TUGAS AKHIR – KI091391

Implementasi Sistem Pendeteksi Gerakan dengan Motion Detection pada Kamera Video Menggunakan AForge .NET

MUHAMMAD REDHA
NRP 5110 100 703

Dosen Pembimbing I
Dwi Sunaryono, S.Kom., M.Kom.

Dosen Pembimbing II
Ridho Rahman H., S.Kom., M.Sc.

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

FINAL PROJECT – KI091391

Implementation of Motion Detector System using Aforge.Net Motion Detection in Video Camera

MUHAMMAD REDHA
NRP 5110 100 703

Dosen Pembimbing I
Dwi Sunaryono, S.Kom., M.Kom.

Dosen Pembimbing II
Ridho Rahman H., S.Kom., M.Sc.

INFORMATICS ENGINEERING DEPARTEMENT
Information Technology Faculty
Sepuluh Nopember Institute of Technology
Surabaya 2014

Implementasi Sistem Pendeteksi Gerakan dengan Motion Detection pada Kamera Video Menggunakan AForge .NET

Nama Mahasiswa	: Muhammad Redha
NRP	: 5110 100 703
Jurusan	: Teknik Informatika FTIf-ITS
Dosen Pembimbing I	: Dwi Sunaryono, S.Kom., M.Kom.
Dosen Pembimbing II	: Ridho Rahman H., S.Kom., M.Sc.

ABSTRAK

Perancangan sistem pendeteksi gerakan adalah suatu program yang dirancang untuk memberikan informasi setiap terjadi gerakan yang terdeteksi pada hasil analisa dari sebuah video. Pemrograman yang dilakukan dengan menggunakan bahasa pemrograman C# dan library AForgr .NET Framework pada Microsoft Visual Studio 2010. AForge .Net Framework ini digunakan karena dirancang khusus untuk memberikan filter pemrosesan gambar pada bahasa pemrograman C#. Perancangan dimulai dengan membuat program pendeteksi gerakan. Setelah program dibuat, percobaan dilakukan terhadap aplikasi agar mendapatkan hasil yang optimal.

Pengujian dilakukan dengan membandingkan hasil pengamatan manual dan hasil deteksi sistem. Hasil pengujian menunjukkan sistem mendeteksi dengan tingkat akurasi yang sangat baik, yaitu 98.1%. Sistem juga dapat mendeteksi perubahan gerakan terhadap batasan region area yang diinginkan untuk memusatkan deteksi pada area tertentu.

Kata kunci: *Motion Detektion, AForge .NET Framework, Pemrograman C#*

Implementation of Motion Detector System using Aforge.Net Motion Detection in Video Camera

Student Name	: Muhammad Redha
NRP	: 5110 100 703
Major	: Informatics Engineering FTIf-ITS
Advisor I	: Dwi Sunaryono, S.Kom., M.Kom.
Advisor II	: Ridho Rahman H., S.Kom., M.Sc.

ABSTRACT

Motion detector system is an application designed to provide notification whenever any motion is detected from a video capture analysis. The programming in this design is performed using C# language and AForge.Net Framework library of Microsoft Visual Studio 2010. AForge.Net is preferred because of its ability to filter the image processing using C#. The design is started by building the motion detector program. After the program is ready to run, several experiments are conducted to provide optimum result.

The experiments are performed by comparing the result of manual observation and system' output. The result suggests that the system's accuracy is very good, with value of 98.1%. The system is also capable of detecting any motion on some area. This area can be specified by user to focus the monitoring of a certain object.

Keyword: *Motion Detection, AForge .NET Framework, C# Programming*

LEMBAR PENGESAHAN

Implementasi Sistem Pendeteksi Gerakan dengan Motion Detection pada Kamera Video Menggunakan AForge .NET

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Rekayasa Perangkat Lunak
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
MUHAMMAD REDHA
NRP : 5110 100 703

Disetujui oleh Dosen Pembimbing Tugas Akhir

Dwi Sunaryono, S.Kom., M.Kom. ..... *(pembimbing 1)*
NIP: *197205281997021001*

Ridho Rahman H., S.Kom., M.Sc. ..... (pembimbing 2)
NIP: 051100123

SURABAYA
JULI 2014

KATA PENGANTAR

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya sehingga penulis dapat menyelesaikan tugas akhir ini yang berjudul:

Implementasi Sistem Pendeteksi Gerakan dengan Motion Detection pada Kamera Video Menggunakan AForge .NET

Melalui lembar ini, penulis hanya ingin menyampaikan ucapan terima kasih dan penghormatan yang sebesar-besarnya kepada:

1. Allah SWT atas segala nikmat dan rahmat yang telah diberikan selama ini.
2. Ayah, Ibu dan keluarga penulis yang tiada henti-hentinya mencurahkan kasih sayang, perhatian, dan doa kepada penulis selama ini.
3. Pak Dwi Sunaryono dan Pak Ridho Rahman selaku dosen pembimbing yang telah memberikan bimbingan, motivasi, dan meluangkan waktu untuk membantu pengerjaan Tugas Akhir ini.
4. Bapak dan Ibu dosen Teknik Informatika ITS yang telah membina dan memberikan ilmu kepada penulis selama menempuh studi di Teknik Informatika ITS.
5. Sahabat dekat yang selalu memberikan dukungan dan pencerahan.
6. Rekan sekontrakan yang memberikan ide, saran dan dukungan baik teknis maupun tidak yaitu Fachri, Royan, Adies, Fazar, Ade, Mansur, Ibeck, Masbar, Tsabit, Djayusman, Nawa dan Prima.

7. Teman-teman angkatan 2010 yang membantu dan mendukung dalam pengerjaan tugas akhir ini.
8. Serta pihak-pihak lain yang namanya tidak dapat penulis sebutkan satu -persatu.

Bagaimanapun juga penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Juli 2014

Muhammad Redha

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

Bagian ini akan dijelaskan hal–hal yang menjadi latar belakang, permasalahan yang dihadapi, batasan masalah, tujuan dan manfaat, metodologi, dan sistematika penulisan yang digunakan dalam pembuatan Tugas Akhir.

1.1 Latar Belakang

Pada saat ini keamanan adalah salah satu hal yang sangat penting. Dengan kemajuan teknologi dalam bidang keamanan sekarang ini, pengamatan suatu objek menjadi lebih praktis. Untuk mengamati suatu objek tidak perlu dilakukan pengamatan secara langsung dan terus menerus. Cukup meletakkan suatu kamera yang mengarah pada objek yang diinginkan dan objek tersebut dapat diamati pada layar monitor. Permasalahan yang timbul selama ini kamera hanya dapat menangkap suatu objek tetapi tidak dapat memberikan informasi tentang pergerakan objek tersebut. Hal ini sangat berguna jika diaplikasikan ke dalam sistem pendeteksi gerakan dengan *motion detection*. Ketika ada objek yang bergerak, maka secara otomatis aplikasi akan memberikan informasi objek asing yang terdeteksi tersebut.

Sistem *motion detection* merupakan suatu sistem yang digunakan untuk menganalisis gerakan pada video pengawas untuk memudahkan pendeteksian gerakan dari suatu kejadian penting. Sistem *motion detection* yang akan dibangun harus dapat mengidentifikasi adanya perubahan dari kondisi awal yang sudah didefinisikan. Hal ini sangat penting, karena apabila terjadi perubahan, maka aplikasi harus bisa mengkonfirmasikan atau memberi informasi perubahan tersebut melalui informasi deteksi. Dengan demikian dapat diketahui jika ada objek yang bergerak. Hasil dari informasi pendeteksian gambar atau objek bergerak

tersebut dapat digunakan sebagai media konkrit untuk melihat adanya perubahan.

Pemrograman dilakukan dengan menggunakan bahasa pemrograman C# dan library framework AForge .NET pada Microsoft Visual Studio 2010. AForge .NET adalah sebuah framework yang dirancang khusus untuk memberikan filter pemrosesan gambar pada bahasa pemrograman C#. Perancangan dimulai dengan membuat program pendeteksi gerakan. Setelah program dibuat, percobaan dilakukan terhadap aplikasi agar mendapatkan hasil yang optimal.

Hasil yang diharapkan semoga dengan adanya sistem ini nantinya dapat memberikan manfaat untuk memudahkan pendeteksian gerakan dari suatu kejadian penting pada video pengawas untuk meningkatkan sistem keamanan menjadi lebih baik.

1.2 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut.

1. Membangun sebuah sistem pendeteksi gerakan dengan *motion detection* pada kamera video.
2. Mengimplementasikan penerapan *framework* AForge .NET pada sistem pendeteksi gerakan.
3. Menganalisis pendeteksian gerakan pada video.

1.3 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut.

1. Bagaimana mendeteksi adanya gerakan pada video.
2. Bagaimana suatu sistem bisa memberikan informasi adanya gerakan pada sebuah video.
3. Bagaimana suatu sistem bisa menganalisis pengecualian perubahan gerakan kecil yang terjadi secara alami.

4. Bagaimana membangun aplikasi yang bisa memilih sebagian area yang memerlukan pendeteksian dari keseluruhan kawasan deteksi.

1.4 Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, di antaranya sebagai berikut.

1. Mengintegrasikan kamera video dengan perangkat lunak pendeteksi gerakan.
2. Wilayah deteksi hanya terbatas pada suatu area yang tidak terlalu luas (sebuah ruangan) dan kondisi ruangan yang relatif tetap.
3. Kamera hanya memantau perubahan yang terjadi pada area yang terbatas tanpa mendefinisikan perubahan tersebut. Bila ada objek baru yang bergerak, maka pendeteksi akan aktif.
4. Bahasa pemrograman yang digunakan adalah C# dengan *framework* .NET.

1.5 Metodologi

Langkah yang ditempuh dalam pengerjaan Tugas Akhir ini adalah sebagai berikut.

1. Studi Literatur

Mengumpulkan literatur yang dibutuhkan dalam proses perancangan dan implementasi sistem yang akan dibangun. Literatur yang dibutuhkan antara lain sebagai berikut.

a. Pengumpulan informasi dari beberapa sumber terkait *framework* AForge .NET.
b. Proses penerapan *motion detection* pada kamera video dan *file* video mengunakan AForge .NET.

2. Analisis Sistem

Melakukan analisa kebutuhan sistem sebagai solusi atas permasalahan yang dihadapi pengguna. Dari proses tersebut selanjutnya dirumuskan rancangan sistem yang akan dibangun dan dapat menangani permasalahan. Langkah pada tahap ini antara lain:

a. Analisa aktor yang terlibat dalam sistem.
b. Perancangan *use case diagram* sebagai analisa kebutuhan fungsional sistem.
c. Analisa kebutuhan non-fungsional.
d. Analisa arsitektur sistem.

3. Perancangan Sistem

Melakukan perancangan sistem dari hasil analisa terhadap sistem. Proses analisa digambarkan dalam bentuk diagram atau bagan untuk mempermudah gambaran rancangan sistem. Langkah perancangan pada tahap ini antara lain:

a. Rancangan sistem berbasis desktop.
b. Rancangan antarmuka apikasi desktop Windows.

4. Implementasi

Pada tahap ini dilakukan pembuatan perangkat lunak berdasarkan rancangan yang telah dibuat pada proses sebelumnya. Yaitu, implementasi rancangan aplikasi berbasis desktop.

5. Pengujian dan evaluasi

Pada tahap ini dilakukan pengujian terhadap perangkat lanak berdasarkan skenario yang telah ditentukan. Pengujian ini bertujuan untuk menguji kebutuhan fungsional yang dibutuhkan, masalah yang timbul, kekurangan program dan kemudahan pengguna dalam menggunakan aplikasi. Tahap pengujian yang akan dilakukan sebagai berikut.

a. Uji coba aplikasi untuk mendeteksi gerakan pada *file* video rekaman cctv dan yang sudah disediakan.

b. Uji coba aplikasi untuk mendeteksi pada *region area* dari video rekaman yang sudah disediakan.

6. Penyusunan Buku Tugas Akhir

Pada tahap ini ditulis buku yang bertujuan untuk mendokumentasikan seluruh konsep, rancangan, dasar teori, literatur, proses yang dilakukan dan hasil yang diperoleh selama pengerjaan tugas akhir. Buku yang ditulis bertujuan untuk memberikan gambaran dari pengerjaan tugas akhir dan berguna untuk pembaca yang tertarik untuk melakukan pengembangan sistem lebih lanjut.

1.6 Sistematika

Pendokumentasian seluruh konsep, rancangan, dasar teori, literature, proses yang dilakukan dan hasil yang diperoleh selama pengerjaan tugas akhir. Buku yang ditulis bertujuan untuk memberikan gambaran dari pengerjaan tugas akhir dan berguna untuk pembaca yang tertarik untuk melakukan pengembangan sistem lebih lanjut.

Buku tugas akhir akan terdiri dari beberapa bagian, yaitu:

Bab I Pendahuluan

Bab ini membahas latar belakang masalah, tujuan pembuatan tugas akhir, rumusan permasalahan, batasan permasalahan, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

Bab II Tinjauan Pustaka

Bab ini membahas teori pendukung dan literatur yang berkaitan dengan bahasan dan mendasari pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan

Bab ini membahas tentang desain dan rancangan dari perangkat lunak. Rancangan dan desain meliputi data, proses, arsitektur.

Bab IV Implementasi

Bab ini membahas tentang implementasi hasil analisis dan perancangan dalam bentuk *coding*. Bab ini membahas proses pembangunan perangat lunak.

Bab V Pengujian dan Evaluasi

Bab ini membahas tentang pengujian aplikasi berdasarkan skenario yang telah ditentukan. Mengevaluasi fitur aplikasi apakah telah memenuhi kebutuhan fungsional.

Bab V Kesimpulan dan Saran

Bab ini berisi kesimpulan baik dari proses pengembangan perangkat lunak dan hasil uji coba.

Daftar Pustaka

Merupakan daftar referensi yang digunakan dalam pembuatan tugas akhir.

Lampiran

Merupakan bab tambahan yang berisi kode–kode sumber yang penting pada aplikasi

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai dasar teori dan literatur yang menjadi dasar pembuatan Tugas Akhir. Diantaranya, pengertian *Motion Detection*, *Framework .NET*, AForge *Framework*, Pemrograman C# dan *Computer vision.*

2.1 Motion Detection

Motion Detection adalah proses mendeteksi perubahan posisi dari suatu objek relatif terhadap sekitarnya atau perubahan lingkungan relatif terhadap suatu objek. *Motion Detection* dapat dicapai oleh kedua metode mekanik dan elektronik [1].

Motion detection melakukan pendekatan dengan membandingkan *frame* pada saat ini dengan *frame* sebelumnya. Mula-mula kamera video akan menangkap gambar dari ruangan yang sedang dipantau. Kemudian membandingkan warna yang terdapat pada *frame* saat ini dengan *frame* sebelumnya. Apabila terdapat perbedaan warna, maka objek tersebut terdeteksi sebagai gerakan. Pendekatan yang dilakukan adalah memisahkan gambar antara area *background* dengan area *foreground* pada area gerak untuk melacak adanya pergerakan.

Pengurangan gambar yang popular adalah satu teknik dalam pengolahan gambar dan visi computer. Pada dasarnya dalam melakukan pengurangan gambar digunakan persamaan 2.1.

$$\Delta I(i,j) = ICurr(i,j) - IPrev(i,j) \qquad (2.1)$$

Dimana $\Delta I(i,j)$ adalah gambar intensitas yang berbeda dari dua *frame* berturut-turut. $ICurr(i,j)$ adalah gambar untuk saat sekarang dan $IPrev(i,j)$ intensitas masing-masing *frame* sebelumnya.

Ada beberapa langkah untuk menerapkan metode pengurangan gambar. Langkah pertama adalah murni membandingkan gambar *pixel* dari dua *frame*. Jika ada perbedaan nilai antara dua *frame* secara konsekuen, maka bisa disimpulkan ada gerakan. Perbedaan antara kedua *frame* mewakili sebagai bagian yang bergerak. Hasil yang ditampilkan terdiri dari kombinasi daerah yang bergerak (yang dapat digambarkan sebagai warna merah) dan latar belakang. Misalnya, gambar *pixel* dialokasikan pada *frame* saat sekarang dan *frame* sebelumnya, dapat dilihat dalam Gambar 2.1. Misalnya gambar memiliki ukuran 7x7 dan perbedaan antara *frame* sebelumnya dan *frame* sekarang adalah benda berbentuk lingkaran.

1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,1	5,2	5,3	5,4	5,5	5,6	5,7
6,1	6,2	6,3	6,4	6,5	6,6	6,7
7,1	7,2	7,3	7,4	7,5	7,6	7,7

Gambar 2.1 Pixel Frame dan Objek Bergerak

Langkah selanjutnya adalah dengan memperkenalkan *threshold* (batas ambang), *threshold* ditentukan sehingga intensitas perbedaan gambar $\Delta I(i,j)$ tidak akan secara otomatis menjadi objek bergerak kecuali perbedaan tersebut lebih besar daripada *threshold.* Dalam teknik ini, *frame* yang akan dipindai dua kali, pertama dari kiri ke kanan baris per baris kedua dari atas ke bawah, kolom per kolom.

Misalnya, *pixel* dari kedua *frame* akan dipindai (1,1) ke (1,7); (2,1) ke (2,7); (3,1) ke (3,7); (4,1) ke (4,7); (5,1) ke (5,7); (6,1) ke (6,7); (7,1) ke (7,7). Ketika setiap baris dipindai maka akan ada dua kondisi yang membuat suatu *pixel* memiliki batas dari objek. Pertama, perbedaan antara dua *pixel* yang sesuai (lokasi (i,j) sama) dari dua *frame* lebih besar daripada *threshold*. Contoh jika A ($[ICurr(4,4) - IPrev(4,4)] > threshold$) maka akan ditentukan Hitam ataukah Merah. Kondisi kedua adalah ketika status dua *pixel* yang berdekatan dapat dibedakan. *Pixel* yang akan dianggap sebagai gerakan jika memenuhi kedua kondisi. Untuk contoh: pada Gambar 2.2 terdapat tujuh *pixel* pada baris kedua dengan hasil (2,1) adalah hitam, (2,2) adalah hitam, (2,3) adalah merah, (2,4) adalah merah, (2,5) adalah hitam, (2,6) adalah hitam dan (2,7) adalah hitam. Dua *pixel* (2,3) dan (2,4) akan dipertimbangkan sebagai gerakan karena memenuhi kondisi yang kedua.

1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,1	5,2	5,3	5,4	5,5	5,6	5,7
6,1	6,2	6,3	6,4	6,5	6,6	6,7
7,1	7,2	7,3	7,4	7,5	7,6	7,7

Gambar 2.2 *Motion detection*

2.2 *Framework* .NET

Framework.NET adalah suatu komponen windows yang terintegrasi yang dibuat dengan tujuan untuk mensupport

pengembangan berbagai macam jenis aplikasi serta untuk dapat mejalankan berbagai macam aplikasi generasi mendatang termasuk pengembangan aplikasi Web Services XML.

