *method* `cari_daerah_x` yang digunakan untuk mencari *index* dari *histogram* 16x16

Baris 2 – 3 : Jika *id_x* lebih dari sama dengan 0 dan kurang dari sama dengan 3 maka *return* 0

Baris 4 – 5 : Jika *id_x* lebih dari sama dengan 4 dan kurang dari sama dengan 7 maka *return* 1

Baris 6 – 7 : Jika *id_x* lebih dari sama dengan 8 dan kurang dari sama dengan 11 maka *return* 2
Baris 8 – 9 : Jika *id_x* lebih dari sama dengan 12 dan kurang dari sama dengan 15 maka *return* 3
Baris 10 : Pendefinisian *method* `cari_daerah_x` yang digunakan untuk mencari *index* dari *histogram* 16x16
Baris 11 – 12 : Jika *id_y* lebih dari sama dengan 0 dan kurang dari sama dengan 3 maka *return* 0
Baris 13 – 14 : Jika *id_y* lebih dari sama dengan 4 dan kurang dari sama dengan 7 maka *return* 4
Baris 15 – 16 : Jika *id_y* lebih dari sama dengan 8 dan kurang dari sama dengan 11 maka *return* 8
Baris 17 – 18 : Jika *id_y* lebih dari sama dengan 12 dan kurang dari sama dengan 15 maka *return* 12
Baris 19 : Pendefinisian method `cari_index_histogram`
Baris 20 : Jika *turunan_x* lebih dari 0 dan *turunan y* lebih dari 0 maka sudut tersebut berada di kuadran 1
Baris 21 : Hitung nilai sudut
Baris 22 – 23 : Jika sudut kurang dari sama dengan 0 dan lebih dari 45 maka *return* 0
Baris 24 – 25 : Jika tidak maka *return* 1
Baris 26 : Jika *turunan_x* kurang dari 0 dan *turunan y* lebih dari 0 maka sudut tersebut berada di kuadran 2
Baris 27 : Hitung nilai sudut
Baris 28 – 29 : Jika sudut lebih dari sama dengan 90 dan kurang dari 125 maka *return* 2
Baris 24 – 25 : Jika tidak maka *return* 3
Baris 32 : Jika *turunan_x* kurang dari 0 dan *turunan y* kurang dari 0 maka sudut tersebut berada di kuadran 3
Baris 33 : Hitung nilai sudut + 180 karena berada di kuadran 3
Baris 34 – 35 : Jika sudut lebih dari sama dengan 180 dan kurang dari 225 maka *return* 4
Baris 36 – 37 : Jika tidak maka *return* 5
Baris 38 : Jika *turunan_x* lebih dari 0 dan *turunan y* kurang dari 0 maka sudut tersebut berada di kuadran 4
Baris 39 : Hitung nilai sudut + 360 karena berada di kuadran 4
Baris 40 – 41 : Jika sudut lebih dari sama dengan 270 dan kurang dari 315 maka *return* 6

Baris 42 – 43 : Jika tidak maka *return* 7

Baris 44 – 45 : *Default* jika tidak ada kondisi yang terpenuhi maka *return* 0

5.2.6 Implementasi Proses *Matching*

Setelah didapatkan *descriptor* dari gambar langkah terakhir adalah melakukan pencocokan dengan *descriptor* dari data latih untuk mengetahui gambar dari sampul buku apakah itu seperti yang telah dijelaskan pada Gambar 4.32. Implementasi sistem untuk proses *keypoint descriptor* ditunjukan dengan potongan kode program pada Kode Program 5.12.

Kode Program 5.12 Implementasi Proses *Matching*

Baris ke-	Kode Program
1	`index = 0`
2	`best_match = 0`
3	`persen_match = 0`
4	`for img in train:`
5	`  print(" ========= "+str(index)+" ============= ")`
6	`  banyak_kp = 0`
7	`  banyak_kp_match = 0`
8	`  for im in img[1]:`
9	`  for kp in im:`
10	`  minn = 1000`
11	`  banyak_kp += 1`
12	`  for im2 in test:`
13	`  for kp2 in im2:`
14	`  tot = 0`
15	`  h = np.abs(np.array(kp) - np.array(kp2))`
16	`  for x in h:`
17	`  for y in x:`
18	`  tot += y`
19	`  if tot < minn:`
20	`  minn = tot`
21	`  if minn <= 200:`
22	`  banyak_kp_match += 1`
23	`  persen = banyak_kp_match/banyak_kp`
24	`  print(persen)`
25	`  if persen > persen_match:`
26	`  best_match = index`
27	`  persen_match = persen`
28	`  index += 1`
29	`client_secret = "*****"`
30	`key = "*****"`
31	`gc = client.GoodreadsClient(client_secret, key)`
32	`book = gc.book(train[best_match][0])`
33	`title = book.title`
34	`rating = book.average_rating`

