TUGAS AKHIR – KI141502

APLIKASI DETEKSI KEMUNCULAN BULAN SABIT BERBASIS ANDROID MENGGUNAKAN METODE FAST RANDOMIZED HOUGH TRANSFORM

ACHMAD RIZKY HAQIQI
5113100075

Dosen Pembimbing
Dr. Agus Zainal Arifin, S. Kom., M. Kom.
Dwi Sunaryono, S. Kom., M. Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

APLIKASI DETEKSI KEMUNCULAN BULAN SABIT BERBASIS ANDROID MENGGUNAKAN METODE FAST RANDOMIZED HOUGH TRANSFORM

ACHMAD RIZKY HAQIQI
5113100075

Dosen Pembimbing
Dr. Agus Zainal Arifin, S. Kom., M. Kom.
Dwi Sunaryono, S. Kom., M. Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[*Halaman ini sengaja dikosongkan*]

FINAL PROJECT - KI141502

LUNAR CRESCENT DETECTOR APPLICATION BASED ON ANDROID USING FAST RANDOMIZED HOUGH TRANSFORM

ACHMAD RIZKY HAQIQI
NRP 5113 100 075

Supervisor I
Dr. Agus Zainal Arifin, S.Kom., M.Kom.

Supervisor II
Dwi Sunaryono, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
Sepuluh Nopember Institute of Technology
Surabaya, 2017

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

APLIKASI DETEKSI KEMUNCULAN BULAN SABIT BERBASIS ANDROID MENGGUNAKAN METODE FAST RANDOMIZED HOUGH TRANSFORM

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Komputasi Cerdas dan Visi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
ACHMAD RIZKY HAQIQI
NRP. 5113 100 075

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Dr. Agus Zainal Arifin, S.Kom., M.Kom.
NIP. 19720809 199512 1 001 ..... (Pembimbing 1)

Dwi Sunaryono, S.Kom., M.Kom.
NIP. 19720528 199702 1 001 ..... (Pembimbing 2)

SURABAYA
JULI, 2017

[*Halaman ini sengaja dikosongkan*]

APLIKASI DETEKSI KEMUNCULAN BULAN SABIT BERBASIS ANDROID MENGGUNAKAN METODE FAST RANDOMIZED HOUGH TRANSFORM

Nama Mahasiswa	: Achmad Rizky Haqiqi
NRP	: 5113 100 075
Jurusan	: Teknik Informatika, FTIf ITS
Dosen Pembimbing 1	: Dr. Agus Zainal Arifin, S.Kom., M.Kom.
Dosen Pembimbing 2	: Dwi Sunaryono, S.Kom., M.Kom.

Abstrak

Kemunculan bulan sabit dapat digunakan untuk menentukan penanggalan awal pada kalender bulan. Namun bentuk bulan sabit pada fase awalnya menimbulkan subjektifitas penglihatan yang tinggi karena bentuknya yang tipis dan cenderung samar pada sore hari.

Salah satu solusi untuk mengatasinya yaitu menggunakan aplikasi mobile pada smartphone yang dapat mendeteksi kemunculan bulan sabit secara objektif. Keberadaan aplikasi mobile memudahkan pengguna dalam pengoperasiannya karena proses pengambilan dan pengolahaan citra dapat dilakukan pada perangkat yang sama. Namun, saat ini kemampuan komputasi pada smartphone masih terbatas sehingga memiliki performa yang lambat dalam menjalankan aplikasi yang memiliki komputasi tinggi. Oleh karena itu, aplikasi mobile yang dikembangkan membutuhkan metode yang efisien dalam hal komputasi sehingga menghasilkan performa yang bagus.

Makalah ini mengembangkan suatu aplikasi mobile dengan menggunakan metode Fast Randomized Hough Transform (FRHT) untuk mendeteksi kemunculan bulan sabit. Awalnya, aplikasi akan mendeteksi tepi dari berbagai objek pada citra yang didapatkan dari kamera smartphone. Selanjutnya, tepi objek tersebut dapat mengarahkan aplikasi untuk medeteksi objek bulan kemudian menghasilkan tanda berupa lingkaran penuh pada citra input dengan menggunakan FRHT. Hasil uji coba pada dua dataset yang memiliki karakteristik berbeda, menunjukkan keberhasilan akurasi pendeteksian sebesar 95% dari 120 citra input uji coba.

Kata kunci: Bulan Sabit, Android, Smartphone, Fast Randomized Hough Transform.

[*Halaman ini sengaja dikosongkan*]

LUNAR CRESCENT DETECTOR APPLICATION BASED ON ANDROID USING FAST RANDOMIZED HOUGH TRANSFORM

Student Name	: Achmad Rizky Haqiqi
Student ID	: 5113 100 075
Department	: Informatics Engineering, FTIf ITS
First Supervisor	: Dr. Agus Zainal Arifin, S.Kom., M.Kom.
Second Supervisor	: Dwi Sunaryono, S.Kom., M.Kom.

Abstract

The appearance of the crescent moon can be used to determine the initial calendar on the lunar calendar. But the shape of the crescent moon in its initial phase gives high visual subjectivity because of its thin shape and tend to be faint in the afternoon.

One solution to overcome that is using mobile applications on the smartphone that can detect the appearance of the crescent moon objectively. The existence of mobile application facilitates the user in operation because the process of capturing and image processing can be done on the same device. However, at this time the ability of computing on smartphones is still limited so that has a slow performance in running applications that have high computing. Therefore, the developed mobile application requires an efficient method in terms of computing so as to produce a good performance.

This paper develops a mobile application using the Fast Randomized Hough Transform (FRHT) method to detect the appearance of the crescent moon. Initially, the application will detect the edge of various objects in the image obtained from the smartphone camera. Furthermore, the edges of the object may direct the application to detect the object of the moon and then generate a full circle sign on the input image using FRHT. The test results on two datasets having different characteristics show the detection accuracy of 95% of the 120 test input images.

Keywords : Crescent Moon, Android, Smartphone, Fast Randomized Hough Transform.

[*Halaman ini sengaja dikosongkan*]

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Segala puji bagi Allah SWT yang telah melimpahkan rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul **"Aplikasi Deteksi Kemunculan Bulan Sabit Berbasis Android Menggunakan Metode Fast Randomized Hough Transform".**

Buku tugas akhir ini disusun dengan harapan dapat memberikan manfaat dalam penilitian mengenai deteksi kemunculan bulan sabit lebih lanjut. Selain itu, penulis berharap dapat memberikan kontribusi positif bagi kampus Teknik Informatika ITS.

Dalam perancangan, pengerjaan, dan penyusunan tugas akhir ini, penulis banyak mendapatkan bantuan dari berbagai pihak. Penulis ingin mengucapkan terima kasih kepada:

1. Bapak Dr. Agus Zainal Arifin, S.Kom., M.Kom. dan Bapak Dwi Sunaryono, S.Kom., M.Kom. selaku dosen pembimbing penulis yang telah memberi ide, nasihat dan arahan sehingga penulis dapat menyelesaikan tugas akhir dengan tepat waktu.
2. Orang tua penulis Bapak Mohammad Affan dan Ibu Fitriya serta Adik penulis yang telah memberikan dukungan moral, spiritual dan material serta senantiasa memberikan doa demi kelancaran dan kemudahan penulis dalam mengerjakan tugas akhir.
3. Seluruh peserta Zemi yang secara tidak langsung memberikan inspirasi dan semangat untuk terus konsisten. Khususnya kepada Mbak Raras, Mbak Riska dan Bapak Saikhu yang telah membantu dalam memberi masukan kepada penulis, sehingga menjadi lebih baik.
4. Teman-teman Burgator (Harry, Ayu, Devira, Budi, Gian dan Franky) dan Admin KCV 2013 (Nida, Nela, Ihsan, Reza,

Rizok) yang secara aktif memberikan semangat, mendukung dan menemani penulis dikala senang maupun susah.

5. Teman-teman di Lab KCV: user TA KCV 2017, Ilham Gurat dan Admin lain yang telah banyak membantu, memberi pencerahan dalam pengerjaan serta teman-teman yang berhabitat hidup di KCV, yang sudah menemani penulis.
6. Pihak-pihak lain yang tidak bisa penulis sebutkan satu-persatu.

Penulis menyadari masih ada kekurangan dalam penyusunan tugas akhir ini. Penulis mohon maaf atas kesalahan, kelalaian maupun kekurangan dalam penyusunan tugas akhir ini. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan ke depan.

Surabaya, Juli 2017

Penulis

DAFTAR ISI

DAFTAR GAMBAR

[*Halaman ini sengaja dikosongkan*]

DAFTAR TABEL

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE PROGRAM

[*Halaman ini sengaja dikosongkan*]

BAB I
PENDAHULUAN

Pada bab ini dibahasn hal-hal yang mendasari tugas akhir. Bahasan meliputi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan tugas akhir.

1.1 Latar Belakang

Penanggalan dalam kalender bulan (lunar calendar) dihitung berdasarkan durasi yang dibutuhkan bulan dalam mengitari bumi selama satu kali. Oleh karena itu, penentuan tanggal 1 pada kalender bulan, dapat dilakukan dengan melihat kemunculan bulan sabit, hilal, pada waktu pergantian siang menuju malam. Ketika, bentuk fase awal bulan sabit muncul, itulah pertanda bahwa pergantian bulan dalam kalender lunar terjadi.

Namun, upaya penglihatan bulan sabit pada fase awalnya tidak semudah layaknya melihat bulan pada fase tengahnya. Hal ini disebabkan oleh bentuk bulan yang tipis, dan latar langit yang masih cerah di sore hari sehingga mudah terlewatkan oleh mata. Oleh karena itu, dibutuhkan teropong bintang agar dapat mendapatkan penglihatan yang lebih besar dan jelas. Selain itu, teropong yang digunakan juga harus memiliki sistem pengarahan secara mekanis, agar dapat mengarahkan teropong pada suatu posisi yang tepat tanpa harus menunggu obyek bulan sabit muncul terlebih dahulu.

Meskipun telah dibantu teropong, faktor subyektifitas manusia masih cukup mempengaruhi, karena kemampuan dari setiap manusia berbeda, contohnya kepekaan dalam melihat warna hingga kerabunan. Sari [1] dalam penelitiannya, mengatasi masalah ini dengan membangun sebuah aplikasi pada komputer yang dapat mendeteksi kemunculan bulan sabit dengan menggunakan video yang telah direkam sebelumnya. Metode Circular Hough Transform (CHT) [2], salah satu metode untuk menemukan bentuk-bentuk geometri yang menggunakan proses

voting, digunakan untuk mendeteksi obyek lingkaran. Pada uji coba, hasilnya bagus dalam mendeteksi kemunculan bulan sabit. Namun aplikasi ini memiliki portabilitas yang minim, karena proses dilakukan secara terpisah. Contohnya mengenai pengambilan data, dimana harus dilakukan oleh kamera, sedangkan proses pendeteksiannya dilakukan oleh komputer, hal ini membutuhkan proses pemindahan file yang menyebabkan minimnya portabilitas.

Untuk mengatasi permasalahan diatas, dibutuhkan perangkat yang dapat melakukan dua tugas sekaligus, yaitu mengambil data dan juga mengolahnya secara langsung. Kemampuan ini dapat ditemukan pada smartphone Android, yang memiliki performa mumpuni pada kamera dan proses komputasinya. Namun, kemampuan smartphone dalam hal kecepatan sangat berbeda dengan komputer, dimana kecepatan pemrosesan pada smartphone lebih lambat dibandingkan komputer karena keterbatasan sumber daya. Hal ini menjadi masalah jika menerapkan algoritma CHT yang memiliki waktu komputasi tinggi serta membutuhkan kebutuhan penyimpanan yang besar. Oleh karena itu, metode CHT tidak dapat berjalan dengan cepat terutama pada smartphone.

Chiu[3], dalam penelitiannya menemukan metode Fast Randomized Hough Transform (FRHT) yang merupakan pengembangan dari Hough Transform Klasik. FRHT memilih sebuah titik pada citra secara acak, kemudian menggunakan titik tersebut untuk mencari 2 titik lain yang berjarak sama. Tiga titik tadi, dilakukan pengecekan apakah tiga titik tersebut ada pada garis lingkaran yang sama. Pada uji coba dengan menggunakan citra yang cukup kompleks, metode ini melakukan deteksi lingkaran dengan performa yang lebih baik dari Hough Transform Klasik, karena waktu komputasi dalam mendeteksi lingkaran dapat dipangkas hingga 99%. Melihat potensi ini, FRHT sangat mungkin diterapkan untuk mendeteksi kemunculan bulan sabit pada smartphone karena masalah portabilitas, dan efisiensi waktu komputasi dapat diatasi.

Penelitian ini mengembangkan suatu aplikasi mobile dengan menggunakan metode Fast Randomized Hough Transform (FRHT) untuk mendeteksi kemunculan bulan sabit. Diharapkan aplikasi dapat memberikan hasil yang akurat namun dengan beban komputasi yang ringan, serta memiliki portabilitas yang tinggi.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana cara mendapatkan objek bulan sabit yang tipis?
2. Bagaimana cara untuk mendapatkan kandidat obyek lingkaran yang memiliki kemungkinan tinggi sebagai busur lingkaran bulan sabit?
3. Bagaimana cara menerapkan metode Fast Randomized Hough Transform pada platform Android untuk deteksi kemunculan bulan sabit?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan antara lain:

1. Citra yang digunakan adalah citra foto dari penglihatan teropong yang diambil dengan kamera smartphone.
2. Bentuk geometris Bulan Sabit yang dideteksi adalah minimal seperempat dari lingkaran utuh.
3. Metode diimplementasikan menggunakan Bahasa Java.

1.4 Tujuan Tugas Akhir

Tujuan dari tugas akhir ini adalah membuat perangkat lunak berbasis Android yang dapat mendeteksi kemunculan bulan sabit pada sebuah citra dengan menggunakan metode Fast Randomized Hough Transform.