Saat ini *framework .NET* umumnya telah terintegrasi dalam distribusi standar Windows (mulai dari Windows Server 2003 dan versi-versi Windows yang lebih baru). Kerangka kerja ini menyediakan sejumlah besar pustaka pemrograman komputer dan mendukung beberapa bahasa pemrograman serta interoperabilitas yang baik sehingga memungkinkan bahasa-bahasa tersebut berfungsi satu dengan lain dalam pengembangan sistem. Perangkat lunak ini adalah kunci penawaran utama dari Microsoft dan dimaksudkan untuk digunakan oleh sebagian besar aplikasi-aplikasi baru yang dibuat untuk *platform* Windows [2].

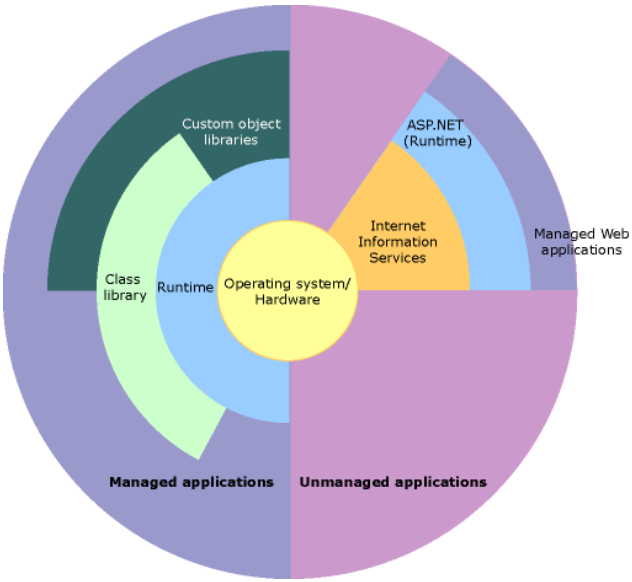
Framework .NET dapat memenuhi beberapa tujuan berikut:

1. Menyediakan lingkungan pemrograman berorientasi objek, apakah kode objek disimpan dan dijalankan secara lokal, dijalankan secara lokal tetapi disebarkan melalui internet atau dijalankan secara *remote* (dijalankan dari suatu tempat).
2. Menyediakan lingkungan untuk menjalankan suatu kode yang meminimalkan konflik saat *software deployment* disebarkan dan *versioning* atau tentang versi.
3. Menyediakan lingkungan untuk menjalankan suatu kode yang menjamin keamanan saat kode dijalankan, termasuk kode yang dibuat oleh pihak yang tidak diketahui atau pihak ketiga yang setengah dipercaya.
4. Menyediakan lingkungan untuk menjalankan suatu kode yang dapat mengeliminasi masalah performa dari lingkungan *scripted* dan *interpreted*.
5. Membuat pengembang memiliki pengalaman yang konsisten dalam berbagai tipe aplikasi berbasis Windows dan aplikasi berbasis *web*.

Sebagai salah satu sarana untuk dapat memenuhi tujuan yang telah dipaparkan, maka dibuatlah berbagai macam bahasa pemrograman yang dapat digunakan dan dapat berjalan di atas

platform framework seperti bahasa C#, VB.NET, C++, J#, Perl.NET dan lain-lain. Masing-masing bahasa tersebut mempunyai kelebihan dan kekurangannya masing-masing, namun yang pasti apapun bahasa pemrograman yang digunakan semua dapat saling berkomunikasi dan saling compatible satu sama lainnya dengan bantuan framework .NET.

Ilustrasi pada Gambar 2.1 menunjukkan hubungan *runtime* bahasa secara umum dan *class library* untuk aplikasi dan sistem secara keseluruhan dalam konteks framework .NET. Ilustrasi ini juga menunjukkan bagaimana kode dikelola untuk beroperasi dalam arsitektur yang lebih besar.



Gambar 2.3 Ilustrasi Hubungan Runtime Bahasa Secara Umum

Framework .NET memungkinkan digunakan untuk menyelesaikan berbagai tugas pemrograman umum, termasuk tugas-tugas seperti *string management*, pengumpulan data,

konektivitas database, dan akses file. Berikut ini layanan dan aplikasi yang dapat dibangun menggunakan framework .NET.

1. *Console Application*, membangun aplikasi konsol.
2. Aplikasi Windows GUI (*Windows Forms*), aplikasi berbasis *desktop*.
3. *Windows Presentation Foundation* (WPF) aplikasi.
4. Aplikasi ASP.NET, mengembangkan aplikasi berbasis web.
5. Layanan Windows (*Windows Service*), aplikasi *Windows Service*.
6. Aplikasi berorientasi layanan menggunakan *Windows Communication Foundation* (WCF).
7. Aplikasi *Workflow-enabled* menggunakan *Windows Workflow Foundation* (WF), membangun alur kerja [3].

Dalam tugas akhir ini untuk membangun sistem pendeteksian gerakan digunakan framework .NET Aplikasi Windows GUI atau aplikasi berbasis *desktop*.

2.3 AForge *Framework*

AForge *Framework* merupakan *framework open source* C# yang dirancang bagi para pengembang dan peneliti di bidang *computer vision* dan *artificial intelligence* yang meliputi pengolahan citra, jaringan saraf tiruan, algoritma genetika, logika *fuzzy*, *machine learning* dan robotika.

Framework ini terdiri dari beberapa *library* dan contoh aplikasi yang ditunjukkan oleh fitur-fitur sebagai berikut:

1. *Aforge.Imaging, library* dengan *image processing* dan *filtering,* untuk mem-*filter* gambar yang akan diproses.
2. *Aforge.Vision, computer vision library, library* untuk proses *motion detection* yang sederhana, membedakan nilai *threshold* dan perhitungan perbedaan piksel.
3. *Aforge.Neuro, neural network computation library,* untuk membuat *arsitekture neural network* (jaringan saraf tiruan).
4. *Aforge.Genetic, evolution programing library,* untuk ilmu komputasi.

5. *Afore.MachineLearning, machine learning library,* mesin pembelajaran.
6. *Aforge.Robotics, library providing support of some robotics kits,* memanipulasi perbedaan Lego Mindsrtom peralatan robotic, mendukung peralatan Lego Mindstrom RCX dan Lego Mindstrom NXT.
7. *Aforge.Video, set of libraries for video processing,* untuk video yang terintegrasi dengan Windows [4].

Pada tugas akhir ini, untuk proses pengolahan video menggunakan *library AForge.Video*. Pengolahan video digunakan untuk mendeteksi *driver* kamera dan menampilkan gambar di *videoSourcePlayer* sehingga dapat digunakan untuk proses pengolahan citra. Dan untuk proses pengolahan citra menggunakan *library AForge.Imaging* dan *library AForge.Vision*, *library AForge.Vision* memiliki berbagai macan *class* dan *class* yang dibutuhkan untuk deteksi gerakan adalah *class MotionDetector* [5].

2.4 Pemrograman C#

C# (dibaca "*See-sharp"*) merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka *framework* .NET. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti Java, Delphi, Visual Basic, dan lain-lain dengan beberapa penyederhanaan. Menurut standar ECMA-334 C# *Language Specification*, nama C# terdiri atas sebuah huruf Latin C (U+0043) yang diikuti oleh tanda pagar yang menandakan simbol # (U+0023).

Keuntungan memilih bahasa pemroraman C# adalah.

1. Sederhana.
2. Modern.
3. Object oriented language.
4. Powerfull dan fleksibel.

5. Efisien.
6. Modular [2].

2.5 *Computer Vision*

Computer vision merupakan proses otomatis yang mengintegrasikan sejumlah besar proses untuk persepsi visual, seperti akuisisi citra, pengolahan citra, pengenalan dan pengambilan keputusan pada data yang berdimensi tinggi dari dunia nyata untuk menghasilkan informasi numerik maupun simbolis [6]. *Computer vision* sering didefinisikan sebagai salah satu cabang ilmu pengetahuan yang mempelajari bagaimana komputer dapat mengenali obyek yang diamati atau diobeservasi. *Computer vision* merupakan kombinasi antara *Image Processing* dan *Pattern Recognition.*

Image Processing (Pengolahan Citra) merupakan bidang yang berhubungan dengan proses transformasi citra atau gambar. Proses ini bertujuan untuk mendapatkan kualitas citra yang lebih baik. Sedangkan *Pattern Recognition* (Pengenalan Pola), merupakan bidang yang berhubungan dengan proses identifikasi obyek pada citra atau interpretasi citra. Proses ini bertujuan untuk mengekstrak informasi atau pesan yang disampaikan oleh gambar.

Computer vision mencoba meniru cara kerja sistem visual manusia yang sangat kompleks. Sebuah komputer yang menyerupai kemampuan manusia dalam menangkap sinyal visual dilakukan dalam empat tahapan proses dasar sebagai berikut:

1. Proses penagkapan citra atau gambar (*image acquisition*),
2. Proses pengolahan citra (*image processing*),
3. Analisa data citra (*image analysis*),
4. Proses pemahaman data citra (*image understanding*) [7].

2.6 Microsoft Visual Studio

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (*suite*) yang dapat digunakan untuk melakukan

pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya. Aplikasi yang dapat dibangun dengan Microsoft Visual Studio antara lain adalah aplikasi bentuk *console*, aplikasi Windows berbasis *desktop* dan aplikasi berbasis *Web*. Visual Studio dapat mencakup beberapa perkakas kerja yaitu kompiler, SDK, Integrated Development Environment (IDE) dan dokumentasi (berupa MSDN *Library*). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro dan Visual SourceSafe.

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam *native code* (bentuk bahasa mesin yang berjalan di atas sistem operasi Windows) ataupun *managed code* (bentuk *Microsoft Intermediate Language* di atas .NET *Framework*). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi *Silverlight*, aplikasi *Windows Mobile* (yang berjalan di atas .NET *Compact Framework)* [8].

Microsoft Visual Studio memiliki banyak versi yang telah dikembangkan dari mulai versi pertamanya. Adapun Visual Studio yang digunakan dalam tugas akhir ini adalah Microsoft Visual Studio 2010.

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN

Pada bab ini dibahas analisa kebutuhan, proses bisnis dan desain dari perangkat lunak yang dibangun dalam tugas akhir ini. Bagian awal bab akan dibahas tentang analisa permasalahan dan kebutuhan masyarakat. Berikutnya dibahas fungsional sistem yang berdasarkan hasil analisa kebutuhan. Bagian berikutnya akan dibahas rancangan perangkat lunak yang ditujukan untuk memberikan gambaran tentang perangkat lunak yang dibuat.

3.1. Analisis

Pada subbab berikut akan dijelaskan analisa pembuatan sistem pendeteksi gerakan (*Motion detection*). Analisa yang dilakukan meliputi analisa permasalahan, kebutuhan umum perangkat lunak, deskripsi umum sistem, arsitektur dan kebutuhan fungsional sistem.

3.1.1. Permasalahan

Pada saat ini keamanan adalah salah satu hal yang sangat penting. Dengan kemajuan teknologi dalam bidang keamanan sekarang ini, pengamatan suatu objek menjadi lebih praktis. Untuk mengamati suatu objek tidak perlu dilakukan pengamatan secara langsung dan terus menerus. Cukup meletakkan suatu kamera yang mengarah pada objek yang diinginkan dan objek tersebut dapat diamati pada layar monitor.

Salah satu sistem keamanan pemantauan objek yang tersedia saat ini adalah sistem keamanan kemera pengawas atau lebih dikenal dengan sebutan *Closed Circuit Television* (CCTV). CCTV adalah sebuah kamera video digital yang difungsikan mengontrol semua kegiatan secara visual (audio visual) pada area tertentu. Yang secara lansung dapat mengawasi, mengamati serta merekam

kejadian di suatu ruangan atau area tertentu dan mengirimkan sinyal video pada ruang tersebut yang kemudian akan diteruskan ke sebuah layar monitor.

Permasalahan yang timbul selama ini kamera hanya dapat menangkap suatu objek tetapi tidak dapat memberikan informasi tentang pergerakan objek tersebut. Begitu juga pada kebanyakan system keamanan CCTV.

Hal ini sangat berguna jika diaplikasikan ke dalam sistem pendeteksi gerakan dengan *motion detection*. Ketika ada objek yang bergerak, maka secara otomatis aplikasi akan memberikan informasi objek asing yang terdeteksi tersebut. Ini akan memberikan kemudahan bagi pengguna untuk menganalisis lebih lanjut video dari rekaman kamera keamanan. Dan sistem ini juga dapat memberikan informasi secara *realtime* dari kamera yang tersambung ke sistem.

3.1.2. Deskripsi Umum Perangkat Lunak

Sistem yang dibangun pada Tugas Akhir ini bernama Pendetaksi Gerakan. Pendeteksi gerakan adalah sistem berbasis desktop yang dapat dioperasikan di sistem operasi Windows. Informasi yang diberikan berupa *record* dari pergerakan sebuah objek dalam bentuk waktu lamanya pergerakan terjadi mulai dari awal bergerak sampai berhenti. Setiap pergerakan yang terdeteksi akan di tampilkan hasil *record*.

Sistem pendeteksi gerakan dapat dijalankan dalam dua mode pendeteksian. Yaitu, deteksi dari file video rekaman dan dateksi langsung dari perangkat kamera yang tersambung ke kistem secara *realtime.* Pada dasarnya kedua mode pendeteksian tersebut tidak ada perbedaan pada cara peroperasian dan juga hasil *record* yang diberikan. Jika deteksi dari file video rekaman, pengguna harus menyediakan file video yang akan dianalisis deteksi gerakannya terlebih dahulu. Sedangkan untuk deteksi secara *realtime* pengguna

harus menyediakan perangkat tambahan berupa perangkat kamera yang dapat dihubungkan dengan sistem.

Sistem pendeteksi gerakan dibangun dengan menggunakan bahasa pemrograman C# dan AForge .NET *framework*. C# digunakan untuk membangun aplikasi secara keseluruhan seperti tampilan antar muka pengguna dan program fungsional aplikasi. Sedangkan AForge .NET *framework* digunakan untuk menjalankan algoritme *motion detection* dan algoritme *motion processing*.

Selain untuk menganalisis gerakan objek secara keseluruhan video, sistem pendeteksi gerakan ini juga dapat mendeteksi *region area* yaitu mendeteksi hanya sebagian wilayah pada video yang telah diberi tanda oleh pengguna sesuai keinginan pengguna. Misalkan pengguna hanya ingin mengamati pergerakan pada sebagian objek tertentu saja dari keseluruhan layar video yang ditampilkan. Maka, pergerakan yang terjadi selain dalam batas area yang telah ditentukan tersebut tidak akan terhitung sebagai gerakan.

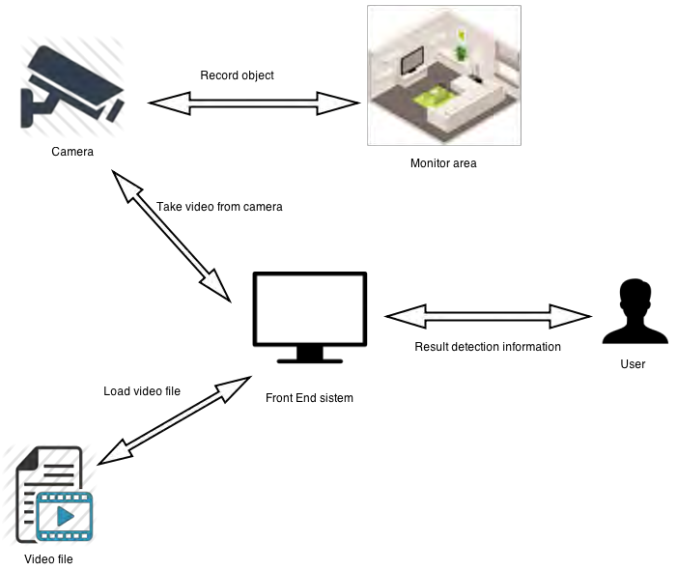
Fungsionalitas sistem pendeteksi gerakan apabila diringkas adalah sebagai berikut.

1. Dapat mendeteksi pergerakan dari file video.
2. Dapat mendeteksi pergerakan secara *realtime.*
3. Dapat memberikan informasi waktu lamanya pergerakan.
4. Dapat mendeteksi pergerakan pada sebagian area yang ditentukan.

3.1.3. Arsitektur Sistem

Sistem pendeteksi gerakan menggunakan arsitektur seperti pada Gambar 3.1. Sistem ini merupakan aplikasi berbasis desktop yang dijalankan di atas sistem operasi Windows. Sistem dapat mendeteksi gerakan dari dua jenis sumber. Sumber pertama berupa berkas video dari komputer pengguna dan sumber kedua berupa

hasil tangkapan langsung perangkat kamera yang disambungkan ke sistem.

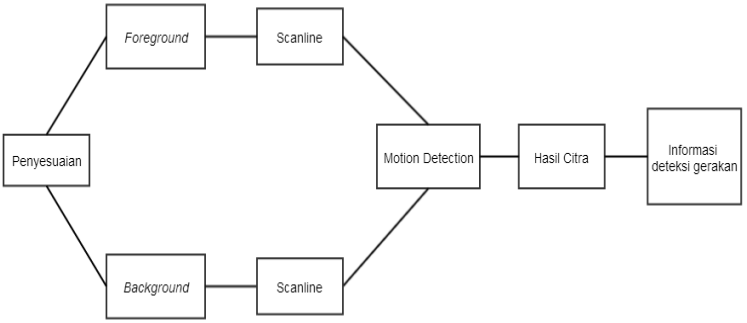


Gambar 3.1 Arsitektur Sistem

Pendekatan *motion detection* dilakukan dengan mengidentifikasi adanya suatu gerakan dengan cara melakukan proses pengurangan nilai- nilai intensitas setiap piksel yang ada pada *background* terhadap nilai-nilai intensitas pada suatu *foreground* (objek) yang diambil secara kontinyu. Secara garis besar, gambaran umum sistem pendeteksi gerakan ditunjukkan pada Gambar 3.2. Penjelasan poin-poin prosesnya sebagai berikut.

1. Proses Penyesuaian
   Pada proses penyesuaian ini, gambar yang ditangkap kamera video sampai pada kondisi normal, yaitu sebuah kondisi dimana tidak akan terjadi perubahan nilai secara signifikan dari intensitas gambar. Dan kondisi yang tetap pada area yang dipantau.

2. Proses Penentuan *Background*
   Pada proses ini *background* yang didapat dari area yang dipantau akan di-*scanline*, yaitu proses pengambilan atau pencarian nilai intensitas yang ada pada suatu gambar (untuk setiap pikselnya) yang memiliki format *bitmap*.
3. Proses Pengambilan *foreground* (objek)
   Setelah penentuan *background* langkah selanjutnya adalah pengambilan gambar yang dilakukan secara kontinyu persatuan waktu. Gambar yang diperoleh akan didefinisikan sebagai *foreground* (objek).
4. Proses *Motion Detection*
   Pada proses *motion detection, background* dan *foreground* (objek) akan di-*scanline*. Angka-angka yang diperoleh merepresentasikan tingkat intensitas gambar yang ada pada *foreground* (objek). Nilai yang diperoleh akan disubstitusikan ke dalam sebuah persamaan untuk menentukan adanya gerakan. Setelah gerakan terdeteksi, maka sistem akan memberikan informasi pendeteksian gerakan yang terjadi.