Keterangan Kode Program 5.12:

Baris 1	:	Variabel *index* untuk menampilkan indeks gambar dari *database*
Baris 2	:	Variabel *best_match* untuk menyimpan *index* dengan kecocokan paling besar
Baris 3	:	Variabel *persen_match* untuk menyimpan nilai kecocokan dalam persen
Baris 4 – 28	:	Perulangan untuk mencocokkan *deskriptor* data uji dengan deskriptor-deskriptor yang ada dalam data latih
Baris 5	:	*Print index* gambar data latih
Baris 6 – 7	:	*Variable* banyak_kp untuk menyimpan nilai total jumlah *keypoint*, dan *banyak_kp_match* menyimpan banyak *keypoint* yang sesuai
Baris 8 – 22	:	Perulangan untuk mencocokkan *keypoint* dari data latih
Baris 10 – 11	:	Variabel *min* untuk mendapatkan nilai paling sedikit dari jarak antara *keypoint* dan *increament* nilai *banyak_kp*
Baris 12 – 20	:	Perulangan untuk mencocokkan keypoint dari data uji
Baris 14	:	Variabel total untuk menyimpan nilai total jarak *keypoint*
Baris 15	:	Menghitung jarak antara *keypoint*
Baris 16 – 18	:	Menjumlahkan total jarak
Baris 19 – 20	:	Jika jarak lebih kecil dari jarak sebelumnya *update* jarak minimal
Baris 21 – 22	:	Jika jarak minimal kurang dari *threshold* tambahkan total jumlah *keypoint* yang cocok
Baris 23	:	Menghitung nilai persen
Baris 24	:	Menampilkan nilai persen
Baris 25 – 27	:	*Update* nilai best_*match* jika menemukan nilai persen kecocokan yang lebih besar
Baris 28	:	Increament indeks gambar data latih
Baris 29 – 30	:	Pendeskripsian client_secret dan *api key* untuk api dari *goodreads*
Baris 31	:	Inisialisasi koneksi *client* ke Goodreads API
Baris 32	:	Mengambil informasi umum buku berdasarkan *index* dari Goodreads
Baris 33 – 34	:	Mengambil informasi judul buku dan *rating* dari buku

BAB 6 PENGUJIAN

Pada bab ini dijelaskan mengenai pengujian dan analisis sistem berdasarkan metodologi, perancangan, dan implementasi yang telah dijelaskan pada bab sebelumnya. Data latih dan data uji ditunjukkan pada lampiran.

6.1 Pengujian Pengaruh Pencahayaan

Pada proses pengujian yang pertama adalah pengujian pengaruh pencahayaan pada gambar. Proses pengujian dilakukan untuk mengetahui apakah pencahayaan yang kurang pada gambar data latih akan berpengaruh pada akurasi ketika diuji menggunakan gambar yang diambil pada pencahayaan terang dan sebaliknya. Pengujian dilakukan dalam dua tahap yang pertama menggunakan data latih pencahayaan kurang dan kedua menggunakan data latih pencahayaan terang. Pengujian dilakukan masing-masing menggunakan data uji yang diambil pada kondisi pencahayaan kurang dan terang.

6.1.1 Pencahayaan Kurang

Pada tahap pertama menggunakan data latih yang diambil pada pencahayaan yang kurang dan data uji pada pencahayaan yang kurang dan terang. Pengujian menggunakan masing-masing 20 data uji. Hasil pengujian pengaruh pencahayaan kurang ditunjukkan pada Tabel 6.1 dan Tabel 6.2.