1.5 Manfaat Tugas Akhir

Manfaat dari tugas akhir ini adalah menghasilkan aplikasi deteksi kemunculan bulan sabit menggunakan Fast Randomized Hough Transform berbasis Android secara akurat. Hasil dari deteksi dapat digunakan untuk menentukan apakah pergantian bulan telah terjadi.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Studi Literatur

   Pada studi literatur, dilakukan pengumpulan data dan studi terhadap sejumlah referensi yang diperlukan dalam pengerjaan tugas akhir. Referensi tersebut didapatkan dari beberapa artikel yang dipublikasikan oleh jurnal. Selain dari artikel, studi literatur juga dilakukan melalui pencarian referensi dari internet yang membahas mengenai informasi yang dibutuhkan, seperti Bulan Sabit, Deteksi Tepi Canny, Android, Open CV, Fast Randomized Hough Transform.

2. Pengumpulan dataset.

   Dataset yang digunakan dalam tugas akhir ini didapat dengan cara mengekstrak video hasil *ru'yatul hilal* menjadi *frame-frame* citra. Kedua, dataset diambil menggunakan kamera smartphone Android.

3. Analisis dan Desain Perangkat Lunak

   Pada tahap ini disusun rancang bangun dari perangkat lunak yang dibangun. Pengguna dapat memasukkan citra dari file, maupun kamera. Kemudian, sistem akan memproses citra dengan melakukan *preprocessing,* dan FRHT. Setelah proses selesai, sistem akan menampilkan citra bulan sabit yang telah ditandai posisinya.

4. Implementasi Perangkat Lunak

   Tahap ini meliputi perancangan algoritma dan perancangan antar muka. Rancangan algoritma berbentuk diagram alir

(flowchart) yang dibuat dengan tool Google Draw IO. Sedangan rancangan antar muka dibuat dengan tool Android Studio. Tahap perancangan perangkat lunak ini merupakan tahapan yang penting karena pengerjaan Tugas Akhir selanjutnya akan mengacu pada hasil rancangan pada tahap ini.

5. Uji Coba dan Evaluasi

   Uji coba dan evaluasi perangkat lunak dilakukan untuk menentukan parameter-parameter yang tepat. Parameter yang tepat adalah parameter yang dapat menghasilkan akurasi yang tinggi ketika dilakukan uji coba pada aplikasi deteksi kemunculan bulan sabit. Selain itu, dilakukan uji coba pada citra real, menggunakan kamera pada smartphone.

1.7 Sistematika Laporan

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Dasar Teori

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan

Bab ini berisi tentang analisis dan perancangan desain Aplikasi deteksi kemunculan bulan sabit berbasis Android menggunakan Fast Randomized Hough Transform

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode yang digunakan untuk proses implementasi.

Bab V Uji Coba dan Evaluasi

Bab ini membahas tahap-tahap uji coba. Kemudian hasil uji coba dievaluasi untuk kinerja dari aplikasi yang dibangun.

Bab VI Kesimpulan dan Saran

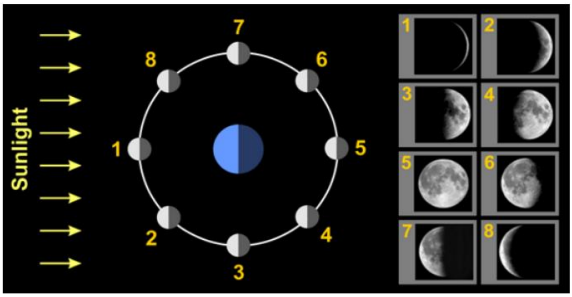
Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan aplikasi ke depannya.

BAB II
DASAR TEORI

Pada bab ini diuraikan mengenai dasar-dasar teori yang digunakan dalam pengerjaan tugas akhir dengan tujuan untuk memberikan gambaran secara umum terhadap penelitian yang dikerjakan. Penjelasan tersebut bertujuan untuk memberikan gambaran mengenai sistem yang akan dibangun dan berguna sebagai pendukung dalam pengembangan perangkat lunak.

2.1 Bulan Sabit

Kalender bulan didasarkan oleh waktu perputaran bulan mengelilingi bumi. Bulan merupakan satelit alami bumi yang tidak mempunyai sumber cahaya sendiri dan cahaya bulan berasal dari pantulan cahaya matahari. Bulan memiliki beberapa fase yakni bulan purnama adalah keadaan ketika bulan nampak bulat sempurna dari bumi. Kebalikannya adalah saat bulan mati, alias tidak nampak apa-apa. Diantara kedua waktu itu terdapat keadaan bulan separuh dan bulan sabit. Fase bulan dapat dilihat pada **Gambar 2.1** [4].



Gambar 2.1 Fase Bulan Sabit [4]

Cahaya bulan sabit sangatlah lemah bila dibandingkan dengan cahaya matahari maupun cahaya senja, sehingga sulit untuk bisa mengamati bulan sabit yang masih berusia sangat muda [5]. Pada bulan sabit yang sangat muda, beda azimut antara bulan dan matahari sangat kecil (akibatnya jarak sudut antara keduanya pun

kecil) demikian pula dengan luas bulan sabit yang memantulkan sinar matahari. Karena dekatnya jarak sudut bulan-matahari ini, bulan sabit akan terbenam beberapa saat setelah matahari terbenam dan dengan tipisnya bulan sabit yang memantulkan sinar matahari berarti diperlukan latar yang gelap untuk bisa mengamati penampakan bulan sabit.

Oleh karena itu, mengamati bulan sabit bukanlah pekerjaan yang ringan, sebab meskipun bulan sabit berada di atas ufuk saat matahari terbenam ia belum tentu bisa diamati. Sebabnya adalah cahaya bulan sabit yang amat lemah itu kalah dengan cahaya senja. Artinya, agar mata manusia dapat mengamati bulan sabit dengan baik diperlukan kondisi langit yang gelap.Bulan adalah satu-satunya satelit alami bumi, dan merupakan satelit alami terbesar kelima di tata surya. Bulan tidak mempunyai sumber cahaya sendiri dan cahaya bulan sebenarnya berasal dari pantulan cahaya matahari.

2.2 Deteksi Tepi Canny

Deteksi tepi *(edge detection)* pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra [6]. Tujuan dari *edge detection* adalah :

1. Untuk menandai bagian yang menjadi detil citra.
2. Untuk memperbaiki detil dari citra yang kabur, yang terjadi karena derau atau adanya efek dari proses akuisisi citra.

Dalam pengolahan citra digital, identifikasi tepi sebuah obyek dapat memberikan informasi yang berguna untuk keperluan segmentasi. Suatu titik*(x,y)* dikatakan sebagai tepi *(edge)* dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya. Tepi merupakan perubahan nilai intensitas derajat keabuan yang mendadak besar dalam jarak yang singkat.

Deteksi tepi ini dilakukan dengan cara mencari *local maxima* gradien citra *f*(*x*,*y*). Gradien dihitung dengan turunan dari filter Gaussian. Filter Gaussian dirumuskan sebagai berikut :

$$G(x) = e^{-\frac{x^2}{2\sigma^2}} \qquad (2.1)$$

Turunan filter Gaussian terhadap *x* adalah:

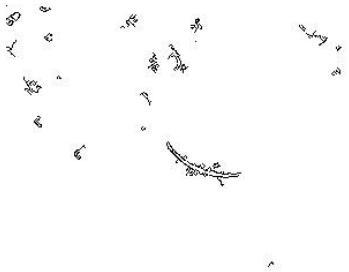
$$G'(x) = \left(-\frac{x}{\sigma^2}\right) e^{-\frac{x^2}{2\sigma^2}} \quad (2.2)$$

Metode ini menggunakan dua *threshold* untuk mendeteksi tepi yang tebal dan yang tipis, serta hanya mengikutkan tepi yang tipis pada citra output jika terhubung dengan tepi yang tebal [1]. Oleh karena itu, metode ini lebih cocok untuk mendeteksi tepi yang tipis.

Secara teknis, metode deteksi tepi Canny bisa dijelaskan sebagai berikut [1] :

1. Citra dihaluskan dengan filter Gaussian dengan standar deviasi yang ditentukan untuk mengurangi *noise.*
2. *Local gradien* $g(x,y) = \left[G_x^2 + G_y^2\right]^{1/2}$, dan arah tepi, $a(x,y) = tan^{-1}\left(G_y/G_x\right)$, dihitung pada tiap titik. Titik tepi didefinisikan sebagai titik yang mempunyai *local maximum* pada arah gradien
3. Titik-titik yang telah ditentukan pada langkah (2) memberikan kenaikan pada bukit gradien citra. Algoritma ini kemudian menelusuri titik sepanjang bukit sampai pada puncaknya dan memberikan nilai 0 pada piksel yang tidak berada pada puncak bukit untuk memberikan output berupa garis tipis. Proses ini dikenal sebagai *nonmaximal suppression*. Piksel-piksel pada bukit kemudian disegmentasi dengan menggunakan dua *threshold*, *T1* dan *T2* dimana *T1* < *T2*. Piksel dengan nilai lebih besar daripada *T2* dikatakan sebagai tepi yang tebal. Sedangkan nilai piksel di antara *T1* dan *T2* disebut tepi yang tipis.

Pada akhirnya, metode ini melakukan penghubungan tepi (*edge linking*) dengan menggabungkan piksel-piksel pada tepi yang tipis yang berketetanggaan 8 dengan piksel pada tepi tebal. Citra hasil deteksi tepi Canny dapat dilihat pada **Gambar 2.2**.



Gambar 2.2 Contoh Citra Tepi

2.3 Android

Android[7] adalah sistem operasi berbasis Linux yang dirancang untuk perangkat bergerak layar sentuh seperti telepon pintar dan komputer tablet. Android awalnya dikembangkan oleh Android, Inc., dengan dukungan finansial dari Google, yang kemudian membelinya pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun 2007, bersamaan dengan didirikannya Open Handset Alliance, konsorsium dari perusahaan-perusahaan perangkat keras, perangkat lunak, dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler. Ponsel Android pertama mulai dijual pada bulan Oktober 2008.

Pada November 2013, Android menguasai pangsa pasar telepon pintar global, yang dipimpin oleh produk-produk Samsung, dengan persentase 64% pada bulan Maret 2013. Pada Juli 2013, terdapat 11.868 perangkat Android berbeda dengan beragam versi. Keberhasilan sistem operasi ini juga menjadikannya sebagai target ligitasi paten "perang telepon pintar" antar perusahaan-perusahaan teknologi. Hingga bulan Mei 2013, total 900 juta perangkat Android telah diaktifkan di seluruh dunia, dan 48 miliar aplikasi telah dipasang dari Google Play. Faktor-faktor di atas telah memberikan kontribusi terhadap perkembangan Android, menjadikannya sebagai sistem operasi telepon pintar yang paling banyak digunakan di dunia, mengalahkan Symbian pada tahun 2010. Android juga menjadi pilihan bagi perusahaan teknologi

yang menginginkan sistem operasi berbiaya rendah, bisa dikustomisasi, dan ringan untuk perangkat berteknologi tinggi tanpa harus mengembangkannya dari awal. Sifat Android yang terbuka juga telah mendorong munculnya sejumlah besar komunitas pengembang aplikasi untuk menggunakan kode sumber terbuka sebagai dasar proyek pembuatan aplikasi, dengan menambahkan fitur-fitur baru bagi pengguna tingkat lanjut atau mengoperasikan Android pada perangkat yang secara resmi dirilis dengan menggunakan sistem operasi lain.

Aplikasi pada Android dikembangkan dalam bahasa pemrograman Java dengan menggunakan kit pengembangan perangkat lunak Android (SDK). SDK ini terdiri dari seperangkat perkakas pengembanga, termasuk debugger, perpustakaan perangkat lunak, emulator handset yang berbasis QEMU, dokumentasi, kode sampel, dan tutorial. Didukung secara resmi oleh lingkungan pengembangan terpadu (IDE) Eclipse, yang menggunakan plugin Android Development Tools (ADT). Perkakas pengembangan lain yang tersedia di antaranya adalah Native Development Kit untuk aplikasi atau ekstensi dalam C atau C++, Google App Inventor, lingkungan visual untuk pemrogram pemula, dan berbagai kerangka kerja aplikasi web seluler lintas platform.

2.4 Open CV

OpenCV [8] (Open Source Computer Vision) adalah sebuah BSD-license (Berkeley Software Distribution-license) open-source library yang mencakup ratusan algoritma visi komputer. OpenCV ini gratis baik untuk penggunaan akademis maupun komersial. Library ini memiliki C++, C, Phyton, dan Java interface yang mendukung Windows, Linux, Mac OS, iOS dan Android. OpenCV dirancang untuk efisiensi komputasi dan dengan fokus yang kuat pada aplikasi real-time. Dituliskan dalam bahasa C/C++ yang optimal, menyebabkan library ini memiliki keuntungan pada multi-core processing. Diaktifkan dengan OpenCL, OpenCV dapat

mengambil keuntungan dari akselerasi hardware pada komputasi pokok platform heterogen. Diadopsi di seluruh dunia, OpenCV memiliki lebih dari 47 ribu orang dari komunitas pengguna dan diperkirakan jumlah download melebihi 9 juta.

Untuk menggunakan OpenCV pada sebuah aplikasi Android, harus meng-install OpenCV Manager sebelumnya. OpenCV Manager adalah sebuah service Android yang ditargetkan untuk mengelola OpenCV binary library pada device end-user. OpenCV Manager ini mengizinkan untuk berbagi OpenCV dynamic library antar aplikasi dengan perangkat yang sama.

