**TUGAS AKHIR – SF141501**

# Rancang Bangun Aplikasi Komputer untuk Manajemen Konten Multimedia Berbasis Webcam sebagai Sensor Gerak Menggunakan Metode *Background Subtraction*

**MUHAMMAD FAHMIL HUDA**
**NRP 1110 100 704**

**Dosen Pembimbing:**
**1. Drs. Bahtera Indarto, M.Si**
**2. Drs. Hasto Sunarno, M.Sc**

**JURUSAN FISIKA**
**Fakultas Matematika dan Ilmu Pengetahuan Alam**
**Institut Teknologi Sepuluh Nopember**
**Surabaya 2015**

*"Halaman ini sengaja dikosongkan"*

**FINAL PROJECT - SF141501**

# Design and Development of Computer Application for Multimedia Content Management Based on Webcam as Motion Sensor Using Background Subtraction

**MUHAMMAD FAHMIL HUDA**
**NRP 1110 100 704**

**Advisor:**
1. **Drs. Bahtera Indarto, M.Si**
2. **Drs. Hasto Sunarno, M.Sc**

**DEPARTMENT OF PHYSICS**
**Faculty Of Mathematics And Science**
**Sepuluh Nopember Institute of Technology**
**Surabaya 2015**

*"Halaman ini sengaja dikosongkan"*

**Rancang Bangun Aplikasi Komputer untuk Manajemen Konten Multimedia Berbasis Webcam sebagai Sensor Gerak Menggunakan Metode *Background Subtraction***

**TUGAS AKHIR**

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Sains
pada
Bidang Fisika Instrumentasi
Program Studi S-1 Jurusan Fisika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember


Oleh:
**MUHAMMAD FAHMIL HUDA**
NRP 1110 100 704


Disetujui oleh Dosen Pembimbing Tugas Akhir

**Drs. Bahtera Indarto, M.Si** (.....)
NIP. 19610404 199102.1.001

**Drs. Hasto Sunarno, M.Sc** (.....)
NIP. 19560331 198003.1.001

**SURABAYA**
**DESEMBER 2015**

*“Halaman ini sengaja dikosongkan”*

# Rancang Bangun Aplikasi Komputer untuk Manajemen Konten Multimedia Berbasis Webcam sebagai Sensor Gerak Menggunakan Metode *Background Subtraction*


| | |
|---|---|
| **Nama** | **: Muhammad Fahmil Huda** |
| **NRP** | **: 1110 100 704** |
| **Jurusan** | **: Fisika, FMIPA-ITS** |
| **Pembimbing** | **: 1. Drs. Bahtera Indarto, M.Si** |
| | **2. Drs. Hasto Sunarno, M.Sc** |



**Abstrak**

*Era computer vision telah cukup lama dimulai. Berbagai algoritma telah dikembangkan sehingga memungkinkan komputer untuk melihat layaknya manusia. Penelitian ini mencoba menerapkan computer vision ke dalam aplikasi manajemen konten multimedia yang diintegrasikan dengan webcam sebagai sensor gerak. Metode yang digunakan adalah background subtraction. Penelitian dilakukan dengan sebuah laptop dengan dukungan webcam dan GPU. Aplikasi dibuat dengan Visual Studio 2012, librari OpenCV, dan CUDA Toolkit. Analisis gerak dilakukan melalui nilai standar deviasi dan kontur hasil background subtraction.. Nilai standar deviasi hasil background subtraction adalah antara 0 – 127,5. Dalam analisis kontur, pergerakan yang direspons aplikasi adalah jika dalam rentang jarak 80 piksel objek memiliki 5-6 gambar yang terekam oleh webcam. Dari hasil percobaan, webcam mampu mendeteksi pergerakan dengan kecepatan antara 0,63 m/s sampai 1,17 m/s dalam jarak 69 cm dari objek.*



***Kata kunci: Analisis Gerak, Background Subtraction, Computer Vision, CUDA, OpenCV.***

*"Halaman ini sengaja dikosongkan"*

**Design and Development of Computer Application for Multimedia Content Management Based on Webcam as Motion Sensor Using Background Subtraction**


| | |
|---|---|
| **Name** | **: Muhammad Fahmil Huda** |
| **NRP** | **: 1110 100 704** |
| **Major** | **: Physics** |
| **Advisor** | **: 1. Drs. Bahtera Indarto, M.Si** |
| | **2. Drs. Hasto Sunarno, M.Sc** |



**Abstract**

*The era of Computer Vision has been started for a long time. Many algorithms have been developed for computer to has vision like human. This research was intended to apply computer vision to a multimedia content management application integrated with webcam as a motion sensor. The method used for this research is background subtraction which the background model is updated gradually. The research was conducted with a laptop supported by webcam and GPU. The application is developed by Visual Studio 2012, OpenCV library, and CUDA. The motion is analyzed by the standard deviation and contours of background subtraction results. The standard deviation varies from 0 – 127.5. In the contours analysis, motion will be responden by the application if in the range of 80 pixel there is object with 5-6 pictures captured by webcam. From the experiment, the webcam can detect motion with the velocity between 0.63 m/s to 1.17 m/s with the distance of 69 cm from the object.*



***Keywords: Background Subtraction, Computer Vision, CUDA, Motion Analysis, OpenCV.***

*"Halaman ini sengaja dikosongkan"*

# KATA PENGANTAR

Alhamdulillah, rasa syukur yang begitu mendalam penulis panjatkan kepada Allah Ta'ala yang telah mencurahkan rahmat, taufik, dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir ini beserta laporannya. Sholawat serta salam tak lupa penulis lantunkan kepada insan panutan sepanjang zaman, Nabi Muhammad ﷺ yang selalu menjadi motivator penulis dalam menjalani kehidupan ini.

Laporan Tugas Akhir ini penulis susun sebagai syarat menyelesaikan pendidikan strata satu Jurusan Fisika Fakultas Matematika dan Ilmu Pengetahuan Institut Teknologi Sepuluh Nopember Surabaya. Begitu banyak suka dan duka yang telah alami demi menyelesaikan pendidikan di kampus perjuangan ini. Namun, berkat dukungan dan doa banyak pihak akhirnya penulis mampu menyelesaikannya.

Oleh karena itu, penulis menyampaikan rasa terima kasih kepada pihak-pihak tersebut terutama kepada:

1. Kedua orang tua tercinta, Bapak Ali Mahfudh dan Ibu Sholichah yang telah mendidik penulis sejak kecil,
2. Mbak Himmah, Faza, Faidlur, dan segenap keluarga besar Bani Mizan, Bani Chadziq, dan Bani Asnawi yang senantiasa mengisi hari-hari bahagia penulis,
3. Segenap guru penulis di Madrasah Al-Khurriyah Besito, Madrasah TBS Kudus, dan Ponpes Darussalam Keputih yang telah membina keimanan, keislaman, dan keilmuan penulis,
4. Kementerian Agama RI, khususnya Ditjen PDPontren yang telah memberikan kesempatan bagi penulis untuk belajar di kampus ini dengan beasiswa,
5. Bapak Ahlus, Bapak Agus, Bapak Darmaji, Bapak Wahyudin, Bapak Mustain, Almarhum Abah Syakur, dan segenap pembina CSSMoRA ITS yang senantiasa memberikan rasa tenteram di hati penulis,
6. Bapak Drs. Bahtera Indarto, M.Si dan Drs. Hasto Sunarno, M.Sc yang telah membimbing penulis dalam tugas akhir, Dr.

M. Zainuri, M.Sc yang telah menjadi dosen wali, membimbing dan mengarahkan penulis selama di ITS, serta segenap dosen S1 Fisika ITS dan dosen-dosen ITS beserta jajarannya yang telah memperluas cakrawala pengetahuan dan wawasan penulis,

7. Teman-teman Fisika ITS 2010, CSSMoRA ITS, CSSMoRA Nasional, santri Ponpes Darussalam Keputih, Himasika ITS, serta EkspresiX Super Team FMIPA ITS yang telah menjadi sahabat terbaik penulis selama di ITS dan semoga seterusnya,
8. Mas Akbar dan segenap warga Keputih yang telah mewarnai kehidupan penulis selama di Surabaya,
9. Para pengunjung blog hdqbasith.blogspot.com dan segenap warga NU yang tiada henti mengamalkan ajaran ahlussunnah waljama'ah dan membuat penulis bangga menjadi bagian darinya,
10. Segenap rakyat Indonesia dan pihak-pihak yang tidak dapat penulis sebutkan satu persatu yang telah membantu dan mendoakan penulis baik secara langsung maupun tidak langsung.

Kepada pihak-pihak tersebut penulis hanya mampu mengucapkan terima kasih teriring doa semoga Allah memberikan balasan kebaikan yang berlipat ganda.

Tiada gading yang tak retak, begitu pula laporan Tugas Akhir ini pastilah banyak kekurangan dan kesalahan. Oleh karena itu, penulis memohon maaf kepada para pembaca beserta kritik dan saran demi perbaikan ke depannya. Akhir kata, semoga Laporan Tugas Akhir ini dapat bermanfaat bagi kita semua baik di dunia maupun di akhirat.

Surabaya, 21 Desember 2014

Penulis

# DAFTAR ISI

*"Halaman ini sengaja dikosongkan"*

# DAFTAR TABEL

*"Halaman ini sengaja dikosongkan"*

# DAFTAR GAMBAR

*"Halaman ini sengaja dikosongkan"*

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang Masalah

Era *computer vision* telah cukup lama dimulai. Di era *computer vision* komputer tidak lagi hanya mengabadikan gambar di dalam memori, namun lebih dari itu saat ini komputer mampu "melihat" benda layaknya manusia. Komputer mampu merepresentasikan arti sebuah gambar dan bertindak sesuai kondisi di gambar tersebut.

Berbagai bidang saat ini telah memanfaatkan teknologi *computer vision*, di antaranya adalah pembuatan peta, kamera keamanan, kamera lalu lintas, sistem inspeksi manufaktur, dan robot tak berawak. Hal ini tak lepas dari perkembangan teknologi yang begitu pesat, baik dalam perangkat keras maupun perangkat lunak.

Perkembangan *computer vision* ini tampaknya tak bisa lepas dari kemunculan OpenCV, sebuah librari *computer vision* yang dikembangkan oleh Intel. Librari ini bersifat *open-source* sehingga dapat digunakan dengan gratis oleh semua orang untuk tujuan apapun, termasuk tujuan komersial. OpenCV memiliki lebih dari 2500 fungsi *computer vision* yang dapat digunakan di berbagai sistem operasi seperti windows, linux, MacOSX, bahkan juga Android dan iOS.

OpenCV mulai dirilis pada tahun 1999. Semenjak saat itu, telah banyak orang yang menggunakannya dalam pengembangan teknologi. Salah satunya adalah robot dari Stanford, "Stanley" yang memenangkan 2 juta dolar pada *DARPA Grand Challenge desert robot race*. Karena usianya yang baru 15 tahun, kesempatan pengembangan aplikasi dengan OpenCV masih terbuka lebar. (Bradski and Kaehler, 2008)

Penelitian ini merupakan salah satu pengaplikasian OpenCV dalam bidang teknologi, yaitu untuk mengatur

pemutaran konten multimedia pada komputer. Pada penelitian ini akan dirancang sebuah aplikasi yang dapat mengatur pemutaran konten multimedia berdasarkan gerakan di depan webcam. Konten akan dijalankan, dijeda, diganti, atau dihentikan berdasarkan gerakan yang yang terekam oleh webcam.

## 1.2 Perumusan Masalah

Permasalahan yang diangkat dalam penelitian ini sebagai berikut:

1. Bagaimana membuat aplikasi manajemen konten multimedia komputer yang terintegrasi dengan webcam sebagai sensor gerak?
2. Bagaimana menganalisis gerak dari hasil *background subtraction*?
3. Bagaimana hasil analisis performa aplikasi yang dibuat?

## 1.3 Batasan Masalah

Penelitian ini memiliki batasan-batasan masalah sebagai berikut:

1. Konten multimedia yang dimaksud mencakup gambar dan audio-video;
2. Komputer dalam penelitian ini menggunakan laptop ASUS K43SV;
3. Webcam yang digunakan adalah webcam laptop ASUS K43SV dengan resolusi maksimum 0,3MP (640x480 piksel);
4. Sistem operasi yang digunakan adalah Windows.
5. Perangkat lunak yang digunakan adalah Visual Studio 2012 Profesional dengan librari OpenCV dan bahasa pemrograman Visual C++;
6. Penelitian tidak membedakan jenis ataupun bentuk objek yang bergerak.

## 1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Membuat aplikasi manajemen konten multimedia komputer yang terintegrasi dengan webcam sebagai sensor gerak.
2. Menganalisis gerak dari hasil *background subtraction*.
3. Menganalisis performa aplikasi yang dibuat.

## 1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah terciptanya sebuah sistem aplikasi Windows yang bisa digunakan untuk manajemen konten multimedia yang terintegrasi dengan webcam sebagai sensor gerak sehingga multimedia dapat diputar tanpa menyentuh komputer.