Gambar 3.2 Gambaran Umum Proses *Motion Detection*

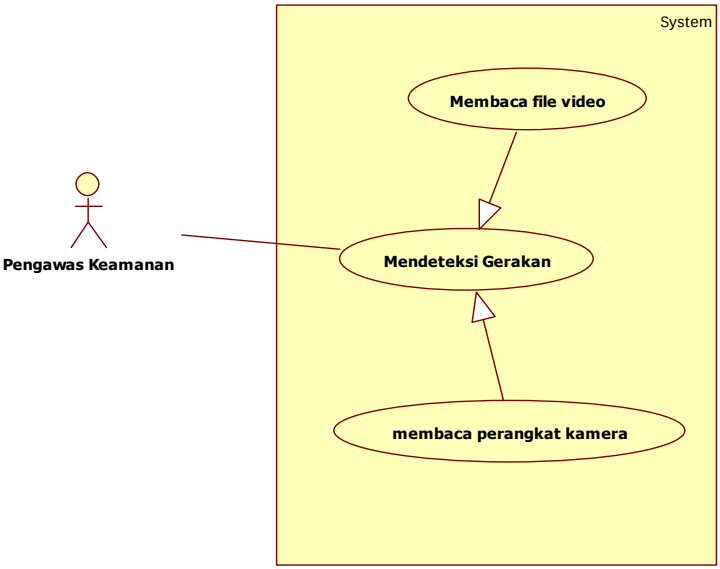
3.1.4. Kebutuhan Fungsional Sistem

Sistem pendeteksi gerakan berfungsi untuk memudahkan menganalisa pergerakan objek pada video dengan memberikan

informasi pergerakan kepada pengguna. Berikut daftar kebutuhan fungsional dari sistem pendeteksi gerakan.

1. Mendeteksi gerakan.
2. Membaca file video.
3. Membaca perangkat kamera.
4. Membuat batasan wilayah deteksi.

Kebutuhan fungsional aplikasi berbasis web digambarkan seperti Gambar 3.3.



Gambar 3.3 Diagram Kasus Penggunaan Sistem Pendeteksi Gerakan

Yang akan menggunakan sistem pendeteksi gerakan adalah pengawas keamanan. Pengawas keamanan dapat menganalisa dan mengelola hasil deteksi dari sistem. Penjelasan lengkap mengenai kasus penggunaan berada di Tabel 3.1.

Tabel 3.1 Deskipsi Kasus Penggunaan Sistem Pendeteksi Gerakan

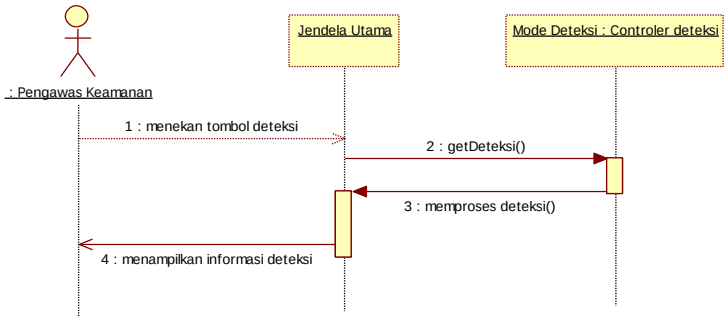
No	Kode	Nama	Keterangan
1	UC-D01	Mendeteksi Gerakan	Pengawas keamanan dapat menentukan untuk menjalankan sistem pendeteksian pergerakan pada video.
2	UC-D02	Membaca File Video	Pengawas keamanan dapat memilih sumber deteksi dari berkas video untuk mendeteksi gerakan.
3	UC-D03	Membaca Perangkat Kamera	Pengawas keamanan dapat memilih sumber deteksi langsung dari perangkat kamera.
4	UC-D04	Membuat batasan wilayah deteksi	Pengawas keamanan dapat membuat batasan wilayah deteksi untuk memudahkan pendeteksian

3.1.4.1. Deskripsi Kasus Kebutuhan UC-D01

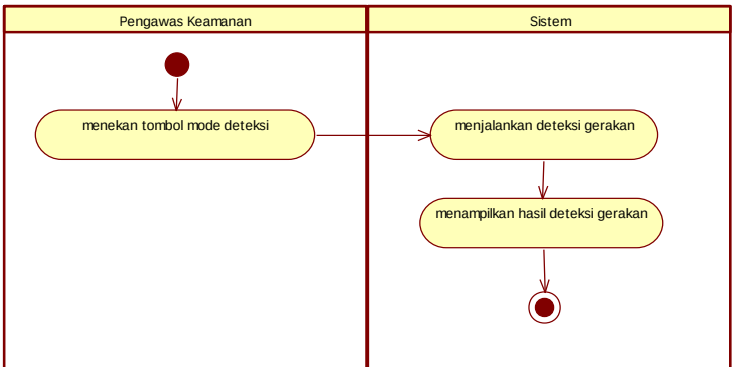
Kasus kebutuhan kode UC-D01 merupakan kasus kebutuhan utama mendeteksi gerakan. Rincian alur kasus mendeteksi gerakan dijelaskan pada Tabel 3.2 dan diagram urut ditampilkan pada Gambar 3.4. Sedangkan Gambar 3.5 adalah gambar diagram aktivitas kasus kebutuhan.

Tabel 3.2 Rincian Alur Kasus Penggunaan Sistem Pendeteksi Gerakan

Nama Use Case	Mendeteksi gerakan
Nomor	UC-D01
Aktor	Pengawas Keamanan
Kondisi Awal	Video belum ditampilkan ke layar dan deteksi belum dijalankan
Kondisi akhir	Informasi catatan deteksi ditampilkan
Alur Normal	1. Pengawas keamanan memilih buka file video. A.1. Pengawas keamanan memilih perangkat kamera. 2. Sistem membuka file direktori. 3. Pengawas keamanan menyalakan mode deteksi gerakan. 4. Sistem menjalankan deteksi gerakan. 5. Sistem menampilkan hasil deteksi gerakan. 6. Selesai mendeteksi gerakan.
Alur Alternatif	A1.1. Sistem menjalankan kamera. A1.2. Selesai menjalankan kamera.



Gambar 3.4 Diagram Urut Kode UC-D01



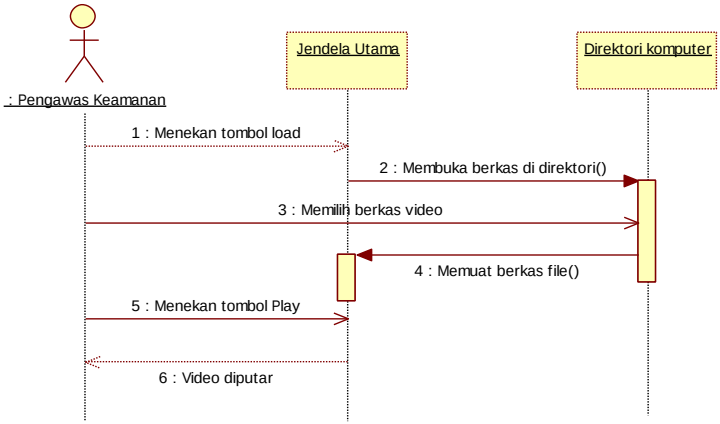
Gambar 3.5 Sub Diagram Aktivitas UC-D01

3.1.4.2. Deskripsi Kasus Kebutuhan UC-D02

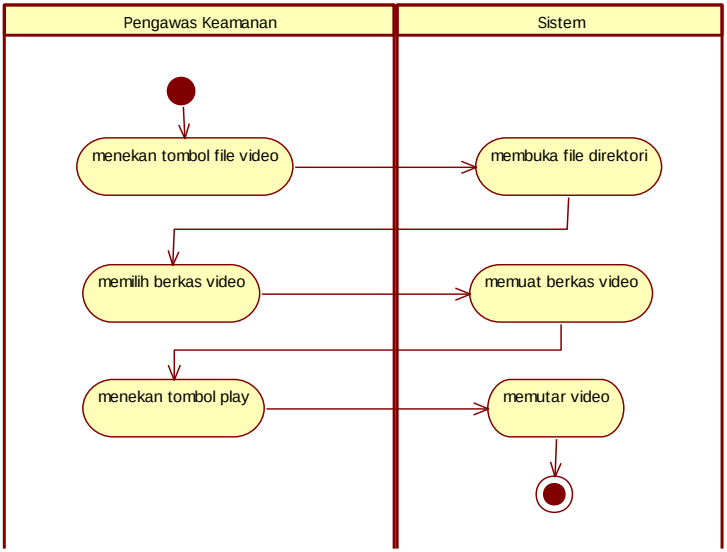
Kasus kebutuhan kode UC-D02 merupakan kasus kebutuhan generalisasi membaca file video. Digunakan sebagai pilihan sumber deteksi dari berkas video untuk menjalankan kasus kebutuhan mendeteksi gerakan. Rincian alur kasus membaca berkas video dijelaskan pada Tabel 3.3 dan diagram urut ditampilkan pada Gambar 3.6. Sedangkan Gambar 3.7 adalah gambar diagram aktivitas kasus kebutuhan.

Tabel 3.3 Rincian Alur Kasus Penggunaan Membaca File Video

Nama	Membaca file video
Nomor	UC-D02
Aktor	Pengawas Keamanan
Kondisi Awal	Video belum dimuat ke sistem
Kondisi akhir	Video berjalan di sistem
Alur Normal	1. Pengawas keamanan menekan tombol load. 2. Sistem membuka berkas di direktori file. 3. Sistem memuat berkas video.
Alur Alternatif	-



Gambar 3.6 Diagram Urut Kode UC-D02



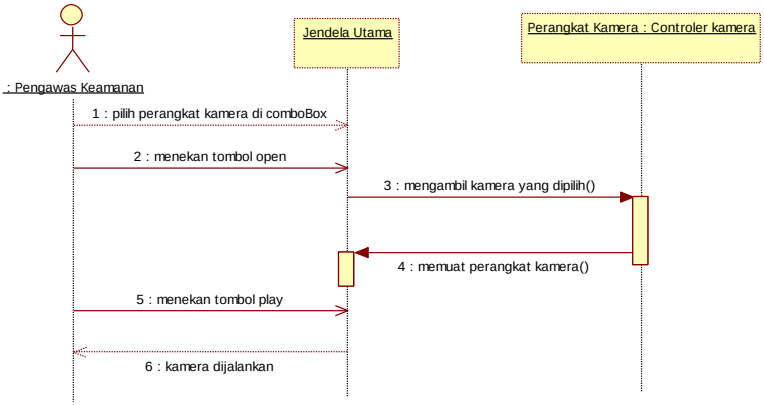
Gambar 3.7 Sub Diagram Aktivitas UC-D02

3.1.4.3. Deskripsi Kasus Kebutuhan UC-D03

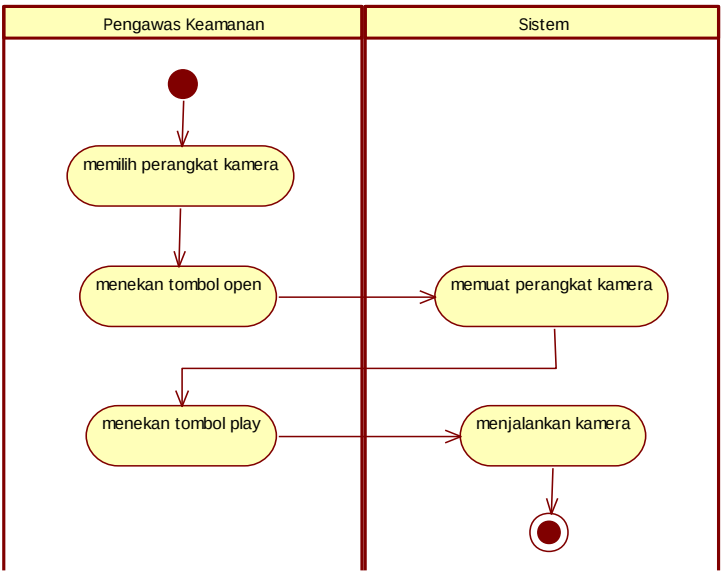
Kasus kebutuhan kode UC-D03 merupakan kasus kebutuhan generalisasi membaca perangkat kamera. Digunakan sebagai pilihan sumber deteksi dari perangkat kamera yang terhubung ke sistem untuk menjalankan kasus kebutuhan mendeteksi gerakan. Rincian alur kasus membaca perangkat kamera dijelaskan pada Tabel 3.4 dan diagram urut ditampilkan pada Gambar 3.8. Sedangkan Gambar 3.9 adalah gambar diagram aktivitas kasus kebutuhan. Diagram aktivitas menggambarkan tahapan alur kerja sistem secara berurutan, bagaimana masing-masing alir berawal, decision yang mungkin terjadi hingga alir berakhir.

Tabel 3.4 Rincian Alur Kasus Penggunaan Membaca Perangkat Kamera

Nama	Membaca perangkat kamera
Nomor	UC-D03
Aktor	Pengawas Keamanan
Kondisi Awal	Gambar tangkapan kamera belum dimuat ke sistem
Kondisi akhir	Hasil gambar tangkapan kamera dijalankan
Alur Normal	1. Pengawas keamanan memilih perangkat kamera di *comboBox*. 2. Pengawas keamanan menekan tombol open. 3. Sistem membaca kamera yang dipilih. 4. Sistem memuat gambar tangkapan kamera ke sistem.
Alur Alternatif	-



Gambar 3.8 Diagram Urut Kode UC-D03



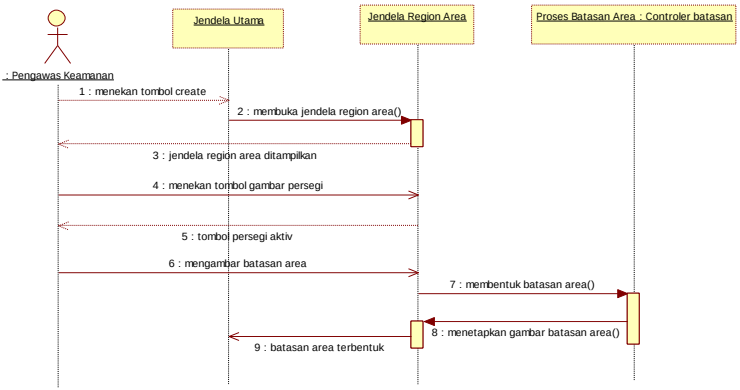
Gambar 3.9 Sub Diagram Aktivitas UC-D03

3.1.4.4. Deskripsi Kasus Kebutuhan UC-D04

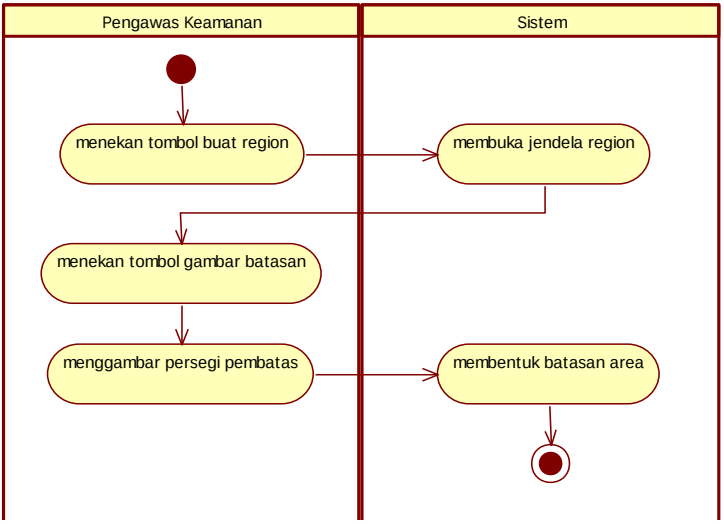
Kasus kebutuhan kode UC-D04 merupakan kasus kebutuhan fitur tambahan yaitu membuat batasan wilayah deteksi. Digunakan sebagai pilihan pendeteksian dengan membatasi wilayah deteksi supaya memudahkan pendeteksian. Rincian alur kasus membuat batasan wilayah deteksi dijelaskan pada Tabel 3.5 dan diagram urut ditampilkan pada Gambar 3.10. Sedangkan Gambar 3.11 adalah gambar diagram aktivitas kasus kebutuhan. Diagram aktivitas menggambarkan tahapan alur kerja sistem secara berurutan, bagaimana masing-masing alir berawal, decision yang mungkin terjadi hingga alir berakhir.

Tabel 3.5 Rincian Alur Kasus Penggunaan Membuat Batasan Wilayah Deteksi

Nama	Membuat batasan wilayah deteksi
Nomor	UC-D04
Aktor	Pengawas Keamanan
Kondisi Awal	Video sudah dimuat ke sistem
Kondisi akhir	Batasan wilayah deteksi digambarkan
Alur Normal	1. Pengawas keamanan menekan tombol *create*. 2. Sistem membuka halaman region area 3. Pengawas keamanan menekan tombol gambar persegi. 4. Pengawas keamanan menggambar persegi sebagai batasan wilayah. 5. Sistem membentuk batasan area. 6. Sistem menetapkan gambar batasan area.
Alur Alternatif	-



Gambar 3.10 Diagram Urut Kode UC-D04



Gambar 3.11 Sub Diagram Aktivitas UC-D04

3.2. Perancangan

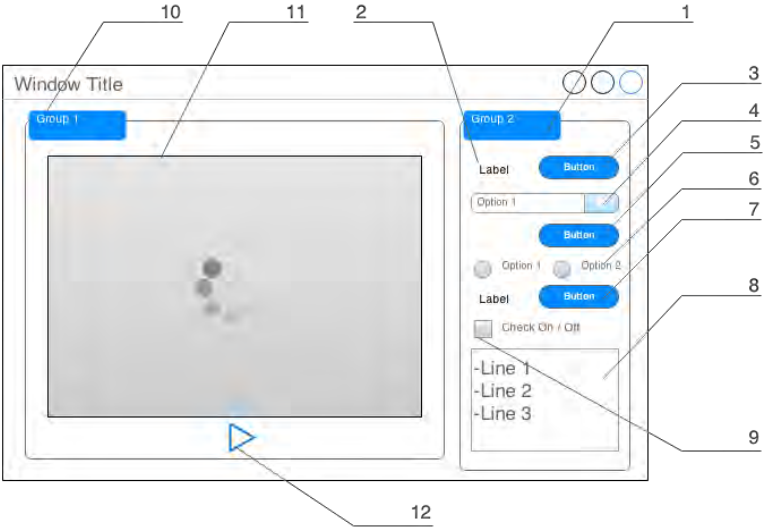
Subbab berikut membahas tentang perancangan dari sistem pendeteksi gerakan. Adapun perancangan yang akan dibahas adalah rancangan antarmuka dan rancangan proses. Pembahasan lebih detil akan dibahas berikut ini.

3.2.1. Perancangan Antarmuka Sistem

Pada subbab ini akan dibahas secara mendetail dari rancangan antarmuka sistem pendeteksi pergerakan.