2.5 Fast Randomized Hough Transform

Fast Randomized Hough Transform [3] merupakan pengembangan dari Hough Transform Klasik, yang merupakan suatu teknik untuk menentukan lokasi suatu bentuk dalam citra. Transformasi Hough dicetuskan pertama kali oleh P.V.C Hough (1962), dilihat potensinya sebagai salah algoritma dalam pemrosesan citra oleh Rosenfeld (1969), kemudian diimplementasikan untuk mendeteksi garis dalam citra oleh Duda(1972), dan sejak itu mengalami perkembangan yang sangat luas karena banyaknya keunggulan dan besarnya potensi untuk pengembangan lebih lanjut yang ditawarkan algoritma ini. Salah satu pengembangannya adalah metode FRHT ini. Dicetuskan oleh Chiu, metode ini diklaim memiliki performa yang lebih tinggi dibandingkan dengan Hough Transform klasik.

2.5.1 Hough Transform

Hough Transform [9] pertama kali diperkenalkan oleh Paul Hough pada tahun 1962 untuk mendeteksi garis lurus. Hough Transform adalah teknik transformasi citra yang dapat digunakan untuk mengisolasi atau dengan kata lain memperoleh fitur dari sebuah citra. Karena tujuan dari sebuah transformasi adalah mendapatkan suatu fitur yang lebih spesifik, Classical Hough Transform merupakan teknik yang paling umum digunakan untuk

mendeteksi objek yang berbentuk kurva seperti garis, lingkaran, elips dan parabola. Keuntungan utama dari transformasi Hough adalah dapat mendeteksi sebuah tepian dengan celah pada batas fitur dan secara relatif tidak dipengaruhi oleh derau atau noise. Jika objek yang dicari berupa lingkaran, maka digunakan transformasi lingkaran Hough. Prosedur yang digunakan dalam mendeteksi lingkaran adalah sama dengan transformasi Hough pada objek garis, tapi dikerjakan pada ruang dimensi yang lebih kompleks, yaitu dalam parameter ruang 3D (x_0, y_0, r). Di mana x_0 dan y_0 merupakan koordinat pusat lingkaran dan r adalah jari-jari lingkaran seperti persamaan berikut:

$$\sqrt{(x - x_0)^2 + (y - y_0)^2} = r \tag{2.3}$$

2.5.2 Randomized Hough Transform

Randomized Hough Transform (RHT) [10] berbeda dengan Hough Transform, karena RHT mencoba untuk menghindari komputasi yang tinggi pada proses voting untuk setiap nonzero pixel dengan memanfaatkan analisis geometri yang ditentukan oleh titik-titk tertentu pada citra sehingga dapat meningkatkan efisiensi waktu komputasi dan mengurangi kebutuhan penyimpanan dari Hough Tranform. Sebagai contoh, sebuah garis lurus dapat ditentukan dari dua titik, dan sebuah ellipse(lingkaran) dapat ditentukan oleh tiga titik. Proses implementasi RHT secara umum adalah:

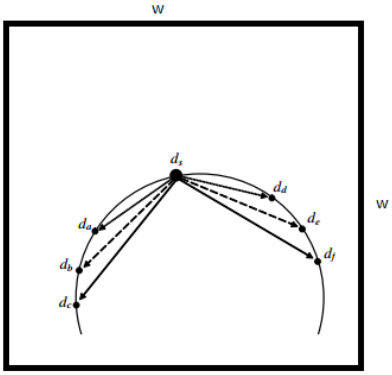
1. Cocokkan lingkaran dengan titik yang terpilih secara acak.
2. Memperbarui akumulator dan nilai yang berkaitan.
3. Memilih dan mengeluarkan lingkaran yang terpilih yaitu lingkaran yang memiliki nilai lebih akumulator tinggi dari ambang batas yang telah ditentukan.

2.5.3 Fast Randomized Hough Transform

Salah satu pembeda antara FRHT dan Randomized Hough Transform (salah satu pengembangan lain dari Hough Transform) adalah pada RHT tiga titik tepi dari citra binary diambil secara

acak, kemudian diperiksa apakah tiga titik tersebut ada dalam lingkaran utuh yang sama. Probabilitas mengambil tiga titik yang berada dalam sebuah lingkaran utuh sangat rendah. Hal ini berbeda jika hanya satu titik yang diambil secara acak yang menyebabkan probabilitas dan performa akan meningkat.

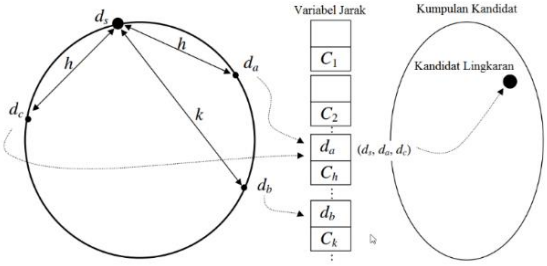
FRHT [3] menggunakan satu titik menjadi seed point yang dipilih secara acak, d_s = (x_s, y_s). Kemudian dicarilah dua titik lain yang memiliki jarak yang sama menuju seed point. Ketika ada dua titik lainnya yang memiliki jarak sama ditemukan, seed point beserta dua titik tadi, dijadikan kandidat obyek sebuah lingkaran. Contoh, d_a dan d_d adalah dua titik dan jarak mereka ke d_s sama, $(\overline{d_s d_a} = \overline{d_s d_d})$. Begitu juga dengan $(\overline{d_s d_b} = \overline{d_s d_e})$ dan $(\overline{d_s d_c} = \overline{d_s d_f})$. Jika terdapat tiga titik dapat menemukan lingakaran, maka $(d_s, d_a, d_d), (d_s, b, d_e)$, dan (d_s, d_c, d_f) dapat digunakan sebagai kandidat lingkaran.



Gambar 2.3 Tiga pasang kandidat lingkaran yang jarak antar pasangan sama

Berdasarkan teori diatas, kita membutuhkan dua titik yang jaraknya sama dengan seed point. Untuk menampung seluruh kandidat, diperlukan variabel untuk menyimpan kandidat-kandidat tersebut dan dapat berfungsi sebagai accumulator. Variabel ini berbasis jarak, sehingga jika terdapat dua titik dengan jarak yang

sama, maka nilainya akan masuk ke variabel yang sama. Ketika sebuah seed point dipilih, diinisiasi variable yang dimaksud tadi, dan dikosongkan. Kotak berukuran simetris dibuat dengan d_s sebagai titik tengahnya. Didalam kotak tersebutlah, setiap titik tepi dilakukan pengecekan satu persatu kecuali d_s dan dihitung jarak antara setiap titik tepi tadi dengan d_s. Jika variabel yang dimaksud tadi kosong, maka data titik tadi langsung dimasukkan kedalam variabel. Sebaliknya, jika tenyata variabel telah ada isinya, titik yang diekstraksi sebelumnya, seed point dan titik saat ini untuk dimasukkan kedalam variabel berbasis jarak tadi. Gambar pada proses diatas ada pada gambar 9.



Gambar 2.4. Pengecekan seed point dan mekanisme akumulator

Jika titik kandidat lingkaran tidak ada dalam variabel, maka kandidat tadi dimasukkan ke dalam pool dengan *accumulator* (accumulator dinilai 1) sebaliknya jika kandidat lingkaran telah ada dalam variabel, maka nilai *accumulator* ditambah 1. Ketika setiap titik tepi dalam kotak(windows) diproses, dipilihlah kandidat dengan nilai akumulasi tertinggi dari variabel kandidat untuk kita jadikan lingkaran yang kita cari. Asumsi sebuah lingkaran yang dipilih dinotasikan (x'_c, y'_c, r_c') dimana (x'_c, y'_c) adalah titik pusat lingkaran dan r_c' adalah jari-jarinya. Titik tepi padahal lingkaran yang dipilih dapat dideskripsikan:

$$\tilde{D} = \left\{ d_i \middle| \sqrt{(x_i - x'_c)^2 + (y_i - y'_c)^2} - r'_c \middle| \leq \varepsilon_r, d_i \in D \right\} \qquad (2.4)$$

Dimana ε_r adalah nilai threshold yang dapat diatur. Berdasarkan paper yang direferensi, nilai ε_r adalah 1. Untuk mendeteksi sebuah kandidat lingkaran adalah lingkaran sebenarnay dapat jika:

$$\frac{N_c}{2r'_c\pi} > R_t \tag{2.5}$$

dimana N_c adalah angka jumlah titik dalam citra, dan R_t adalah perbandingan antara jumlah titik dengan jumlah lingkaran yang sesuai.

2.6 Metode Perhitungan Akurasi

Perhitungan akurasi pada Tugas Akhir ini digunakan untuk menghitung kinerja algoritma dalam mendeteksi kemunculan bulan sabit pada citra masukan, yakni apakah di dalam sebuah citra terdapat kemunculan bulan sabit dengan mendeteksi adanya obyek bulan sabit pada citra tersebut. Perhitungan akurasi ini menggunakan persamaan (2.6) hingga (2.8) [11]:

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.6}$$

$$Specificity = \frac{TN}{TN + FP} \tag{2.7}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{2.8}$$

Dimana,

- TP *(True Positive)* : pada citra masukan terdapat komponen bulan sabit, pada citra keluaran dideteksi terdapat bulan sabit.
- TN *(True Negative)* : pada citra masukan tidak terdapat komponen bulan sabit, pada citra keluaran dideteksi tidak terdapat bulan sabit.

- FP *(False Positive)* : pada citra masukan terdapat komponen bulan sabit, pada citra keluaran dideteksi tidak terdapat bulan sabit.
- FN *(False Negative)* : pada citra masukan tidak terdapat komponen bulan sabit, pada citra keluaran dideteksi terdapat bulan sabit.

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN

Bab analisis dan perancangan berisi analisis kebutuhan dan perancangan aplikasi yang akan dibangun. Perancangan ini meliputi perancangan data, perancangan proses, dan perancangan sistem, serta juga akan dijelaskan tentang analisis implementasi metode secara umum pada sistem.

3.1 Deskripsi Umum Sistem

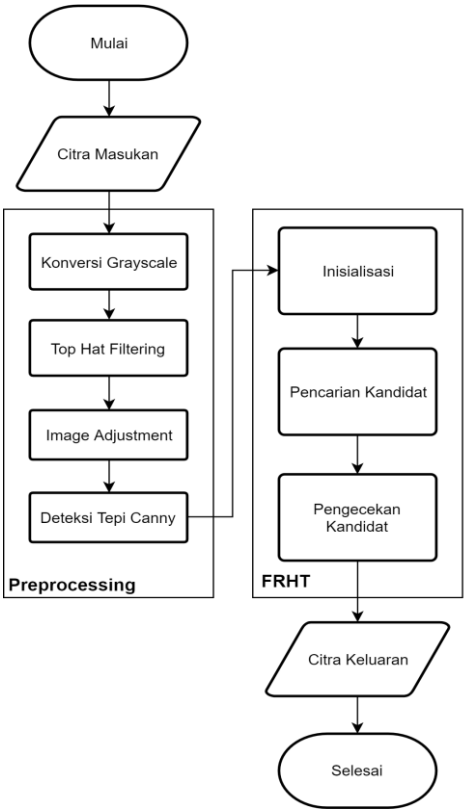
Pada tugas akhir ini akan dibangun sebuah sistem untuk mendeteksi kemunculan bulan sabit menggunakan Fast Randomized Hough Transform pada Android. Secara garis besar, terdapat dua tahap proses dalam melakukan deteksi kemunculan bulan sabit yaitu, preprocessing dan Fast Randomized Hough Transform(FRHT).

Pada tahap preprocessing, citra masukan yang didapatkan dari kamera diubah kedalam bentuk grayscale, kemudian dilakukan tophat filtering agar perbedaan obyek bulan dan langit terlihat lebih jelas. Setelah itu dilakukan deteksi tepi canny, untuk mendapatkan titik tepi yang kemudian menjadi input pada tahap FRHT.

Berikutnya merupakan tahap FRHT dimana pada tahap ini kandidat-kandidat lingkaran diekstraksi dari citra tepi. Setelah itu penghitungan jumlah kandidat (voting) dilakukan untuk menentukan kandidat lingkaran yang paling relevan dengan citra. Namun, tak hanya memilki nilai voting tertinggi, kandidat tersebut harus memiliki nilai rasio lingkaran yang sudah ditentukan sebelumnya. Program akan berhenti jika telah mencapai iterasi maksimum atau telah menemukan lingkaran dengan rasio minimum.

Terakhir, program akan menampilkan citra dengan lingkaran yang menandai posisi dari obyek bulans abit yang

terdeteksi. Diagram Alir rancangan umum sistem terdapat pada **Gambar 3.1**



Gambar 3.1 Diagram Alir Aplikasi

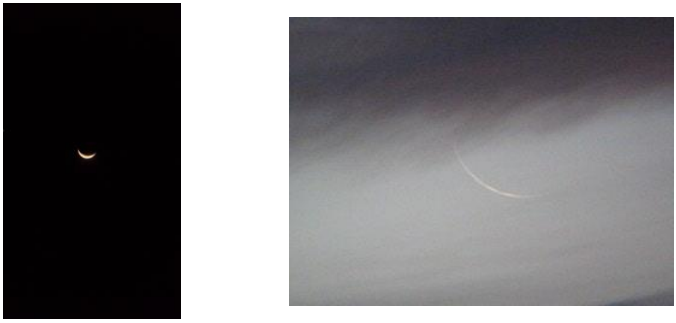
3.2 Perancangan Data

Pada subbab ini akan dibahas mengenai perancangan data yang merupakan bagian penting karena data sebagai obyek yang akan diolah oleh perangkat lunak dan menghasilkan sebuah informasi. Data yang akan digunakan pada sistem ini adalah data masukan

(*input*), dan data keluaran (*output*) yang memberikan hasil pengolahan sistem ini untuk pengguna.