## 1.6 Sistematika penulisan

Penulisan Proposal Tugas akhir ini terdiri dari uraian singkat yang berisi gambaran umum dari penelitian ini. Bab I pendahuluan yang memuat latar belakang, perumusan masalah, tujuan penelitian, manfaat penelitian, serta sistematika penulisan. Bab II tinjauan pustaka berisi tentang dasar-dasar teori yang digunakan sebagai acuan dari penelitian, Bab III metodologi penelitian, Bab IV analisis dan pembahasannya, dan Bab V kesimpulan dan saran.

*"Halaman ini sengaja dikosongkan"*

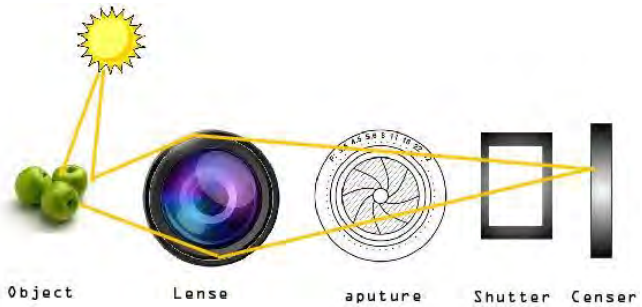
# BAB II
# TINJAUAN PUSTAKA

## 2.1 Webcam

Webcam adalah sebuah kamera video digital kecil yang dihubungkan ke komputer melalui (biasanya) port USB ataupun port COM. Dahulu kamera digital memiliki dimensi yang cukup besar dan harga yang cukup mahal. Namun, saat ini kamera digital sangat mudah ditemui pada berbagai jenis laptop, tablet, smartphone, juga ponsel. Hal ini tak lepas dari perkembangan sensor dan teknologi digital yang semakin pesat.

**Gambar 2.1 Webcam**

Gambar 2.1 di atas menunjukkan salah satu jenis webcam yang sering digunakan untuk komputer. Pada prinsipnya, tidak ada perbedaan yang mencolok antara kamera digital dan kamera analog, karena teknologi dasar yang dikandungnya sebenarnya sederhana saja. Sebuah kamera analog menggunakan film seluloid, mempunyai tiga elemen dasar, masing-masing adalah elemen optikal berupa berbagai ragam lensa, elemen kimia berupa film seluloidnya sendiri, dan elemen mekanik yang merupakan badan kamera itu sendiri. Elemen kimia pada kamera digital sekarang ini tergantikan menjadi elemen chips yang bisa berupa CCD *(Charge Coupled Device)* maupun CMOS *(Complementary Metal Oxide Semiconductor)* yang mengatur sensitivitas pencahayaan dan menjadi "film digital" pada kamera-kamera modern sekarang ini.



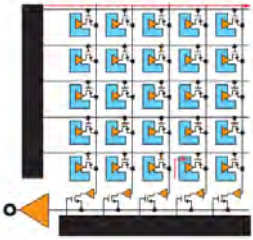

**Gambar 2.2 Proses perjalanan cahaya pada kamera**

Gambar 2.2 di atas memperlihatkan proses penangkapan gambar pada kamera. Jika cahaya mengenai suatu objek, maka objek tersebut akan memantulkannya. Pantulan cahaya ini kemudian difokuskan melalui lensa kamera. Lensa yang digunakan adalah lensa cembung, namun untuk hasil yang lebih baik biasanya ditambahkan lagi beberapa lensa di belakangnya. Banyak sedikitnya cahaya yang masuk ke dalam kamera diatur oleh *aperture* (lubang bidik). Cahaya ini diarahkan menuju sensor kamera. Sensor kamera membutuhkan waktu beberapa milidetik untuk menerima cahaya dan mengolahnya menjadi sinyal-sinyal listrik. Agar cahaya ini tidak berlebih atau kurang, maka digunakan *shutter* yang akan membuka ketika sensor siap menerima cahaya dan akan menutup setelah sensor selesai menerima cahaya yang cukup.

Sensor kamera yang digunakan dalam webcam umumnya adalah sensor CMOS. Hal ini dikarenakan CMOS membutuhkan daya yang lebih sedikit daripada CCD. Biaya produksi CMOS juga lebih rendah. Selain itu, CMOS memiliki efek *blooming* yang lebih kecil dibandingkan CCD.

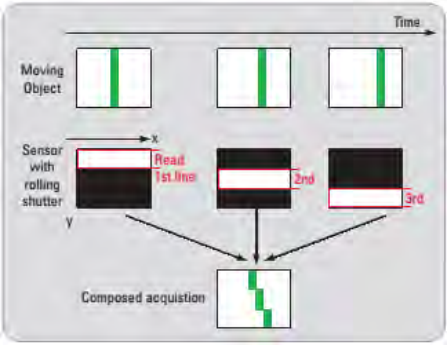
CMOS menggunakan fotosensor dioda silikon atau disebut juga piksel untuk menangkap cahaya. Jika cahaya foton dengan energi cukup mengenai sensor tersebut dan diserap di daerah deplesi, maka sebuah electron akan

dilepaskan yang dapat dideteksi sebagai adanya muatan. Jumlah muatan ini sebanding dengan intensitas cahaya yang diterima oleh sensor. Sensor ini tidak dapat membedakan warna sehingga di depan sensor dipasang filter warna.



**Gambar 2.3 Struktur Sensor CMOS**

Gambar 2.3 di atas menunjukkan struktur sensor CMOS. Pada CMOS muatan yang didapatkan di setiap piksel akan dikonversi menjadi beda potensial. Beda potensial tiap piksel diberi alamat melalui sebuah matriks dan dikuatkan dengan *amplifier* untuk setiap kolom lalu dikonversi ke data digital dengan *A/D converter*. Pembacaan data pada piksel dilakukan per kolom sehingga hal ini terkadang dapat menimbulkan gambar yang bergeser.




**Gambar 2.4 Pengambilan gambar yang bergeser pada sensor CMOS**

Gambar 2.4 adalah contoh pengambilan gambar objek yang bergeser. Garis hijau pada gambar tersebut seharusnya

berupa garis lurus vertikal, namun karena objek bergerak ketika dipotret, maka garis hijau tersebut tidak tertangkap sebagai garis lurus. Hal ini dikarenakan CMOS mengambil gambar objek per baris, tidak langsung seluruhnya.(Weber, 2012)

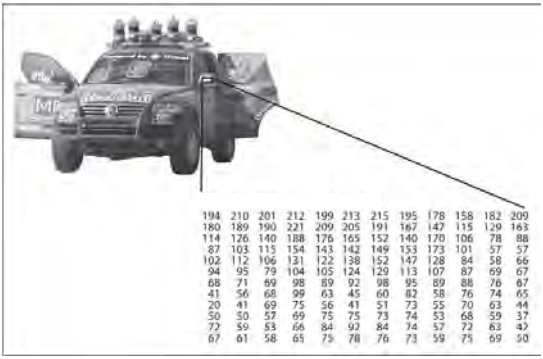
## 2.2 *Computer Vision*

*Computer Vision* (penglihatan komputer) adalah transformasi data dari gambar atau video menjadi sebuah keputusan atau representasi. Transformasi yang dilakukan bisa bermacam-macam, sementara keputusan misalnya "ada seseorang di dalam gambar ini" atau "ada 14 sel tumor di dalam film ini". Sedangkan representasi dapat berarti mengubah sebuah gambar berwarna menjadi gambar abu-abu.

Berhubung manusia adalah "makhluk visual", maka akan sangat mudah untuk dibohongi bahwa *computer vision* sangat mudah untuk dilakukan. Namun, kenyataannya *computer vision* sangat sulit dilakukan. Mengapa demikian? Hal ini dapat dilihat melalui gambar di bawah. Manusia dapat mengenali gambar di bawah sebagai gambar mobil karena informasi yang didapat dari mata dapat diolah dengan kecepatan yang luar biasa oleh otak, di mana impuls yang diterima dibandingkan dengan berbagai gambar yang ada di dalam memori otak yang diperoleh selama bertahun-tahun. Berbagai algoritma di dalam otak berjalan secara otomatis dan terintegrasi secara sempurna dengan mata sehingga manusia dapat mengenali gambar tersebut dengan mudah.

Sedangkan dalam sistem penglihatan mesin, komputer menganggap gambar sebagai kumpulan angka-angka dari kamera atau memori penyimpanan (*disk*). Selain itu, dalam komputer tidak ada pengenal pola, pengontrol fokus dan celah otomatis kamera, serta kemampuan untuk belajar dari pengalaman yang terpasang secara otomatis begitu komputer

tersebut diciptakan. Apa yang komputer lihat hanyalah angka-angka tak bermakna seperti pada gambar di bawah jika tanpa algoritma yang tepat. Algoritma yang diterapkan pada komputer untuk mengenali pola juga harus dinamis karena angka-angka yang dilihat komputer tidak akan sama 100% sekalipun gambar tersebut diambil dengan kamera yang sama dengan jeda waktu hanya milidetik. Hal inilah yang membuat *computer vision* menjadi topik riset khusus dalam beberapa dekade.



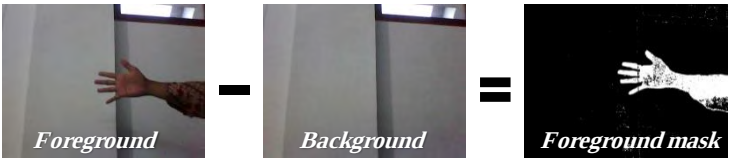

**Gambar 2.5 Interpretasi komputer terhadap gambar**

Gambar 2.5 menunjukkan bagaimana komputer melihat kumpulan warna. Warna-warna tersebut bagi komputer hanyalah kumpulan angka-angka. Angka-angka ini tidak memiliki pola yang khusus sehingga sulit untuk menginterpretasikan angka-angka tersebut sebagai gambar sebuah objek. (Bradski and Kaehler, 2008)

## 2.3 *Background subtraction*

*Background subtraction* (pengurangan gambar latar belakang) adalah salah satu metode dasar pengolahan citra dalam analisis gerak pada video. Metode ini mudah dilakukan karena hanya membandingkan 2 gambar yang diperoleh dari kamera yang diam. Langkah pertama yang

perlu dilakukan adalah menentukan gambar yang akan dijadikan latar belakang (*background*). Kemudian gambar mutakhir (*foreground*) yang diperoleh dibandingkan dengan *background* tersebut melalui pengurangan nilai warna setiap piksel pada posisi yang sama. Dengan demikian jika terdapat nilai piksel yang tidak nol berarti pada daerah tersebut terjadi pergerakan.




**Gambar 2.6 Contoh *Background Subtraction***

Gambar 2.6 menunjukkan bagaimana *background subtraction* terjadi. Nilai setiap piksel pada gambar foreground dikurangkan dengan nilai setiap piksel pada gambar dengan background di posisi yang sama. Setelah itu, gambar hasil pengurangan nilai piksel ditransformasi ke dalam gambar hitam-putih atau abu-abu.

Meskipun terkesan simpel, namun dalam metode ini terdapat beberapa kelemahan. Salah satunya adalah jika sebuah benda seperti gelas di tempat pengambilan gambar (*scene*) dipindahkan, maka akan terdapat dua pergerakan yang terjadi terus-menerus, yaitu tempat asal gelas dan tempat gelas diletakkan. Kelemahan lain adalah jika pencahayaan dalam *scene* berubah, maka semua area di *scene* akan terdeteksi sebagai gerakan. (Bradski and Kaehler, 2008)

Memperbarui gambar *background* secara berkala saja tidak akan menyelesaikan masalah dalam *background subtraction*. Sebab, terkadang ada objek dalam *scene* yang selalu bergerak setiap saat seperti pohon yang tertiup angin. Metode ini tentu tidak efektif dalam kondisi tersebut. Oleh

karena itu, gambar *background* tidak boleh statis, namun harus bisa beradaptasi dengan perubahan-perubahan pada *scene*. Adaptasi tersebut mencakup adaptasi terhadap perubahan penerangan baik langsung ataupun berangsur-angsur, gerakan yang kontinu seperti air laut dan dahan pohon, osilasi pada kamera, serta perpindahan barang-barang dalam *scene*.

Berbagai algoritma dasar telah dikembangkan untuk memperbarui gambar *background* secara otomatis, di antaranya adalah *average*, *median*, dan *running average*. Ketiga metode ini cukup bagus, namun kurang begitu bagus diterapkan dalam *scene* dengan pencahayaan yang berubah secara berangsur-angsur. Selain itu, metode *average* dan *median* menghabiskan memori yang cukup besar. Oleh karena itu, diperlukan sebuah algoritma dengan level yang lebih tinggi untuk mengatasi hal-hal tersebut. (Piccardi, 2004)

## 2.4 OpenCV *(Open Source Computer Vision)* dan CUDA

Salah satu perusahaan yang fokus dalam riset *computer vision* adalah perusahaan mikroprosesor Intel melalui paket librari OpenCV. OpenCV memulai pra-rilis pertamanya pada tahun 1999 dengan dukungan bahasa C yang dapat dijalankan pada berbagai platform Sistem Operasi seperti Windows, Linux, dan MacOSX. Sejak kemunculan pertamanya OpenCV telah digunakan dalam banyak aplikasi, produk, dan riset. hal ini tak lepas dari banyaknya fungsi yang dimiliki oleh OpenCV yang sangat mudah digunakan dalam pengolahan citra ataupun *computer vision*.