3.2.1.1. Antarmuka Jendela Utama

Gambar 3.12 merupakan gambar rancangan antarmuka jendela utama. Pada jendela ini terbagi atas dua kotak grup. Di sebelah kiri adalah kotak grup untuk layar video sedangkan di sebelah kanan adalah kotak grup menu yang dapat digunakan untuk menjalankan sistem.



Gambar 3.12 Rancangan Antarmuka Jendela Utama

Gambar 3.12 merupakan rancangan antarmuka jendela utama. Berikut penjelasan masing-masing nomor yang tertera pada Gambar 3.12.

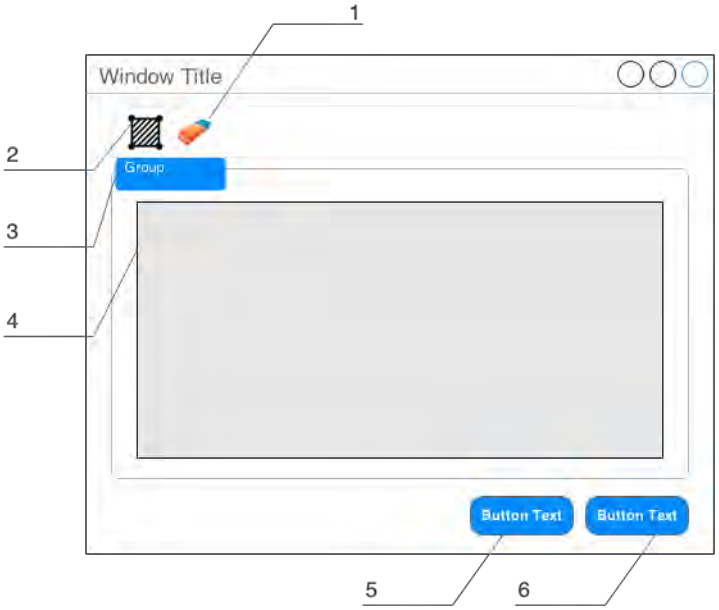
1. Kotak grup untuk menu.
2. Berupa *label* untuk memberikan nama fungsi membuka file video.
3. Berupa tombol untuk membuka file video dari computer pengguna.
4. Berupa *combobox* untuk menampilkan pilihan perangkat kamera yang terhubung ke sistem.
5. Berupa tombol untuk menjalankan pilihan perangkat kamera.
6. Berupa *radiobutton* untuk menghidupkan dan mematikan mode deteksi.
7. Berupa *label* untuk membrikan nama fungsi membuat *region area* deteksi dan tombol untuk menjalankan fungsi membuat *region area* deteksi.
8. Berupa *listbox* untuk menampilkan informasi hasil deteksi.
9. Berupa *checkbox* untuk menghidupkan dan mematikan simpan tangkapan gambar.
10. Kotak grup untuk layar video.
11. Merupakan layar video untuk memutar file video atau gambar tangkapan dari perangkat kamera.
12. Berupa tombol untuk memutar video yang telah dimuat.

3.2.1.2. Antarmuka Jendela Region Area

Gambar 3.13 merupakan gambar rancangan antarmuka untuk jendela region area. Pada jendela ini pengguna akan membuat region area dengan cara menggambarkan persegi pada area objek yang ingin dipantau khusus. Persegi yang digambarkan tersebut nantinya akan menjadi sebagai batasan area deteksi.

Gambar 3.13 merupakan rancangan antarmuka jendela region area. Berikut penjelasan masing–masing nomor yang tertera pada Gambar 3.13.

1. Berupa tombol dengan gambar penghapus untuk menghapus persegi yang sudah digambarkan sebelumnya.
2. Berupa tombol dengan gambar persegi untuk menggambarkan persegi pada area objek yang ingin dipantau khusus.
3. Kotak grup untuk layar.
4. Merupakan layar tempat menggambarkan area yang dipilih.
5. Berupa tombol untuk menyetujui aksi dari pengguna.
6. Berupa tombol untuk membatalkan aksi dari pengguna.



Gambar 3.13 Rancangan Antarmuka Jendela Region Area

3.2.2. Perancangan Proses pada Jendela Utama

Pada subbab ini akan dibahas secara mendetail dari rancangan proses yang terdapat pada jendela utama untuk memenuhi kebutuhan fungsionalnya.

3.2.2.1. Perancangan Proses Deteksi Gerakan

Proses ini merupakan proses mendeteksi gerakan. Proses deteksi gerakan pada sistem pendeteksi gerakan dapat dilakukan dari dua sumber deteksi yaitu, mendeteksi pergerakan dari file video dan mendeteksi gerakan dari perangkat kamera.

3.2.2.1.1. Proses Mendeteksi Pergerakan dari File Video

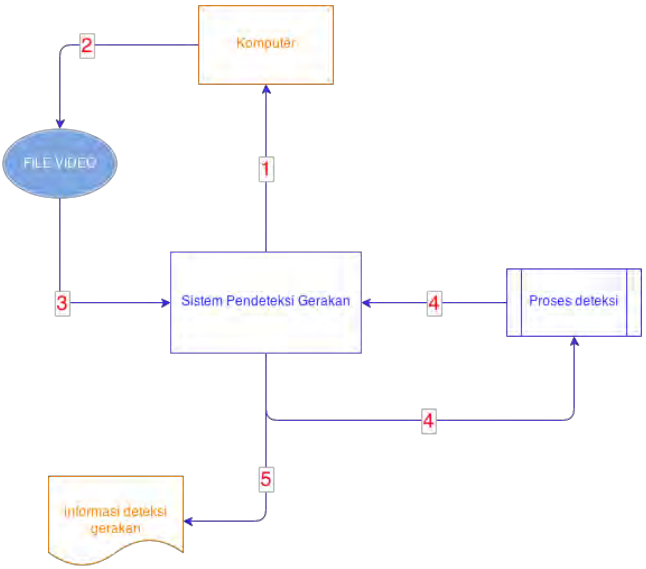
Proses ini merupakan proses mendeteksi pergerakan terhadap berkas video dari komputer pengguna. Mendeteksi gerakan bergantung pada sumber deteksi dari dipilih pengguna. Gambar 3.14 adalah gambar alur proses mendeteksi pergerakan dari file video.

Secara singkat urutan proses mendeteksi pergerakan dari file video adalah sebagai berikut.

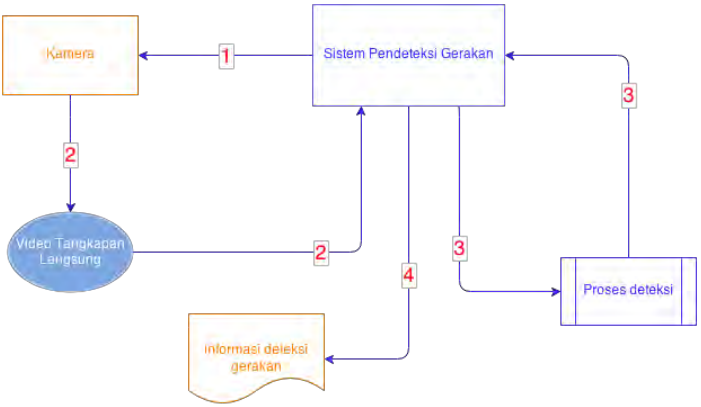
1. Sistem membuka berkas video pada komputer.
2. Sistem mengambil berkas video yang dipilih.
3. Berkas video yang dipilih dimuat ke sistem.
4. Pengguna menekan tombol untuk memutar video.
5. Sistem membaca dan mendeteksi gerakan pada video.
6. Sistem menampilkan informasi deteksi gerakan.

3.2.2.1.2. Proses Mendeteksi Pergerakan dari Perangkat Kamera

Proses ini merupakan proses mendeteksi pergerakan yang bersumber dari tangkapan langsung perangkat kamera yang terhubungkan pada sistem. Gambar 3.15 adalah gambaran alur proses mendeteksi pergerakan dari perangkat kamera.



Gambar 3.14 Alur Proses Mendeteksi Pergerakan dari Berkas Video



Gambar 3.15 Alur Proses Mendeteksi Pergerakan dari Perangkat Kamera

Alur proses mendeteksi pergerakan dari perangkat kamera adalah sebagai berikut.

1. Sistem menjalankan perangkat kamera.
2. Sistem membaca gambar tangkapan langsung kamera.
3. Sistem mendeteksi gerakan pada gambar tangkapan langsung kamera.
4. Sistem menampilkan informasi deteksi gerakan.

3.2.2.2. Perancangan Proses Memuat Berkas Video

Proses ini merupakan proses dimana pengguna harus menentukan sumber untuk mendeteksi dari berkas video. Untuk mendeteksi dari berkas video maka pengguna harus memuat berkas video terlebih dahulu ke sistem pendeteksi gerakan. Alur proses untuk memuat berkas video adalah sebagai berikut.

1. Pengguan menekan tombol untuk memuat berkas video pada daftar menu di jendela utama.
2. Sistem membuka berkas video pada komputer.
3. Sistem mengambil berkas video yang dipilih.
4. Berkas video yang dipilih dimuat ke sistem.

3.2.2.3. Perancangan Proses Memuat Perangkat Kamera

Proses ini merupakan proses dimana penguna harus menentukan sumber yang deteksi selain dari sumber berkas video yaitu dari sumber tangkapan langsung perangkat kamera. Untuk mendeteksi dari perangkat kamera maka pengguna harus menghubungkan perangkat kamera ke sistem pendeteksi gerakan. Alur proses untuk memuat perangkat kamera adalah sebagai berikut.

1. Penguna memilih perangkat kamera yang tersedia pada *ListBox* yang terdapat pada jendela utama.
2. Pengguna menekan tombol untuk menjalankan perangkat kamera.
3. Sistem menjalankan perangkat kamera.

4. Sistem membaca gambar tangkapan langsung kamera.

3.2.2.4. Perancangan Proses Membuat Region Area

Proses ini merupakan proses dimana pengguna ingin membuat batasan wilayah deteksi untuk memudahkan pemantauan pada sebagian wilayah tertentu saja. Pada jendela utama proses ini hanya untuk membuka jendela region area. Jendela region area adalah jendela untuk menentukan batasan wilayah deteksi seperti yang telah dijelaskan sebelumnya. Alur proses membuat region area adalah sebagai berikut.

1. Pengguna menekan tombol untuk membuat region area yang terdapat pada daftar menu di jendela utama.
2. Sistem membuka jendela region area.
3. Pengguna menentukan batasan wilayah deteksi.
4. Pengguna menekan tombol *OK*.
5. Sistem menetapkan batasan wilayah deteksi yang telah ditentukan.

3.2.2.5. Perancangan Proses Menampilkan Informasi Deteksi

Proses ini merupakan proses menampilkan informasi hasil deteksi. Informasi deteksi yang ditampilkan tergantung pada proses deteksi dari sistem pendeteksi gerakan terhadap video yang telah ditentukan oleh pengguna. Secara singkat urutan proses menampilkan informasi deteksi adalah sebagai berikut.

1. Berkas video yang dipilih pengguna atau video hasil tangkapan langsung dari kamera dimuat ke sistem.
2. Sistem membaca dan mendeteksi gerakan pada video.
3. Sistem menampilkan informasi deteksi gerakan.

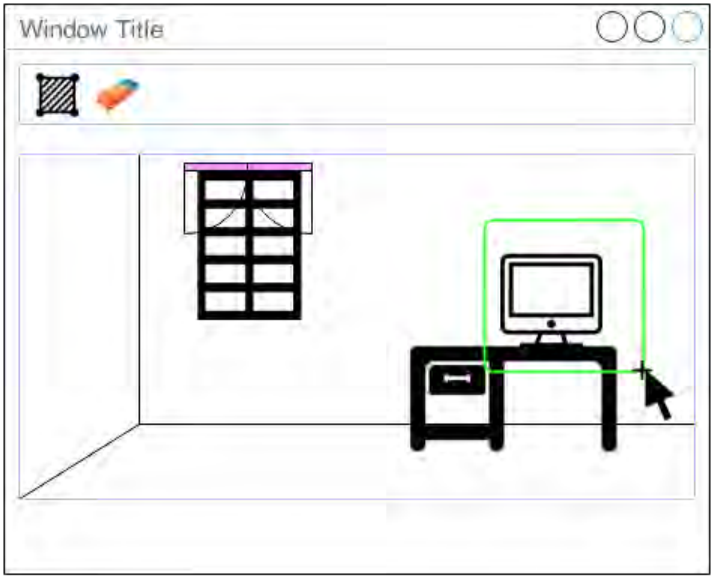
3.2.3. Perancangan Proses pada Jendela Region Area

Pada subbab ini akan dibahas secara mendetail dari rancangan proses yang terdapat pada jendela region area untuk memenuhi

kebutuhan fungsionalnya. Jendela region area dibuka apabila pengguna ingin menentukan batasan wilayah deteksi.

3.2.3.1. Perancangan Proses Menggambar Persegi

Proses menggambar persegi dilakukan ketika pengguna ingin menentukan batasan wilayah deteksi untuk memudahkan pemantauan pada sebagian wilayah yang diinginkan. Setelah pengguna memilih tombol persegi untuk memulai menggambar, pertama pengguna menentukan dimana wilayah yang ingin dibatasi kemudian pengguna tinggal menahan *click* sambil menggeser *mouse* untuk membentuk persegi pada wilayah tersebut. Gambaran proses menggambar persegi terdapat pada Gambar 3.16.



Gambar 3.16 Gambaran Proses Menggambar Persegi

Secara singkat urutan proses menampilkan informasi deteksi adalah sebagai berikut.

1. Pengguna memilih tombol persegi.

2. Pengguna menggambar persegi pada wilayah tertentu pada layar tampilan.
3. Sistem menetapkan batasan wilayah deteksi terhadap gambar persegi.

3.2.3.2. Perancangan Proses Menghapus Persegi

Proses ini merupakan proses menghapus persegi yang telah digambarkan sebelumnya. Semua persegi yang sudah tergambar pada layar akan terhapus ketika pengguna menekan tombol penghapus. Setelah layar bersih dari persegi maka deteksi akan dijalankan kembali pada keseluruhan layar. Pengguna bisa kembali menentukan batasan wilayah deteksi dengan menggambar persegi seperti yang telah dijelaskan pada Subbab proses menggambar persegi.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas implementasi dari rancangan sistem yang ditulis pada Bab 3. Namun, tidak menutup kemungkinan adanya perubahan-perubahan dari rancangan tersebut apabila memang diperlukan.

4.1. Lingkungan Implementasi

Dalam merancang perangkat lunak ini digunakan beberapa perangkat pendukung sebagai berikut.

4.1.1. Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem adalah komputer dengan spesifikasi *notebook* HP Pavilion dv3, Intel Core i3-M330 2,13Ghz dan 4GB memory.

4.1.2. Lingkungan Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk pengembangan sistem adalah sebagai berikut:

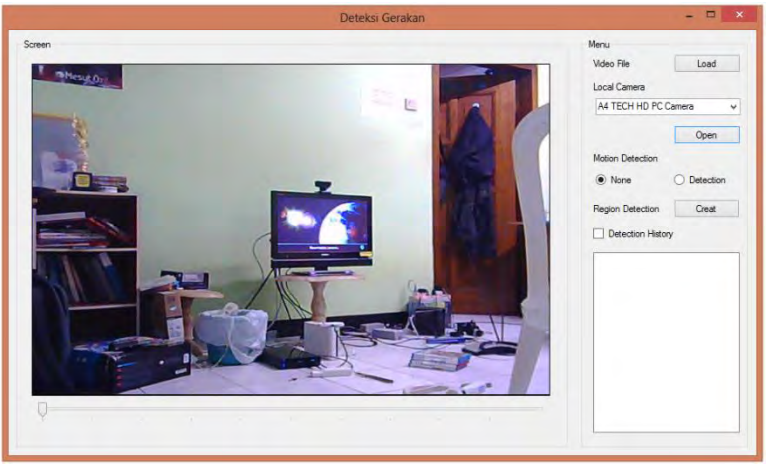
- Microsoft Windows 8 sebagai sistem operasi.
- Microsoft Visual Studio 2010 sebagai IDE untuk implementasi sistem.
- StarUML 5.0 untuk merancang design analisis.

4.2. Lingkungan Implementasi Antarmuka

Pada subbab ini akan dibahas implementasi antarmuka berdasarkan rancangan antarmuka yang telah dibahas pada bab 3. Antarmuka yang akan dibahas merupakan antarmuka dari sistem pendeteksi gerakan yang akan digunakan oleh pengawas keamanan. Pengawas keamanan akan menjalani sistem ini untuk mendeteksi gerakan dari rekaman berkas video.

4.2.1. Implementasi Antarmuka Jendela Utama

Jendela utama dari sistem pendeteksi gerakan seperti pada Gambar 4.1. Semua fungsi utama dari sistem terdapat pada jendela utama ini. Berikut implementasi dari rancangan jendela utama ketika pertama kali sistem dijalankan.