3.2.1 Data Masukan

Data masukan merupakan data awal yang akan diproses oleh sistem untuk mendeteksi kemunculan bulan sabit. Data masukan pertama merupakan citra yang diambil menggunakan kamera smartphone berukuran 1080 x 1920 pixel. Contoh data terdapat pada **Gambar 3.2** (a). Data masukan kedua merupakan citra hasil ekstrasi frame video ru'yatul hilal yang disimpan dalam citra RGB dengan ukuran 320 x 240 piksel. Contoh citra data masukan kedua ditunjukkan pada **Gambar 3.2** (b).



(a) (b)

Gambar 3.2 Contoh Citra Masukan. (a) Citra Dataset 1. (b) Citra Dataset 2.

3.2.2 Data Keluaran

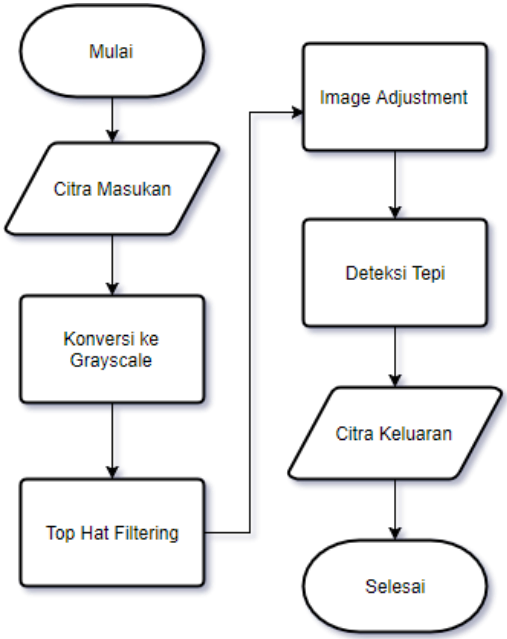
Data keluaran dari sistem ini berupa kandidat lingkaran yang memenuhi syarat-syarat yang telah ditentukan. Kandidat lingkaran ini selanjutnya digunakan sebagai dasar menggambar lingkaran pada citra awal, untuk membantu user dalam mengidentifikasi posisi bulan sabit yang terdeteksi.

3.3 Perancangan Proses

Pada subbab ini akan dibahas mengenai perancangan proses yang dilakukan untuk memberikan gambaran secara rinci pada

setiap alur aplikasi deteksi kemunculan bulan sabit menggunakan Fast Randomized Hough Transform berbasis Android. Alur tersebut nantinya akan diimplementasikan pada aplikasi.

3.3.1 Tahap Preprocessing



Gambar 3.3 Diagram Alir Preprocessing

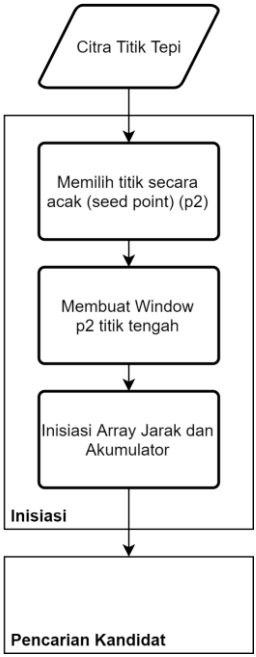
Pada tahap preprocessing, citra bulan sabit yang didapatkan dari video maupun kamera, akan diproses guna mempersiapkan citra sebelum masuk ke tahap berikutnya. Pada awalnya, citra akan diubah dalam bentuk grayscale kemudian Tophat Filtering dilakukan guna menyelaraskan background yang memiliki banyak noise (awan), sehingga obyek bulan sabit lebih jelas perbedaannya. Berikutnya, citra dilakukan adjustment untuk menambah ketegasan antara obyek bulan sabit dan background langit. Bagian terakhir dari Preprocessing merupakan deteksi tepi

Canny. Canny dipilih karena menghasilkan tepi yang tipis dibandingkan dengan deteksi titik tepi lainnya. Diagram alir mengenai tahap ini dapat dilihat pada **Gambar 3.3**.

3.3.2 Tahap Fast Randomized Hough Transform

Tahap ini merupakan tahap inti dari proses pendeteksian. Input FRHT ini adalah citra hasil proses preprocessing berupa citra hasil deteksi tepi canny yang berisi edge point/garis-garis tepi dari citra aslinya. FRHT dapat dibagi menjadi tiga bagian, yaitu inisiasi, pencarian kandidat, dan pengecekan kandidat.

3.3.2.1 Inisiasi

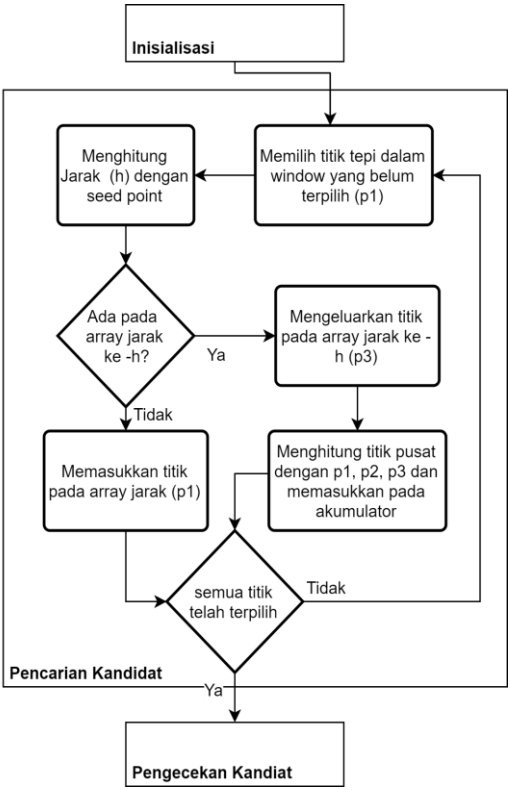


Gambar 3.4 Diagram Alir Inisiasi

Pada bagian ini, satu titik tepi akan dipilih secara acak pada citra titik tepi kemudian titik tersebut dijadikan sebuah seed point (p2). Selanjutnya window w x w akan dibuat dengan titik p2

sebagai titik pusatnya. Setelah itu array jarak dan akumulator di inisiasi. Variabel array jarak akan digunakan dalam pencarian kandidat lingkaran, dan akumulator sebagai penampung serta penghitung kandidat lingkaran. Proses ini dijalankan secara iterasi terus menerus hingga bertemu dengan minimal kriteria lingkaran (rasio lingkaran) yang dicari atau mencapai batas iterasi maksimum.

3.3.2.2 Pencarian Kandidat

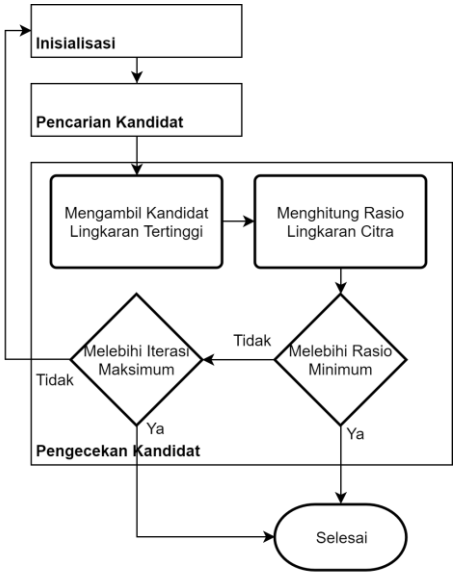


Gambar 3.5 Diagram Alir Pencarian Kandidat

Pada bagian ini, setiap titik yang ada didalam window diproses. Proses yang dilalui yaitu, setiap titik(p1) akan dicari

jaraknya dengan seed point. Kemudian dilakukan pengecekan apakah array jarak ke-h kosong, jika iya, titik tersebut akan dimasukkan ke dalam array index ke-h dan kembali memproses titik berikutnya. Namun, jika tidak, titik pada array ke-h tadi akan dikeluarkan sebagai(p3) dan dilakukan penghitungan titik pusat dan jari-jari dengan parameter p1, p2 dan p3 untuk menjadi kandidat lingkaran. Kemudian hasilnya akan dimasukkan kedalam akumulator, jika pada iterasi berikutnya menghasilkan kandidat lingkaran yang sama, maka nilai voting kandidat tersebut meningkat. Ketika semua proses telah dilakukan, data kandidat akan diteruskan kepada bagian selanjutnya, yaitu pengecekan kandidat.

3.3.2.3 Pengecekan Kandidat



Gambar 3.6 Diagram Alir Pengecekan Kandidat

Pada bagian ini, data kandidat yang memiliki voting tertinggi dipilih. Kemudian titik tadi dihitung rasio lingkarannya menggunakan persamaan (2.5), dan nilai rasionya dibandingkan

dengan nilai rasio minimum (2.6). Jika rasio melebihi syarat yang diberikan, maka iterasi dihentikan, namun jika tidak, maka dilakukan pengecekan, apakah telah melebihi iterasi maksimum atau tidak. Jika belum, kembali ke bagian inisiasi untuk mengulangi proses yang sama. Iterasi maksimum digunakan untuk menghindarkan program pada iterasi yang tidak diperlukan dalam arti tidak menghasilkan kandidat yang signifikan sehingga dapat dikatakan pada citra tersebut tidak ada obyek bulan sabit. Sedangkan nilai rasio minimum digunakan untuk merepresentasikan ukuran busur lingkaran terkecil yang dikategorikan sebagai bulan sabit.

BAB IV
IMPLEMENTASI

Pada bab ini diuraikan mengenai implementasi perangkat lunak dari rancangan metode yang telah dibahas pada Bab 3 meliputi kode program dalam perangkat lunak. Selain itu, implementasi dari tiap proses dan beberapa keterangan yang berhubungan dengan program juga akan dijelaskan.

4.1 Lingkungan Implementasi

Obyek citra yang akan diolah pada implementasi tugas akhir ini adalah kumpulan citra bulan sabit. Obyek citra bulan sabit adalah gambar tipe *jpg*.

Dalam implementasi aplikasi deteksi kemunculan bulan sabit menggunakan Fast Randomized Hough Transform berbasis Android, digunakan perangkat-perangkat sebagai berikut:

4.1.1 Perangkat Keras

Lingkungan implementasi pada tugas akhir ini adalah sebuah *personal computer* (PC). Perangkat PC yang digunakan adalah laptop bermerk DELL 5458.

Spesifikasi dari PC yang digunakan pada tugas akhir ini adalah memiliki prosesor Intel Core i3 5005U dengan kecepatan 2,0 GHz dan *Random Access Memory* (RAM) untuk proses menjalankan program sebesar 8,00 GB.

4.1.2 Perangkat Lunak

Lingkungan implementasi pada tugas akhir ini adalah sebuah *personal computer* (PC). Spesifikasi PC dari sisi perangkat lunak menggunakan *software* Eclipse Java EE dan Android Studio. Penggunaan software didukung dengan satu *library* utama yaitu Open CV.

Selain itu, pada tugas akhir ini dalam melakukan pengolahan angka didukung dengan *software Microsoft Excel*.

4.2 Implementasi Proses

Implementasi proses dilakukan berdasarkan perancangan proses yang sudah dijelaskan pada bab analisis dan perancangan. Implementasi ini menggunakan *library* bantuan OpenCV Java API untuk tahap preprocessing, dan Fast Randomized Hough Transform.

4.2.1 Implementasi Tahap Preprocessing

Pada subbab ini akan dijelaskan mengenai implementasi pada tahap preprocessing. Implementasi tahap ini dimulai dengan menyesuaikan citra input menjadi grayscale. Berikut ini adalah implementasi proses konversi citra kedalam bentuk grayscale yang ditunjukkan oleh **Kode Program 4.1**.

```
1.  //convert to grayscale
2.  Mat grayscaleMat = new Mat(matSrc.size(),matSrc.type())
    ;
3.  Imgproc.cvtColor(matSrc, grayscaleMat, Imgproc.COLOR_BG
    R2GRAY);
```

Kode Program 4.1. Konversi Grayscale

Pada baris 2, dilakukan inisiasi terlebih dahulu Matrix untuk menampung output citranya. Ukuran dari matrix disamakan dengan citra masukan. Pada baris 3, dilakukan konversi menggunakan menuju grayscale, proses ini dibantu oleh library OpenCV.