Sejak rilis 2.0 OpenCV mulai mendukung bahasa C++ dan mulai merambah bahasa pemrograman lainnya. Saat ini OpenCV telah dikembangkan untuk berbagai *interface* pemrograman seperti Visual Studio, Qt, Matlab, Python, Ruby, dan Java. Selain itu, OpenCV juga bisa diaplikasikan

ke dalam sistem operasi *gadget* seperti iOS dan Android. Bahkan sejak 2010 OpenCV telah mendukung pemrograman dengan GPU *(Graphics Processing Unit)* NVIDIA yang meningkatkan pemrosesan algoritma pada OpenCV dengan kecepatan yang sangat signifikan.


**Gambar 2.7 Logo OpenCV**

Gambar 2.7 adalah logo librari OpenCV. OpenCV memiliki lebih dari 2500 fungsi dalam *computer vision* yang bisa diaplikasikan dalam berbagai bidang, seperti pengurangan derau pada citra medis, analisis objek, otomasi video keamanan, sistem inspeksi manufaktur, kalibrasi kamera, aplikasi militer, dan kendaraan tanpa awak baik di darat, air, maupun udara. OpenCV juga bisa digunakan untuk pengenal suara dan musik. Berbagai riset robotik juga telah menggunakan OpenCV karena kemudahan dan efisiensi dalam penggunaannya.

Bahasa yang digunakan dalam OpenCV adalah bahasa tingkat tinggi, artinya pengguna cukup menggunakan satu fungsi dan OpenCV akan mampu memberikan beberapa *output*. Hal ini sangat memudahkan para pemula yang ingin belajar *computer vision* dan juga sangat menghemat waktu dan tenaga mereka. Terlebih lagi OpenCV berada di bawah lisensi BSD yang artinya dapat digunakan secara bebas untuk tujuan riset ataupun komersial. Hal ini tentunya juga sangat menghemat biaya. (Bradski and Kaehler, 2008)

Seiring dengan berkembangnya teknologi, sejak 2010 OpenCV mendukung pemrograman dengan GPU. Dahulu

GPU hanya digunakan untuk mengolah grafis. Namun seiring dengan perkembangan pasar dan teknologi, pada 2006 NVIDIA mulai memperkenalkan CUDA *(Compute Unified Device Architecture)*, platform GPGPU *(General-purpose computing on graphics processing units)* yang memungkinkan pemrograman dilakukan di dalam GPU. (www.nvidia.com, 2014)

GPU adalah perangkat keras (*hardware*) yang dirancang khusus untuk pengolahan data secara paralel dan intensif (biasanya berupa pengolahan grafis) pada komputer. Berbeda dengan CPU *(Central Processing Unit)*, GPU memiliki inti yang lebih banyak dari CPU sehingga memungkinkan pemrograman secara paralel. Hal ini membuat GPU dapat memproses data yang cukup besar dalam waktu yang relatif lebih singkat daripada CPU, meskipun setiap inti GPU memiliki frekuensi yang lebih kecil dibandingkan inti CPU.


**Gambar 2.8 Contoh GPU CUDA (kiri) dan CPU (kanan)**

Salah satu contoh GPU dan CPU dapat dilihat pada gambar 2.8 di atas. Secara teknis, GPU lebih sering dijadikan sebagai komponen tambahan pada komputer. GPU sering dipakai untuk mendapatkan performa komputer yang maksimal dalam pemrosesan gambar atau video.

Karena arsitektur GPU yang berbeda dengan CPU, OpenCV tidak bisa menjalankan semua algoritmanya lewat GPU. Kebanyakan fungsi OpenCV yang mendukung GPU adalah fungsi yang mendukung pengolahan data secara paralel dan intensif, sementara fungsi yang lebih

membutuhkan *flow control* diolah dengan CPU. Hal ini tentunya lebih efisien, terlebih antara CPU dan GPU dapat dilakukan data transfer. (Aranda et al., 2013)

## 2.5 *Background Subtraction* dalam OpenCV

Dalam opencv terdapat beberapa fungsi yang bisa digunakan untuk background subtration, salah satunya adalah fungsi BackgroundSubtractorMOG2(). Fungsi ini didasarkan pada metode *Adaptive Gaussian Mixture Model* (GMM) yang dikembangkan oleh Zivkovic pada 2004. Kelebihan fungsi ini dibandingkan fungsi yang lain adalah eksekusi algoritma *background subtraction* yang lebih cepat. Selain itu, pembaruan gambar *background* dilakukan secara berkala dengan fungsi eksponensial sehingga cocok digunakan di tempat yang pencahayaannya berubah-ubah.

Umumnya setiap objek yang memasuki *scene* akan menyebabkan adanya nilai tidak nol di dalam *foreground mask*. Nilai tidak nol ini menunjukkan adanya pergerakan. Jika objek ini terus diam berada di dalam *scene* maka nilai tidak nol ini akan selalu ada sehingga pergerakan selalu terdeteksi di dalam *scene* meskipun objek sebenarnya diam. Jika *background* diperbarui beberapa saat setelah objek memasuki *scene*, maka ketika objek ini keluar dari *scene*, bekas tempat objek tersebut akan terdeteksi sebagai gerakan sampai *background* diperbarui kembali.

Keadaan sebagaimana tersebut di atas tentunya tidak efektif jika dianalisis dengan *background subtraction* yang tidak adaptif. Namun, dengan menggunakan fungsi *BackgroundSubtractorMOG2()* hal ini dapat diatasi dengan baik. Caranya adalah dengan menentukan koefisien alfa ($\alpha$) yang dirumuskan sebagai:

$$\alpha = 1 - 10^{\frac{\log 0{,}9}{frame_{rate} \times t}} \qquad (2.1)$$

dengan $frame_{rate}$ adalah kecepatan pemrosesan gambar per detik dan t adalah waktu yang dibutuhkan untuk menjadikan objek yang diam sebagai bagian dari *background*.

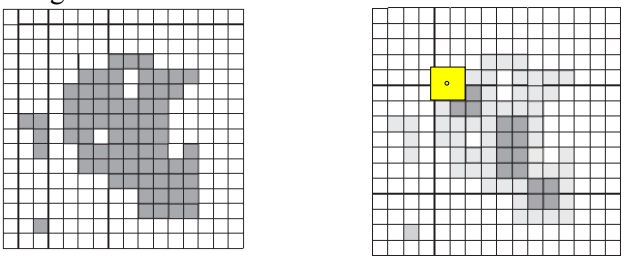
Misalkan $\alpha = 0{,}001$ dan kecepatan proses adalah 21 gambar per detik, maka objek akan dikenali sebagai *background* dalam waktu 5 detik. Artinya setelah 5 detik terdeteksi sebagai gerakan, objek tersebut tidak lagi terdeteksi sebagai gerakan. Jika objek tersebut keluar dari scene setelah 5 detik maka bekas tempat objek tersebut juga tidak akan terdeteksi sebagai gerakan. (www.opencv.org, 2014)

## 2.6 Erosi dan Dilasi

Hasil dari *background subtraction* sering kali memiliki banyak *noise*, titik-titik yang tidak perlu dianalisis lebih lanjut. *Noise* tersebut dapat dihilangkan dengan menggunakan erosi, yang dalam opencv dikenal dengan fungsi *erode()*. Secara matematis, erosi A oleh B dinyatakan dengan

$$A \ominus B = \{z|(B)_z \subseteq A\} \tag{2.2}$$

Atau dapat dikatakan bahwa erosi A oleh B adalah kumpulan titik z sehingga titik B yang ditranslasikan oleh z termuat dalam A. Cara kerja fungsi ini dapat dijelaskan melalui gambar berikut:

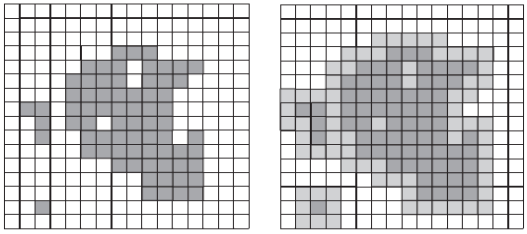


(a) (b)

**Gambar 2.9 Erosi pada sebuah gambar (a) gambar asli (b) hasil erosi**

Pada gambar 2.9 (a) titik-titik berwarna hitam adalah bagian yang akan dikenakan erosi. Setiap titik hitam yang bersebelahan dengan titik berwarna lain (dalam hal ini putih) akan dihapus (dijadikan putih). Besarnya kedalaman penghapusan ini bergantung pada ukuran penstrukturan elemen *(structuring element)* yang digunakan. Elemen yang biasa digunakan adalah persegi atau lingkaran. Panjang dan lebar elemen ini harus bernilai ganjil sehingga dapat dicari titik tengahnya. Dalam gambar di atas digunakan elemen persegi 3x3 piksel (warna kuning). Elemen persegi tersebut bisa dikatakan sebagai sebuah penghapus yang akan mengubah setiap titik yang dilaluinya menjadi putih. Titik tengah elemen persegi ini akan menyisir setiap titik putih sehingga setiap titik hitam yang tertutupi oleh persegi ini akan diubah menjadi putih. Pada gambar 2.9 (b) di atas hasil erosi ditunjukkan dengan warna abu-abu cerah.

Dari ilustrasi di atas terlihat bahwa erosi dapat menghilangkan titik-titik hitam yang berukuran kecil (*noise*) yang tidak diperlukan dalam analisis lebih lanjut. Namun, erosi juga memperkecil ukuran gambar yang akan dianalisis. Oleh karena itu, untuk mengembalikan gambar tersebut ke ukuran yang mendekati aslinya, dapat diterapkan fungsi yang prinsip kerjanya berlawanan dengan erosi, yaitu dilasi.



(a) (b)

**Gambar 2.10 Dilasi pada sebuah gambar (a) gambar asli (b) hasil dilasi**

Dilasi dalam opencv dikenal dengan fungsi *dilate()*. Prinsip kerjanya sama dengan erosi, hanya saja bagian yang disisir adalah titik-titik hitam. Jika ada bagian titik putih yang beririsan dengan elemen persegi, maka titik tersebut akan diubah menjadi hitam. Gambar 2.10 di atas menunjukkan bagaimana dilasi terjadi. Gambar (a) adalah gambar asli, sedangkan gambar (b) adalah gambar yang sudah dikenakan dilasi. (Jähne, 2005)

## 2.7 Gaussian Blur

*Blurring* atau dikenal juga sebagai *smoothing* banyak digunakan dalam pengolahan citra untuk menghaluskan tepi suatu gambar. Dengan *blurring* suatu gambar bisa dibuat agar tidak memiliki perbedaan yang sangat kontras antara piksel yang bersebelahan. Misalnya seperti piksel putih dengan intensitas 255 yang bersebelahan dengan piksel hitam dengan intensitas 0, maka gambar akan terlihat seperti gambar diskrit, seperti pada papan catur. Namun dengan mengaplikasikan *blurring*, gambar tersebut akan bisa terlihat kontinu sebagaimana kertas yang terkena tinta, hitam pekat di tengah dan memudar seiring bertambahnya jarak dari pusat tinta.

Secara sederhana proses blurring pada gambar dapat dijelaskan melalui gambar berikut.

$$\begin{matrix}\vdots & \vdots & \vdots & \vdots & \\ \cdots & 0 & 0 & 1 & 1 & \cdots \\ \cdots & 0 & 0 & 1 & 1 & \cdots \\ \cdots & 0 & 0 & 1 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{matrix} * \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{matrix}\vdots & \vdots & \vdots & \vdots & \\ \cdots & 0 & 1/3 & 2/3 & 1 & \cdots \\ \cdots & 0 & 1/3 & 2/3 & 1 & \cdots \\ \cdots & 0 & 1/3 & 2/3 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{matrix}$$

**Gambar 2.11 Proses Gaussian Blur dengan Filter kotak 1x3**

Pada gambar 2.11 di atas, sebuah matriks berukuran 4x4 dikalikan dengan matriks berukuran 1x3. Matriks 1x3 ini disebut sebagai filter kotak. Faktor 1/3 pada filter kotak tersebut adalah hasil penjumlahan konvolusi untuk menjaga

nilai hasil perkalian tidak melebihi nilai maksimum yang ada. Hasil perkalian matriks tersebut sebagaimana terlihat pada gambar di atas menunjukkan perubahan yang signifikan. Matriks yang awalnya hanya memiliki nilai 0 dan 1 berubah menjadi nilai yang naik dari 0 sampai 1 dengan interval 1/3. Jika matriks tersebut adalah sebuah gambar, tentunya akan dihasilkan gambar yang memiliki nilai piksel dari hitam ke putih tanpa adanya batas yang jelas di mana perubahan nilai piksel dari hitam ke putih terjadi. (Jähne, 2005)

Metode di atas dikenal dengan metode *box average*, di mana filter yang digunakan adalah matriks/gambar dengan intensitas seragam. Metode ini sangat sederhana namun terkadang batas antar piksel masih terlihat jelas. Untuk mengatasi masalah ini, dapat digunakan filter gaussian, seperti pada gambar berikut:

**Gambar 2.12 (a) Filter kotak (b) Filter Gaussian**

Secara matematis, filter gaussian di atas diperoleh dari persamaan densitas probabilitas:

$$p(f) = (2\pi\sigma^2)^{-1/2} \exp\left\{-\frac{f^2}{2\sigma^2}\right\} \tag{2.3}$$

di mana f menyatakan koordinat piksel dan $\sigma$ menyatakan ukuran lebar filter yang digunakan. Semakin besar nilai $\sigma$ maka gambar akan semakin kabur dan tidak jelas.