Gambar 4.1 Jendela Utama Sistem Pendeteksi Gerakan

Pada jendela utama sistem pendeteksi gerakan seperti pada Gambar 4.1 sisi kanan merupakan daftar menu yang dapat diakses oleh pengawas keamanan yang dapat menjalankan fungsi-fungsi tertentu. Sedangkan di sisi kiri merupakan layar untuk menampilkan video yang sedang dijalankan. Kode sumber berikut ini merupakan implementasi untuk membentuk antarmuka jendela utama.

```
1   private void InitializeComponent()
2   {
3           this.components = new
4   System.ComponentModel.Container();
5           this.groupBox1 = new
6   System.Windows.Forms.GroupBox();
7           this.panel1 = new
8   System.Windows.Forms.Panel();
9           this.videoSourcePlayer = new
10  AForge.Controls.VideoSourcePlayer();
11          this.groupBox2 = new
12  System.Windows.Forms.GroupBox();
13          this.checkReport = new
14  System.Windows.Forms.CheckBox();
15          this.listReport = new
16  System.Windows.Forms.ListBox();
17          this.button1 = new
18  System.Windows.Forms.Button();
19          this.label4 = new
20  System.Windows.Forms.Label();
21          this.radioMotion_Detect = new
22  System.Windows.Forms.RadioButton();
23          this.radioNone_Detect = new
24  System.Windows.Forms.RadioButton();
25          this.label3 = new
26  System.Windows.Forms.Label();
27          this.comboBox_Camera = new
28  System.Windows.Forms.ComboBox();
29          this.btn_Load = new
30  System.Windows.Forms.Button();
31          this.label2 = new
32  System.Windows.Forms.Label();
33          this.label1 = new
34  System.Windows.Forms.Label();
35          this.btn_Open = new
36  System.Windows.Forms.Button();
37          this.openFileDialog = new
38  System.Windows.Forms.OpenFileDialog();
39          this.alarmTimer = new
40  System.Windows.Forms.Timer(this.components);
41          this.timer = new
42  System.Windows.Forms.Timer(this.components);
43          this.saveFileDialog1 = new
44  System.Windows.Forms.SaveFileDialog();
45          this.trackBar1 = new
46  System.Windows.Forms.TrackBar();
47          this.groupBox1.SuspendLayout();
48          this.panel1.SuspendLayout();
49          this.groupBox2.SuspendLayout();
```

```
50          ((System.ComponentModel.ISupportInitialize)(t
51  his.trackBar1)).BeginInit();
52          this.SuspendLayout();
53
```

Kode Sumber 4. 1 Kode sumber inisialisasi komponen antarmuka jendela utama

```
1   //
2   // videoSourcePlayer
3   //
4   this.videoSourcePlayer.Anchor =
5   ((System.Windows.Forms.AnchorStyles)((((System.Windo
6   ws.Forms.AnchorStyles.Top |
7   System.Windows.Forms.AnchorStyles.Bottom)
8                           |
9   System.Windows.Forms.AnchorStyles.Left)
10                          |
11  System.Windows.Forms.AnchorStyles.Right)));
12  this.videoSourcePlayer.BackColor =
13  System.Drawing.SystemColors.ControlDark;
14  this.videoSourcePlayer.Location = new
15  System.Drawing.Point(13, 12);
16  this.videoSourcePlayer.Name = "videoSourcePlayer";
17  this.videoSourcePlayer.Size = new
18  System.Drawing.Size(486, 303);
19  this.videoSourcePlayer.TabIndex = 0;
20  this.videoSourcePlayer.TabStop = false;
21  this.videoSourcePlayer.VideoSource = null;
22  this.videoSourcePlayer.NewFrame         +=          new
23  AForge.Controls.VideoSourcePlayer.NewFrameHandler(th
24  is.videoSourcePlayer_NewFrame);
```

Kode Sumber 4. 2 Kode sumber detail *properties* komponen videoSourcePlayer

```
1   //
2   // radioMotion_Detect
3   //
4   this.radioMotion_Detect.AutoSize = true;
5   this.radioMotion_Detect.Location = new
6   System.Drawing.Point(116, 167);
7   this.radioMotion_Detect.Name = "radioMotion_Detect";
8   this.radioMotion_Detect.Size = new
9   System.Drawing.Size(71, 17);
10  this.radioMotion_Detect.TabIndex = 7;
11  this.radioMotion_Detect.TabStop = true;
```

```
12  this.radioMotion_Detect.Text = "Detection";
13  this.radioMotion_Detect.UseVisualStyleBackColor =
14  true;
15  this.radioMotion_Detect.CheckedChanged     +=     new
16  System.EventHandler(this.radioMotion_Detect_CheckedC
17  hanged);
```

Kode Sumber 4. 3 Kode sumber detail *properties* komponen radioButton mode deteksi

```
1   //
2   // comboBox_Camera
3   //
4   this.comboBox_Camera.FormattingEnabled = true;
5   this.comboBox_Camera.Location = new
6   System.Drawing.Point(18, 74);
7   this.comboBox_Camera.Name = "comboBox_Camera";
8   this.comboBox_Camera.Size = new
9   System.Drawing.Size(181, 21);
10  this.comboBox_Camera.TabIndex = 4;
```

Kode Sumber 4. 4 Kode sumber detail *properties* komponen comboBox perangkat kamera

```
1   //
2   // btn_Load
3   //
4   this.btn_Load.Location = new
5   System.Drawing.Point(116, 19);
6   this.btn_Load.Name = "btn_Load";
7   this.btn_Load.Size = new System.Drawing.Size(83,
8   23);
9   this.btn_Load.TabIndex = 3;
10  this.btn_Load.Text = "&Load";
11  this.btn_Load.UseVisualStyleBackColor = true;
12  this.btn_Load.Click                +=                new
13  System.EventHandler(this.btn_Load_Click);
```

Kode Sumber 4. 5 Kode sumber detail *properties* komponen tombol memuat file video

```
1   //
2   // btn_Open
3   //
4   this.btn_Open.Location = new
5   System.Drawing.Point(116, 108);
```

```
6   this.btn_Open.Name = "btn_Open";
7   this.btn_Open.Size = new System.Drawing.Size(83,
8   23);
9   this.btn_Open.TabIndex = 0;
10  this.btn_Open.Text = "&Open";
11  this.btn_Open.UseVisualStyleBackColor = true;
12  this.btn_Open.Click                 +=                 new
13  System.EventHandler(this.btn_Open_Click);
```

Kode Sumber 4. 6 Kode sumber detail *properties* komponen tombol memuat perangkat kamera

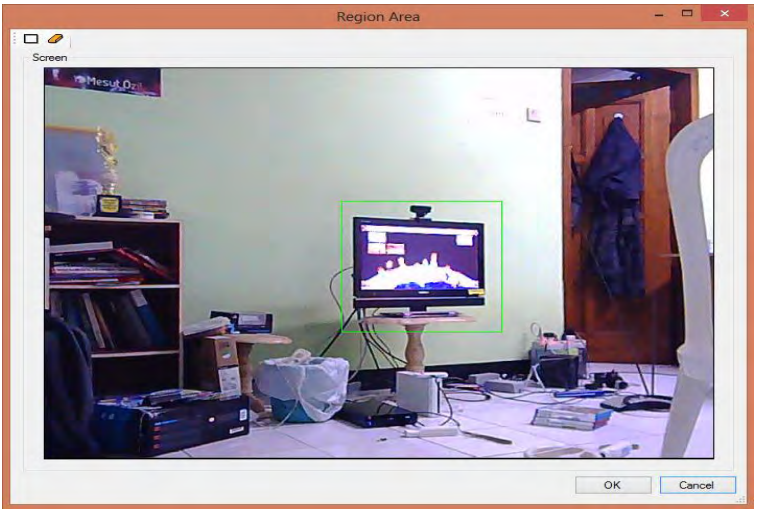
```
1   //
2   // MainForm
3   //
4   this.AutoScaleDimensions = new
5   System.Drawing.SizeF(6F, 13F);
6   this.AutoScaleMode =
7   System.Windows.Forms.AutoScaleMode.Font;
8   this.ClientSize = new System.Drawing.Size(773, 419);
9   this.Controls.Add(this.groupBox2);
10  this.Controls.Add(this.groupBox1);
11  this.Name = "MainForm";
12  this.ShowIcon = false;
13  this.StartPosition =
14  System.Windows.Forms.FormStartPosition.CenterScreen;
15  this.Text = "Deteksi Gerakan";
16  this.groupBox1.ResumeLayout(false);
17  this.panel1.ResumeLayout(false);
18  this.panel1.PerformLayout();
19  this.groupBox2.ResumeLayout(false);
20  this.groupBox2.PerformLayout();
21  ((System.ComponentModel.ISupportInitialize)(this.tra
22  ckBar1)).EndInit();
23  this.ResumeLayout(false);
```

Kode Sumber 4. 7 Kode sumber detail *properties* jendela MainForm

4.2.2. Implementasi Antarmuka Jendela Region Area

Berikut implementasi dari jendela region area sistem pendeteksi gerakan seperti pada Gambar 4.2. Jendela region area merupakan jendela untuk menggambarkan batas area deteksi. Jendela region area tampak pada Gambar 4.2 berikut ini.

Pada jendela region area seperti pada Gambar 4.2 yang menjadi area objek pantauan khusus adalah gambar persegi berwarna hijau yaitu pada bagian televisi. Jika terjadi perubahan di dalam area persegi tersebut maka akan dianggap sebagai gerakan. Pada jendela region area ini terdapat beberapa fungsi seperti fungsi menentukan area dengan gambar persegi dan juga fungsi menghapus gambar persegi.



Gambar 4.2 Jendela Region Area

Kode sumber berikut ini merupakan implementasi untuk membentuk antarmuka jendela region area.

```
1   private void InitializeComponent()
2   {
3           System.ComponentModel.ComponentResourceManage
4   r resources = new
5   System.ComponentModel.ComponentResourceManager(typeo
6   f(RegionForm));
7           this.toolStrip1 = new
8   System.Windows.Forms.ToolStrip();
9           this.bntGambar = new
10  System.Windows.Forms.ToolStripButton();
11
```

```
12          this.btnHapus = new
13  System.Windows.Forms.ToolStripButton();
14          this.btnCancel = new
15  System.Windows.Forms.Button();
16          this.btnOK = new
17  System.Windows.Forms.Button();
18          this.groupBox1 = new
19  System.Windows.Forms.GroupBox();
20          this.regionControl = new
21  DeteksiGerakan.RegionControl();
22          this.toolStrip1.SuspendLayout();
23          this.groupBox1.SuspendLayout();
            this.SuspendLayout();
```

Kode Sumber 4. 8 Kode sumber inisialisasi komponen antarmuka jendela region area

```
1   //
2   // bntGambar
3   //
4   this.bntGambar.DisplayStyle =
5   System.Windows.Forms.ToolStripItemDisplayStyle.Image
6   ;
7   this.bntGambar.Image =
8   ((System.Drawing.Image)(resources.GetObject("bntGamb
9   ar.Image")));
10  this.bntGambar.ImageTransparentColor =
11  System.Drawing.Color.Magenta;
12  this.bntGambar.Name = "bntGambar";
13  this.bntGambar.Size = new System.Drawing.Size(23,
14  22);
15  this.bntGambar.Text = "toolStripButton1";
16  this.bntGambar.Click              +=              new
17  System.EventHandler(this.bntGambar_Click);
```

Kode Sumber 4. 9 Kode sumber detail *properties* komponen tombol menggambar persegi

```
1   //
2   // btnHapus
3   //
4   this.btnHapus.DisplayStyle =
5   System.Windows.Forms.ToolStripItemDisplayStyle.Image
6   ;
7   this.btnHapus.Image =
8   ((System.Drawing.Image)(resources.GetObject("btnHapu
9   s.Image")));
```

```
this.btnHapus.ImageTransparentColor =
System.Drawing.Color.Magenta;
this.btnHapus.Name = "btnHapus";
this.btnHapus.Size = new System.Drawing.Size(23,
22);
this.btnHapus.Text = "toolStripButton2";
this.btnHapus.Click                 +=                 new
System.EventHandler(this.btnHapus_Click);
```

Kode Sumber 4. 10 Kode sumber detail *properties* komponen tombol menghapus persegi

```
//
// regionControl
//
this.regionControl.BackColor =
System.Drawing.SystemColors.ControlDark;
this.regionControl.BackgroundImage = null;
this.regionControl.Cursor =
System.Windows.Forms.Cursors.Default;
this.regionControl.DrawingMode =
DeteksiGerakan.DrawingMode.None;
this.regionControl.Location = new
System.Drawing.Point(20, 19);
this.regionControl.Name = "regionControl";
this.regionControl.Rectangles = new
System.Drawing.Rectangle[0];
this.regionControl.Size = new
System.Drawing.Size(383, 242);
this.regionControl.TabIndex = 0;
```

Kode Sumber 4. 11 Kode sumber detail *properties* komponen memanggil class region control

```
//
// RegionForm
//
this.AutoScaleDimensions = new
System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(446, 339);
this.Controls.Add(this.btnOK);
this.Controls.Add(this.btnCancel);
this.Controls.Add(this.toolStrip1);
this.Controls.Add(this.groupBox1);
this.Name = "RegionForm";
```

```
14  this.ShowIcon = false;
15  this.StartPosition =
16  System.Windows.Forms.FormStartPosition.CenterScreen;
17  this.Text = "Region Area";
18  this.toolStrip1.ResumeLayout(false);
19  this.toolStrip1.PerformLayout();
20  this.groupBox1.ResumeLayout(false);
21  this.ResumeLayout(false);
22  this.PerformLayout();
23
```

Kode Sumber 4. 12 Kode sumber detail *properties* jendela region area

4.3. Implementasi Proses pada Jendela Utama

Pada subbab ini akan dibahas berbagai fungsi atau proses yang berjalan pada jendela utama sistem pendeteksi gerakan seperti pada Gambar 4.1. Seluruh proses dan fitur-fitur tambahan sistem deteksi gerakan dapat diakses pada jendela utama ini. Pemusatan akses ini dilakukan dengan tujuan untuk memudahkan pengawas keamanan mengunakan sistem ini.

4.3.1. Implementasi Proses Memuat Berkas Video

Berikut ini implementasi dari proses memuat berkas video. Memuat berkas video pada sistem pendeteksi gerakan dengan cara menekan tombol *load* pada barisan menu yang terdapat pada jendela utama. Setelah tombol *load* ditekan sistem akan menuju ke *file system* computer pengguna. Dan sistem pendeteksi gerakan akan memuat berkas video yang telah disediakan oleh pengguna terlebih dahulu setelah pengguna memilih video yang ingin dideteksi dan menekan tombol *open*, berkas video akan dimuat ke sistem seperti pada Kode Sumber 4.13. Sedangkan Kode Sumber 4.14 merupakan kode sumber untuk membuka video ke layar.

```
1  private void btn_Load_Click(object sender, EventArgs
2  e)
3  {
4          if (openFileDialog.ShowDialog() ==
5  DialogResult.OK)
```

```
6           {
7                   //bikin video source
8                   FileVideoSource fileSource = new
9   FileVideoSource(openFileDialog.FileName);
10
11                  //buka
12                  OpenVideoSource(fileSource);
13          }
14  }
```

Kode Sumber 4. 13 Kode sumber memuat berkas video

```
1   private void OpenVideoSource(IVideoSource source)
2   {
3           this.Cursor = Cursors.WaitCursor;
4
5           CloseVideoSource();
6
7           videoSourcePlayer.VideoSource = new
8   AsyncVideoSource(source);
9           videoSourcePlayer.Start();
10
11          timer.Start();
12          watch.Start();
13          videoSourcePlayer.PlayingFinished += new
14  PlayingFinishedEventHandler(videoSourcePlayer_Playin
15  gFinished);
16          alarmTimer.Start();
17
18          videoSource = source;
19
20          this.Cursor = Cursors.Default;
21  }
```

Kode Sumber 4. 14 Kode sumber membuka video ke layar

4.3.2. Implementasi Proses Memuat Perangkat Kamera

Berikut ini implementasi dari proses memuat perangkat kamera. Sistem akan menyaring perangkat kamera yang telah terhubung dengan sistem pendeteksi gerakan, daftar kamera tersebut akan ditampilkan pada *dropdown list* yang terdapat pada barisan menu jendela utama. Memuat perangkat kamera pada sistem pendeteksi gerakan dengan cara memilih daftar kamera yang ada pada *dropdown list* dan menekan tombol *open*. Kode

Sumber 4.15 merupakan kode sumber untuk menampilkan perangkat kamera yang tersedia atau telah terhubung ke sistem.

```
Try
{
        videoDevices = new
FilterInfoCollection(FilterCategory.VideoInputDevice
);

        if (videoDevices.Count == 0)
                throw new ApplicationException();

        //add semua device camera ke combobox
        foreach (FilterInfo device in videoDevices)
        {

        comboBox_Camera.Items.Add(device.Name);
        }

}
catch (ApplicationException)
{
        comboBox_Camera.Items.Add("Tidak Ada Camera
Yang Tersambung");
        comboBox_Camera.Enabled = false;
        btn_Open.Enabled = false;
}

comboBox_Camera.SelectedIndex = 0;
```

Kode Sumber 4. 15 Implementasi Menampilkan Perangkat Kamera

Setelah memilih perangkat kamera dan tombol *open* ditekan sistem akan menjalankan kamera secara *realtime* seperti pada Kode Sumber 4.16.

```
private void btn_Open_Click(object sender, EventArgs
e)
{
        VideoCaptureDevice device = new
VideoCaptureDevice(videoDevices[comboBox_Camera.Sele
ctedIndex].MonikerString);
        OpenVideoSource(device);
}
```

Kode Sumber 4. 16 Implementasi memuat perangkat kamera

Sedangkan untuk implementasi membuka video tangkapan ke layar sama halnya pada Kode Sumber 4.14 pada pembahasan implementasi proses memuat berkas video.

4.3.3. Implementasi Proses Deteksi Gerakan

Berikut ini implementasi dari proses deteksi gerakan. Proses ini berlangsung secara otomatis ketika pengguna telah memuat berkas video ataupun berkas dari perangkat kamera dan pengguna memilih mode deteksi. Proses deteksi gerakan mendeteksi apakah terjadi perubahan gerakan terhadap video yang ingin dianalisis. Proses ini memanfaatkan *framework* AForge .NET. Kode Sumber 4.17 merupakan implementasi untuk melakukan pemanggilan algoritma *motion detection* untuk permintaan deteksi gerakan pada video. Sedangkan Kode Sumber 4.18 merupakan kode sumber untuk memanggil algoritma *motion processing*.

```
1   private void
2   SetMotionDetectionAlgorithm(IMotionDetector
3   detectionAlgorithm)
4   {
5           lock (this)
6           {
7                   detector.MotionDetectionAlgorithm =
8   detectionAlgorithm;
9                   motionReport.Clear();
10
11                  if (detectionAlgorithm is
12  TwoFramesDifferenceDetector)
13                  {
14                          if (
15
16          (detector.MotionProcessingAlgorithm is
17  MotionBorderHighlighting) ||
18
19          (detector.MotionProcessingAlgorithm is
20  BlobCountingObjectsProcessing))
21                          {
22
23          SetMotionProcessingAlgorithm(new
24  MotionAreaHighlighting());
25                          }
26                  }
```

```
27          }
28  }
```

Kode Sumber 4. 17 Implementasi memanggil algoritma *motion detection*

```
1   private void
2   SetMotionProcessingAlgorithm(IMotionProcessing
3   processingAlgorithm)
4   {
5           lock (this)
6           {
7                   detector.MotionProcessingAlgorithm =
8   processingAlgorithm;
9           }
10  }
```

Kode Sumber 4. 18 Implementasi memanggil algoritma *motion processing*

Proses memanggil algoritma untuk deteksi ini berjalan ketika mode deteksi dipilih oleh penguna. Dan pengecekan objek bergerak pada layar oleh sistem, jika terdapat objek maka akan terjadi perubahan pada layar dan dianggap sebuah gerakan yang terdeteksi. Untuk menjalankan mode deteksi implementasinya seperti pada Kode Sumber 4.19. Sedangkan Kode Sumber 4.20 merupakan kode sumber untuk mengecek objek bergerak pada layar.

```
1   private void
2   radioMotion_Detect_CheckedChanged(object sender,
3   EventArgs e)
4   {
5           SetMotionDetectionAlgorithm(new
6   TwoFramesDifferenceDetector());
7           SetMotionProcessingAlgorithm(new
8   MotionAreaHighlighting());
9   }
```

Kode Sumber 4. 19 Implementasi menjalankan mode deteksi

```
1   private void videoSourcePlayer_NewFrame(object
2   sender, ref Bitmap image)
3   {
4           lock (this)
5           {
6                   if (detector != null)
7                   {
8                           float motionLevel =
9   detector.ProcessFrame(image);
10
11  //ketika motion terdeteksi
12                          if (motionLevel >
13  motionAlarmLevel && udahnemunih == false)
14                          {.....}
15
16                          if (motionLevel <
17  motionAlarmLevel && udahnemunih == true)
18                          {.....}
19
20  // pengecekan objek
21                          if
22  (detector.MotionProcessingAlgorithm is
23  BlobCountingObjectsProcessing)
24                          {
25
26          BlobCountingObjectsProcessing
27  countingDetector =
28  (BlobCountingObjectsProcessing)detector.MotionProces
29  singAlgorithm;
30                                  detectedObjectsCount =
31  countingDetector.ObjectsCount;
32                          }
33                          else
34                          {
35                                  detectedObjectsCount =
36  -1;
37                          }
38                  }
39          }
40  }
```

Kode Sumber 4. 20 Implementasi mengecek objek bergerak pada layar

4.3.4. Implementasi Proses Membuat Region Area

Berikut ini implementasi dari proses membuat region area. Membuat region area pada sistem pendeteksi gerakan dengan

menekan tombol *Cread* yang terdapat pada daftar menu di jendela utama. Setelah tombol ditekan maka sistem akan memanggil jendela region area untuk membuat batasan wilayah deteksi. Untuk memanggil jendela region area implementasinya seperti pada Kode Sumber 4.21.

```
1   private void button1_Click(object sender, EventArgs
2   e)
3   {
4           if (videoSourcePlayer.VideoSource != null)
5           {
6                   Bitmap currentVideoFrame =
7   videoSourcePlayer.GetCurrentVideoFrame();
8
9                   if (currentVideoFrame != null)
10                  {
11                          RegionForm form = new
12  RegionForm();
13                          form.VideoFrame =
14  currentVideoFrame;
15                          form.MotionRectangles =
16  detector.MotionZones;
17
18                          if (form.ShowDialog(this) ==
19  DialogResult.OK)
20                          {
21                                  Rectangle[] rects =
22  form.MotionRectangles;
23
24                                  if (rects.Length == 0)
25                                          rects = null;
26
27                                  detector.MotionZones =
28  rects;
29                          }
30                          return;
31                  }
32          }
33          MessageBox.Show("Tentukan video source untuk
34  pendeteksian. Pilih dari sumber deteksi (berkas
35  video atau perangkat kamera) terlebih dahulu untuk
36  menentukan motion region area.",
37                  "Message", MessageBoxButtons.OK,
38  MessageBoxIcon.Information);
39  }
```

Kode Sumber 4. 21 Implementasi Membuat Region Area

Proses membuat region area pada jendela utama hanya sebatas untuk membuka jendela region area. Dan untuk membuat batasan wilayah deteksi dilakukan pada implementasi proses jendela region area.