Setelah diubah menjadi citra grayscale, langkah selanjutnya adalah melakukan top hat filtering terhadap citra grayscale tersebut untuk menyelaraskan background dengan obyek agar background menjadi seragam dan obyek bulan sabit menjadi jelas. Implementasi dari proses ini dapat dilihat pada **Kode Program 4.2.** Proses filtering ini dibantu oleh library OpenCV seperti pada baris 5. Proses pertama dalam top hat filtering adalah menginisiasi structuring element yang merupakan salah satu parameter fungsi tophat filtering. Operasi tophat pada library OpenCV merupakan operasi ke-5 dari fungsi tersebut, sehingga isi variabel operation

adalah 5. Baris 6 pada kode program berfungsi untuk mengubah pixel hitam menjadi putih dan sebaliknya.

```
1.  // TopHat
2.  int operation = 5;
3.  Mat dst = new Mat();
4.
5.  Imgproc.morphologyEx(src, dst, operation, element);
6.  Core.bitwise_not(dst, dst);
7.  Imgcodecs.imwrite("saved2TopHat.jpg", dst);
```

Kode Program 4.2. Tophat Filtering

Langkah berikutnya dari proses preprocessing ini adalah Image Adjustment. Hal ini dilakukan untuk mempertegas bentuk obyek bulan sabit sehingga semakin terlihat berbeda dengan background langit. Implementasi pada sistem untuk langkah ini ditunjukkan pada **Kode Program 4.3**. Adjustment disini menggunakan threshold dengan nilai 0 atau 255.

```
1.  Imgproc.threshold(dst, dst, 245, 255, Imgproc.THRESH_BI
    NARY);
```

Kode Program 4.3. Image Adjustment

Langkah terakhir pada tahap ini adalah deteksi tepi Canny. Hasil dari tahap ini merupakan citra tepi tipis dari obyek bulan dan bentuk lainnya. Implementasi pada langkah ini dapat dilihat pada **Kode Program 4.4**. Proses deteksi tepi Canny diawali dengan proses blurring seperti pada baris baris 3. Baris 7, menunjukkan proses pemanggilan library OpenCV untuk melakukan deteksi tepi Canny. Nilai threshold pada parameter deteksi tepi Canny merupakan batas bawah dan atas untuk memberi titik tepi. Nilai threshold Canny diatur sebesar 19.4 karena dapat menghasilkan garis-garis tepi obyek bulan sabit yang baik dan garis-garis noise yang minim. Output dari proses ini akan menjadi input pada tahap

berikutnya. Kemudian invers citra dilakuakn kembali untuk mengubah pixel putih menjadi hitam dan sebaliknya seperti yang ditunjukkan pada baris 8 program.

```
1.  public static Mat edgeDetector(Mat src) {
2.    Mat dst = new Mat();
3.    Imgproc.blur(src, dst, new Size(3, 3));
4.    Imgcodecs.imwrite("saved5Blurred.jpg", src);
5.
6.    double thres = 19.4;
7.    Imgproc.Canny(dst, dst, thres, thres*3);
8.    Core.bitwise_not(dst, dst);
9.    Imgcodecs.imwrite("saved6Edge.jpg", dst);
10.
11.   return dst;
12. }
```

Kode Program 4.4 Deteksi Tepi Canny

4.2.2 Implementasi Tahap FRHT

Pada subbab ini akan dijelaskan mengenai implementasi tahap Fast Randomized Hough Transform. Implementasi tahap ini terdapat tiga bagian, yaitu inisialisasi, pencarian kandidat dan pengecekan kandidat. Input dari tahap ini adalah citra tepi dari tahap preprocesssing sebelumnya.

4.2.2.1 Inisialisasi

```
1.  // extract points
2.  Mat dst = new Mat();
3.  src.copyTo(dst);
4.
5.  ArrayList<Point> edge = new ArrayList<Point>();
6.  for (int i = 0 + w; i < src.rows() - w; i++)
7.    for (int j = 0 + w; j < src.cols() - w; j++)
8.      if (src.get(i, j)[0] == 0)
9.         edge.add(new Point(j, i));
10. if(edge.size()==0){
11.   return "";
12. }
```

Kode Program 4.5 Pemindahan titik menuju array

Pada bagian awal yaitu tahap inisialisasi ini, program akan melakukan pemilihan titik secara acak pada citra tepi untuk dijadikan seed point pada setiap iterasinya. Demi menghemat beban komputasi, edge point dipindahkan terlebih dahulu kedalam array satu dimensi yang dijelaskan pada baris 6 hingga 10 **Kode Program 4.5**.

```
1.  for (int n = 0; n < nrandom; n++) {
2.
3.    //pick random seed
4.    Random rand = new Random();
5.    int pos = rand.nextInt((edge.size() - 1) - 0 + 1) + 0
    ;
6.    Point selected = edge.get(pos);
7.
8.    Map<Double, Point> counter = new HashMap<Double, Poin
    t>();
9.
10.   Map<String, Integer> cans = new HashMap<String, Integ
    er>();
11.
12.   String high = "";
13.   int nhigh = 0;
14.
15.   ...
```

Kode Program 4.6. Randomize dan Inisiasi

Setelah itu, seed point dipilih secara acak dari array yang telah dibuat pada proses sebelumnya. Hal ini diulangi sebanyak iterasi maksimum yang disimpan pada variabel nrandom. Implementasi proses ini dapat dilihat pada **Kode Program 4.6.** Proses pemilihan seed point secara acak dapat dilihat pada baris 1 hingga 6. Setelah memilih seed point secara acak, array untuk menampung jarak serta akumulator kandidat juga harus diinisiasi seperti yang ditunjukkan pada baris 8 hingga 10 program. Akumulator akan berisi nilai-nilai voting kandidat lingkaran.

4.2.2.2 Pencarian Kandidat

Bagian kedua dari tahap ini adalah pencarian kandidat. Pada awalnya, window akan dibuat sebesar w x w dengan seed point. Implementasi pada proses ini dapat dilihat pada **Kode Program 4.7**.

```
1.  for (int i = selected.y - w; i < (selected.y - w) + 2 *
     w; i++) {
2.    for (int j = selected.x - w; j < (selected.x - w) + 2
     * w; j++) {
3.      if (src.get(i, j)[0] == 0) {
4.        Double d = Math.sqrt(Math.pow((selected.y - i), 2
    ) + Math.pow((selected.x - j), 2));
5.        if (counter.get(d) == null) {
6.          counter.put(d, new Point(j, i));
7.        } else {
8.          String key = findCenter(counter.get(d), new Poi
    nt(j,i), selected);
9.          if(key=="") continue;
10.         if (cans.get(key) == null) {
11.             cans.put(key, 1);
12.         } else {
13.           cans.put(key, cans.get(key) + 1);
14.         }
15.         if (nhigh <= cans.get(key)) {
16.           nhigh = cans.get(key);
17.           high = key;
18.         }
19.         counter.remove(d);
20.       }
21.     }
22.   }
23. }
```

Kode Program 4.7 Proses Pencarian Kandidat

Seed point ditunjukkan oleh variabel selected yang berfungsi sebagai titik pusat window. Proses pembatasan dengan window ada pada baris 1 dan 2 yang sekaligus berfungsi untuk mengiterasi setiap titik pada window. Setiap titik tepi itu kemudian dihitung jaraknya dengan seed point yang direpresentasikan

sebagai d, menggunakan euclidean distance. Proses pengitungan terdapat pada baris 4 program. Setelah itu dilakukan pengecekan apakah array jarak ke-d kosong maupun tidak sekaligus proses peletakan pada array jarak. Proses ini dapat dapat dilihat pada baris 5 hingga 19.

```
1.  public static String findCenter(Point a, Point b, Point
     c){
2.      Integer A = a.x*(b.y - c.y)
3.              - a.y*(b.x-c.x)
4.              + b.x*c.y
5.              - c.x*b.y;
6.
7.      Integer B = (a.x*a.x + a.y*a.y) * (c.y-b.y)
8.              + (b.x*b.x + b.y*b.y) * (a.y-c.y)
9.              + (c.x*c.x + c.y*c.y) * (b.y-a.y);
10.
11.     Integer C = (a.x*a.x+a.y*a.y) * (b.x-c.x)
12.             +(b.x*b.x+b.y*b.y) * (c.x-a.x)
13.             +(c.x*c.x+c.y*c.y) * (a.x-b.x);
14.
15.     Integer D = (a.x*a.x+a.y*a.y)*(c.x*b.y-b.x*c.y)
16.             +(b.x*b.x+b.y*b.y)*(a.x*c.y-c.x*a.y)
17.             +(c.x*c.x+c.y*c.y)*(b.x*a.y-a.x*b.y);
18.
19.     if(A == 0){
20.         return "";
21.     }
22.     Integer x =  Math.round(-1*B/(2*A));
23.     Integer y =  Math.round(-1*C/(2*A));
24.     Double r = Math.sqrt((B*B + C*C - 4*A*D)/(4*A*A));
25.     if(r < 60){
26.         return "";
27.     }
28.     String key = x.toString() + ";" + y.toString() + ";
    "
29.      + r.toString();
30.     return key;
31. }
```

Kode Program 4.8. Pencarian titik pusat dan jari-jari lingkaran

Setelah itu dilakukan proses penghitungan titik pusat lingkaran berdasarkan 3 titik yang telah didapatkan pada garis lingkaran. Rumus matematika yang digunakan digunakan rumus matematika menggunakan determinan [12], kemudian diubah menjadi baris program seperti terlihat pada **Kode Program 4.8.**

4.2.2.3 Pengecekan Kandidat

```
1.  if(high=="") continue;
2.  String[] det = high.split(";");
3.  //System.out.println(high);
4.
5.  int count = 0;
6.  for (int i = selected.y - w; i < (selected.y - w) + 2 *
     w; i++) {
7.
8.    for (int j = selected.x - w; j < (selected.x - w) + 2
    * w; j++) {
9.
10.     if (src.get(i, j)[0] == 0) {
11.
12.       Double circ =  Math.sqrt(
13.       Math.pow((i - Integer.valueOf(det[1])), 2) + Math
    .pow((j - Integer.valueOf(det[0])), 2));
14.       Double diff = circ - Double.valueOf(det[2]);
15.       if ((diff <= 1) && (diff >= -1)) {
16.         count++;
17.       }
18.     }
19.   }
20. }
21.
22. Double ratio = (double) count / (2 * Math.PI * Double.v
    alueOf(det[2]));
23.
24. if(ratio>mratio){
25.   System.out.println(count + " " + ratio);
26.   return high;
27. }
```

Kode Program 4.9. Pengecekan Kandidat

Bagian terakhir adalah pengecekan kandidat lingkaran ini. Implementasi dapat dilihat pada **Kode Program 4.9**. Proses menghitung titik yang bersinggungan dengan lingkaran dapat dilihat pada baris 6 – 20. Sedangkan penghitungan rasio lingkaran serta pengecekan apakah telah melebihi rasio minimum terdapat pada baris 22 – 26.

4.2.3 Implementasi Main Program

Pada bagian ini, program – program non inti akan dijelaskan. Kode Program pada bagian ini merupakan program untuk menyatukan setiap bagian dan fungsionalitas pada aplikasi deteksi kemunculan bulan sabit.

4.2.3.1 FRHT Main

FRHT Main digunakan untuk menerima citra input dari fungsi lain, kemudian memanggil setiap proses deteksi sesuai dengan urutannya. Implementasi program ini terdapat pada **Kode Program 4.10**. Bagian ini juga berfungsi untuk menandai area mana yang berhasil dideteksi sebagai bulan sabit, hal ini dilakukan pada baris 10 hingga 14.

```
1.  Mat matSrc  = Imgcodecs.imread("data/Input/"+String.val
    ueOf(i)+".jpg", Imgcodecs.CV_LOAD_IMAGE_COLOR);
2.  Mat pred = preprocess(dst);
3.  Mat edged = edgeDetector(pred);
4.
5.  String res = doFRHT(edged, nrandom,  window, mratio);
6.  Mat finale = new Mat();
7.  if(res !=""){
8.    String[] det = res.split(";");
9.    Scalar warna = new Scalar(255,255,255);
10.   Scalar warna1 = new Scalar(0,0,0);
11.   matSrc.copyTo(dst);
12.   Imgproc.circle(dst, new org.opencv.core.Point(Double.
    valueOf(det[0]),Double.valueOf(det[1])), Double.valueOf
    (det[2]).intValue(), warna);
13.
14.   return dst;
15. }
```

Kode Program 4.10 FRHT Main

4.2.3.2 MainActivity2

Activity ini berfungsi menjadi menu utama dari program. Pada bagian ini terdapat tombol untuk mengambil data baik dari file maupun kamera. Pemanggilan proses FRHT Main dilakukan pada Activity ini. Implementasi bagian ini terdapat pada **Kode Program 4.11**. Sedangkan layout XML terdapat pada **Kode Program 4.12**.

```
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main2);
  chooseImage=(ImageView) findViewById(R.id.chooseImage
);
  resultImage=(ImageView) findViewById(R.id.resultImage
);
  Button load = (Button)findViewById(R.id.btnLoad);
  Button proses = (Button)findViewById(R.id.btnProses);

  load.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
      openGallery();
    }
  });
  proses.setOnClickListener(new View.OnClickListener()
{

    @Override
    public void onClick(View view) {

      Mat src = new Mat();
      Utils.bitmapToMat(getBitmap, src);
      Mat dst = new Mat();
      try {
        dst = FRHT.main(src);
      } catch (IOException e) {
        e.printStackTrace();
      }
      Utils.matToBitmap(dst,getBitmap);
      resultImage.setImageBitmap(getBitmap);
```

```
29.     }
30.
31.   });
32. }
33. private void openGallery(){
34.   Intent photoPickerIntent = new Intent(Intent.ACTION_P
    ICK);
35.   photoPickerIntent.setType("image/*");
36.   startActivityForResult(photoPickerIntent, SELECT_PHOT
    O);
37. }
38.
39. @Override
40. protected void onActivityResult(int requestCode, int re
    sultCode, Intent data) {
41.   switch(requestCode) {
42.     case SELECT_PHOTO:
43.     if(resultCode == RESULT_OK){
44.       Uri selectedImage = data.getData();
45.       if(selectedImage !=null){
46.         chooseImage.setImageURI(selectedImage);
47.         try {
48.           InputStream image_stream;
49.           try {
50.             image_stream = getApplicationContext().getC
    ontentResolver().openInputStream(selectedImage);
51.             getBitmap = BitmapFactory.decodeStream(imag
    e_stream);
52.           } catch (FileNotFoundException e) {
53.             e.printStackTrace();
54.           }
55.         }
56.         catch (Exception e) {
57.           e.printStackTrace();
58.         }
59.       }
60.     }
61.   }
62. }
```

Kode Program 4.11 MainActivity2.java

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.      xmlns:tools="http://schemas.android.com/tools"
4.      android:id="@+id/activity_main2"
5.      android:layout_width="match_parent"
6.      android:layout_height="match_parent"
7.      android:orientation="vertical"
8.      android:paddingBottom="@dimen/activity_vertical_margin"
9.      android:paddingLeft="@dimen/activity_horizontal_margin"
10.     android:paddingRight="@dimen/activity_horizontal_margin"
11.     android:paddingTop="@dimen/activity_vertical_margin"
12.     tools:context="com.hqq.lunaria.Main2Activity">
13.     <LinearLayout
14.         android:id="@+id/button"
15.         android:orientation="horizontal"
16.         android:layout_width="match_parent"
17.         android:layout_height="wrap_content">
18.         <Button
19.             android:id="@+id/btnLoad"
20.             android:layout_width="wrap_content"
21.             android:layout_height="wrap_content"
22.             android:text="LOAD"/>
23.         <Button
24.             android:id="@+id/btnProses"
25.             android:layout_width="wrap_content"
26.             android:layout_height="wrap_content"
27.             android:text="PROCESS"/>
28.     </LinearLayout>
29.     <LinearLayout
30.         android:layout_below="@id/button"
31.         android:orientation="horizontal"
32.         android:layout_width="match_parent"
33.         android:layout_height="wrap_content">
34.         <ImageView
35.             android:layout_weight="1"
36.             android:layout_width="0px"
37.             android:layout_height="match_parent"
38.             android:id="@+id/chooseImage"/>
```

```
        <ImageView
            android:layout_weight="1"
            android:layout_width="0px"
            android:layout_height="match_parent"

            android:id="@+id/resultImage"/>
    </LinearLayout>

</RelativeLayout>
```

Kode Program 4.12 MainActivity2.xml

4.2.3.3 MainActivity

Activity ini berfungsi untuk mengambil citra melalui kamera dan mengembalikannya pada MainActivity2.java untuk dilakukan proses FRHT. Implementasi bagian ini dapat dilihat pada **Kode Program 4.13**. Berikut juga dengan layout xml activity ini yang dijelaskan pada **Kode Program 4.14**.