Dalam opencv, penggunaan filter gaussian dalam blurring dikenal dengan fungsi gaussianblur(). Dalam fungsi ini, besarnya ukuran filter yang digunakan disebut sebagai

kernel. Kernel ini tidak harus berbentuk persegi, namun harus bernilai ganjil. Persebaran intensitas di dalam kernel ditentukan dari nilai $\sigma$. Nilai $\sigma$ ini dapat ditentukan sendiri atau ditentukan secara otomatis melalui persamaan:

$$\sigma_x = \left(\frac{n_x}{2} - 1\right) \cdot 0.30 + 0.80,$$

$$\sigma_y = \left(\frac{n_y}{2} - 1\right) \cdot 0.30 + 0.80, \tag{2.4}$$

dengan $n_x$ dan $n_y$ menyatakan ukuran kernel secara horizontal dan vertikal. (Paris, 2015)

## 2.8 Deteksi Tepi Canny

Deteksi tepi adalah salah satu metode yang digunakan untuk mendapatkan informasi dari suatu gambar. Deteksi tepi dapat digunakan untuk mencari batas antara satu benda dengan benda yang lain di dalam gambar. Ada beberapa metode deteksi tepi, namun yang paling sering digunakan adalah deteksi tepi Canny yang dikembangkan oleh John F. Canny pada 1986. Deteksi tepi Canny ini memiliki 3 keunggulan,

1. Rata-rata error yang kecil, mendeteksi setiap tepi dalam gambar dengan akurat.
2. Lokalisasi yang baik, jarak antara tepi piksel yang terdeteksi dengan tepi piksel sesungguhnya bernilai minimum.
3. Respons minimal, setiap tepi terdeteksi hanya sekali.

   Deteksi tepi Canny memiliki 4 langkah proses:

   a. Memfilter noise menggunakan gaussian filter. Misalnya dengan kernel berukuran 5x5 seperti berikut:

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{2.5}$$

b. Mencari gradien intensitas dari gambar. Proses ini memiliki 2 tahap:
   (a) Menerapkan sepasang *convolution mask* dalam arah x dan y

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$
$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \quad (2.6)$$

   (b) Mencari kekuatan gradien dan arahnya dengan

$$G = \sqrt{G_x^2 + G_y^2}$$
$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \quad (2.7)$$

   Arah gradien dibulatkan ke dalam satu dari 4 arah yang mungkin (0, 45, 90, atau 135 derajat).

c. Penekanan non-maksimum atau disebut juga penipisan tepi. Proses ini akan menghapus piksel di sekitar tepi yang tidak termasuk tepi sehingga tepi yang didapatkan berukuran tipis.
d. Histerisis, menentukan piksel yang termasuk tepi atau tidak dengan cara thresholding. Canny menggunakan 2 threshold (upper dan lower). Jika gradien piksel lebih tinggi daripada threshold upper, maka piksel dianggap sebagai tepi. Jika gradien piksel lebih rendah daripada lower threshold, maka piksel bukanlah tepi. Jika gradien piksel berada di antara keduanya (upper dan lower threshold), maka piksel hanya akan dianggap sebagai tepi jika bersambung dengan piksel yang bernilai di atas upper threshold.

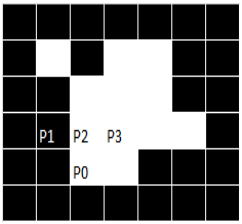
Berikut adalah contoh gambar yang dikenakan deteksi tepi:

**Gambar 2.13 Deteksi Tepi Canny (a) gambar asli (b) hasil deteksi tepi**

(www.wikipedia.org, 2014)

## 2.9 Pencarian Kontur

Hasil dari deteksi tepi Canny selanjutnya dapat digunakan untuk mencari kontur di dalam gambar. Kontur adalah serangkaian titik yang merupakan batas luar kumpulan piksel. Pencarian kontur ini diperoleh dengan mengaplikasikan fungsi findcontour() dari librari opencv terhadap hasil deteksi tepi Canny.

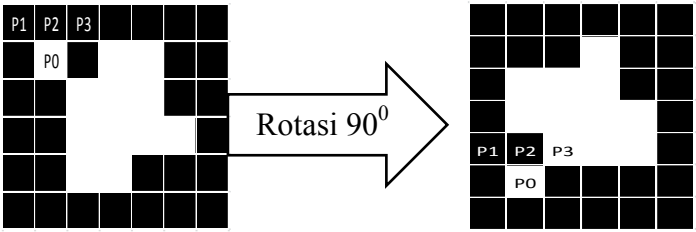



**Gambar 2.14 Titik P0, P1, P2, dan P3 dalam pencarian kontur**

Pencarian kontur dimulai dengan penentuan titik awal. Titik awal ini diperoleh dengan meninjau setiap baris piksel dari pojok bawah kiri. Jika ditemukan piksel putih, maka piksel tersebut dijadikan titik awal. Pada gambar 2.14 di atas, titik awal adalah P0. Lalu, peninjauan akan dilakukan pada titik-titik di sekitar titik awal tersebut. Peninjauan dimulai dari titik di atas-kiri titik awal tersebut (P1). Jika titik P1

adalah piksel putih, maka titik tersebut akan dijadikan titik awal selanjutnya. Jika tidak, maka peninjauan akan dilakukan terhadap titik di atas titik awal (P2). Jika titik P2 juga bukan merupakan piksel putih, maka peninjauan akan dilakukan pada titik P3.

Mulai titik awal kedua, pencarian titik awal selanjutnya dimulai dari titik di sebelah kiri, baru dilanjutkan pada titik P1 sampai titik P3. Jika sampai titik P3 tidak ditemukan titik putih, maka gambar akan diputar $90^0$ dan pencarian dilakukan kembali.




**Gambar 2.15 Rotasi gambar $90^0$ saat P3 bukan titik putih**

Jika gambar sudah diputar 2 kali dan tidak ditemukan piksel putih, maka pencarian akan dihentikan. Hal ini menunjukkan bahwa titik tersebut adalah titik tunggal, sehingga tidak termasuk kontur. Selain itu, jika titik-titik yang didapatkan tidak berupa bidang (hanya garis) maka titik-titik tersebut juga tidak dianggap sebagai kontur. Pencarian juga akan dihentikan jika pencarian telah kembali ke titik awal pertama. (Chen, 2010)

Fungsi *findcontour()* juga dapat mencari kontur di dalam kontur. Kontur yang berada di dalam kontur disebut anak kontur dan kontur yang memuat kontur disebut induk kontur. Jika kontur yang ditemukan adalah kumpulan titik-titik hitam, maka kontur tersebut disebut *hole* (lubang). Struktur kontur yang bertingkat ini dikenal dengan istilah *hierarchy*. Berdasarkan *hierarchy*-nya, fungsi *findcontour()* ini memiliki 4 mode, yaitu:

- CV_RETR_EXTERNAL, hanya mencari kontur terluar dan mengabaikan anak kontur.
- CV_RETR_LIST, mencari semua kontur baik induk kontur maupun anak kontur namun tanpa memperhatikan tingkatan (semua dianggap satu tingkat).
- CV_RETR_CCOMP, mencari semua kontur dengan 2 tingkat *hierarchy*. Tingkat pertama adalah kontur dari piksel putih (baik induk maupun anak kontur) dan tingkat kedua adalah *hole*.
- CV_RETR_TREE, mencari semua kontur dengan *hierarchy* sebagaimana adanya.(www.opencv.org, 2014)

## 2.10 Ekstraksi Fitur Histogram

Ekstraksi fitur histogram merupakan metode pengambilan ciri berdasarkan pada karakteristik histogram citra. Di mana histogram menunjukkan probabilitas kemunculan nilai derajat keabuan pixel pada suatu citra. Jika x menyatakan tingkat keabuan pada suatu citra, maka probabilitas P(x) dinyatakan sebagai berikut :

$$P(x) = \frac{banyaknya\ titk-titik\ yang\ memiliki\ tingkat\ keaauan\ x}{total\ banyaknya\ titik\ pada\ daerah\ suatu\ citra} \quad (2.8)$$

dengan x=0,1,2,3…,L-1

Ekstraksi fitur histogram yang digunakan dalam penelitian ini hanya ada 2, yaitu :

a. Mean (μ)

Mean merupakan ukuran disperse dari suatu citra

$$\mu = \sum_n f_n p(f_n) \quad (2.9)$$

dengan $f_n$ merupakan suatu nilai intensitas keabuan, sementara p(fn) menunjukkan nilai histogramnya (probabilitas kemunculan intensitas tersebut pada citra)

b. Standar Deviasi (σ)

Standar Deviasi merupakan cerminan dari rata-rata penyimpangan data dari *mean*.

$$\sigma = \sum_n |f_n - \mu| \ (f_n) \quad (2.10)$$

(Darma, Putera, 2010)

## 2.11 Bandul Matematis

Suatu benda titik yang digantungkan pada sebuah tali sehingga dapat diayunkan disebut sebagai bandul matematis. Pada bandul matematis, massa tali dapat diabaikan dan panjang tali harus jauh lebih besar daripada ukuran geometris dari bandul. Saat bandul diayunkan, terjadi gerak harmonis sederhana, yaitu gerak bolak-balik benda melalui suatu titik keseimbangan tertentu dengan periode yang tetap.

Pada bandul matematis yang diayunkan berlaku prinsip konservasi energi mekanik, di mana jumlah energi potensial gravitasi dan energi kinetiknya di setiap titik selalu sama. Secara matematis dapat dinyatakan:

$$E_m = E_p + E_k = \text{konstan} \qquad (2.11)$$

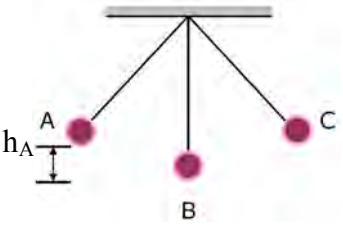
di mana $E_m$ adalah energi mekanik, $E_p$ adalah energi potensial gravitasi, dan $E_k$ adalah energi kinetik, dengan

$$E_p = mgh \qquad (2.12)$$

dan

$$E_k = \frac{1}{2}mv^2 \qquad (2.13)$$

di mana m adalah massa bandul, g adalah percepatan gravitasi bumi (9,8 m/s$^2$), h adalah ketinggian bandul, dan v adalah kecepatan bandul.




**Gambar 2.16 Bandul Matematis**

Gambar 2.16 di atas menunjukkan contoh bandul matematis. Jika bandul diayunkan dari titik A ke titik C melalui titik B, maka konservasi energi yang terjadi pada titik A dan titik B adalah

$$E_{m_A} = E_{m_A} \quad (2.14)$$

$$E_{p_A} + E_{k_A} = E_{p_B} + E_{k_B} \quad (2.15)$$

Pada titik A energi kinetiknya adalah nol, sedangkan pada titik B energi potensialnya adalah nol, sehingga persamaan di atas menjadi:

$$mgh_A + 0 = 0 + \frac{1}{2}mv_B{}^2 \quad (2.16)$$

$$gh_A = \frac{1}{2}v_B{}^2 \quad (2.17)$$

$$v_B = \sqrt{2gh_A} \quad (2.18)$$

(Halliday and Resnick, 2003)

*"Halaman ini sengaja dikosongkan"*

# BAB III
# METODOLOGI

## 3.1 Alat dan Bahan

Peralatan yang dibutuhkan dalam penelitian ini mencakup perangkat keras *(hardware)* dan perangkat lunak *(software)*. Adapun perangkat keras yang dibutuhkan dalam penelitian ini adalah:

1. Satu unit laptop ASUS seri K43SV dengan dukungan webcam dan CUDA Geforce 520M.
2. Bola kasti dengan diameter 6,5 cm.
3. Benang jahit
4. Penyangga
5. Meteran

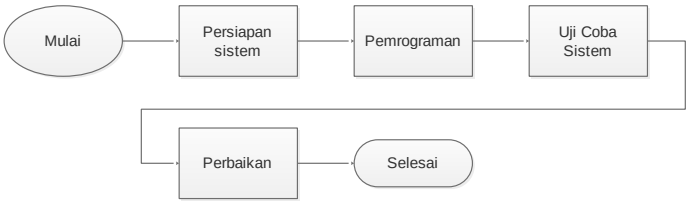
Sedangkan perangkat lunak yang dibutuhkan adalah:

1. Sistem Operasi Windows,
2. Visual Studio 2012 Profesional,
3. CUDA Toolkit 6.5.14,
4. Librari OpenCV 2.4.9,
5. cmake 3.1.0.