Tabel 6.1 Pencahayaan Terang

Gambar ke-	Judul asli	Judul Terdeteksi	Hasil
1	*Freshwater*	*Computer Vision*	X
2	*The Atomic City Girl*	*The Atomic City Girl*	√
3	*To Kill a Mockingbird*	*To Kill a Mockingbird*	√
4	*Which is not true ?*	*Which Is Not True ?*	√
5	*The Giving Tree*	*The Giving Tree*	√
6	*On the Road*	*On the Road*	√
7	*The Joy Luck Club*	*The Joy Luck Club*	√
8	*Louder Than a Whisper*	*Louder Than A Whisper*	√
9	*Computer Vision*	*Computer Vision*	√
10	*Signal Processing for Computer Vision*	*Signal Processing for Computer Vision*	√
11	*Computer Vision for Electronics Manufacturing*	*Computer Vision for Electronics Manufacturing*	√
12	*Pyramidal Architectures for Computer Vision*	*Pyramidal Architectures For Computer Vision*	√
13	*Arduino Computer Vision Programming*	*Arduino Computer Vision Programming*	√
14	*Parallel Computer Vision*	*Parallel Computer Vision*	√

Tabel 6.1 Pencahayaan Terang (Lanjutan)

15	*Learning OpenCV 3*	*Learning OpenCV 3*	√
16	*Big Mushy Happy Lump*	*Big Mushy Happy Lump*	√
17	*My Favorite Thing is Monters*	*My Favorite Thing Is Monster*	√
18	*The Sun and Her Flowers*	*The Sun and Her Flowers*	√
19	*Adultolescence*	*Adultolescence*	√
20	*A List of Cages*	*A List Of Cages*	√
Total			19
Akurasi			95%

Tabel 6.2 Pencahayaan kurang

Gambar ke-	Judul asli	Judul Terdeteksi	Hasil
1	*Freshwater*	*Computer Vision*	X
2	*The Atomic City Girl*	*The Atomic City Girl*	√
3	*To Kill a Mockingbird*	*To Kill a Mockingbird*	√
4	*Which is not true ?*	*Which Is Not True ?*	√
5	*The Giving Tree*	*The Giving Tree*	√
6	*On the Road*	*On the Road*	√
7	*The Joy Luck Club*	*The Joy Luck Club*	√
8	*Louder Than a Whisper*	*Louder Than A Whisper*	√
9	*Computer Vision*	*A List of Cages*	X
10	*Signal Processing for Computer Vision*	*Signal Processing for Computer Vision*	√
11	*Computer Vision for Electronics Manufacturing*	*Computer Vision for Electronics Manufacturing*	x
12	*Pyramidal Architectures for Computer Vision*	*Pyramidal Architectures For Computer Vision*	√
13	*Arduino Computer Vision Programming*	*Arduino Computer Vision Programming*	√
14	*Parallel Computer Vision*	*Parallel Computer Vision*	√
15	*Learning OpenCV 3*	*Learning OpenCV 3*	√
16	*Big Mushy Happy Lump*	*Big Mushy Happy Lump*	√
17	*My Favorite Thing is Monters*	*My Favorite Thing Is Monster*	√
18	*The Sun and Her Flowers*	*The Sun and Her Flowers*	x
19	*Adultolescence*	*Adultolescence*	√
20	*A List of Cages*	*A List Of Cages*	√
Total			16
Akurasi			80%

Berdasarkan Tabel 6.1 dan 6.2 hasil akurasi pada data latih yang diambil pada pencahayaan terang akan menghasilkan akurasi yang baik, karena pencahayaan yang terang akan memudahkan untuk mendeteksi *keypoint-keypoint* yang ada. Sedangkan ketika data uji diambil pada tempat dengan pencahayaan yang kurang akan

menyebabkan sistem lebih sulit untuk mendeteksi adanya *keypoint*. Semakin sedikit keypoint yang terdeteksi maka semakin sedikit pula tingkat kecocokannya.

6.1.2 Pencahayaan Terang

Pada tahap kedua menggunakan data latih yang diambil pada pencahayaan terang dan data uji pada pencahayaan yang kurang dan terang. Pengujian menggunakan masing-masing 20 data uji. Hasil pengujian pengaruh pencahayaan di dalam ruangan ditunjukkan pada Tabel 6.3 dan Tabel 6.4.