```
private Camera mCamera = null;
private CameraView mCameraView = null;
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  try{
    mCamera = Camera.open();
  }catch (Exception e){
  }
  if(mCamera!=null){
    mCameraView = new CameraView(this, mCamera
    );
    FrameLayout camera_view = (FrameLayout)findViewById
(R.id.camera_view);
    camera_view.addView(mCameraView);
  }
  ImageButton imgClose = (ImageButton)findViewById(R.id
.imgClose);
  imgClose.setOnClickListener(new View.OnClickListener(
) {
    @Override
```

```
20.     public void onClick(View view) {
21.       System.exit(0);
22.     }
23.   });
24. ImageButton btn = (ImageButton) findViewById(R.id.imgTa
    ke);
25. final Camera.PictureCallback jpegCallback = new Camera.
    PictureCallback() {
26.   public void onPictureTaken(byte[] data, Camera camera
    ) {
27.     mCamera.startPreview();
28.     FileOutputStream outStream = null;
29.     try {
30.       outStream = new FileOutputStream(String.format("/
    sdcard/%d.jpg", System.currentTimeMillis()));
31.       outStream.write(data);
32.       Mat src  = Imgcodecs.imdecode(new MatOfByte(data)
    , Imgcodecs.CV_LOAD_IMAGE_UNCHANGED);
33.
34.       Mat dst = new Mat();
35.       try {
36.         dst = FRHT.main(src);
37.       } catch (IOException e) {
38.         e.printStackTrace();
39.       }
40.       outStream.close();
41.     } catch (FileNotFoundException e) {
42.       e.printStackTrace();
43.     } catch (IOException e) {
44.       e.printStackTrace();
45.     } finally {
46.    }
47. };
48. ImageButton btn = (ImageButton) findViewById(R.id.imgTa
    ke);
49. btn.setOnClickListener(new View.OnClickListener() {
50.   public void onClick(View v) {
51.     mCamera.takePicture(null, null, jpegCallback);
52.   }
53. });
```

Kode Program 4.13 MainActivity.java

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.
3.  <FrameLayout
4.      xmlns:android="http://schemas.android.com/apk/res/a
    ndroid"
5.      xmlns:tools="http://schemas.android.com/tools"
6.      android:layout_width="match_parent"
7.      android:layout_height="match_parent"
8.      tools:context=".MainActivity">
9.
10.     <FrameLayout
11.
12.         android:id="@+id/camera_view"
13.         android:layout_width="match_parent"
14.         android:layout_height="match_parent">
15.
16.     </FrameLayout>
17.
18.     <ImageButton
19.
20.         android:layout_width="wrap_content"
21.         android:layout_height="wrap_content"
22.         android:id="@+id/imgClose"
23.         android:layout_gravity="right|top"
24.         android:background="@android:drawable/ic_menu_c
    lose_clear_cancel"
25.         android:padding="20dp"/>
26.
27.
28.     <ImageButton
29.         android:layout_width="wrap_content"
30.         android:layout_height="wrap_content"
31.         android:id="@+id/imgTake"
32.         android:layout_gravity="center|bottom"
33.         android:background="@android:drawable/ic_menu_c
    amera"
34.         android:padding="20dp"/>
35.
36. </FrameLayout>
```

Kode Program 4.14 MainActivty.xml

4.2.3.4 Camera View

Class ini berguna untuk menjalankan fungsi kamera pada aplikasi. Implementasi dari program ini dapat dilihat pada **Kode Program 4.15**.

```
1.  public class CameraView extends SurfaceView implements
    SurfaceHolder.Callback {
2.
3.      private SurfaceHolder mHolder;
4.      private Camera mCamera;
5.
6.
7.      public CameraView(Context context, Camera camera) {

8.          super(context);
9.          mCamera= camera;
10.         mCamera.setDisplayOrientation(90);
11.         mHolder = getHolder();
12.         mHolder.addCallback(this);
13.         mHolder.setType(SurfaceHolder.SURFACE_TYPE_NORM
    AL);
14.     }
15.
16.     @Override
17.     public void surfaceCreated(SurfaceHolder surfaceHol
    der) {
18.         try{
19.             mCamera.setPreviewDisplay(surfaceHolder);
20.             mCamera.startPreview();
21.         }
22.         catch (IOException e){
23.             Log.d("ERROR", "Camera error on surfaceCrea
    ted "+e.getMessage());
24.         }
25.     }
26.
27.     @Override
28.     public void surfaceChanged(SurfaceHolder surfaceHol
    der, int i, int i1, int i2){
29.         if(mHolder.getSurface()==null){
30.             return;
31.         }
32.         try{
```

```
33.             mCamera.stopPreview();
34.         }
35.         catch(Exception e){
36.         }
37.         try{
38.             mCamera.setPreviewDisplay(mHolder);
39.             mCamera.startPreview();
40.         }catch (IOException e){
41.             Log.d("ERROR", "Camera error on surfaceChan
    ged "+e.getMessage());
42.         }
43.     }
44.
45.     @Override
46.     public void surfaceDestroyed(SurfaceHolder surfaceH
    older) {
47.         mCamera.stopPreview();
48.         mCamera.release();
49.     }
50. }
```

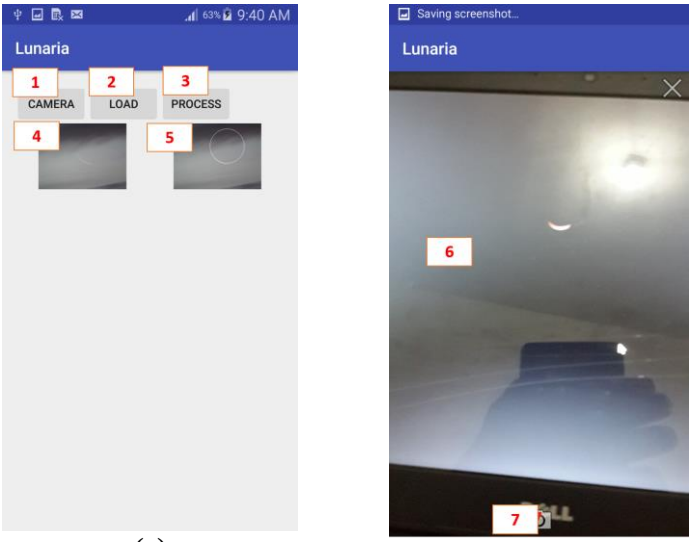
Kode Program 4.15 Camera View

4.3 Implementasi Antar Muka

Pada Implementasi Antar Muka, aplikasi ini didesain menjadi dua tampilan utama. Pertama adalah layout halaman utama yang berguna untuk menampilkan citra sebelumnya, menampilkan hasil citra serta tombol-tombol pemrosesan yang dibutuhkan. Layout ini ditunjukkan pada **Gambar 4.1** (a). Desain tampilan pada layout ini, dibentuk oleh file xml pada **Kode Program 4.12** dan fungsionalitasnya dijalankan oleh **Kode Program 4.11**.

Layout kedua berisi preview dari kamera yang menampilkan citra yang sedang dilihat oleh kamera. Terdapat tombol untuk menangkap gambar yang terletak pada sisi bawah. Implementasi dari menu ini terdapat pada **Gambar 4.1** (b). Desain tampilan pada layout ini, dibentuk oleh file xml pada **Kode Program 4.14** dan fungsionalitasnya dijalankan oleh **Kode Program 4.13**.

Setiap tombol serta bagian-bagian dari layout memiliki fungsinya masing-masing dan penjelasan dari setiap bagian tersebut dijelaskan pada **Tabel 4.1**



(a) (b)

Gambar 4.1 Implementasi Antar Muka. (a) Halaman Utama. (b) Halaman Kamera

Tabel 4.1 Daftar kegunaan fitur

No	Fungsi
1	Tombol untuk menampilkan halaman pengambilan kamera
2	Tombol untuk membuka gallery untuk memuat gambar dari file
3	Tombol untuk mengeksekusi FRHT
4	Tampilan Citra Awal
5	Tampilan Hasil Deteksi
6	Tampilan yang didapatkan dari Kamera
7	Tombol mengambil Gambar

BAB V
UJI COBA DAN EVALUASI

Dalam bab ini dibahas mengenai hasil uji coba sistem yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan.

5.1 Lingkungan Uji Coba

Lingkungan uji coba pada tugas akhir ini adalah sebuah *personal computer* (PC) dan Smartphone Android. Spesifikasi PC dari sisi perangkat keras adalah memiliki prosesor Intel Core i3 5200 dengan kecepatan 3,4 GHz dan memori untuk proses sebesar 8,00 GB. PC yang digunakan memiliki sistem operasi Windows 10. Spesifikasi smartphone yang digunakan adaah memilki prosesor Exynos 5420 Octa-core (4x1.9 GHz Cortex-A15 & 4x1.3 GHz Cortex-A7) dan memori untuk proses sebesar 3 GB. Sistem Operasi yang digunakan adalah Android Marshmallow 5.1. Kamera untuk pengambilan data memiliki resolusi 13 MP.

Pada sisi perangkat lunak, uji coba pada tugas akhir ini dilakukan dengan menggunakan *software* Eclipse Java. Penggunaan Java didukung dengan library *OpenCV*. Dokumentasi hasil uji coba dilakukan dengan menggunakan *Microsoft Paint.*

5.2 Data Uji Coba

Data uji coba yang digunakan sebagai masukan adalah citra bulan sabit yang diambil menggunakan kamera smartphone dan citra hasil ekstraksi video yang menjelaskan mengapa menggunakan dua dataset, dataset 1 dan dataset 2.

Dataset 1 merupakan citra yang diambil menggunakan kamera smartphone dan berukuran 1920 x 1080 pixel. Pada dataset ini, terdapat 50 citra yang digunakan dalam uji coba. Lebih rinci lagi, 45 citra merupakan ground truth positif seperti pada **Gambar 5.1** (a) dan 5 citra merupakan ground truth negatif seperti pada **Gambar 5.1** (b). Dataset ini diambil pada malam hari, sehingga mayoritas background berwarna gelap dan obyek bulan sabit terlihat lebih kontras.

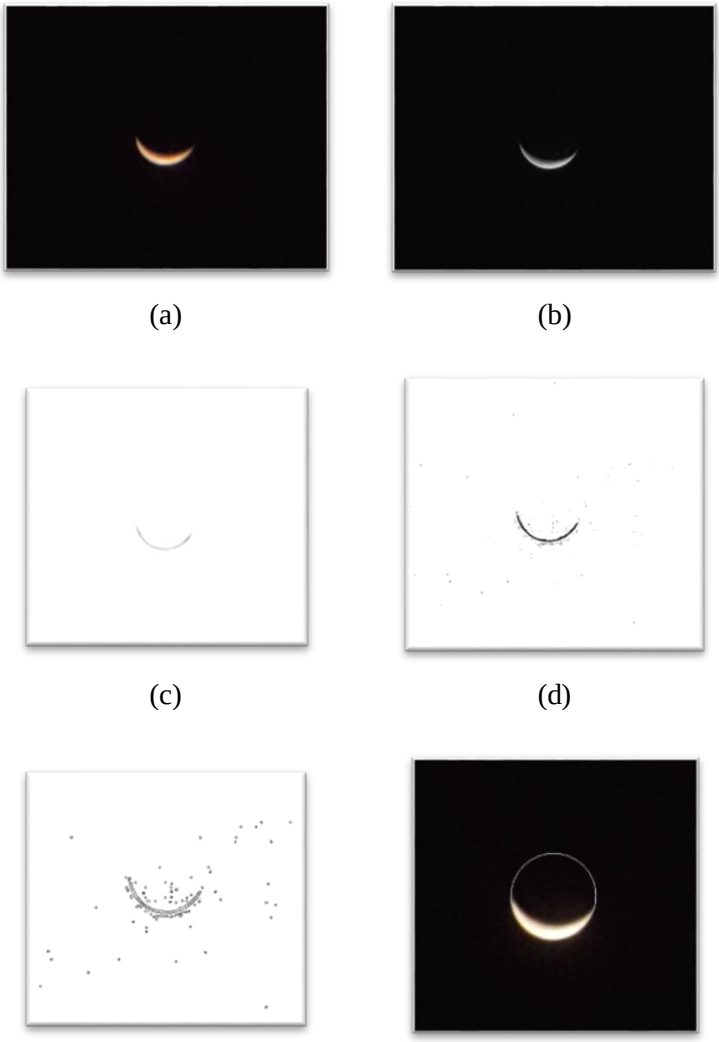
(a) (b)

Gambar 5.1 Citra Dataset 1. (a) Positif. (b) Negatif.