## 3.2 Langkah Kerja

### 3.2.1 Langkah Kerja Penelitian

Secara garis besar, penelitian ini dilakukan sebagaimana diagram alir berikut ini:

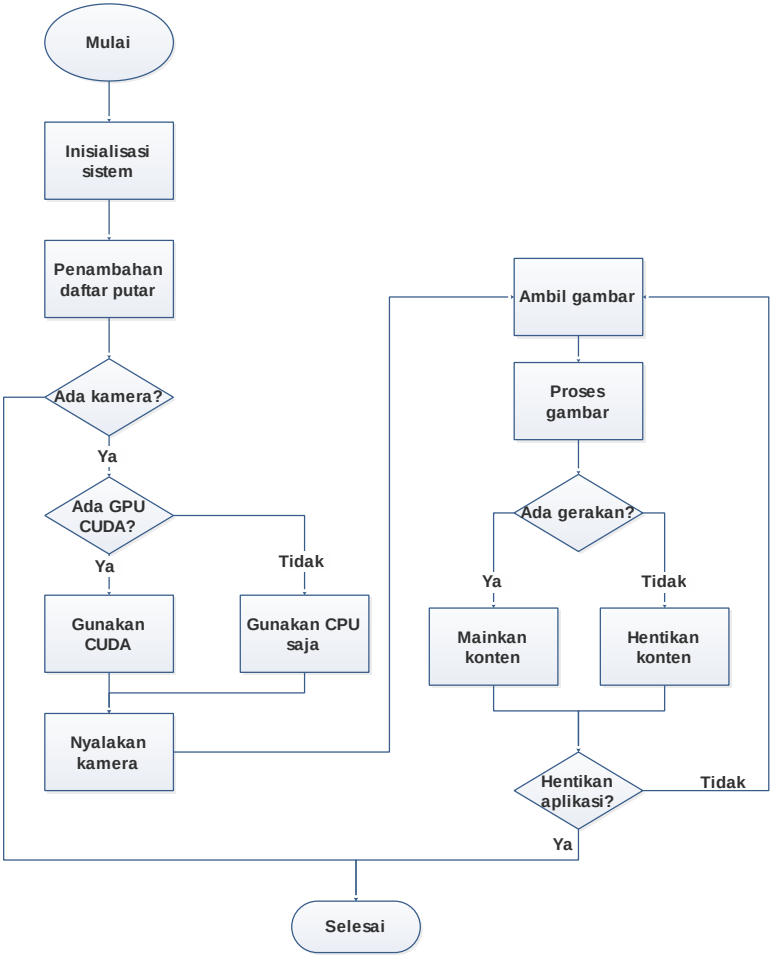



**Gambar 3.1 Diagram Alir Pembuatan Sistem**

1. Persiapan Aplikasi
   Persiapan aplikasi dimulai dengan instalasi Visual Studio 2012 dan CUDA Toolkit. Proses ini membutuhkan waktu sekitar 30 menit sampai 1 jam. Setelah itu, librari OpenCV di-*build* ulang menggunakan cmake dan selanjutnya di-*compile* menggunakan Visual Studio. Proses ini membutuhkan waktu sekitar 1 sampai 2 jam. Lalu librari OpenCV yang telah di-*build* ulang diintegrasikan ke dalam librari Visual Studio.
2. Pemrograman
   Pemrograman dilakukan dengan Visual Studio. Bahasa yang digunakan adalah Visual C++ yang memudahkan dalam pembuatan tampilan GUI *(Graphical User Interface)*. Aplikasi ini menggunakan *plugin Windows Media Player* untuk memainkan video. Resolusi gambar yang digunakan adalah 640x480 piksel.
3. Uji Coba Aplikasi
   Pengujian sistem dilakukan di kamar penulis di Ponpes Darussalam Keputih Surabaya. Pengujian dilakukan menggunakan bandul matematis dari kelereng.
4. Perbaikan
   Perbaikan sistem dilakukan agar memperoleh hasil yang baik. Perbaikan ini meliputi efisiensi kode pemrograman, pemecahan masalah *error* pada aplikasi, koreksi parameter-parameter yang digunakan, serta perbaikan tampilan GUI.

### 3.2.2 Alur Kerja Aplikasi

Alur kerja aplikasi dapat dilihat pada diagram alir berikut:




**Gambar 3.2 Diagram Alir untuk Alur Kerja Aplikasi**

Penjelasan prosesnya adalah sebagai berikut:

1. Inisialisasi aplikasi
   Dalam proses ini, semua parameter yang digunakan diatur terlebih dahulu nilainya.
2. Penambahan daftar putar
   Pengguna aplikasi harus menambahkan daftar putar terlebih dahulu. Daftar putar diberikan 2 opsi, yaitu audio-video atau gambar. Setelah daftar putar diisi, tombol mulai akan aktif.
3. Pengecekan webcam dan CUDA
   Aplikasi akan mengecek apakah komputer yang digunakan sudah terpasang webcam atau belum. Jika belum maka aplikasi tidak akan dilanjutkan. Jika sudah, maka webcam akan dinyalakan. Setelah itu, aplikasi akan mengecek keberadaan CUDA. Jika ada CUDA maka proses akan dilakukan dengan bantuan CUDA, jika tidak maka proses dilakukan dengan CPU saja.
4. Pemrosesan gambar
   Aplikasi akan mengambil 1 gambar dengan webcam, setelah itu gambar akan melalui beberapa proses.
   a. *Background subtraction*. Dari *background subtraction* ini didapatkan output berupa gambar *foreground mask*.
   b. Untuk konten audio-video, foreground mask dicari nilai standar deviasinya dan pemrosesan gambar selesai. Sedangkan untuk konten gambar tidak ada pencarian nilai standar deviasi dan proses masih berlanjut.

c. Erosi untuk menghilangkan noise pada foreground mask.
d. Dilasi untuk mengembalikan ukuran piksel yang akan diolah.
e. Gaussian Blur untuk membuat batas piksel yang akan diolah lebih halus.
f. Deteksi tepi Canny untuk mencari batas tepi piksel yang akan diolah.
g. Pencarian kontur dari deteksi tepi canny.
h. Pemfilteran kontur agar kontur yang diproses hanya kontur dengan luas terbesar.
i. Pencarian titik tengah kontur untuk analisis gerak.

5. Analisis gerak

Untuk konten audio-video analisis gerak dilakukan melalui nilai standar deviasi. Jika nilai standar deviasi melebihi nilai yang ditentukan, maka terjadi gerak. Sedangkan jika nilai standar deviasi di bawah nilai yang ditentukan tidak akan terjadi gerak.

Untuk konten gambar analisis gerak dilakukan melalui titik tengah kontur yang didapatkan. Jika titik tengah kontur berpindah posisi secara horizontal dalam jarak tertentu dan waktu tertentu maka gambar akan berganti.

6. Respons aplikasi

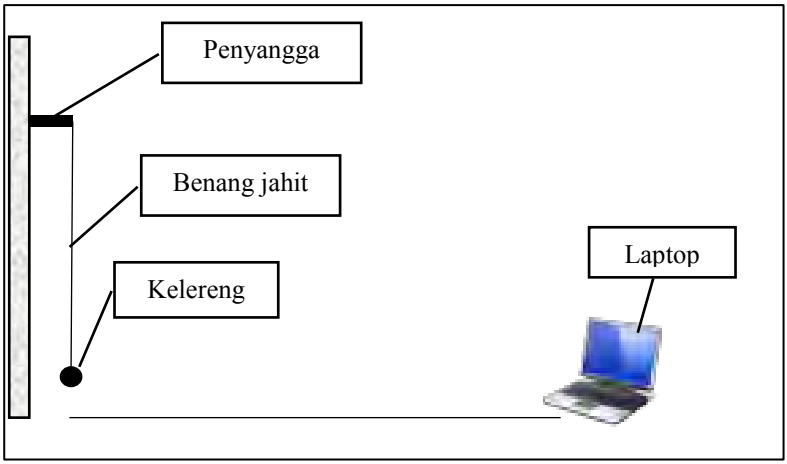
Untuk konten audio-video respons yang diberikan adalah memainkan konten jika ada gerakan dan menjedanya jika tidak ada gerakan. Selain itu, layar monitor juga akan dimatikan jika tidak ada gerakan dan dinyalakan kembali jika ada gerakan.

Untuk konten gambar respons yang diberikan adalah pergantian gambar selanjutnya untuk gerak ke kanan dan pergantian gambar sebelumnya untuk gerak ke kiri.

7. Penghentian aplikasi
   Aplikasi akan berhenti jika pengguna mengeklik tombol stop atau menutup aplikasi.

### 3.2.3 Rangkaian Alat untuk Pengujian Aplikasi

Peralatan disusun seperti berikut:




**Gambar 3.3 Setup Alat Penelitian**

### 3.2.4 Langkah Kerja Pengujian Aplikasi

Bandul dari kelereng diikat dengan tali sepanjang 28,5 cm dan dikatikan pada penyangga. Lalu bandul diayunkan dengan variasi tinggi 1 cm sampai 10 cm dengan interval 1 cm. Keadaan bandul saat diayunkan direkam dengan webcam laptop yang

berjarak 69 cm dari bandul. Resolusi gambar yang digunakan dalam perekaman gerakan bandul adalah 320x240 piksel. Daerah yang direkam adalah antara piksel kolom ke-120 sampai kolom ke-200. Dari percobaan ini akan didapatkan jumlah gambar yang mampu dideteksi oleh webcam pada daerah tersebut.

*"Halaman ini sengaja dikosongkan"*

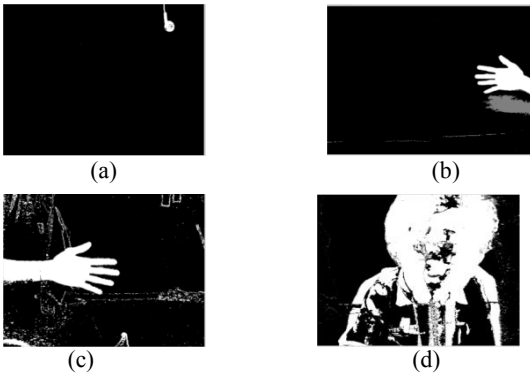
# BAB IV
# HASIL DAN PEMBAHASAN

## 4.1 Analisis Gerak

Analisis gerak dari *background subtraction* pada penelitian ini dilakukan melalui 2 cara, yaitu dengan analisis standar deviasi dan analisis kontur.

### 4.1.1 Analisis Gerak Melalui Standar Deviasi

*Foreground mask* hasil *background subtraction* dapat digunakan untuk menganalisis gerak dengan mencari standar deviasinya. Dengan menggunakan resolusi webcam 640x480 piksel, nilai standar deviasi hasil *background subtraction* memiliki nilai minimum nol jika *foreground mask* hanya berisi piksel berwarna putih saja atau hitam saja. Sedangkan nilai maksimumnya adalah 127,5 yaitu ketika separuh *foreground mask* berupa piksel warna putih dan separuhnya lagi berupa piksel warna hitam.




**Gambar 4.1 Foreground mask dengan standar deviasi (a) 16 (b) 51 (c) 85 (d) 124**

Sebagaimana terlihat pada gambar 4.1 di atas, nilai standar deviasi yang kecil menunjukkan adanya

sedikit perubahan di dalam *scene*, seperti headset yang masuk ke dalam *scene*. Sedangkan nilai standar deviasi yang besar menunjukkan banyak perubahan terjadi di dalam *scene*, seperti orang yang lewat di depan webcam. Pada gambar 4.1 di atas, *foreground mask* dari sebuah *headset* memiliki standar deviasi 16, telapak tangan 51, tangan 85, dan wajah beserta dada 124.

Analisis gerak melalui nilai standar deviasi sifatnya sangat sederhana. Analisis ini tidak dapat menentukan arah pergerakan, lokasi pergerakan, ataupun jumlah pergerakan. Analisis ini hanya sebatas indikator apakah ada objek yang masuk atau keluar *scene*.

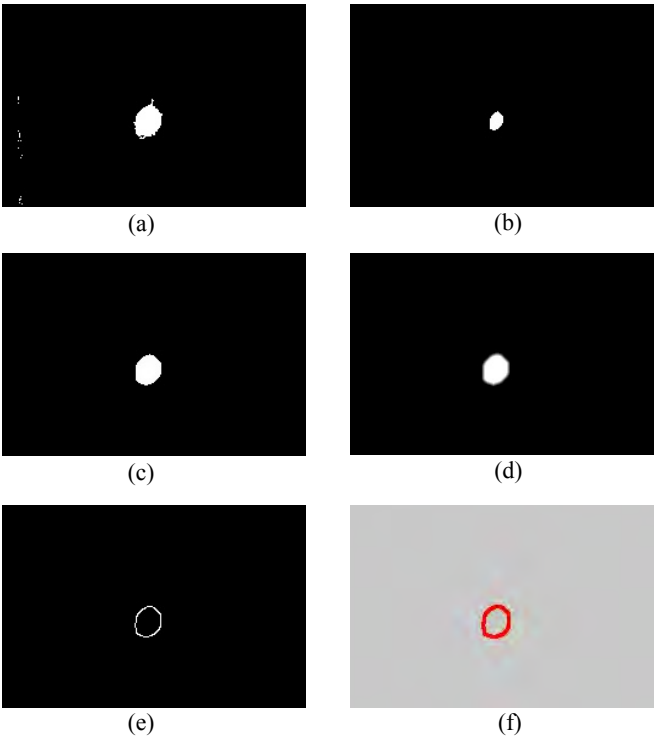
Analisis gerak melalui nilai standar deviasi dari *foreground mask* ini sangat cocok diterapkan untuk manajemen konten audio-video, terlebih audio-video iklan atau pengumuman yang dipasang di tempat umum sehingga audio-video dapat berhenti otomatis jika tidak ada orang di depannya dan juga menyala otomatis saat ada orang lewat.