4.3.5. Implementasi Proses Menampilkan Informasi Hasil Deteksi

Berikut ini implementasi dari proses menampilkan cataan hasil deteksi. Proses ini berjalan berdasarkan hasil dari deteksi gerakan. Informasi hasil deteksi ditentukan terhadap lama waktu terjadinya deteksi dari video yang dipantau. Misalkan ada video yang berdurasi 05:00 menit dan memiliki perubahan deteksi mulai menit ke-02:40 sampai dengen menit ke-04:00. Maka, sistem akan menampilkan informasi hasil deteksi "Ada Gerakan pada 2:40–4:00".

Informasi hasil deteksi ditentukan dengan cara sistem membandingkan nilai *threshold* yang ditentukan dengan nilai perubahan *pixel* yang terjadi ketika terdapat perubahan gerakan atau terdetaksi sebuah gerakan. Nilai *threshold* yang ditentukan oleh penulis adalah 0.015.

Untuk menentukan informasi hasil deteksi perbandingan nilai *threshold* terhadap nilai perubahan *pixel*, jika nilai perubahan *pixel* lebih besar dari pada nilai *threshold* maka akan disimpan sebagai waktu awal terjadi gerakan. Dan jika nilai perubahan *pixel* lebih kecil dari pada nilai *threshold* maka gerakan berhenti dan waktu akhir disimpan. Kode Sumber 4.22 merupakan kode untuk menampilkan informasi hasil deteksi.

```
1  {....}
2  if (detector != null)
3  {
4          float motionLevel =
5  detector.ProcessFrame(image);
6
7          //ketika motion terdeteksi
8
```

```
9           if (motionLevel > motionAlarmLevel &&
10  udahnemunih == false)
11          {
12                  udahnemunih = true;
13                  detikAwal = (int)
14  watch.Elapsed.TotalSeconds;
15          }
16          if (motionLevel < motionAlarmLevel &&
17  udahnemunih == true)
18          {
19                  int detikAkhir = (int)
20  watch.Elapsed.TotalSeconds;
21
22                  if ((detikAkhir - detikAwal) > 1)
23                  {
24                          udahnemunih = false;
25
26          listReport.BeginInvoke((Action)(() =>
27                          {
28                                  int menit1 = detikAwal
29  / 60;
30                                  int detik1 = detikAwal
31  % 60;
32
33                                  int menit2 =
34  detikAkhir / 60;
35                                  int detik2 =
36  detikAkhir % 60;
37
38          listReport.Items.Add("Ada Gerakan Pada: " +
39  menit1 + ":" + detik1 + "  --  " + menit2 + ":" +
40  detik2);
41                          })));
42                  }
43          }
44  {....}
```

Kode Sumber 4. 22 Implementasi Menampilkan Informasi Hasil Deteksi

Saat Pertama video dijalankan, secata bersamaan sistem akan menjalankan waktu dalam satuan detik. Jika terjadi perubahan pergerakan pada video seperti yang dijelaskan di atas sebelumnya, sistem akan menyimpan waktu awal perubahan terjadi dan menyimpan waktu akhir perubahan terjadi.

4.4. Implementasi Proses pada Jendela Region Area

Pada subbab ini akan dibahas berbagai fungsi atau proses yang berjalan pada jendela region area sistem pendeteksi gerakan seperti pada Gambar 4.2. Terdapat dua implementasi pada subbab ini yakni implementasi proses menggambar persegi untuk region area deteksi dan implementasi proses menghapus persegi yang telah digambarkan. Instruksi yang mengatur kedua proses tersebut terdapat pada implementasi kelas *RegionControl*.

Ketika pertama kali jendela region area terbuka, sistem akan membuat gambar latar yang menjadi tempat mengambar persegi sebagai batas wilayah deteksi. Gambar latar tersebut diambil dari tangkapan video yang sedang diputar pada layar utama. Proses ini dilakukan untuk memudahkan pengguna menetapkan dimana batasan wilayah yang diinginkan. Kode Sumber 4.23 merupakan kode sumber ketika jendela region area pertama dibuka untuk membuat gambar latar.

```
1   protected override void OnLoad(EventArgs e)
2   {
3           if (regionControl.BackgroundImage != null)
4           {
5                   int imageWidth =
6   regionControl.BackgroundImage.Width;
7                   int imageHeight =
8   regionControl.BackgroundImage.Height;
9
10                  regionControl.Size = new
11  Size(imageWidth + 2, imageHeight + 2);
12
13                  this.Size = new Size(imageWidth + 2 +
14  26, imageHeight + 2 + 118);
15          }
16
17          base.OnLoad(e);
18  }
```

Kode Sumber 4. 23 Implementasi Menggambar Persegi

4.4.1. Implementasi Proses Menggambar Persegi

Berikut ini merupakan implementasi dari proses menggambar persegi yang bertujuan untuk menentukan batasan wilayah deteksi. Menggambar persegi pada sistem pendeteksi gerakan dengan memilih tombol persegi yang terdapat pada *bar* di jendela region area. Kode Sumber 4.24 merupakan kode sumber untuk menggambar persegi.

```
1   private void bntGambar_Click(object sender,
2   EventArgs e)
3   {
4           DrawingMode currentMode =
5   regionControl.DrawingMode;
6
7           // ganti mode
8           currentMode = (currentMode ==
9   DrawingMode.Rectangular) ? DrawingMode.None :
10  DrawingMode.Rectangular;
11          // perbaharui mode
12          regionControl.DrawingMode = currentMode;
13          // ganti status tombol
14          bntGambar.Checked = (currentMode ==
15  DrawingMode.Rectangular);
16  }
```

Kode Sumber 4. 24 Implementasi Menggambar Persegi

4.4.2. Implementasi Proses Menghapus Persegi

Berikut ini merupakan implementasi dari proses menghapus persegi yang telah digambarkan sebelumnya dimana persegi tersebut bertujuan untuk menentukan batasan wilayah deteksi. Menghapus persegi pada sistem pendeteksi gerakan dengan cara pengguna cukup menekan tombol penghapus yang terdapat pada *bar* di jendela region area. Dan sistem akan menghapus semua persegi yang ada pada layar. Kode Sumber 4.25 merupakan kode sumber untuk menggambar persegi.

```
1  private void btnHapus_Click(object sender,
2  EventArgs e)
3  {
4          regionControl.RemoveAllRegions();
5  }
6
```

Kode Sumber 4. 25 Implementasi Menghapus Persegi

Perintah ini akan menghapus semua persegi yang telah ditentukan oleh pengguna.

4.4.3. Implementasi Kelas RegionControl

Berikut ini implementasi dari kelas RegionControl. Kelas RegionControl merupakan kelas yang mengatur instruksi kedua proses proses yang terdapat pada jendela region area yaitu proses menggambar persegi dan proses menghapus persegi.

Instruksi yang diatur oleh kelas ini meliputi instruksi untuk membuat persegi mulai dari koordinat posisi awal gambar persegi mulai dibuat, instruksi ketika *mouse* bergeser untuk membentuk persegi hingga persegi tergambar dan juga instruksi untuk menghapus persegi yang telah dibuat. Berikut ini merupakan implementasi pada kelas RegionControl. Kode Sumber 4.26 merupakan kode sumber untuk normalisasi titik koordinat.

```
1  private void NormalizePoints(ref Point point1, ref
2  Point point2)
3  {
4          Point t1 = point1;
5          Point t2 = point2;
6
7          point1.X = Math.Min(t1.X, t2.X);
8          point1.Y = Math.Min(t1.Y, t2.Y);
9          point2.X = Math.Max(t1.X, t2.X);
10         point2.Y = Math.Max(t1.Y, t2.Y);
11 }
```

Kode Sumber 4. 26 Menormalisasi koordinat *Point*

Normalisasi ini bertujuan untuk menginstruksikan titik koordinat ketika mulai menggambar persegi hingga persegi selesai digambar. Dan untuk menetapkan titik koordinat pertama (x, y)

harus sekecil mungkin supaya titik (x, y) pertama membentuk sudut persegi yang digambar.

Kode Sumber 4.27 merupakan kode sumber untuk memastikan titik koordinat berada pada wilayah menggambar.

```
1   private void CheckPointsInClient(ref Point point)
2   {
3           if (point.X < 1)
4           {
5                   point.X = 1;
6           }
7           if (point.Y < 1)
8           {
9                   point.Y = 1;
10          }
11          if (point.X >= ClientRectangle.Width - 1)
12          {
13                  point.X = ClientRectangle.Width - 2;
14          }
15          if (point.Y >= ClientRectangle.Height - 1)
16          {
17                  point.Y = ClientRectangle.Height - 2;
18          }
19  }
```

Kode Sumber 4. 27 Memastikan Titik Koordinat pada Wilayah Menggambar

Instruksi ini bertujuan untuk memastikan jika titik-titik koordinat berada pada wilayah tempat pengguna menggambar persegi.

Kode Sumber 4.28 merupakan kode sumber untuk menggambar persegi.

```
1   protected override void OnPaint(PaintEventArgs pe)
2   {
3           Graphics g = pe.Graphics;
4           Rectangle rect = this.ClientRectangle;
5
6           using (Pen pen = new Pen(borderColor, 1))
7           {
8                   g.DrawRectangle(pen, rect.X, rect.Y,
9   rect.Width - 1, rect.Height - 1);
10          }
```

```
11
12          if (backImage != null)
13          {
14                  g.DrawImage(backImage, 1, 1,
15  rect.Width - 2, rect.Height - 2);
16          }
17          else
18          {
19                  using (Brush backBrush = new
20  SolidBrush(backColor))
21                  {
22                          g.FillRectangle(backBrush, 1,
23  1, rect.Width - 2, rect.Height - 2);
24                  }
25          }
26          using (Pen pen = new Pen(rectsColor, 1))
27          {
28                  foreach (Rectangle r in rectangles)
29                  {
30                          g.DrawRectangle(pen, r.X + 1,
31  r.Y + 1, r.Width - 1, r.Height - 1);
32                  }
33          }
34          base.OnPaint(pe);
35  }
```

Kode Sumber 4. 28 Instruksi Menggambar Persegi

Kode Sumber 4.29 merupakan kode sumber untuk menggambar persegi ketika *mouse* ditekan. Sedangkan Kode Sumber 4.30 merupakan kode sumber ketika *mouse* bergeser untuk membentuk persegi pada layar menggambar. Dan Kode Sumber 4.31 merupakan kode sumber ketika *mouse* dilepas setelah menggambar persegi.

```
1   protected override void OnMouseDown(MouseEventArgs
2   e)
3   {
4           if (drawingMode == DrawingMode.Rectangular)
5           {
6                   if (e.Button == MouseButtons.Left)
7                   {
8                           dragging = true;
9                           this.Capture = true;
10
11
```

```
12                          startPoint.X = endPoint.X =
13  e.X;
14                          startPoint.Y = endPoint.Y =
15  e.Y;
16
17
18          ControlPaint.DrawReversibleFrame(new
19  Rectangle(e.X, e.Y, 1, 1), Color.Green,
20  FrameStyle.Dashed);
21                  }
22                  else if (e.Button ==
23  MouseButtons.Right)
24                  {
25                  }
26          }
    }
```

Kode Sumber 4. 29 Instruksi Ketika Mouse Ditekan

```
1   protected override void OnMouseMove(MouseEventArgs
2   e)
3   {
4           if (dragging == true)
5           {
6                   DrawSelectionRectangle();
7
8                   endPoint.X = e.X;
9                   endPoint.Y = e.Y;
10
11                  DrawSelectionRectangle();
12          }
13  }
```

Kode Sumber 4. 30 Instruksi Ketika Mouse Bergeser

```
1   protected override void OnMouseUp(MouseEventArgs e)
2   {
3           if ((drawingMode == DrawingMode.Rectangular)
4   && (dragging == true))
5           {
6                   dragging = false;
7                   this.Capture = false;
8
9                   drawingMode = DrawingMode.None;
10                  this.Cursor = Cursors.Default;
11
12                  DrawSelectionRectangle();
13
```

```
14                  NormalizePoints(ref startPoint, ref
15  endPoint);
16
17                  CheckPointsInClient(ref startPoint);
18                  CheckPointsInClient(ref endPoint);
19
20                  Rectangle rect = new
21  Rectangle(startPoint.X - 1, startPoint.Y - 1,
22  endPoint.X - startPoint.X + 1, endPoint.Y -
23  startPoint.Y + 1);
24                  rectangles.Add(rect);
25
26                  if (OnNewRectangle != null)
27                  {
28                          OnNewRectangle(this, rect);
29                  }
30
31                  this.Invalidate();
32          }
33  }
```

Kode Sumber 4. 31 Instruksi Ketika Mouse Dilepas

Ketika pengguna selesai menggambar persegi, selanjutnya sistem akan membuat persegi pada layar untuk dijadikan batasan wilayah deteksi. Kode Sumber 4.32 merupakan kode sumber untuk membuat persegi pada layar.

```
1   private void DrawSelectionRectangle()
2   {
3           Point start = startPoint;
4           Point end = endPoint;
5
6           NormalizePoints(ref start, ref end);
7
8           CheckPointsInClient(ref start);
9           CheckPointsInClient(ref end);
10
11          Point screenStartPoint =
12  this.PointToScreen(start);
13          Point screenEndPoint =
14  this.PointToScreen(end);
15
16          ControlPaint.DrawReversibleFrame(
17                  new Rectangle(
18                          screenStartPoint.X,
19  screenStartPoint.Y,
```

```
20                          screenEndPoint.X -
21  screenStartPoint.X + 1, screenEndPoint.Y -
22  screenStartPoint.Y + 1),
23                  selectionColor, FrameStyle.Dashed);
24  }
```

Kode Sumber 4. 32 Instruksi membuat persegi pada layar

Instruksi ini dilakukan dengan cara merubah titik koordinat yang telah ditentukan pengguna ketika menggambarkan persegi ke titik koordinat pada layar latar. Kemudian sistem menggambar ulang persegi yang telah ditentukan pada layar latar.

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas rangkaian pengujian dan evaluasi terhadap sistem yang dibangun pada tugas akhir ini. Pengujian dilakukan secara menyeluruh untuk memeriksa apakah fungsi telah berjalan sesuai kebutuhan sistem. Bab ini mencakup lingkungan, skenario, hasil serta evaluasi terhadap pengujian.

5.1. Lingkungan Pengujian

Lingkungan pengujian dalam pembuatan tugas akhir ini meliputi perangkat keras dan perangkat lunak yang digunakan untuk melakukan proses pendeteksian utama dan proses pendeteksian pada region area. Perangkat keras yang digunakan berupa komputer dengan spesifikasi *notebook* HP Pavilion dv3, Intel Core i3-M330 2,13Ghz dan 4GB memory. Uji coba berjalan di atas sistem operasi Windows 8 Professional 64-bit dan pembangunan sistem yang menggunakan perangkat pengembang Microsoft Visual Studio 2010.

5.2. Skenario Pengujian

Pada subbab ini akan dibahas pengujian sistem pendeteksi gerakan untuk menguji fungsionalitas dari sistem tersebut. Pengujian didokumentasikan secara sistematis sebagai tolak ukur keberhasilan sistem.

5.2.1. Skenario Pengujian Pendeteksian Utama

Pada subbab ini akan dibahas pengujian untuk pendeteksian utama yang digunakan untuk mendeteksi gerakan dari dua sumber deteksi yaitu, deteksi dari sumber berkas video dan deteksi dari perangkat kamera serta menampilkan informasi hasil deteksi. Berkas video yang digunakan untuk deteksi adalah hasil rekaman dari kamera pengawas. Berkas video harus disiapkan terlabih

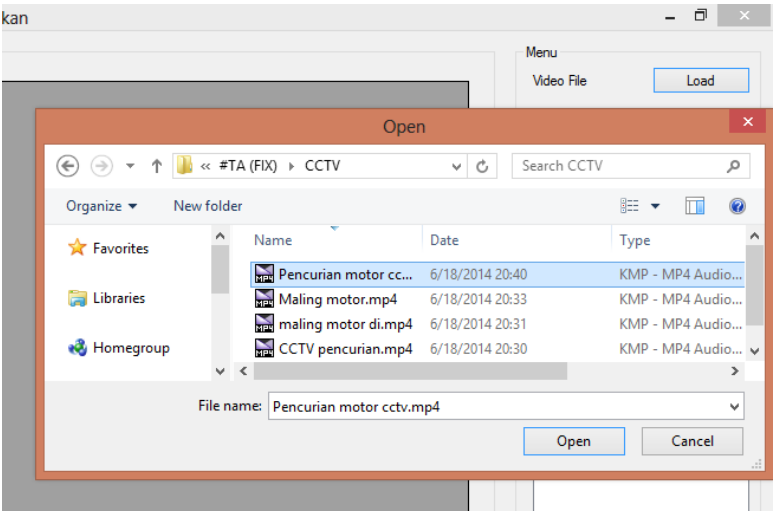
dahulu oleh penguna untuk dideteksi. Sedangkan perangkat kamera pengguna harus menyediakan perangkat tambahan berupa kamera yang dapat disambungkan ke sistem pendeteksian gerakan.