Dataset 2 merupakan citra yang diambil dari video kamera smartphone dan berukuran 320 x 240 pixel. Pada dataset ini, terdapat 70 citra yang digunakan dalam uji coba. Lebih rinci lagi, 55 citra merupakan ground truth positif seperti pada **Gambar 5.2Gambar 5.1**(a) dan 15 citra merupakan ground truth negatif seperti pada **Gambar 5.2**(b). Dataset ini berlatar sore hari, sehingga mayoritas background berwarna keabuan. Terdapat banyak awann yang dianggap noise karena menutupi obyek bulan sabit.

(a) (b)

Gambar 5.2 Citra Dataset 2. (a) Positif. (b) Negatif.



(a) (b)

(c) (d)

(e) (f)

Gambar 5.3 (a) Citra Masukan. (b) Citra hasil Grayscale. (c) Citra hasil Tophat. (d) Citra hasil adjustment. (e) Citra hasil deteksi tepi

5.3 Skenario Uji Coba

Uji coba dilakukan untuk mengetahui nilai-nilai parameter yang tepat untuk digunakan pada masing-masing proses. Nilai parameter yang tepat penting untuk diketahui karena penggunaan parameter yang tepat akan memberikan hasil yang terbaik pada keluaran tiap proses. Selain menentukan nilai parameter, uji coba berguna melihat performa sistem dengan metode lainnya.
Skenario pengujian terdiri dari lima macam yaitu:

1. Uji coba penentuan parameter maksimum iterasi.
2. Uji coba penentuan parameter minimum ratio.
3. Uji coba pembandingan dengan metode lain.
4. Uji coba pembandingan waktu eksekusi dengan metode lain.

5.4 Proses Uji Coba

Proses uji coba dilakukan sesuai skenario yang telah dijelaskan pada sub bab sebelumnya. Hasil pada uji coba dapat dilihat pada masing-masing sub bab, namun secara keseluruhan, program berjalan dengan proses yang sama untuk setiap tahapnya. Hasil dari setiap tahap proses yang dilakukan oleh aplikasi dapat dilihat pada **Gambar 5.3**.

5.4.1 Uji Coba penentuan parameter maksimum iterasi

Uji coba menentukan iterasi maksimum merupakan hal yang penting agar program dapat menghindari iterasi yang tidak perlu. Iterasi yang tidak perlu terjadi ketika hampir semua titik edge diproses namun tidak ada kandidat yang relevan dengan system dan memilih titik yang sama. Oleh karena itu, program harus dihentikan sedini mungkin ketika sudah tidak ada titik yang relevan supaya program dapat melakukan proses berikutnya.

Hasil Uji coba, dapat dilihat pada **Tabel 5.1** dan **Tabel 5.2**. Dari tabel tersebut dapat dilihat bahwa nilai parameter terbaik untuk dataset 1 dan 2 adalah 100 karena performa mencapai konvergensi dengan nilai akurasi 100% untuk dataset 1 dan 91% pada dataset 2.

Tabel 5.1 Dataset 1 Uji Nilai Iterasi Maksimum

Nilai Iterasi	Accuracy	Specificity	Sensitivity
60	96%	100%	96%
70	96%	100%	96%
80	98%	100%	98%
90	98%	100%	98%
100	**100%**	**100%**	**100%**

Tabel 5.2 Performa Dataset 2 Uji Nilai Iterasi Maksimum

Nilai Iterasi	Accuracy	Spesificity	Sensitivity
60	87%	93%	85%
70	89%	100%	85%
80	87%	80%	89%
90	89%	87%	89%
100	**91%**	**100%**	**89%**
110	89%	100%	85%
120	89%	93%	87%

5.4.2 Uji Coba penentuan parameter minimum ratio

Uji Coba menentukan nilai minimum rasio sangat penting agar program dapat mendeteksi obyek bulan sabit yang tepat dan menghindari kandidat lingkaran kecil yang tidak relevan. Karena program didesain untuk berhenti ketika menemui kandidat yang relevan, nilai yang paling optimum dibutuhkan sehingga program dapat berhenti tepat ketika menemukan obyek bulan sabit yang dituju.

Hasil uji coba dapat dilihat pada **Tabel 5.3** dan **Tabel 5.4** di bawah ini. Dari tabel tersebut dapat dilihat bahwa nilai rasio minimum yang paling optimal untuk dataset 1 adalah 0.08 dan 0.1 untuk dataset 2.

Tabel 5.3 Performa Dataset 1 Uji Nilai Rasio Minimum

Nilai Rasio	Accuracy	Specificity	Sensitivity
0.06	96%	80%	98%
0.07	98%	80%	100%
0.08	**100%**	**100%**	**100%**
0.09	100%	100%	100%
0.10	98%	100%	98%
0.11	98%	100%	98%
0.12	98%	100%	98%

Tabel 5.4 Performa Dataset 2 Uji Nilai Rasio Minimum

Nilai Rasio	Accuracy	Specificity	Sensitivity
0.08	86%	73%	89%
0.09	87%	93%	85%
0.10	**90%**	**100%**	**87%**
0.11	87%	100%	84%
0.12	86%	100%	82%

5.4.3 Uji Coba dengan metode pembanding

Uji coba ini diperlukan guna melihat performa aplikasi dengan penelitian sebelumnya yang dilakukan oleh Ike. Pada percobaan ini, dilakukan dengan menggunakan dataset 2 karena dataset ini digunakan oleh penelitian Ike dalam programnya yang mengimplementasi Circular Hough Transform(CHT). Nilai yang akan diukur sama seperti uji coba sebelumnya, yaitu akurasi, spesifisity dan juga sensitivity. Pada percobaan, aplikasi yang diuji menggunakan parameter optimum yang telah didapatkan yaitu 0.1 untuk rasio minimum dan iterasi maksimum 100.

Berdasarkan hasil uji coba yang dapat dilihat pada **Tabel 5.5**, performa aplikasi yang diuji menghasilkan hasil yang lebih

baik pada akurasi (90%) dan specificity (100%), namun aplikasi dalam penelitian Ike(2012) unggul dalam sensitivity (93%).

Tabel 5.5 Komparasi Performa Program

Program	Accuracy	Specificity	Sensitivity
FRHT	**90%**	**100%**	87%
CHT	89%	73%	**93%**

5.4.4 Uji Coba pembandingan waktu eksekusi

Uji coba ini diperlukan guna melihat performa aplikasi dari sisi waktu eksekusi yang dibutuhkan untuk mendeteksi kemunculan bulan sabit. Percobaan dilakukan pada dua dataset yang memiliki karakteristik yang berbeda sesuai deskripsi pada Sub-bab 5.2. Nilai yang akan diukur adalah kecepatan dalam satuan millisecond (ms) pada setiap proses eksekusi. Semakin cepat waktu eksekusi, maka semakin baik. Metode pembanding yang digunakan adalah Circular Hough Transform (CHT) milik OpenCV dimana CHT telah dilakukan optimasi dari versi aslinya. Pada hasil uji coba yang dapat dilihat pada **Tabel 5.6**, waktu eksekusi FRHT lebih unggul dibandingkan dengan CHT OpenCV pada dataset 1 namun waktu eksekusi CHT OpenCV lebih unggul pada dataset 2.

Tabel 5.6 Komparasi Waktu Eksekusi

Dataset	Time (ms)	
	Dataset 1	Dataset 2
FRHT	**23**	49
CHT OpenCV	33	**8**

5.5 Evaluasi

Pada subbab ini akan dijelaskan hasil dari serangkaian uji coba yang dilakukan dan kendala yang dihadapi selama proses pengerjaan. Evaluasi yang dilakukan meliputin nilai parameter

iterasi maksimum, nilai rasio minimum, performa menggunakan parameter optimum dan perbandingan metode lain.

5.5.1 Evaluasi nilai parameter iterasi maksimum

Pada dasarnya, dikarenakan algoritma Fast Randomized Hough Transform yang digunakan berjalan secara random dalam pemilihan titik-titiknya, nilai iterasi maksimum sangat penting digunakan guna menghasilkan performa yang terbaik dari sisi akurasi dan waktu pemrosesan. Semakin kecil nilai iterasi maksimum, maka semakin sedikit titik – titik yang diproses untuk pencarian kandidat lingkaran yang menyebabkan proses pendeteksian semakin cepat. Namun, semakin besar nilai iterasi maksimum juga akan meningkatkan performa karena lebih banyak titik yang dapat diproses. Oleh karena itu, nilai iterasi maksimum yang dicari adalah nilai paling minimum parameter dimana performa aplikasi menghasilkan hasil yang terbaik.

Pada hasil uji coba yang dapat dilihat pada **Tabel 5.1** dan **Tabel 5.2**, didapatkan hasil terbaik untuk kedua dataset yaitu nilai iterasi 100. Nilai 100 dipilih karena pada dataset 1, sistem telah mendeteksi secara benar seluruh dataset 1 dengan nilai akurasi, spesificity dan sensitivity sebesar 100%. Begitu pula dengan dataset 2 dimana ketika iterasi maksimum pada nilai 100, akurasi berada pada titik tertinggi yaitu 91%. Specificity juga berada pada titik tertinggi yaitu 100% dimana hal ini membuktikan bahwa system tidak mengalami kesalahan dalam mendeteksi citra yang tidak ada bulan sabit di dalamnya. Nilai iterasi maksimum yang teroptimal ini menjadi acuan pada uji coba lainnya.

5.5.2 Evaluasi Nilai Parameter Rasio Minimum

Nilai rasio minimum merupakan batas bawah apakah sebuah kandidat lingkaran yang muncul adalah lingkaran bulan sabit yang sedang dicari. Namun, walaupun dataset diambil pada waktu yang sama, nilai rasio minimum tetap bervariasi. Hal ini disebabkan oleh noise awan yang ada pada citra yang menutupi sebagian dari obyek bulan sabit. Pada saat awan menutupi sebagian dari obyek bulan sabit, nilai rasio juga menurun. Oleh karena itu, dibutuhkan rasio

minimum yang paling optimal dimana rasio bulan sabit terkecil karena paling banyak terganggu oleh noise.

Berdasarkan hasil uji coba nilai minimum rasio yang ditunjukkan pada **Tabel 5.3** dan **Tabel 5.4**, didapatkan nilai terbaik untuk dataset 1 yaitu 0.08 dimana menghasilkan akurasi 100%, begitu juga dengan specifity dan sensitivity. Hal ini berbeda dengan dataset 2 dimana nilai rasio minimum yang paling optimal yaitu 0.10 dimana performa aplikasi mencapai puncaknya dengan nilai akurasi 90%.

5.5.3 Evaluasi Ketepatan Deteksi

Hasil uji coba menggunakan nilai parameter optimum yang didapatkan pada uji coba sebelumnya digunakan untuk melihat berapa performa maksimal yang dapat dicapai oleh program pada kedua dataset. Capaian maksimum program cukup dilihat pada **Tabel 5.3** dan **Tabel 5.4** karena pada uji coba tersebut telah menggunakan nilai iterasi maksimum yang optimal. Hasil uji coba menunjukkan performa aplikasi pada dataset 1 mencapai akurasi, specificity dan sensitivity sebesar 100%. Namun pada dataset 2, performa akurasi menjadi lebih rendah dari dataset 1 yaitu 90% pada akurasi, 100% pada specificity dan 87% pada sensitivity. Pada dataset 2, kemampuan sistem dalam mendeteksi citra yang mengandung bulan sabit kurang baik. Hal ini disebabkan oleh noise awan yang banyak ditemui pada dataset 2. Sedangkan pada dataset 1 yang backgroundnya cenderung gelap dan noise awan minim, system dapat mendeteksi dengan sangat baik.

5.5.4 Evaluasi Perbandingan dengan metode lain

Pada uji coba dalam membandingkan aplikasi dengan metode lain yaitu aplikasi dalam penelitian Ike yang menggunakan Circular Hough Transform, berdasarkan data pada **Tabel 5.5** performa aplikasi ini lebih baik dalam akurasi maupun specificity. Namun, pada tingkat sensitivity, program pada penelitian Ike Mardiya Sari (2012) lebih unggul dengan nilai 93%.

Berdasarkan pengamatan, citra yang mengandung banyak noise dapat dideteksi oleh program dalam penelitan Ike. Hal ini disebabkan oleh proses morfologi yang dilakukan oleh Ike pada

tahap segmentasi. Proses morfologi Ike menghasilkan deteksi lebih bagus karena dapat mengatasi noise(awan) yang menyebabkan obyek citra lebih mudah dideteksi karena noise awan dapat diminimalir. Sedangkan dalam aplikasi ini, tidak dilakukan proses segmentasi karena metode Fast Randomized Hough Transform langsung menggunakan citra tepi dari deteksi tepi canny, dan untuk menghemat proses komputasi. Hal ini membuktikan bahwa program pada penelitian Sari(2012) lebih unggul dalam sisi mendeteksi citra yang mengandung bulan sabit sehingga nilai sensitivitynya unggul.

Pengamatan berikutnya yaitu dari sisi komputasi yang dapat dilihat pada **Tabel 5.6**, dimana waktu eksekusi Fast Randomized Hough Transform (FRHT) pada dataset 1 lebih baik dari Circular Hough Transform (CHT) OpenCV dengan selisih 10 ms lebih cepat. Namun, pada dataset 2, CHT OpenCV lebih cepat dibandingkan dengan metode FRHT. Berdasarkan karakteristik dari setiap dataset, dataset 1 memiliki ukuran 1920 x 1080 pixel, sedangan dataset 2 memilki ukuran 320 x 240 pixel. Dari data tersebut dapat disimpulkan bahwa, FRHT lebih efisien dalam menangani citra dengan ukuran pixel besar. Namun CHT OpenCV lebih baik jika citra yang digunakan memiliki ukuran kecil. Namun, CHT OpenCV yang digunakan telah dilakukan modifikasi dari versi asli Circular Hough Transform dengan tujuan efisiensi sehingga waktu eksekusi CHT OpenCV jauh lebih cepat dibandingkan dengan CHT yang asli.