### 4.1.2 Analisis Gerak melalui Kontur

Analisis gerak yang lebih baik dapat dilakukan melalui analisis kontur dari *foreground mask*. Tidak seperti pada analisis standar deviasi yang menganalisis gambar *foreground* secara keseluruhan, analisis kontur hanya dilakukan pada piksel-piksel berwarna putih saja.

Untuk mendapatkan kontur dari gambar *foreground mask*, perlu dilakukan beberapa proses terlebih dahulu, yaitu erosi, dilasi, *gaussian blur*, dan deteksi tepi Canny. Setelah itu, baru dilakukan pencarian kontur. Proses-proses ini dilakukan untuk mendapatkan gambar kontur yang tanpa *noise* dan

lebih halus. Gambaran proses-proses ini dapat dilihat pada gambar berikut:




**Gambar 4.2 Gambar hasil (a) *background subtraction* (b) Erosi (c) Dilasi (d) *Gaussian Blur* (e) Deteksi Tepi Canny (f) Pencarian Kontur**

Hasil *background subtraction* seperti pada gambar 4.2 (a) masih memiliki *noise*. Setelah dilakukan dilasi *noise* tersebut dapat hilang seperti terlihat pada gambar 4.2 (b). Hasil dilasi menyebabkan ukuran objek menjadi lebih kecil, sehingga untuk mengembalikan ukurannya dilakukan dilasi dan hasilnya terlihat seperti pada gambar 4.2 (c).

Dari hasil dilasi sebenarnya sudah bisa dilakukan pencarian kontur, namun karena hasil dilasi terkadang menimbulkan tepi yang tajam, maka dilakukan *gaussian blur* untuk menghaluskannya. Hasilnya seperti terlihat pada gambar 4.2 (d). Setelah itu dilakukan deteksi tepi Canny untuk mendapatkan tepi gambar objek seperti pada gambar 4.2 (e). Dari sini barulah dilakukan pencarian kontur dan didapatkan hasil seperti pada gambar 4.2 (f).

Sekilas tidak terlihat perbedaan antara hasil deteksi tepi Canny dengan kontur yang didapatkan. Namun, pada nyatanya keduanya berbeda. Deteksi tepi Canny menghasilkan gambar utuh seperti pada gambar 4.2 (e) sementara kontur pada gambar 4.2 (f) adalah kumpulan titik-titik piksel berwarna merah. Karena hanya berupa kumpulan titik-titik piksel yang terbatas, kontur pada gambar tersebut dapat digambarkan di atas *background* berwarna abu-abu.

Dalam background subtraction, kontur mewakili objek yang bergerak. Dari kontur ini dapat diketahui posisi objek dalam satuan piksel. Dengan mengetahui posisi objek, maka dapat diketahui posisi objek setiap saat dan arah pergerakannya.

Dalam analisis gerak dengan kontur ini digunakan bandul sebagai alat uji coba. Pertama-tama, perlu diketahui terlebih dahulu kecepatan maksimum bandul pada setiap ketinggian tertentu. Untuk itu, digunakan persamaan (2.18) yang menunjukkan kecepatan maksimum bandul, yaitu saat di titik terendah. Dari persamaan tersebut, dapat ditulis hubungan antara ketinggian bandul $h_A$ dengan kecepatan bandul $v_b$ sebagai berikut:

**Tabel 4.1 Nilai kecepatan yang diperoleh dari nilai ketinggian bandul**

| No. | Ketinggian ($h_A$) | Kecepatan ($v_B$) |
|---|---|---|
| 1 | 1 cm | 0,44 m/s |
| 2 | 2 cm | 0,63 m/s |
| 3 | 3 cm | 0,77 m/s |
| 4 | 4 cm | 0,89 m/s |
| 5 | 5 cm | 0,99 m/s |
| 6 | 6 cm | 1,08 m/s |
| 7 | 7 cm | 1,17 m/s |
| 8 | 8 cm | 1,25 m/s |
| 9 | 9 cm | 1,33 m/s |
| 10 | 10 cm | 1,40 m/s |

Pada penelitian ini jarak bandul dari webcam adalah 69 cm. Resolusi gambar yang digunakan dalam analisis kontur ini adalah 320x240 piksel. Resolusi maksimum 640x480 piksel tidak digunakan karena waktu eksekusi proses akan membutuhkan waktu yang lama. Pengambilan gambar dilakukan pada daerah antara piksel ke-120 sampai piksel ke-200. Pada daerah ini kecepatan bandul diasumsikan konstan sebagaimana pada tabel 4.1 di atas.

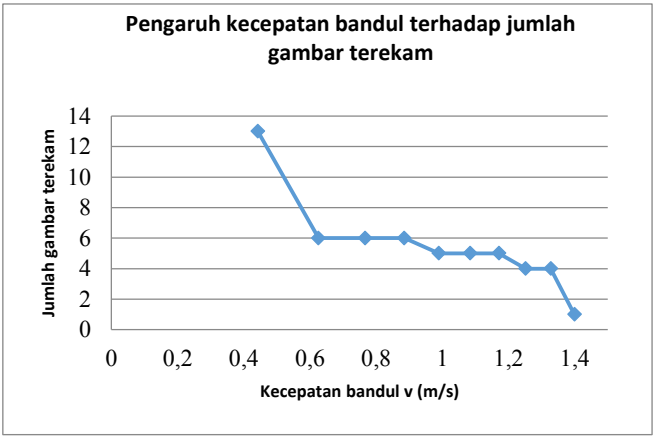
Dari hasil percobaan didapatkan bahwa jumlah gambar objek berupa bandul yang mampu ditangkap webcam untuk setiap kecepatan adalah berbeda-beda, seperti terlihat pada tabel berikut:

**Tabel 4.2 Hubungan kecepatan bandul dengan jumlah gambar terekam**

| No. | Kecepatan | Jumlah Gambar terekam |
|---|---|---|
| 1 | 0,44 m/s | 13 |
| 2 | 0,63 m/s | 6 |
| 3 | 0,77 m/s | 6 |
| 4 | 0,89 m/s | 6 |
| 5 | 0,99 m/s | 5 |
| 6 | 1,08 m/s | 5 |

| No. | Kecepatan | Jumlah Gambar terekam |
|---|---|---|
| 7 | 1,17 m/s | 5 |
| 8 | 1,25 m/s | 4 |
| 9 | 1,33 m/s | 4 |
| 10 | 1,40 m/s | 1 |

Dari tabel 4.2 tersebut dapat dibuat grafik sebagai berikut:




**Grafik 4.1 Pengaruh kecepatan bandul terhadap jumlah gambar yang mampu dideteksi kamera**

Dari tabel 4.2 dan grafik 4.1 di atas terlihat bahwa semakin cepat pergerakan bandul, maka jumlah gambar yang dapat terekam webcam semakin sedikit. Dari sini dapat dipahami hal yang sebaliknya, yaitu jika jumlah gambar yang terekam sedikit, berarti bandul bergerak dengan cepat. Sebaliknya, jika jumlah gambar yang terekam sangat banyak, berarti bandul bergerak pelan.

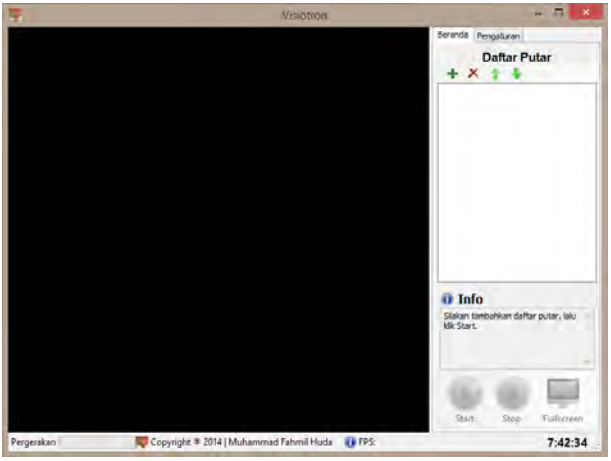
Pada aplikasi ini, objek dikatakan bergerak dengan cepat jika dalam rentang jarak 80 piksel objek memiliki kurang dari 5 gambar yang terekam webcam. Sedangkan objek dikatakan bergerak lambat jika

dalam rentang jarak 80 piksel objek memiliki lebih dari 6 gambar yang terekam. Dengan demikian, pada jarak 69 cm dari webcam kecepatan minimum yang masih bisa dideteksi webcam adalah 0,63 m/s dan kecepatan maksimumnya adalah 1,17 m/s.

Untuk mengetahui arah pergerakan objek, posisi objek dibandingkan pada setiap gambar. Jika posisi horizontal objek selalu lebih besar daripada posisi sebelumnya maka objek dikatakan bergerak ke kanan. Sedangkan jika posisi objek selalu lebih kecil dari posisi sebelumnya, berarti objek bergerak ke kiri.

## 4.2 Tampilan GUI Aplikasi

Berikut adalah tampilan antarmuka aplikasi yang telah dibuat:




**Gambar 4.3 Tampilan Aplikasi**


**Gambar 4.4 Tampilan Aplikasi saat Berjalan**

Antarmuka aplikasi ini dibuat seminimalis mungkin agar pengguna tidak bingung. Pada tab "Beranda", pengguna hanya perlu menambahkan daftar konten yang akan diputar. Pengguna dapat menambahkan beberapa *file* sekaligus atau menambahkan semua *file* yang ada dalam satu folder. Pengguna juga dapat menyimpan daftar konten tersebut ke dalam sebuah *file* berekstensi *.hdq dan dapat dibuka sewaktu-waktu.

Agar aplikasi bisa menyesuaikan dengan selera pengguna, maka pengguna diberi kebebasan dalam menentukan urutan konten yang akan diputar. Selain itu, pengguna juga bisa menghapus daftar konten yang sekiranya tidak ingin dilihatnya. Jika daftar putar masih terlalu sedikit, pengguna juga bisa menambahkan daftar konten ke dalam daftar putar.

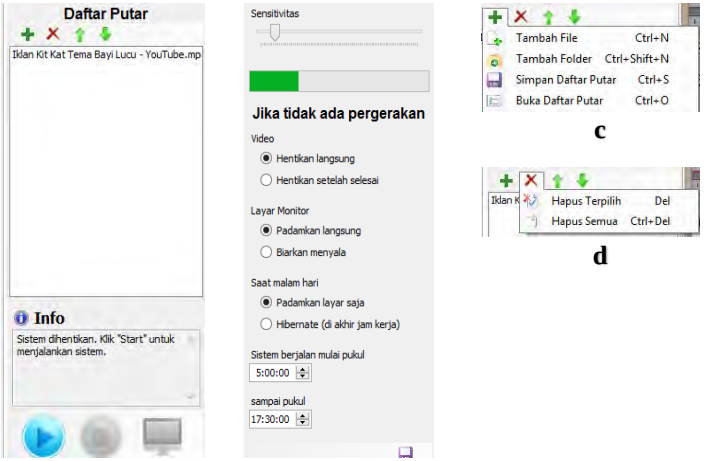
Setelah dafar putar terisi, maka tombol "Start" akan aktif. Dengan mengeklik tombol "Start" tersebut, webcam akan segera menyala, memainkan konten, dan sebuah jendela berisi gambar foreground mask akan terbuka. Saat inilah

aplikasi akan bekerja. Jika webcam mendeteksi adanya pergerakan, maka konten akan terus diputar. Sedangkan jika webcam tidak mendeteksi adanya pergerakan maka konten akan dijeda. Indikator tingkat gerakan dapat dilihat pada batang progres di kiri bawah.

Ketika aplikasi sedang berjalan, akan muncul sebuah jendela kecil yang menampilkan gambar *foreground mask*. Jendela ini tidak dapat ditutup oleh pengguna karena merupakan kebutuhan dasar librari OpenCV agar bisa berjalan. Jendela ini akan tertutup dengan sendirinya begitu aplikasi dihentikan.

Untuk melihat konten dalam keadaan layar penuh (hanya menampilkan video saja), pengguna cukup mengeklik tombol "Fullscreen" dan mengeklik video untuk kembali ke layar normal. Untuk menghentikan aplikasi pengguna dapat mengeklik tombol "Stop" atau menutup aplikasi.