5.2.1.1. Pengujian Memuat Berkas Video

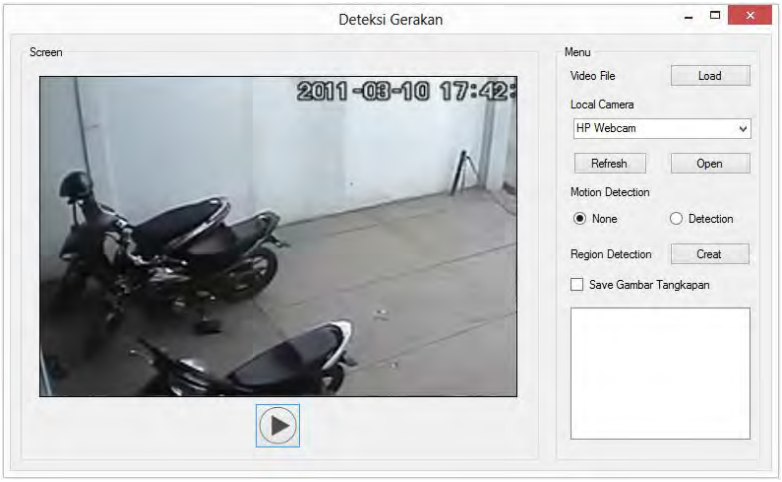
Pengujian memuat berkas video ke sistem pendeteksi gerakan dengan skenario pengawas keamanan telah masuk ke sistem dan memilih untuk memuat berkas video. Administrator memilih menu tombol memuat berkas video dan memilih berkas video yang ingin dideteksi serta memuat video ke sistem untuk kemudian dideteksi. Skenario rinci pengujian ini dijelaskan pada Tabel 5.1 dan hasil pengujian ditunjukkan pada Gambar 5.1 dan Gambar 5.2.

Tabel 5.1 Skenario Pengujian Memuat Berkas Video

Nomor	PD-01
Nama	Memuat Berkas Video.
Use Case	UC-D02
Tujuan	Memeriksa fungsi untuk memuat berkas video ke layar sistem berfungsi atau tidak.
Kondisi awal	Layar pada sistem belum terisi video.
Skenario	1. Pengawas keamanan memilih tombol menu *load* video. 2. Pengawas keamanan memilih berkas video yang ingin dideteksi. 3. Pengawas keamanan memilih tombol buka untuk memuat video.
Masukan	Berkas video
Keluaran yang diharapkan	Berkas video yang dimuat ditampilkan di layar sistem.
Hasil Pengujian	Berhasil



Gambar 5.1 Pengujian Memuat Berkas Video



Gambar 5.2 Memuat Berkas Video Berhasil

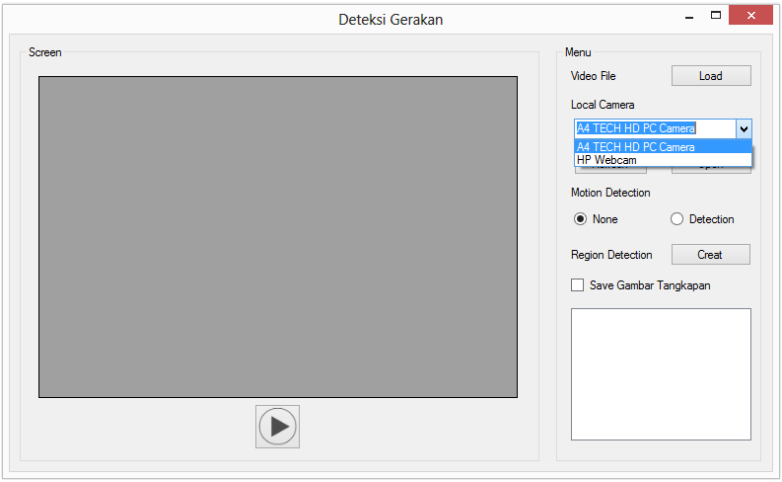
Dalam pengujian memuat berkas video, hampir semua bentuk format dari berkas video dapat dimuat ke sistem pendeteksi gerakan diantaranya .mp4, .avi, .mkv, .mov dan flv.

5.2.1.2. Pengujian Memuat Perangkat Kamera

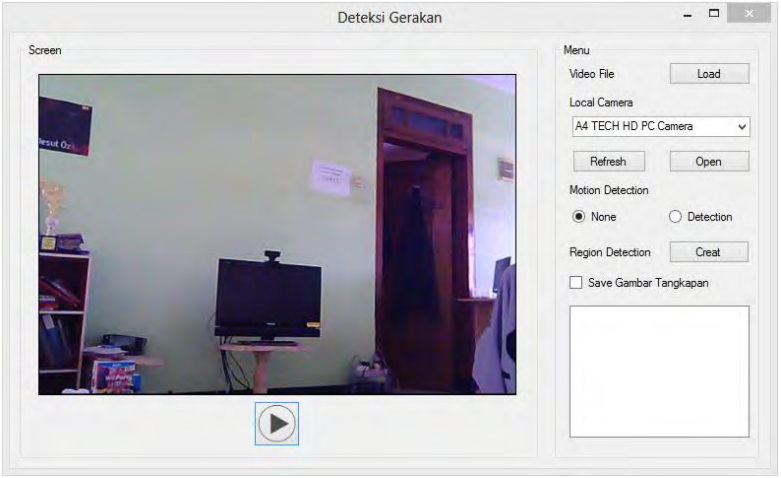
Skenario pengujian memuat perangkat kamera hampir sama dengan memuat berkas video. Hal yang membedakan adalah pengawas keamanan harus menyediakan perangkat tambahan berupa kamera yang dapat disambungkan ke sistem pendeteksi gerakan. Skenario rinci pengujian ini dijelaskan pada Tabel 5.2 dan hasil pengujian ditunjukkan pada Gambar 5.3 dan Gambar 5.4.

Tabel 5.2 Skenario Memuat Perangkat Kamera

Nomor	PD-02
Nama	Memuat Perangkat Kamera.
Use Case	UC-D03
Tujuan	Memeriksa fungsi untuk memuat perangkat kamera ke sistem berfungsi atau tidak.
Kondisi awal	Layar pada sistem belum terisi video.
Skenario	1. Pengawas keamanan memilih tombol menu *open* perangkat kamera. 2. Pengawas keamanan memilih perangkat kamera yang tersedia. 3. Pengawas keamanan memilih tombol buka untuk memuat tangkapan kamera ke layar.
Masukan	Tangkapan perangkat kamera
Keluaran yang diharapkan	Hasil tangkapan kamera berhasil ditampilkan di layar sistem
Hasil Pengujian	Berhasil



Gambar 5.3 Pengujian Memuat Perangkat Kamera



Gambar 5.4 Memuat Perangkat Kamera Berhasil

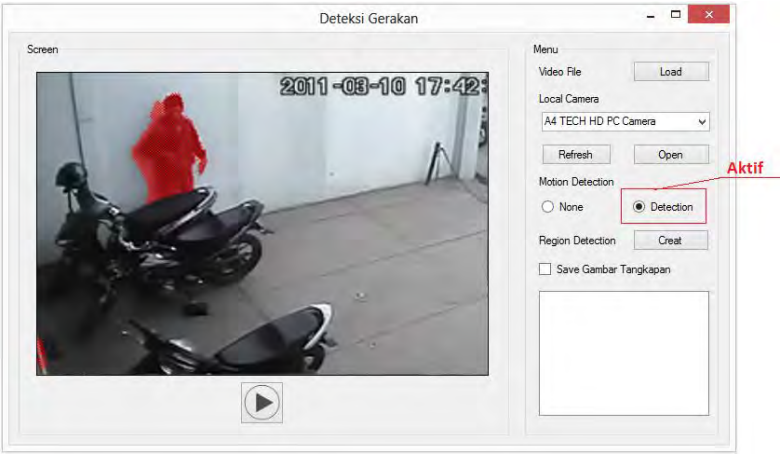
5.2.1.3. Pengujian Deteksi Gerakan

Berikut ini pembahasan pengujian deteksi gerakan. Pengujian deteksi gerakan dapat dilakukan bila pengguna telah memilih salah satu sumber deteksi yang disediakan deteksi melalui berkas video ataupun deteksi melalui perangkat kamera serta video telah dimuat ke layar sistem. Selanjutnya pengguna mengaktiftan mode deteksi dan sistem akan memproses deteksi terhadap video.

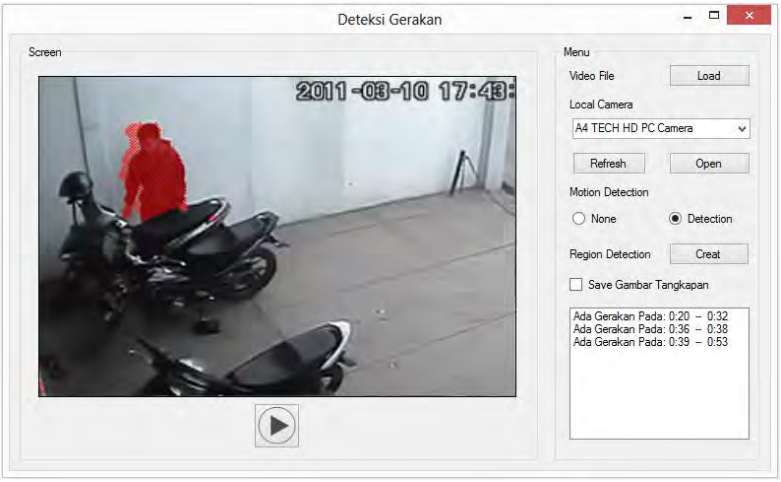
Untuk mengetahui keberhasilan sistem mendeteksi gerakan adalah dengan cara sistem akan menampilkan informasi hasil deteksi. Informasi hasil deteksi ditampilkan berdasarka rentang waktu terjadinya perubahan gerakan. Skenario rinci pengujian ini dijelaskan pada Tabel 5.3 dan hasil pengujian ditunjukkan pada Gambar 5.5 dan Gambar 5.6.

Tabel 5.3 Skenario Pengujian Deteksi Gerakan

Nomor	PD-03
Nama	Deteksi Gerakan
Use Case	UC-D01
Tujuan	Memeriksa fungsi untuk deteksi gerakan berfungsi atau tidak.
Kondisi awal	Video telah dimuat ke layar sistem.
Skenario	1. Pengawas keamanan mengaktifkan mode deteksi. 2. Pengawas keamanan menganalisis informasi hasil deteksi.
Masukan	-
Keluaran yang diharapkan	Informasi hasil deteksi ditampilkan pada *ListBox*.
Hasil Pengujian	Berhasil



Gambar 5.5 Pengujian Mendeteksi Gerakan



Gambar 5.6 Pengujian Menampilkan Informasi Hasil Deteksi

5.2.2. Skenario Pengujian Pendeteksian Region Area

Pada subbab ini akan dibahas pengujian pendeteksian pada region area. Pendeteksian region area adalah mendeteksi gerakan

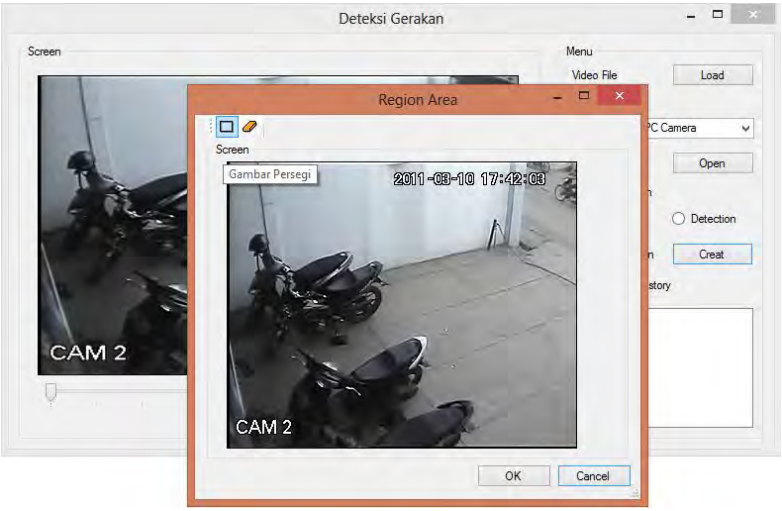
terhadap sebagian wilayah yang ditentukan saja. Melalui mode pendeteksian region area ini pengguna dapat lebih mudah untuk menganalisa perubahan gerakan pada video.

5.2.2.1. Pengujian Membuat Region Area

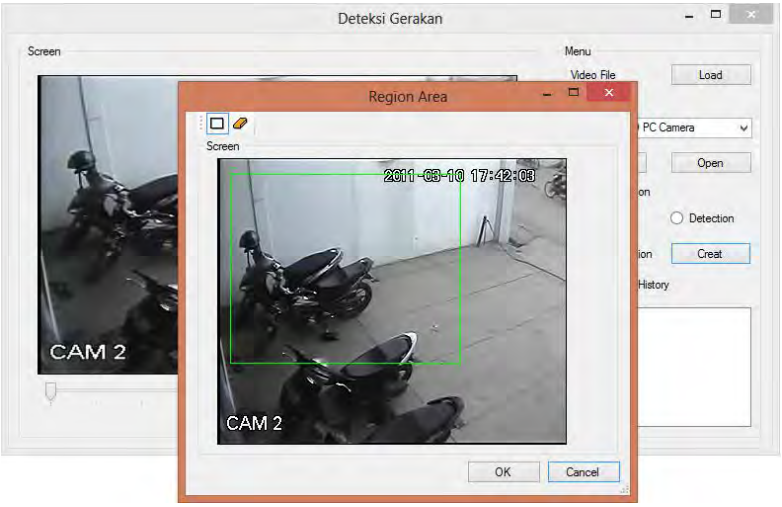
Berikut ini pembahasan pengujian membuat region area. Membuat regian area adalah menentukan batasan wilayah unutk pendeteksian dengan cara membuat persegi sebagai batasannya. Sistem hanya akan mendeteksi pergerakan yang terjadi di dalam persegi tersebut sedangkan pergerakan yang terjadi selain di dalam persegi akan diabaikan. Skenario rinci pengujian ini dijelaskan pada Tabel 5.4 dan hasil pengujian ditunjukkan pada Gambar 5.7 dan Gambar 5.8.

Tabel 5.4 Skenario Pengujian Membuat Region Area

Nomor	PA-01
Nama	Membuat Region Area
Use Case	-
Tujuan	Memeriksa fungsi untuk membuat region area berfungsi atau tidak.
Kondisi awal	Video telah dimuat ke layar sistem.
Skenario	1. Pengawas keamanan memilih tombol menu membuat region area. 2. Pengawas keamanan masuk ke jendela region area. 3. Pengawas keamanan memilih tombol menu menggambar persegi. 4. Pengawas keamanan menggambar persegi. 5. Pengawas keamanan menekan tombol *ok*.
Masukan	Menggambar persegi
Keluaran yang diharapkan	Persegi tergambar pada layar region area.
Hasil Pengujian	Berhasil



Gambar 5.7 Pengujian Membuat Region Area



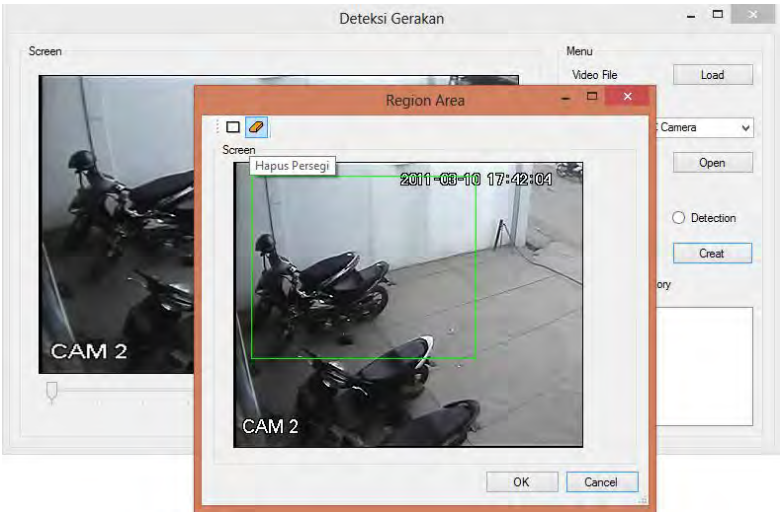
Gambar 5.8 Membuat Region Area Berhasil

5.2.2.2. Pengujian Menghapus Region Area

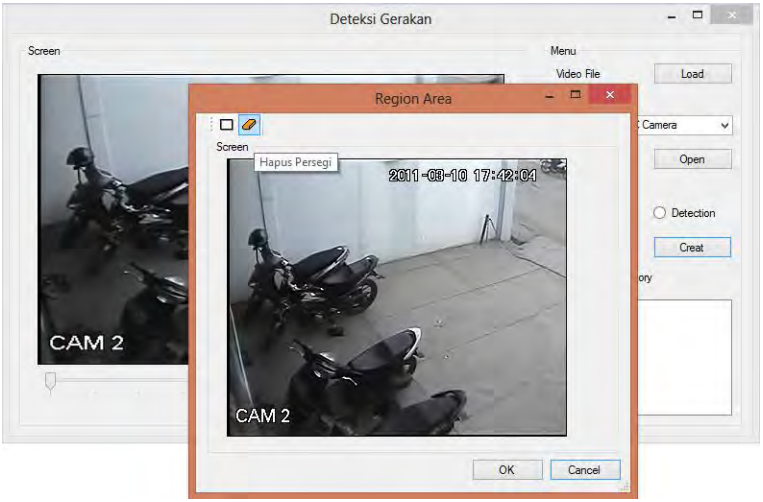
Berikut ini pembahasan pengujian menghapus region area. Jika pengguna ingin kembali mendeteksi secara keseluruhan wilayah pada video maka penguna bisa menghapus persegi yang telah digambarkan sebagai pembatas deteksi. Adapun caranya yaitu dengan kembali masuk ke jendela region area dan memilih tombol menu hapus region area. Skenario rinci pengujian ini dijelaskan pada Tabel 5.5 dan hasil pengujian ditunjukkan pada Gambar 5.9 dan Gambar 5.10.