Tabel 6.3 Pencahayaan Terang

Gambar ke-	Judul asli	Judul Terdeteksi	Hasil
1	*Freshwater*	*Freshwater*	√
2	*The Atomic City Girl*	*A List of Cages*	*X*
3	*To Kill a Mockingbird*	*To Kill A Mockingbird*	√
4	*Which is not true ?*	*Which Is Not True ?*	√
5	*The Giving Tree*	*The Giving Tree*	√
6	*On the Road*	*On the Road*	√
7	*The Joy Luck Club*	*The Joy Luck Club*	√
8	*Louder Than a Whisper*	*Louder Than A Whisper*	√
9	*Computer Vision*	*A List of Cages*	*X*
10	*Signal Processing for Computer Vision*	*Signal Processing for Computer Vision*	√
11	*Computer Vision for Electronics Manufacturing*	*Computer Vision For Electronics Manufacturing*	√
12	*Pyramidal Architectures for Computer Vision*	*Pyramidal Architectures For Computer Vision*	√
13	*Arduino Computer Vision Programming*	*Arduino Computer Vision Programming*	√
14	*Parallel Computer Vision*	*Parallel Computer Vision*	√
15	*Learning OpenCV 3*	*Learning OpenCV 3*	√
16	*Big Mushy Happy Lump*	*Big Mushy Happy Lump*	√
17	*My Favorite Thing is Monters*	*My Favorite Thing Is Monster*	√
18	*The Sun and Her Flowers*	*The Sun And Her Flowers*	√
19	*Adultolescence*	*Adultolescence*	√
20	*A List of Cages*	*A List Of Cages*	√
Total			*18*
Akurasi			*90%*

Tabel 6.4 Pencahayaan Kurang

Gambar ke-	Judul asli	Judul Terdeteksi	Hasil
1	*Freshwater*	*Freshwater*	√
2	*The Atomic City Girl*	*A List of Cages*	X
3	*To Kill a Mockingbird*	*To Kill A Mockingbird*	√
4	*Which is not true ?*	*Which Is Not True ?*	√
5	*The Giving Tree*	*The Giving Tree*	√
6	*On the Road*	*On the Road*	√
7	*The Joy Luck Club*	*The Joy Luck Club*	√
8	*Louder Than a Whisper*	*Louder Than A Whisper*	√
9	*Computer Vision*	*A List of Cages*	X
10	*Signal Processing for Computer Vision*	*Signal Processing for Computer Vision*	√
11	*Computer Vision for Electronics Manufacturing*	*Computer Vision For Electronics Manufacturing*	√
12	*Pyramidal Architectures for Computer Vision*	*Pyramidal Architectures For Computer Vision*	√
13	*Arduino Computer Vision Programming*	*Arduino Computer Vision Programming*	√
14	*Parallel Computer Vision*	*Parallel Computer Vision*	√
15	*Learning OpenCV 3*	*Learning OpenCV 3*	√
16	*Big Mushy Happy Lump*	*Big Mushy Happy Lump*	√
17	*My Favorite Thing is Monters*	*My Favorite Thing Is Monster*	√
18	*The Sun and Her Flowers*	*The Sun And Her Flowers*	√
19	*Adultolescence*	*Adultolescence*	√
20	*A List of Cages*	*A List Of Cages*	√
Total			18
Akurasi			90%

Berdasarkan Tabel 6.3 dan Tabel 6.4 akurasi dari pencocokan gambar dari data uji dengan gambar pada data *train* cukup baik. Hal ini dikarenakan ruangan terbuka akan menyediakan cukup cahaya yang diperlukan untuk menghasilkan gambar yang lebih jelas. Sedangkan akurasi dari sistem dalam menentukan gambar yang sesuai untuk data uji yang diambil dalam ruang tertutup adalah 75%. Hal ini dikarenakan data uji yang diambil pada ruangan tertutup akan mendapatkan intensitas cahaya yang lebih rendah. Intensitas cahaya yang rendah akan memengaruhi nilai-nilai piksel.

6.2 Pengujian Data Gambar dengan Rotasi

Pada pengujian pengaruh rotasi menggunakan data uji yang didapatkan dari gambar yang diambil dalam ruang terbuka dengan melakukan rotasi pada gambar. Pengujian ini dilakukan dalam dua tahap yaitu seperti yang telah dijelaskan pada perancangan pengujian 4.3.3. Pengujian menggunakan 20 data uji untuk masing-masing tahapan.

6.2.1 Rotasi dengan Sudut Ekstrem

Pada pengujian pertama dari pengaruh rotasi dilakukan rotasi pada gambar dengan menggunakan sudut-sudut esktrem. Pengujian pengaruh rotasi dengan sudut ekstrem ditunjukkan pada Tabel 6.5.