Pada tab "Pengaturan", pengguna dapat mengatur tingkah laku aplikasi. Di bagian "Sensitivitas" pengguna dapat mengatur kepekaan aplikasi terhadap gerakan yang terdeteksi. Semakin ke kanan tingkat kepekaan akan semakin berkurang, begitu pula sebaliknya. Selanjutnya pada opsi "Video" pengguna bisa menentukan apakah video akan langsung dijeda saat gerakan tidak terdeteksi ataukah satu video tersebut akan dimainkan sampai akhir. Begitu pula pada opsi "Layar monitor" pengguna bisa memilih ingin mematikan layar jika tidak ada gerakan terdeteksi ataukah membiarkannya menyala.




**Gambar 4.5 Tampilan (a) Tab Beranda (b) Tab Pengaturan (c) Menu Tambah (d) Menu Hapus**

Pada opsi "Saat malam hari" pengguna bisa memilih apakah komputer akan dimatikan atau tidak. Jika pengguna ingin mematikan komputer maka pengguna bisa memilih opsi "Hibernate" dan komputer akan dipadamkan pada saat akhir jam kerja. Akhir jam kerja aplikasi ditentukan pada opsi "sampai pukul". Jika pengguna memilih opsi "Padamkan layar saja" maka komputer tidak akan dipadamkan saat jam kerja berakhir, namun aplikasi akan mengirimkan sinyal untuk mematikan layar komputer setiap 10 menit. Selain itu, aplikasi juga akan me-nonaktif-kan webcam sampai dimulainya kembali jam kerja aplikasi. Awal jam kerja aplikasi dapat diatur lewat opsi "Aplikasi berjalan mulai pukul".

Semua pengaturan yang telah dipilih oleh pengguna dapat disimpan dengan mengeklik tombol "Save". Pengaturan ini akan secara otomatis digunakan oleh aplikasi saat aplikasi

dibuka kembali setelah ditutup. Jadi pengguna tidak perlu repot-repot mengatur aplikasi setiap kali menjalankannya.

Untuk beralih ke pemutaran gambar, pengguna cukup mengeklik tulisan “Daftar Putar: Video”, begitu pula sebaliknya. Untuk pemutaran gambar ini, tab “Pengaturan” tidak berpengaruh apapun karena pemutaran gambar hanya ditujukan untuk presentasi.

## 4.3 Performa Aplikasi

Performa aplikasi yang telah dibuat dapat dibagi menjadi 3 kategori, yaitu input (daftar putar), proses (pemrosesan data), dan output aplikasi.

### 4.3.1 Daftar Putar

Untuk penambahan daftar putar, pengguna bisa menambahkan berapa pun jumlah video atau gambar yang diinginkan. Konten akan diputar urut dari yang paling atas sampai yang paling bawah dan jika sudah sampai bawah maka akan diputar kembali dari atas. Penambahan daftar putar bisa dilakukan ketika aplikasi belum dijalankan ataupun saat aplikasi sedang berjalan.

Untuk pengurangan daftar putar, pengguna bisa menghapus beberapa item sekaligus. Namun, jika penghapusan dilakukan saat aplikasi sedang berjalan dan tidak ada daftar putar yang tersisa maka aplikasi akan langsung dihentikan karena tidak ada konten yang bisa diputar.

Untuk pengurutan daftar putar, pengguna bisa mengurutkan daftar putar baik saat aplikasi sedang berjalan atau tidak. Namun, urutan daftar putar yang dimainkan adalah berdasarkan indeks, bukan berdasarkan nama daftar putar sehingga jika aplikasi telah memainkan 3 konten maka yang akan diputar

selanjutnya adalah konten urutan ke-4 di daftar putar, bukan konten di bawah konten terakhir yang dimainkan. Akan tetapi, pengguna bisa juga melakukan dobel klik pada konten yang ingin diputar dan aplikasi akan memainkan konten tersebut dan mengatur ulang indeksnya.

Adapun ekstensi file untuk video yang didukung oleh aplikasi meliputi .mp4, .mkv, 3gp, .mpg, .flv, .ogg, .webm, dan .mpeg. Sedangkan ekstensi untuk gambar adalah .jpg, .jpeg, png, .gif, dan .bmp.

### 4.3.2 Pemrosesan Data

- Untuk Konten Video

Pengujian aplikasi dilakukan dalam ruangan pada kecepatan webcam 30 fps *(frame per second)* dan 6 fps, dengan dan tanpa CUDA. Berikut adalah data hasil pengujian aplikasi untuk konten berupa video:

**Tabel 4.3 Data Pengujian Aplikasi**

| No. | Parameter | Dengan CUDA | Tanpa CUDA |
|---|---|---|---|
| 1 | Kecepatan rata-rata proses dengan kecepatan webcam 30 fps | 30 proses/detik | 22 proses/detik |
| 2 | Kecepatan rata-rata proses dengan kecepatan webcam 6 fps | 6 proses/detik | 6 proses/detik |
| 3 | Penggunaan memori RAM sebelum aplikasi berjalan | 40 MB | 27 MB |
| 4 | Penggunaan memori RAM saat aplikasi berjalan | 400 MB | 105 MB |
| 5 | Penggunaan memori RAM saat aplikasi dihentikan | 384 MB | 58 MB |

Dari tabel di atas terlihat bahwa kecepatan proses pada siang hari dan malam hari sangat berbeda. Pada webcam dengan kecepatan pengambilan gambar maksimum 30 fps, aplikasi dengan CUDA mampu melakukan 30 proses per detik sedangkan aplikasi tanpa CUDA hanya mampu melakukan 22 proses per detik. Hal ini menunjukkan bahwa penggunaan GPU tidak mempengaruhi kecepatan webcam dalam mengambil gambar sementara penggunaan CPU tanpa GPU menurunkan kecepatan webcam sampai 22 fps.

Meskipun pada kecepatan webcam 30 fps GPU lebih unggul, namun untuk kecepatan 6 fps antara GPU dan CPU tidak terlalu berbeda. Keduanya sama-sama memiliki kecepatan 6 proses per detik. Dengan demikian, penggunaan GPU tidak terlalu berpengaruh.

Dari kecepatan proses yang didapat ini kemudian dapat diperkirakan nilai $\alpha$ yang sesuai untuk aplikasi. Nilai $\alpha$ ini menentukan seberapa lama objek menjadi *foreground*. Misalkan objek yang masuk ke *scene* akan dianggap sebagai *background* jika objek tersebut tidak bergerak selama 5 detik, maka nilai $\alpha$ yang dibutuhkan untuk kecepatan webcam 30 fps adalah:

- Untuk CPU ➔ $\frac{\log(0,9)}{\log(1-\alpha)} = 22\ x\ 5\ detik$ ➔ $\alpha \approx 0{,}001$
- Untuk GPU ➔ $\frac{\log(0,9)}{\log(1-\alpha)} = 30\ x\ 5\ detik$ ➔ $\alpha \approx 0{,}0007$

Jika aplikasi hendak dijalankan dengan kecepatan webcam 6 fps, maka nilai $\alpha$ yang digunakan baik dengan atau tanpa GPU adalah:

$\frac{\log(0,9)}{\log(1-\alpha)} = 6\ fps\ x\ 5\ detik$ ➔ $\alpha \approx 0{,}0035$

Dari tabel

**Tabel 4.3 Data Pengujian Aplikasi**di atas terlihat bahwa GPU menggunakan memori RAM jauh lebih banyak daripada CPU. Hal ini sangat wajar mengingat GPU menjalankan beberapa proses secara bersamaan sehingga setiap inti akan membutuhkan memori untuk menyimpan data hasil pengolahan. Hal ini berbeda dengan CPU yang memiliki inti sedikit sehingga memori yang dibutuhkan juga sedikit.

Dari data-data ini selanjutnya pengguna dapat menentukan apakah akan menggunakan CUDA atau tidak. Jika menggunakan CUDA, pengguna akan mendapatkan respons aplikasi yang sangat cepat namun memori yang dibutuhkan cukup besar. Sebaliknya jika tidak ingin menggunakan CUDA pengguna hanya membutuhkan memori yang sedikit namun responsnya tidak secepat jika menggunakan CUDA. Semua kembali kepada selera masing-masing orang.

- Untuk Konten Gambar

Untuk konten gambar, nilai $\alpha$ yang digunakan adalah 0,01. Pemrograman tidak dilakukan untuk GPU karena adanya penambahan algoritma yang tidak bisa dilakukan dengan GPU. Jika pemrograman dilakukan dengan GPU maka akan terdapat banyak pertukaran data dari CPU ke GPU dan sebaliknya sehingga hal tersebut tidak efisien.

### 4.3.3 Output Aplikasi

- Untuk Konten Video

Setelah standar deviasi minimum aplikasi diatur dengan baik, maka aplikasi akan merespons pergerakan sesuai pengaturan yang telah ditetapkan pengguna. Jika pengguna memilih untuk menghentikan video ketika tidak ada pergerakan,

maka seketika itu juga video akan dijeda. Jika pengguna ingin satu video diputar sampai selesai saat ada pergerakan, maka video akan dijeda setelah selesai. Begitu gerakan terdeteksi kembali, pemutaran video akan dilanjutkan. Penghentian dan pemutaran video ini berjalan dengan baik tanpa ada masalah.

Jika pengguna memilih untuk memadamkan layar monitor jika tidak ada pergerakan, maka aplikasi pun akan segera mengirimkan sinyal kepada layar monitor untuk padam. Namun, pemadaman layar monitor ini ternyata menimbulkan satu masalah besar. Layar monitor membutuhkan jeda waktu beberapa milidetik sampai beberapa detik untuk dapat padam. Jika sinyal penyalaan layar monitor dikirim sebelum layar monitor benar-benar padam, maka layar hanya akan berkedip dan tidak menyala. Masalah ini menjadi jauh lebih besar jika sinyal pemadaman dikirim secara berulang-ulang tanpa jeda waktu yang cukup. Akibatnya layar monitor akan berkedip sebanyak jumlah sinyal yang dikirimkan. Mengingat kinerja aplikasi yang bisa mencapai 56 kali per detik, maka layar monitor bisa saja berkedip hampir selama 1 menit untuk pengiriman sinyal pemadaman selama satu detik.

Jika sinyal pemadaman telah dikirimkan, maka sinyal tersebut tidak bisa dibatalkan. Untuk mengatasi masalah ini, aplikasi perlu diberikan waktu tunda selama setidaknya satu detik begitu sinyal pemadaman dikirim (dalam rentang waktu ini aplikasi tidak bisa merespons klik atau *keyboard*). Dengan demikian pergerakan baru bisa terdeteksi lagi paling cepat setelah satu detik. Satu detik ini bisa jadi tidak cukup untuk membuat layar monitor padam. Oleh karena itu, sinyal untuk menghidupkan layar monitor dikirim 10 kali selama 2 detik seraya video diputar kembali.

Jam kerja aplikasi diatur standar mulai pukul 6.30 pagi sampai 17.00, namun pengguna bisa menyesuaikan jam kerja tersebut. Yang menjadi fokus persoalan adalah bahwa aplikasi hanya akan berjalan dalam rentang jam kerja tersebut. Di luar jam kerja tersebut aplikasi tidak akan bisa menyalakan kamera ataupun memutar video. Pengguna juga bisa membuat aplikasi memadamkan komputer di akhir jam kerja. Pemadaman ini sifatnya adalah hibernasi di mana keadaan komputer setelah dinyalakan kembali akan sama seperti sebelum dipadamkan.

- Untuk Konten Gambar

Pada konten berupa gambar, konten akan berganti jika aplikasi mendeteksi pergerakan yang tidak terlalu cepat dan tidak terlalu lambat. Sebagaimana disebutkan di atas, gerakan yang bisa dideteksi aplikasi adalah jika dalam rentang jarak 80 piksel objek memiliki 5-6 gambar yang terekam oleh webcam.

Jika arah gerakan adalah ke kanan, maka aplikasi akan mengganti konten ke konten selanjutnya dan jika sudah mencapai konten terakhir, maka aplikasi tidak merespons gerakan ke kanan. Sebaliknya jika arah gerakan ke kiri maka aplikasi akan mengganti konten ke konten sebelumnya dan jika sudah mencapai konten pertama maka aplikasi tidak akan merespons gerakan ke kiri.

# BAB V
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Dari penelitian ini dapat disimpulkan hal-hal sebagai berikut:

1. Aplikasi yang dibuat mendeteksi gerak berdasarkan analisis standar deviasi untuk konten audio-video dan analisis kontur untuk konten gambar.
2. Nilai standar deviasi hasil *background subtraction* adalah antara 0 – 127,5.
3. Dalam analisis kontur, pergerakan yang direspons aplikasi adalah jika dalam rentang jarak 80 piksel objek memiliki 5-6 gambar yang terekam oleh webcam;
4. Dalam jarak 69 cm dari webcam, kecepatan objek minimum yang bisa dideteksi adalah 0,63 m/s dan kecepatan maksimumnya adalah 1,17 m/s;
5. Nilai $\alpha$ yang digunakan dalam aplikasi untuk kecepatan webcam 30 fps adalah $\alpha = 0{,}0007$ jika menggunakan CUDA, $\alpha = 0{,}001$ jika tanpa CUDA, dan untuk kecepatan webcam 6 fps adalah $\alpha = 0{,}0035$.