Tabel 5.5 Skenario Menghapus Region area

Nomor	PA-02
Nama	Menghapus Region Area
Use Case	-
Tujuan	Memeriksa fungsi untuk menghapus region area berfungsi atau tidak.
Kondisi awal	Persegi sebagai region area tergambar
Skenario	1. Pengawas keamanan memilih tombol menu membuat region area. 2. Pengawas keamanan masuk ke jendela region area. 3. Pengawas keamanan memilih tombol menu hapus persegi. 4. Pengawas keamanan menekan tombol *ok.*
Masukan	-
Keluaran yang diharapkan	Perseri terhapus dari layar sistem
Hasil Pengujian	Berhasil



Gambar 5.9 Pengujian Menghapus Region Area



Gambar 5.10 Menghapus Region Area Berhasil

5.2.2.3. Pengujian Deteksi Gerakan pada Region Area

Berikut ini pembahasan pengujian deteksi gerakan pada region area yang ditentukan. Pendeteksian ini dapat dilakukan setelah pengguna membuat batasan wilayah untuk deteksi. Gerakan yang terjadi di luar batasan wilayah deteksi yang telah ditentukan akan diabaikan atau tidak dianggap sebagai gerakan. Dan sebaliknya apabila gerakan terjadi di dalam wilayah batasan yang telah ditentukan maka akan ditetapkan sebagai gerakan dan sistem akan menampilkan informasi hasil deteksi yang terjadi pada batasan wilayah tersebut. Hasil pengujian ditunjukkan pada Gambar 5.11, Gambar 5.12 dan Gambar 5.13.

Gambar 5.11 Memilih Wilayah Deteksi

Gambar 5.12 Objek Bergerak Tidak pada Wilayah Deteksi

Gambar 5.13 Objek Bergerak Berada pada Wilayah Deteksi

Untuk menentukan nilai ambang batas atau nilai *threshold*. Hasil uji coba untuk menentukan nilai *threshold* yang akan digunakan terdapat pada Tabel 5.6.

Tabel 5.6 Hasil Uji Coba Menentukan Nilai *Threshold*

Threshold	Terdeteksi / Tidak	Keterangan
1	Tidak	Tidak terdeteksi adanya pergerakan karena `motionLevel` kurang dari 1.
0.5	Tidak	Tidak terdeteksi adanya pergerakan karena `motionLevel` kurang dari 0.5.
0.1	Tidak	Tidak terdeteksi adanya pergerakan karena `motionLevel` kurang dari 0.1.
0.05	Terdeteksi	terdeteksi adanya pergerakan karena `motionLevel` lebih dari 0.05
0.02	Terdeteksi	terdeteksi adanya pergerakan karena `motionLevel` lebih dari 0.02
0.01	Terdeteksi	terdeteksi adanya pergerakan karena `motionLevel` lebih dari 0.01
0.001	Terdeteksi	terdeteksi adanya pergerakan karena `motionLevel` lebih dari 0.001
0.005	Terdeteksi	terdeteksi adanya pergerakan karena `motionLevel` lebih dari 0.005

Nilai threshold dapat mempengaruhi gerakan yang terjadi dimana semakin kecil nilai threshold yang digunakan maka perbedaan pixel yang ditangkap semakin besar. Jika perbedaan nilai pixel yang ditangkap besar, maka akan terdeteksi sebagai gerakan.

Dari hasil uji coba pada Table 5.6 dapat disimpulkan bahwa pergerakan akan terdeteksi jika nilai `motionLevel` lebih tinggi dari nilai *threshold*. Gerakan dapat terdeteksi jika nilai *threshold* berkisar antada 0.05 sampai dengan 0.005, tetapi nilai yang tepat untuk menentukan gerakan adalah 0.01 sampai dengan 0.02. karena jika lebih besar dari 0.02 hanya pergerakan yang lebih besar saja yang terdeteksi. Dan jika lebih kecil dari pada 0.01 pergerakan yang lebih kecil juga akan terdeteksi seperti gerakan yang disebabkan oleh angin secara alami.

$$Threshold = \frac{0.01 + 0.02}{2} = 0.015$$

Maka dapat ditentukan nilai *threshold* yang tepat untuk mendeteksi gerakan adalah 0.015.

5.2.3. Pengujian Akurasi Deteksi Gerakan

Hasil pengujian yang didapat ditentukan dari perhitungan *precision* dan *recall*. Secara umum perumusan *precision* dan *recall* ditunjukkan seperti pada Tabel 5.7.

Tabel 5.7 Perumusan *Precision* dan *Recall*

		Nilai Sebenarnya	
		TRUE	FALSE
Nilai Prediksi	TRUE	TP (True Positive)	FP (False Positive)
	FALSE	FN (False Negative)	TN (True Negative)

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$accurasy = \frac{TP + TN}{TP + TN + FP + FN}$$

Rumus untuk menghitung *precision* dan *recall* adalah sebagai berikut.

$$precision = \frac{jml\ deteksi\ sama}{jml\ deteksi\ sistem} \times 100\%$$

$$recall = \frac{jml\ deteksi\ sama}{jml\ deteksi\ seharusnya} \times 100\%$$

Precision merupakan kemampuan sistem untuk tidak mendeteksi gerakan yang tidak relevan. Sedangkan *recall* berhubungan dengan kemampuan sistem untuk mendeteksi pergerakan yang relevan.

Hasil uji coba untuk perhitungan *precision* deteksi gerakan pada sistem terdapat pada Tabel 5.7.

Tabel 5.8 Hasil Prehitungan Nilai *Precision*

File Video	Durasi	Jumlah Deteksi Sistem	Deteksi Sama	Precision
Filecoba.avi	00:59	4	4	100%
Curanmor.avi	03:01	11	10	90.9%
FILE0003.avi	30:59	62	61	98.4%
FILE0005.avi	33:53	50	50	100%
FILE0006.avi	30:00	85	83	97.6%
Rata-rata		212	208	**98.1%**

Hasil uji coba untuk perhitungan *precision* deteksi gerakan pada sistem terdapat pada Tabel 5.8.

Tabel 5.9 Hasil Perhitungan Nilai *recall*

File Video	Durasi	Deteksi Seharusnya	Deteksi Sama	recall
Filecoba.avi	00:59	4	4	100%
Curanmor.avi	03:01	10	10	100%
FILE0003.avi	30:59	61	61	100%
FILE0005.avi	33:53	50	50	100%
FILE0006.avi	30:00	83	83	100%
Rata-rata		208	208	**100%**

Tabel 5.7 dan Tabel 5.8 menunjukkan hasil perhitungan didapat untuk *precision* adalah 98.1% dan *recall* adalah 100%. Dengan demikian dapat dihitung akurasi deteksi sebagai berikut.

$$akurasi = \frac{TP + TN}{TP + TN + FP + FN}$$
$$= \frac{208 + 0}{208 + 0 + 4 + 0} = 98.1\%$$

Hasil perhitungan menunjukkan sistem mendeteksi gerakan dengan tingkat akurasi 98.1%.

[Halaman ini sengaja dikosongkan]

BAB VI
PENUTUP

Pada bab ini akan dijelaskan kesimpulan yang dapat diambil dalam pengerjaan tugas akhir dan saran tentang kemungkinan pengembangan yang dapat dilakukan pada tugas akhir ini di masa yang akan datang.

6.1. Kesimpulan

Dari proses analisa, perancangan, implementasi dan pengujian sistem pendeteksi gerakan dapat ditarik kesimpulan sebagai berikut.

1. Deteksi gerakan dilakukan dengan membandingkan intensitas gambar sebelumnya (*background*) dan intensitas gambar yang sekarang (*foreground*). Jika perbedaan intensitas gambar lebih besar dibandingkan nilai *threshold* yang ditentukan maka akan dideteksi sebagai gerakan.
2. Informasi gerakan dapat ditampilkan dengan mencetak rentang waktu terjadinya gerakan pada *listBox* sehingga pengguna mengetahui jika terdapat gerakan dan dapat melakukan validasi deteksi gerakan tersebut pada hasil tangkapan video.
3. Sistem menganalisis pengecualian perubahan gerakan kecil yang terjadi secara alami dengan cara menentukan nilai *threshold* atau nilai ambang batas. Nilai yang tepat untuk menentukan gerakan adalah 0.01 - 0.02 karena jika nilai *threshold* yang digunakan lebih besar dari 0.02 maka hanya pergerakan yang lebih besar saja yang terdeteksi. Jika nilai *threshold* yang digunakan kurang dari 0.01, maka pergerakan yang lebih kecil juga akan terdeteksi seperti gerakan yang disebabkan oleh angin

secara alami. Maka dengan nilai *threshold* 0.015 yang ditentukan penulis, pergerakan kecil yang terjadi secara alami tidak akan terdeteksi.

4. Sistem yang dibuat mampu memilih sebagian area yang membutuhkan pendeteksian dari keseluruhan kawasan deteksi dengan cara menggambar persegi sebagai pembatas area yang membutuhkan pendeteksian. Fitur ini memungkinkan sistem untuk mampu memusatkan deteksi pada area tertentu.

6.2. Saran

Berikut ini beberapa saran untuk kemungkinan pengembangan sistem pendeteksi gerakan di masa mendatang berdasarkan hasil rancangan, implementasi dan uji coba yang telah dilakukan.

1. Sistem pendeteksi gerakan ini dikembangkan melalui .NET Framework berbasis desktop. .NET Framework juga mendukung pembangunan aplikasi berbasis *web*. Karenanya dimungkinkan untuk mengembangkan perangkat lunak ini ke sistem berbasis *web* sehingga pengguna dapat mengakses sistem kapan saja dan dimana saja.
2. Sistem pendeteksi gerakan ini belum bisa membedakan antara gerakan manusia dan selain manusia. Disarankan adanya pengembangan berupa kecerdasan buatan yang dapat mengidentifikasi perbedaan pergerakan objek manusia.
3. Sistem pendeteksi gerakan ini masih memiliki banyak kekurangan. Sistem ini masih belum bisa untuk mempercepat deteksi video dikarenakan kelemahan dari *framework* yang digunakan. Disarankan adanya pengembangan sistem dengan menggunakan *framework* ataupun metode lain.

DAFTAR PUSTAKA

[1]. Spacek, L.A., *Edge detection and motion detection.* Image and vision computing, 1986. **4**(1): p. 43-56.

[2]. Kurniawan, A., et al., "*Pengenalan Bahasa C#"*, 1st ed, 2004. Project Otak.

[3]. Microsoft, "*Overview of the .NET Framework"*, Tersedia di: http://msdn.microsoft.com/en-us/library/zw4w595w.aspx. [Dikutip 21 Juni 2014].

[4]. Aforgenet, "*AForge Framework"*, 2012. Tersedia di: http://www.aforgenet.com/aforge/framework. [Dikutip 25 Frebuari 2014].

[5]. Purwantara, R.M., R.Y. Hakkun, and Setiawardana, "*Capture Image Dengan Penanda Jari"*, 2011. Politeknik Elektronika Negri Surabaya.

[6]. Klette, R., "*Concise Computer Vision"*, 1st ed, 2014. London. Springer.

[7]. Sonka, M., V. Hlavac, and R. Boyle, "*Image processing, analysis, and machine vision"*, 4th ed, 2014. Cengage Learning.

[8]. Guckenheimer, S. and J.J. Perez, "*Software Engineering with Microsoft Visual Studio Team System (Microsoft. NET Development Series)"*, 2006. Addison-Wesley Professional.

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Penulis lahir di Aceh, 24 Desember 1991. Penulis menempuh pendidikan di SD Negeri 1 Padang Tiji, SMP YPPU Sigli, dan MA Babun Najah Banda Aceh. Penulis melanjutkan pendidikan sarjana di Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya. Selama kuliah penulis aktif dalam berbagai kegiatan dan organisasi. Penulis diberi amanah sebagai Ketua Departemen Kominfo Community Santri Scholars Ministry of Religious Affairs Institut Teknologi Sepuluh Nopember (CSS MORA ITS) pada periode kepengurusan 2011 – 2012.

Penulis dalam menyelesaikan pendidikan S1 Teknik Informatika ITS mengambil bidang minat Rekayasa Perangkat Lunak (*Software Engineering*), penulis memiliki ketertarikan dalam pengembangan aplikasi atau perangkat lunak. Bidang di luar tersebut yang menjadi ketertarikan penulis adalah wirausaha, dan *design grafis*. Penulis dapat dihubungi melalui alamat email redha24@live.com.

LAMPIRAN

Table 1 Pengamatan Deteksi Gerakan pada File Video Filecoba.avi

Kejadian ke	Pengamatan Manual	Deteksi Sistem	Deteksi Sama
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
Total	4	4	4

Table 2 Pengamatan Deteksi Gerakan pada File Video Curanmor.avi

Kejadian ke	Pengamatan Manual	Deteksi Sistem	Deteksi Sama
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	1	1	1
7	1	1	1
8	0	1	0
9	1	1	1
10	1	1	1
11	1	1	1
Total	10	11	10

Table 3 Pengamatan Deteksi Gerakan pada File Video FILE0003.avi

Kejadian ke	Pengamatan Manual	Deteksi Sistem	Deteksi Sama
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	1	1	1
7	1	1	1
8	1	1	1
9	1	1	1
10	1	1	1
11	1	1	1
12	1	1	1
13	1	1	1
14	1	1	1
15	1	1	1
16	1	1	1
17	1	1	1
18	1	1	1
19	1	1	1
20	1	1	1
21	1	1	1
22	1	1	1
23	1	1	1
24	1	1	1
25	1	1	1
26	1	1	1
27	1	1	1
28	1	1	1

29	1	1	1
30	1	1	1
31	1	1	1
32	1	1	1
33	1	1	1
34	1	1	1
35	1	1	1
36	1	1	1
37	0	1	0
38	1	1	1
39	1	1	1
40	1	1	1
41	1	1	1
42	1	1	1
43	1	1	1
44	1	1	1
45	1	1	1
46	1	1	1
47	1	1	1
48	1	1	1
49	1	1	1
50	1	1	1
51	1	1	1
52	1	1	1
53	1	1	1
54	1	1	1
55	1	1	1
56	1	1	1
57	1	1	1
58	1	1	1
59	1	1	1
60	1	1	1

61	1	1	1
62	1	1	1
Total	61	62	61

Table 4 Pengamatan Deteksi Gerakan pada File Video FILE0005.avi

Kejadian ke	Pengamatan Manual	Deteksi Sistem	Deteksi Sama
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	1	1	1
7	1	1	1
8	1	1	1
9	1	1	1
10	1	1	1
11	1	1	1
12	1	1	1
13	1	1	1
14	1	1	1
15	1	1	1
16	1	1	1
17	1	1	1
18	1	1	1
19	1	1	1
20	1	1	1
21	1	1	1
22	1	1	1
23	1	1	1
24	1	1	1

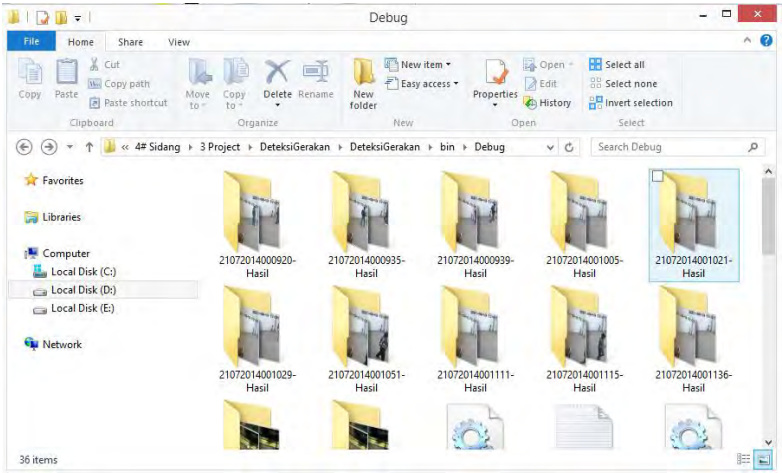
25	1	1	1
26	1	1	1
27	1	1	1
28	1	1	1
29	1	1	1
30	1	1	1
31	1	1	1
32	1	1	1
33	1	1	1
34	1	1	1
35	1	1	1
36	1	1	1
37	1	1	1
38	1	1	1
39	1	1	1
40	1	1	1
41	1	1	1
42	1	1	1
43	1	1	1
44	1	1	1
45	1	1	1
46	1	1	1
47	1	1	1
48	1	1	1
49	1	1	1
50	1	1	1
Total	50	50	50

Table 5 Pengamatan Deteksi Gerakan pada File Video FILE0006.avi

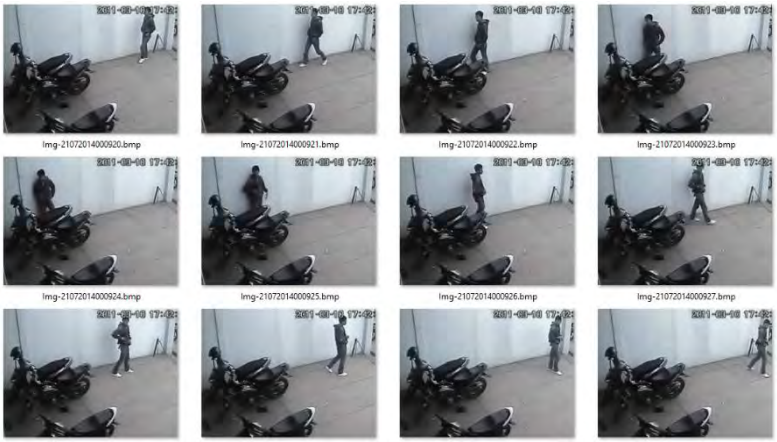
Kejadian ke	Pengamatan Manual	Deteksi Sistem	Keterangan
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	1	1	1
7	1	1	1
8	1	1	1
9	1	1	1
10	1	1	1
11	0	1	0
12	1	1	1
13	1	1	1
14	1	1	1
15	1	1	1
16	1	1	1
17	1	1	1
18	1	1	1
19	1	1	1
20	1	1	1
21	1	1	1
22	1	1	1
23	1	1	1
24	1	1	1
25	0	1	0
26	1	1	1
27	1	1	1
28	1	1	1

29	1	1	1
30	1	1	1
31	1	1	1
32	1	1	1
33	1	1	1
34	1	1	1
35	1	1	1
36	1	1	1
37	1	1	1
38	1	1	1
39	1	1	1
40	1	1	1
41	1	1	1
42	1	1	1
43	1	1	1
44	1	1	1
45	1	1	1
46	1	1	1
47	1	1	1
48	1	1	1
49	1	1	1
50	1	1	1
51	1	1	1
52	1	1	1
53	1	1	1
54	1	1	1
55	1	1	1
56	1	1	1
57	1	1	1
58	1	1	1
59	1	1	1
60	1	1	1

61	1	1	1
62	1	1	1
63	1	1	1
64	1	1	1
65	1	1	1
66	1	1	1
67	1	1	1
68	1	1	1
69	1	1	1
70	1	1	1
71	1	1	1
72	1	1	1
73	1	1	1
74	1	1	1
75	1	1	1
76	1	1	1
77	1	1	1
78	1	1	1
79	1	1	1
80	1	1	1
81	1	1	1
82	1	1	1
83	1	1	1
84	1	1	1
85	1	1	1
Total	83	85	83



Gambar 1 Hasil Tangkapan Gambar Dikelompokkan Berdasarkan Gerakan Terdeteksi



Gambar 2 Hasil Tangkapan Gambar Berdasarkan Rentang Terjadi Deteksi pada File Video Curanmor.avi

Gambar 3 Hasil Tangkapan Gambar Berdasarkan Rentang Terjadi Deteksi pada File Video Filecoba.avi