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah. Selain itu juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Kesimpulan yang diperoleh dari uji coba dan evaluasi adalah sebagai berikut:

1. Dalam mendapatkan titik tepi dari bulan sabit yang tipis, dilakukan preprocessing dengan top hat filtering, image adjustment dan deteksi tepi Canny.
2. Fast Randomized Hough Transform digunakan untuk mendapatkan kandidat lingkaran tertinggi sebagai representasi dari bulan sabit. Penggunaan FRHT menggunakan nilai parameter iterasi maksimum sebesar 100 untuk kedua dataset serta nilai minimum rasio lingkaran sebesar 0.08 untuk dataset 1 dan 0.1 untuk dataset 2.
3. Sistem diimplementasikan pada perangkat Android dengan bahasa java dan dapat memberikan akurasi deteksi bulan sabit sebesar 100% untuk dataset 1 dan 90% untuk dataset 2.
4. Hasil pembandingan ketepatan deteksi dengan CHT pada penelitian Ike (2012), FRHT dapat mendeteksi lebih baik Ike pada dataset 1. Namun pada dataset 2, CHT memiliki performa yang lebih baik karena terdapat proses morfologi yang dapat mengatasi noise(awan) sehingga performa menjadi meningkat.
5. Hasil pembandingan waktu eksekusi dengan metode CHT OpenCV menunjukkan bahwa FRHT dapat mendeteksi lebih cepat untuk citra yang memiliki ukuran pixel yang besar.

6.2 Saran

Saran yang dapat diberikan untuk penelitian selanjutnya adalah dibutuhkan penelitian lebih lanjut mengenai metode morfologi yang dapat berjalan ringan pada platform Android agar dapat mengatasi noise awan yang menutupi obyek sehingga performa aplikasi dapat meningkat.

DAFTAR PUSTAKA

[1] I. M. Sari, A. Z. Arifin, dan A. Yuniarti, "Implementasi Circular Hough Transform untuk Deteksi Kemunculan Bulan Sabit," *J. Tek. POMITS*, vol. Vol. 1, hal. 1–5, 2012.

[2] M. Rizon *et al.*, "Object detection using circular Hough transform," 2005.

[3] S.-H. Chiu, J.-J. Liaw, dan K.-H. Lin, "A fast randomized Hough transform for circle/circular arc recognition," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 24, no. 3, hal. 457–474, 2010.

[4] "Pengertian, Rotasi, Revolusi, dan Fase Bulan | Ilmu Pengetahuan." [Daring]. Tersedia pada: http://www.softilmu.com/2015/10/Pengertian-Fungsi-Proses-Teori-Terbentuknya-Rotasi-Revolusi-Fase-Bulan-Adalah.html. [Diakses: 17-Jul-2017].

[5] *Sekilas Pengetahuan Kriteria Visibilitas Hilal*. Judhistira Aria Utama.

[6] R. C. Gonzalez, *Digital Image Processing*. Pearson Education, 2009.

[7] "Android (operating system) - Wikipedia." [Daring]. Tersedia pada: https://en.wikipedia.org/wiki/Android_(operating_system). [Diakses: 10-Jul-2017].

[8] "OpenCV," *Wikipedia*. 25-Mei-2017.

[9] "Hough transform," *Wikipedia*. 15-Jun-2017.

[10] T.-C. Chen dan K.-L. Chung, "An Efficient Randomized Algorithm for Detecting Circles," *Comput. Vis. Image Underst.*, vol. 83, no. 2, hal. 172–191, Agu 2001.

[11] "Sensitivity and specificity," *Wikipedia*. 05-Jul-2017.

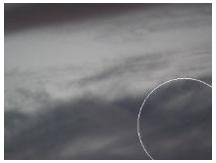
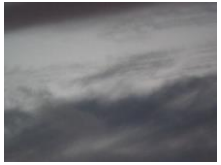
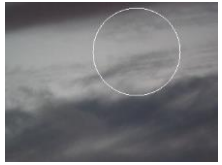
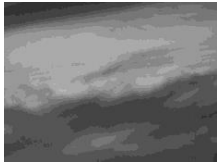
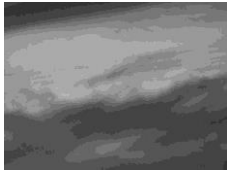
[12] "Circle defined by 3 points." [Daring]. Tersedia pada: http://www.ambrsoft.com/TrigoCalc/Circle3D.htm. [Diakses: 14-Jul-2017].

[*Halaman ini sengaja dikosongkan*]

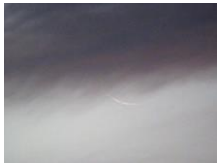
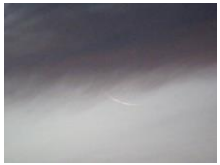
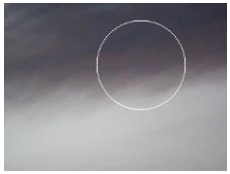
LAMPIRAN

A. Sampel Hasil Deteksi : Dataset 1 (kiri) dan Dataset 2 (kanan)

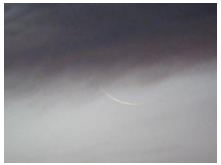
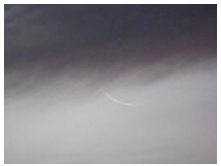
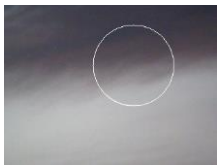
Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TP)**

Citra Input	Citra Output	Hasil Deteksi
		**Salah (FP)**
		**Salah (FP)**
		**Benar (TN)**

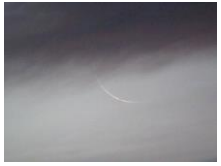
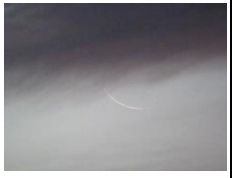
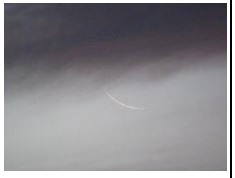
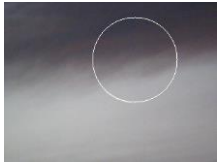
Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TP)**

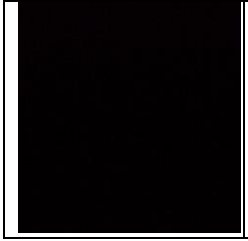
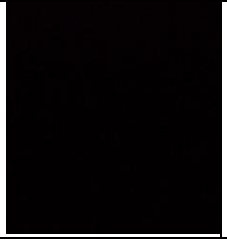
Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TP)**

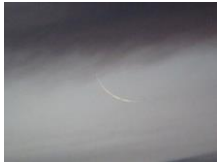
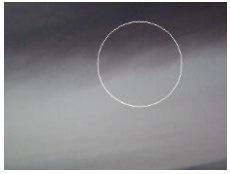
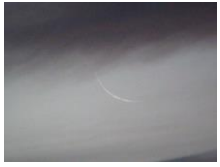
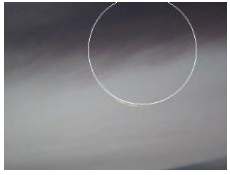
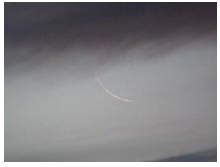
Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TP)**

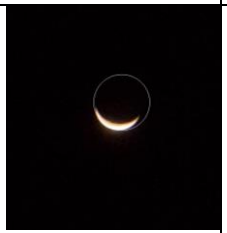
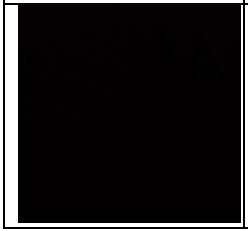
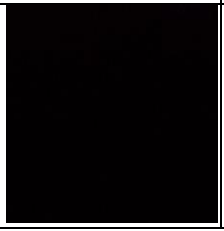
Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TP)**

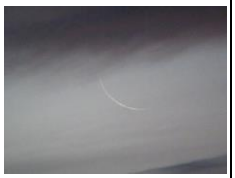
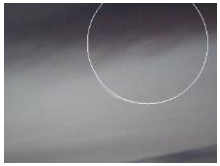
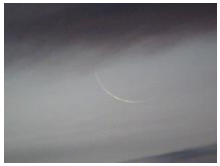
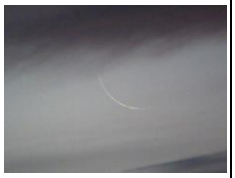
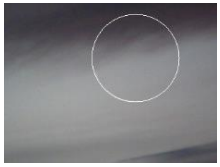
Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TP)**

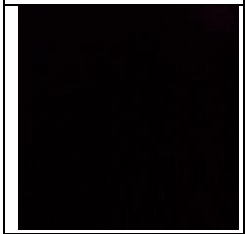
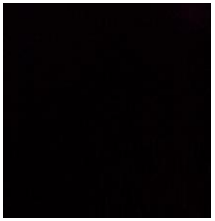
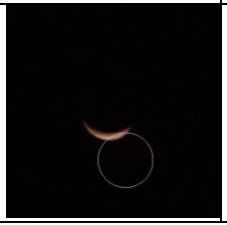
Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TP)**

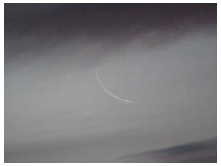
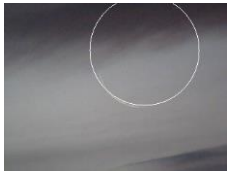
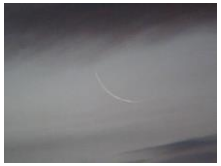
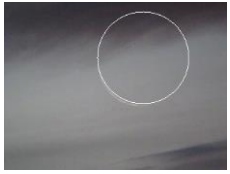
Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TN)**

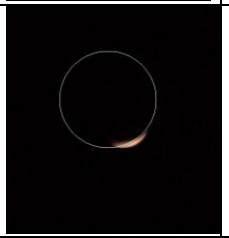
Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TP)**

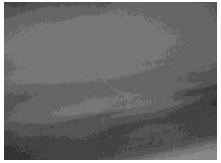
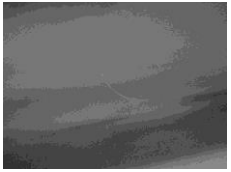
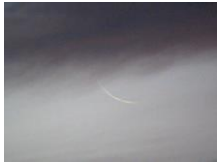
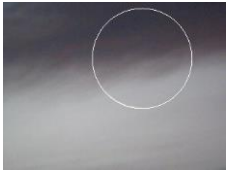
Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TN)**

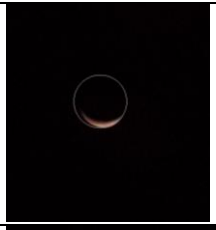
Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TP)**

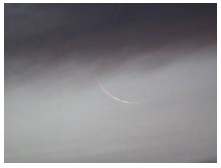
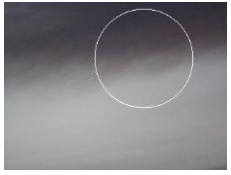
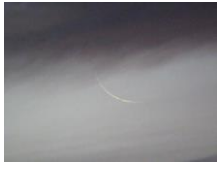
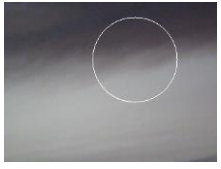
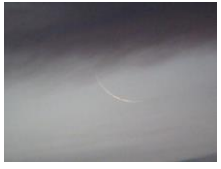
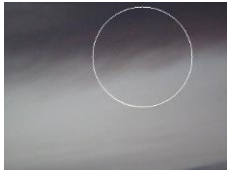
Citra Input	Citra Output	Hasil Deteksi
		**Benar (TN)**
		**Benar (TP)**
		**Benar (TP)**

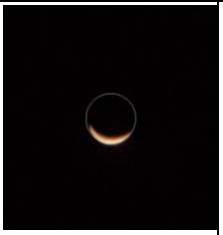
Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TP)**

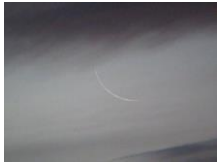
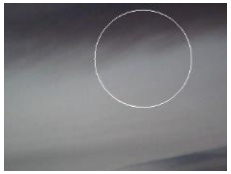
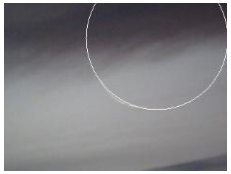
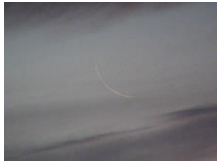
Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TP)**

Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Salah (FN)**
		**Benar (TP)**

Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TP)**

Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TP)**

Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TP)**

Citra Input	Citra Output	Hasil Deteksi
		**Benar (TP)**
		**Benar (TP)**
		**Benar (TP)**

BIODATA PENULIS

Achmad Rizky Haqiqi lahir di Lumajang, 24 Februari 1995. Dalam menjalani pendidikan, penulis menempuh pendidikan di SD Negeri Banyuajuh 3 dan lulus tahun 2007, SMP Negeri 7 Balikpapan dan lulus tahun 2011, SMA Negeri 2 Bandar Lampung dan lulus tahun 2013. Penulis melanjutkan pendidikan S1 Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember (ITS) pada tahun 2013. Selama menjadi mahasiswa, penulis pernah menjadi staff Departemen Kewirausahaan Mahasiswa Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) ITS periode 2014-2015, staff NST SCHEMATICS HMTC ITS tahun 2014, staff ahli NST SCHEMATICS HMTC ITS tahun 2015. Kemudian penulis pernah diamanahi untuk menjabat sebagai Kepala Departemen Keilmuan Keluarga Muslim Informatika periode 2015-2016.