Tabel 6.5 Pengujian Pengaruh Rotasi Gambar dengan Sudut Ekstrem

Gambar ke-	Judul asli	Judul Terdeteksi	Hasil
1	*Freshwater*	*A List of Cages*	x
2	*The Atomic City Girl*	*The Atomic City Girl*	√
3	*To Kill a Mockingbird*	*To Kill A Mockingbird*	√
4	*Which is not true ?*	*Which Is Not True ?*	√
5	*The Giving Tree*	*The Giving Tree*	√
6	*On the Road*	*On the Road*	√
7	*The Joy Luck Club*	*The Joy Luck Club*	√
8	*Louder Than a Whisper*	*Louder Than A Whisper*	√
9	*Computer Vision*	*Computer Vision*	√
10	*Signal Processing for Computer Vision*	*Signal Processing for Computer Vision*	√
11	*Computer Vision for Electronics Manufacturing*	*Computer Vision For Electronics Manufacturing*	√
12	*Pyramidal Architectures for Computer Vision*	*Pyramidal Architectures For Computer Vision*	√
13	*Arduino Computer Vision Programming*	*Arduino Computer Vision Programming*	√
14	*Parallel Computer Vision*	*Parallel Computer Vision*	√
15	*Learning OpenCV 3*	*Learning OpenCV 3*	√
16	*Big Mushy Happy Lump*	*Big Mushy Happy Lump*	√
17	*My Favorite Thing is Monters*	*My Favorite Thing Is Monster*	√
18	*The Sun and Her Flowers*	*The Sun And Her Flowers*	√
19	*Adultolescence*	*Adultolescence*	√
20	*A List of Cages*	*A List Of Cages*	√
Total			19
Akurasi			95%

Berdasarkan Tabel 6.5, hasil dari pengujian menggunakan sudut ekstrem didapatkan akurasi dari sistem yang baik. Hal ini dikarenakan pada data latih juga terdapat data dengan rotasi menggunakan sudut ekstrem. Pada pengujian ini hanya terdapat satu data uji yang hasilnya tidak sesuai dengan yang dihasilkan oleh sistem. Data uji dengan judul buku *Freshwater* menunjukkan hasil *matching* terbesar dengan data latih gambar *A List of Cages*. Hal ini dikarenakan *keypoint* yang terdapat pada gambar latih tidak terlalu banyak dan ada banyak *keypoint* dari gambar latih yang melebihi *threshold* untuk dianggap sesuai dengan *keypoint* yang terdapat pada gambar uji.

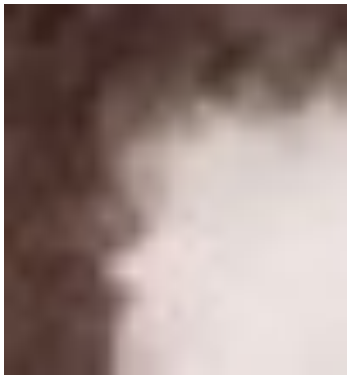
6.2.2 Rotasi dengan Selain Sudut Ekstrem

Pada pengujian kedua dari pengaruh rotasi dilakukan rotasi pada gambar dengan menggunakan sudut-sudut selain sudut esktrem. Pengujian pengaruh rotasi dengan sudut selain sudut ekstrem ditunjukkan pada Tabel 6.6.

Tabel 6.6 Pengujian Pengaruh Rotasi Gambar

Gambar ke-	Judul asli	Judul Terdeteksi	Sudut	Hasil
1	*Freshwater*	*A List of Cages*	15	X
2	*The Atomic City Girl*	*A List of Cages*	30	X
3	*To Kill a Mockingbird*	*A List of Cages*	45	X
4	*Which is not true ?*	*A List of Cages*	60	X
5	*The Giving Tree*	*A List of Cages*	75	X
6	*On the Road*	*On the Road*	105	√
7	*The Joy Luck Club*	*The Joy Luck Club*	120	√
8	*Louder Than a Whisper*	*Computer Vision*	135	X
9	*Computer Vision*	*A List of Cages*	150	X
10	*Signal Processing for Computer Vision*	*A List of Cages*	165	X
11	*Computer Vision for Electronics Manufacturing*	*Computer Vision For Electronics Manufacturing*	195	√
12	*Pyramidal Architectures for Computer Vision*	*A List of Cages*	210	X
13	*Arduino Computer Vision Programming*	*Computer Vision*	225	X
14	*Parallel Computer Vision*	*A List of Cages*	240	X
15	*Learning OpenCV 3*	*The Joy Luck Club*	255	X
16	*Big Mushy Happy Lump*	*A List of Cages*	285	X
17	*My Favorite Thing is Monters*	*A List of Cages*	300	X
18	*The Sun and Her Flowers*	*Computer Vision*	315	X
19	*Adultolescence*	*A List Of Cages*	330	X
20	*A List of Cages*	*A List Of Cages*	345	√
Total				4
Akurasi				20%