## 5.2 Saran

Penelitian ini memang masih belum sempurna, masih banyak kekurangan di mana-mana. Oleh karena itu, dalam penelitian selanjutnya sebaiknya hal-hal berikut diperhatikan supaya sistem yang dibuat bisa lebih baik:

1. Efisiensi energi listrik dalam sistem hendaknya diperhatikan agar didapatkan sistem yang hemat energi;
2. Aplikasi hendaknya dibuat lebih interaktif sehingga pemirsa videotron dapat memilih konten yang ingin dilihatnya;

3. Jika memungkinkan algoritma pemadaman dan penyalaan layar monitor sebaiknya diperbarui dengan yang lebih baik;
4. Manajemen memori sebaiknya turut diperhatikan agar sistem tidak menghabiskan memori;
5. Pengembangan selanjutnya sebaiknya langsung diaplikasikan ke ranah yang lebih besar seperti videotron sesungguhnya yang terintegrasi dengan pendingin udara.

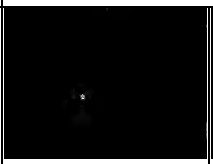
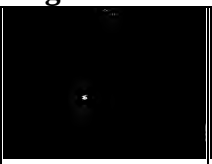
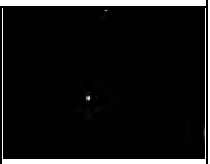
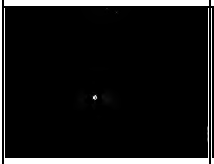
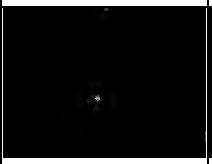
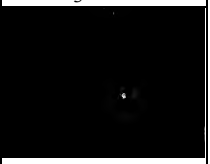
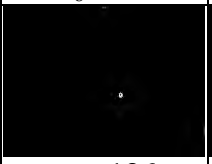
# DAFTAR LAMPIRAN

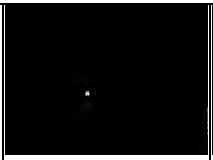
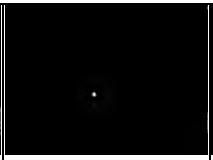
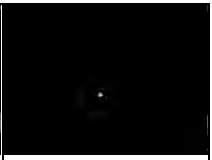
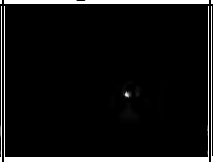
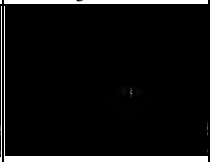
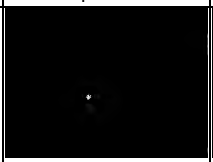
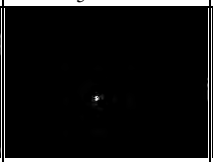
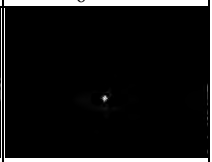
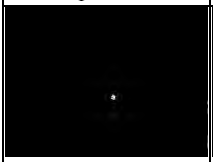
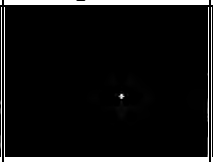
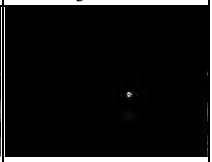
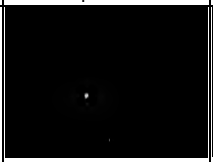
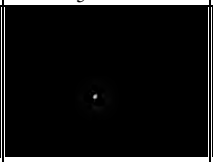
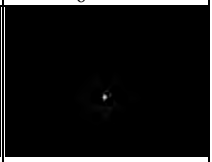
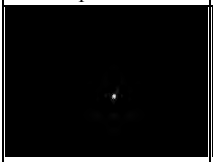
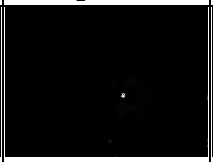
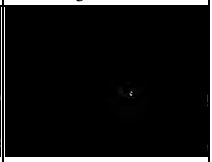
*"Halaman ini sengaja dikosongkan"*

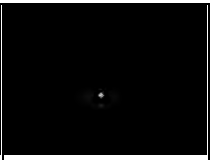
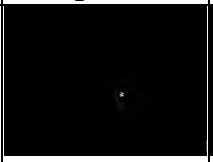
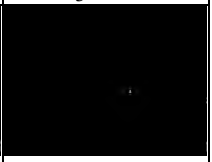
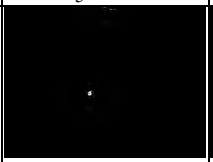
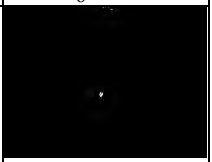
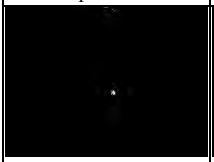
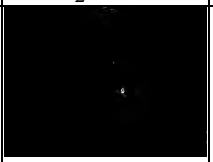
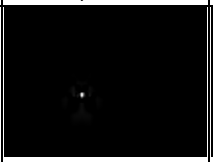
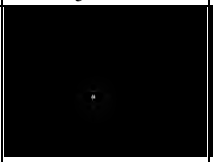
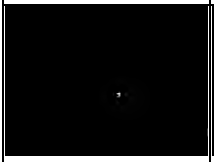
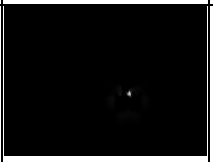
# LAMPIRAN

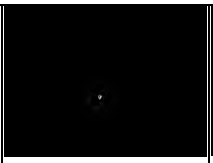
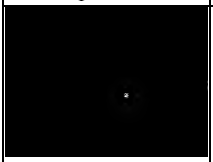
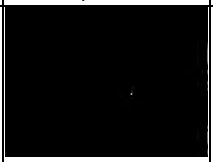
## 1. Gambar Uji Coba Aplikasi

**Tabel 1 Hasil Uji Coba Aplikasi**

| No. | h | Hasil background subtraction | | |
|---|---|---|---|---|
| 1 | 1 cm |  |  |  |
| | | $x_1$=123 | $x_2$=128 | $x_3$=134 |
| | |  |  |  |
| | | $x_4$=141 | $x_5$=148 | $x_6$=161 |
| | |  |  |  |
| | | $x_7$=164 | $x_8$=172 | $x_9$=174 |
| | |  |  |  |
| | | $x_{10}$=175 | $x_{11}$=186 | $x_{12}$=196 |
| | |  | | |
| | | $x_{13}$=198 | | |

| | | | | |
|---|---|---|---|---|
| 2 | 2 cm |  |  |  |
| | | $x_1$=129 | $x_2$=139 | $x_3$=150 |
| | |  |  |  |
| | | $x_4$=182 | $x_5$=192 | $x_6$=198 |
| 3 | 3 cm |  |  |  |
| | | $x_1$=131 | $x_2$=144 | $x_3$=157 |
| | |  |  |  |
| | | $x_4$=170 | $x_5$=183 | $x_6$=194 |
| 4 | 4 cm |  |  |  |
| | | $x_1$=127 | $x_2$=142 | $x_3$=156 |
| | |  |  |  |
| | | $x_4$=171 | $x_5$=186 | $x_6$=198 |

| | | | | |
|---|---|---|---|---|
| 5 | 5 cm |  |  |  |
| | | $x_1$=122 | $x_2$=135 | $x_3$=152 |
| | |  |  |  |
| | | $x_4$=169 | $x_5$=185 | $x_6$=198 |
| 6 | 6 cm |  |  |  |
| | | $x_1$=121 | $x_2$=136 | $x_3$=152 |
| | |  |  | |
| | | $x_4$=170 | $x_5$=187 | |
| 7 | 7 cm |  |  |  |
| | | $x_1$=123 | $x_2$=140 | $x_3$=159 |
| | |  |  | |
| | | $x_4$=179 | $x_5$=197 | |

| | | | | |
|---|---|---|---|---|
| 8 | 8 cm |  |  |  |
| | | $x_1$=131 | $x_2$=150 | $x_3$=170 |
| | |  | | |
| | | $x_4$=190 | | |
| 9 | 9 cm |  |  |  |
| | | $x_1$=123 | $x_2$=144 | $x_3$=166 |
| | |  | | |
| | | $x_4$=187 | | |
| 10 | 10 cm |  | | |
| | | $x_1$=199 | | |

Keterangan: x menyatakan posisi horizontal bandul dalam piksel.

## 2. Data Pengambilan Kecepatan Proses

**Tabel 1 Data pengambilan kecepatan proses per detik pada siang hari**

| No. | Dengan CUDA | Tanpa CUDA |
|---|---|---|
| 1 | 30 | 21 |
| 2 | 31 | 22 |
| 3 | 30 | 21 |
| 4 | 30 | 21 |
| 5 | 31 | 22 |
| 6 | 29 | 21 |
| 7 | 30 | 24 |
| 8 | 30 | 23 |
| 9 | 30 | 23 |
| 10 | 31 | 23 |
| 11 | 29 | 21 |
| 12 | 31 | 23 |
| 13 | 30 | 22 |
| 14 | 30 | 22 |
| 15 | 30 | 22 |
| 16 | 30 | 23 |
| 17 | 31 | 21 |
| 18 | 30 | 21 |
| 19 | 29 | 23 |
| 20 | 31 | 23 |
| **Rata-rata** | **30,15** | **22,1** |
| **Standar Deviasi** | **0,67** | **0,97** |
| **Keseksamaan** | **97,78%** | **95,62%** |

**Tabel 2 Data pengambilan kecepatan proses per detik pada malam hari**

| No. | Dengan CUDA | Tanpa CUDA |
|---|---|---|
| 1 | 6 | 6 |
| 2 | 6 | 6 |
| 3 | 6 | 6 |
| 4 | 6 | 6 |
| 5 | 6 | 6 |
| 6 | 6 | 7 |
| 7 | 6 | 5 |
| 8 | 6 | 6 |
| 9 | 6 | 6 |
| 10 | 7 | 6 |
| 11 | 5 | 6 |
| 12 | 6 | 6 |
| 13 | 6 | 6 |
| 14 | 6 | 6 |
| 15 | 6 | 7 |
| 16 | 6 | 5 |
| 17 | 6 | 6 |
| 18 | 6 | 6 |
| 19 | 6 | 6 |
| 20 | 6 | 6 |
| **Rata-rata** | **6** | **6** |
| **Standar Deviasi** | **0,32** | **0,46** |
| **Keseksamaan** | **94,59%** | **92,35%** |

# DAFTAR PUSTAKA


Aranda, R., Hernandez-Lopez, F., Francisco Madrigal, 2013. Tutorial: OpenCV & CUDA.

Bradski, G., Kaehler, A., 2008. Learning OpenCV. O'Reilly Media, Inc., United States of America.

Chen, W., 2010. Real-Time Palm Tracking and Hand Gesture Estimation Based on Fore-Arm Contour. National Taiwan University of Science and Technology, Taiwan.

Halliday, D., Resnick, R., 2003. Fundamental of Physics. John Wiley & Sons, Inc, USA.

Jähne, B., 2005. Digital Image Processing. Springer-Verlag Berlin Heidelberg, Netherlands.

Paris, S., 2015. Naive Image Smoothing: Gaussian Blur. www.people.csail.mit.edu.

Piccardi, M., 2004. Background subtraction techniques: a review.

Weber, K., 2012. CMOS: Ready for Broadcast Today. www.grassvalley.com, USA.

www.nvidia.com, 2014. CUDA C PROGRAMMING GUIDE.

www.opencv.org, 2014. The OpenCV Tutorials.

www.wikipedia.org, 2014. Canny edge detector.

*"Halaman ini sengaja dikosongkan"*

# BIODATA

Muhammad Fahmil Huda dilahirkan di Kudus, 10 Desember 1991, anak kedua dari 4 bersaudara. Pendidikan formal penulis ditempuh di MI NU Al-Khurriyah 01 Besito (1998-2004), MTs NU TBS Kudus (2004-2007), MA NU TBS Kudus (2007-2010), dan Jurusan Fisika ITS Surabaya (2010-2015). Pendidikan penulis di perguruan tinggi ditempuh melalui beasiswa PBSB dari kemenag RI.

Penulis pernah aktif dalam jurnalistik madrasah (2008-2010), Koran Pelajar Radar Kudus (Edisi perdana Januari 2010), Himasika ITS (2012-2013), dan CSSMoRA ITS (2010-2013). Saat ini penulis aktif menulis di blog hdqbasith.blogspot.com. Beberapa karya penulis dapat ditemukan di blog tersebut. Penulis juga dapat dihubungi melalui blog tersebut atau melalui email fahmil.huda@gmail.com.

Saat ini penulis berencana untuk membuat beberapa karya di bidang pemrograman, terutama pemrograman web. Hal ini tak lepas dari kecintaan penulis terhadap hal-hal berbau komputer. Salah satu karya pertama yang ingin penulis realisasikan adalah sistem manajemen online di Madrasah NU TBS Kudus yang akan menjadi tempat penulis untuk mengabdikan ilmunya.