Berdasarkan Tabel 6.6 didapatkan akurasi system yang begitu rendah. Hal ini dikarenakan algoritme SIFT tidak mampu untuk mencocokkan data dengan perbedaan rotasi. Rotasi akan menyebabkan perbedaan piksel yang diambil oleh untuk dijadikan *histogram* dari *keyppoint*. Misalkan sebuah gambar pada latih mengambil *histogram keypoint* pada titik seperti Gambar 6.1 dan gambar data latih mengambil *histogram keypoint* pada titik seperti Gambar 6.2 maka akan menghasilkan komposisi *histogram* yang berbeda. Hal ini menyebabkan sistem menjadi lemah dalam mencari gambar yang paling sesuai.



Gambar 6.1 Piksel Pembentukan *Histogram Keypoint* Data Latih

Gambar 6.2 Piksel Pembentukan *Histogram Keypoint* Data Test dengan Rotasi

6.3 Pengujian Data Gambar dengan Perbedaan Skala

Pada pengujian pengaruh perbedaan skala menggunakan data uji yang didapatkan dari gambar yang diambil dalam ruang terbuka dengan gambar dari jarak yang berbeda-beda. Pengujian ini dilakukan dalam dua tahap seperti yang telah dijelaskan pada perancangan pengujian 4.3.4. Pengujian menggunakan 20 data uji untuk masing-masing tahapan.

6.3.1 Jarak 15 Cm

Pengujian pertama dari pengaruh perbedaan skala menggunakan gambar data uji yang diambil pada jarak 15 cm dari objek. Hasil pengujian pengaruh jarak 15 cm ditunjukkan pada Tabel 6.7.

Tabel 6.7 Pengujian Pengaruh Perbedaan Skala pada Jarak 15cm

Gambar ke-	Judul asli	Judul Terdeteksi	Hasil
1	*Freshwater*	*A List of Cages*	X
2	*The Atomic City Girl*	*A List of Cages*	X
3	*To Kill a Mockingbird*	*The Atomic City Girl*	√
4	*Which is not true ?*	*Which Is Not True ?*	√
5	*The Giving Tree*	*Computer Vision*	X

Tabel 6.7 Pengujian Pengaruh Perbedaan Skala pada Jarak 15cm (Lanjutan)

6	*On the Road*	*On the Road*	√
7	*The Joy Luck Club*	*The Joy Luck Club*	√
8	*Louder Than a Whisper*	*Louder Than A Whisper*	√
9	*Computer Vision*	*Computer Vision*	√
10	*Signal Processing for Computer Vision*	*Signal Processing for Computer Vision*	√
11	*Computer Vision for Electronics Manufacturing*	*Computer Vision For Electronics Manufacturing*	√
12	*Pyramidal Architectures for Computer Vision*	*Pyramidal Architectures For Computer Vision*	√
13	*Arduino Computer Vision Programming*	*Arduino Computer Vision Programming*	√
14	*Parallel Computer Vision*	*Parallel Computer Vision*	√
15	*Learning OpenCV 3*	*Learning OpenCV 3*	√
16	*Big Mushy Happy Lump*	*Big Mushy Happy Lump*	√
17	*My Favorite Thing is Monters*	*My Favorite Thing Is Monster*	√
18	*The Sun and Her Flowers*	*The Sun And Her Flowers*	√
19	*Adultolescence*	*Adultolescence*	√
20	*A List of Cages*	*A List Of Cages*	√
	Total		17
	Akurasi		85%

Berdasarkan Tabel 6.7, hasil pengujian menunjukkan akurasi yang cukup baik. Hal ini dikarenakan jarak 15 cm akn dapat membuat sebuah gambar uji penuh dengan objek. Ketika gambar uji penuh dengan objek yang dideteksi maka akan mengurangi *noise* atau objek-objek yang tidak perlu untuk dideteksi *keypoint*-nya. Pada data Uji pertama dan kedua menunjukkan hasil yang sama yaitu mempunyai kesamaan terbesar dengan data latih gambar *A List of Cages*, hal ini dikarenakan pada data latih gambar tersebut tidak terdapat terlalu banyak *keypoint* dan *keypoint-keypoint* tersebut mempunyai banyak kemiripan dengan data uji pertama dan kedua.

6.3.2 Jarak 30 Cm

Pengujian kedua dari pengaruh perbedaan skala menggunakan gambar data uji yang diambil pada jarak 30 cm dari objek. Hasil pengujian pengaruh jarak 30 cm ditunjukkan pada Tabel 6.8.

Tabel 6.8 Pengujian Pengaruh Perbedaan Skala pada Jarak 30cm

Gambar ke-	Judul asli	Judul Terdeteksi	Hasil
1	*Freshwater*	*On the Road*	X
2	*The Atomic City Girl*	*The Sun And Her Flowers*	X
3	*To Kill a Mockingbird*	*Computer Vision*	X
4	*Which is not true ?*	*Computer Vision For Electronics Manufacturing*	X

Tabel 6.8 Pengujian Pengaruh Perbedaan Skala pada Jarak 30cm (Lanjutan)

5	*The Giving Tree*	*Computer Vision*	X
6	*On the Road*	*On the Road*	√
7	*The Joy Luck Club*	*Signal Processing for Computer Vision*	X
8	*Louder Than a Whisper*	*Louder Than A Whisper*	√
9	*Computer Vision*	*The Atomic City Girl*	X
10	*Signal Processing for Computer Vision*	*Ted Talks*	X
11	*Computer Vision for Electronics Manufacturing*	*The Joy Luck Club*	X
12	*Pyramidal Architectures for Computer Vision*	*My Favorite Thing Is Monster*	X
13	*Arduino Computer Vision Programming*	*Arduino Computer Vision Programming*	√
14	*Parallel Computer Vision*	*Parallel Computer Vision*	√
15	*Learning OpenCV 3*	*The Joy Luck Club*	X
16	*Big Mushy Happy Lump*	*A List of Cages*	X
17	*My Favorite Thing is Monters*	*Arduino Computer Vision Programming*	X
18	*The Sun and Her Flowers*	*A List Of Cages*	X
19	*Adultolescence*	*Adultolescence*	√
20	*A List of Cages*	*A List Of Cages*	√
Total			6
Akurasi			30%

Berdasarkan Tabel 6.8, hasil akurasi sistem sangat sedikit. Hal ini dikarenakan jarak 30 cm akan membuat objek terlihat kecil dalam gambar. Kecilnya objek yang terdeteksi menyebabkan lemahnya sistem dan algoritme dalam mendeteksi *keypoint* yang ada. Objek yang terlihat kecil pada gambar akan menyebabkan perbedaan piksel yang digunakan untuk menyusun *histogram* dari *keypoint*. Ketika mengambil gambar objek dari jarak yang cukup jauh akan menyebabkan objek-objek lain yang berada di sekitarnya ikut terambil dalam foto. Hal ini menyebabkan terbentuknya *keypoint-keypoint* yang tidak diperlukan dan akan menimbulkan *noise*.

6.3.3 Jarak 10 Cm

Pengujian kedua dari pengaruh perbedaan skala menggunakan gambar data uji yang diambil pada jarak 10 cm dari objek. Hasil pengujian pengaruh jarak 10 cm ditunjukkan pada Tabel 6.9.

Tabel 6.9 Pengujian Pengaruh Perbedaan Skala pada Jarak 30cm

Gambar ke-	Judul asli	Judul Terdeteksi	Hasil
1	*Freshwater*	*The Atomic City Girl*	X
2	*The Atomic City Girl*	*Ted Talks*	X

Tabel 6.9 Pengujian Pengaruh Perbedaan Skala pada Jarak 30cm (Lanjutan)

3	*To Kill a Mockingbird*	*Computer Vision For Electronics Manufacturing*	X
4	*Which is not true ?*	*Pyramidal Architectures for Computer Vision*	X
5	*The Giving Tree*	*Arduino Computer Vision Programming*	X
6	*On the Road*	*Learning OpenCV 3*	X
7	*The Joy Luck Club*	*The Joy Luck Club*	√
8	*Louder Than a Whisper*	*A List of Cages*	√
9	*Computer Vision*	*Computer Vision*	√
10	*Signal Processing for Computer Vision*	*Computer Vision*	X
11	*Computer Vision for Electronics Manufacturing*	*The Joy Luck Club*	X
12	*Pyramidal Architectures for Computer Vision*	*The Joy Luck Club*	X
13	*Arduino Computer Vision Programming*	*A List of Cages*	X
14	*Parallel Computer Vision*	*On the Road*	X
15	*Learning OpenCV 3*	*The Atomic City Girl*	X
16	*Big Mushy Happy Lump*	*Parallel Computer Vision*	X
17	*My Favorite Thing is Monters*	*Computer Vision*	X
18	*The Sun and Her Flowers*	*The Joy Luck Club*	X
19	*Adultolescence*	*Adultolescence*	√
20	*A List of Cages*	*A List Of Cages*	√
Total	5		
Akurasi	25%		

Berdasarkan Tabel 6.9, hasil akurasi dari sistem untuk pengambil gambar pada jarak 10 cm hanya sebesar 25%. Hal ini dikarenakan pengambilan gambar dari jarak 10 cm tidak dapat mengambil keseluruhan bagian dari gambar. Akibatnya nilai piksel-pikselnya pun akan berbeda-beda juga yang berpengaruh kepada *histogram* dari *keypoint* yang terbentuk. Hal ini akan menyulitkan saat melakukan pencocokan *keypoint* gambar dengan yang ada pada data latih.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis hasil yang sudah dilakukan maka kesimpulan yang dapat diambil adalah:

1. Memanfaatkan sampul sebuah buku untuk memperoleh judul dan *rating* dapat diterapkan dengan cara SIFT sebagai algoritme untuk mendapatkan *keypoint* yang yang ada pada sampul buku tersebut untuk kemudian dicocokkan dengan *keypoint* yang telah ada pada data latih. Serta penggunaan algoritme *brute force* untuk mencocokkan *keypoint* yang ada pada sampul dengan *keypoint* yang ada pada data latih.
2. Akurasi dari sistem untuk menentukan judul buku yang tepat diperoleh dengan cara pengujian pengaruh pencahayaan, pengujian pengaruh rotasi, dan pengujian pengaruh skala. Untuk pengujian pencahayaan diperoleh hasil akurasi terbaik adalah dengan menggunakan pencahayaan yang cukup pada ruang terbuka dengan akurasi 95%. Pengujian pengaruh rotasi menghasilkan akurasi terbaik ketika gambar berada pada orientasi 0^{o}, 90^{o}, 180^{o}, 270^{o} dengan akurasi 95%. Pengujian terkahir untuk pengaruh skala menunjukkan hasil akurasi terbaik dengan mengambil gambar dari jarak 15cm dengan akurasi menunjukkan 85%.

7.2 Saran

Berdasarkan penelitian yang telah dilakukan dapat beberapa saran yang dapat digunakan untuk pengembangan penelitian ini secara lebih lanjut yakni dengan menggunakan memperbaiki lagi perhitungan algoritme untuk *scaling* dan rotasi. Pencahayan juga mempunyai peran yang sangat penting dalam menentukan hasil dari *keypoint* yang dapat berdampak pada akurasi *matching*. Diharapkan juga pengembangan lebih lanjut mengenai algoritme dalam menanggulangi masalah perbedaan intensitas cahaya.

Daftar Pustaka

Agustina, S. E. & Mukhlash, I., 2012. SIFT. *Implementasi Metode Scale-Invariant Feature Transform (SIFT) dan Metode Continously Adaptive Mean-Shift (Camshift) Pada Penjejakan Objek Bergerak,* pp. 1-2.

Almeida, J., Torres, R. D. S. & Goldenstein, S., 2009. SIFT applied to CBIR. *SIFT applied to CBIR.*

Almeida, J., Torres, R. d. S. & Goldenstein, S., 2009. *SIFT Applied to CBIR.*

Grega, M., Matiolanski, A., Guzik, P. & Leszczuk, M., 2015. Firearms and Knives Detection. *Automated Detection of Firearms and Knives in a CCTV Image,* pp. 1-2.

Lowe, D. G., 2004. Distinctive Image Features from Scale-Invariant Keypoints. *Distinctive Image Features from Scale-Invariant Keypoints.*

Lowe, D. G., 2004. SIFT. *DIstinctive Image Features from Scale-Invariant Keypoints,* pp. 1-20.

Reghava, R. & Muhammad, N., 2016. A Comparative Study of Sift and PCA for Content Based Image Retrieval. *A Comparative Study of Sift and PCA for Content Based Image Retrieval.*

Setiyawan, A. & Basuki, R. S., 2013. SIFT dan Arc Cosinus. *Pecocokan Citra Berbasis Scale-Invariant Feature Transform (SIFT) Menggunakan Arc Consinus,* pp. 1-2.

Sinha, U., n.d. *SIFT : Theory and Practice.* [Online]
Available at: http://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/

Tores, R. D. S. & Falcao, A. X., 2006. CBIR Theory and Application. *CBIR Theory and Application.*

Zhai, C., 2015. *Coursera.* [Online]
Available at: https://www.coursera.org/lecture/text-retrieval/lesson-1-3-text-retrieval-problem-CXoWB