TUGAS AKHIR – KI141502

DETEKSI API BERBASIS SENSOR VISUAL MENGGUNAKAN METODE *SUPPORT VECTOR MACHINES*

HAMDI AHMADI MUZAKKIY
NRP 5112100091

Dosen Pembimbing I
Prof. Ir. Handayani Tjandrasa, M.Sc.,Ph.D

Dosen Pembimbing II
Dr. Chastine Fatichah, S.Kom, M.Kom

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

TUGAS AKHIR – KI141502

DETEKSI API BERBASIS SENSOR VISUAL MENGGUNAKAN METODE *SUPPORT VECTOR MACHINES*

HAMDI AHMADI MUZAKKIY
NRP 5112100091

Dosen Pembimbing I
Prof. Ir. Handayani Tjandrasa, M.Sc.,Ph.D

Dosen Pembimbing II
Dr. Chastine Fatichah, S.Kom, M.Kom

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

FIRE DETECTION BASED ON VISION SENSOR USING SUPPORT VECTOR MACHINES

HAMDI AHMADI MUZAKKIY
NRP 5112100091

Supervisor I
Prof. Ir. Handayani Tjandrasa, M.Sc.,Ph.D

Supervisor II
Dr. Chastine Fatichah, S.Kom, M.Kom

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2015

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

DETEKSI API BERBASIS SENSOR VISUAL MENGGUNAKAN METODE SUPPORT VECTOR MACHINES

TUGAS AKHIR

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

Oleh
HAMDI AHMADI MUZAKKIY
NRP : 5112 100 091

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Prof. Ir. Handayani Tjandrasa, M.Sc.,Ph.D .....
   NIP: (Pembimbing 1)

2. Dr. Chastine Fatichah, S.Kom, M.Kom .....
   NIP: (Pembimbing 2)

SURABAYA
DESEMBER, 2015

[Halaman ini sengaja dikosongkan]

Deteksi Api Berbasis Sensor Visual Menggunakan Metode Support Vector Machines

Nama Mahasiswa	**: HAMDI AHMADI MUZAKKIY**
NRP	**: 5112100091**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Prof. Ir. Handayani Tjandrasa, M.Sc.,Ph.D**
Dosen Pembimbing 2	**: Dr. Chastine Fatichah, S.Kom, M.Kom**

Abstrak

Kebakaran adalah salah satu bencana yang sering terjadi. Penyebab sering terjadinya kebakaran yaitu karena kelalaian manusia, dan hubungan arus pendek listrik. Bencana kebakaran tidak hanya merusak bangunan bahkan menimbulkan banyak korban. Saat ini banyak alat pendeteksi api menggunakan sensor panas, ion, infrared. Namun penggunaan sistem alarm ini tidak akan bekerja hingga partikel mencapai sensor. Oleh kerena itu diperlukan sistem deteksi api yang dapat mendeteksi kebakaran dengan cepat.

Dalam Tugas Akhir ini diimplementasikan perangkat lunak pendeteksi api menggunakan deteksi gerak, deteksi warna menggunakan probabilitas warna, region growing, dan deteksi bentuk menggunakan fitur wavelet dengan metode klasifikasi support vector machines. Hasil dari deteksi bentuk akan digunakan dalam proses penentuan api.

Dataset yang digunakan dalam proses uji coba berisi lima puluh tiga video api dengan panjang video enam sampai enambelas detik yang diambil dari berbagai sumber. Performa terbaik yang dihasilkan adalah true positif sebesar 92.96%, false negatif sebesar 0.54% dan missing rate sebesar 6.50%

Kata Kunci: Deteksi Gerak, Deteksi Warna, Probabilitas, Wavelet, Support Vector Machines.

[Halaman ini sengaja dikosongkan]

FIRE DETECTION BASED ON VISION SENSOR USING SUPPORT VECTOR MACHINES

Student's Name	**: HAMDI AHMADI MUZAKKIY**
Student's ID	**: 5112100091**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Prof. Ir. Handayani Tjandrasa, M.Sc.,Ph.D**
Second Advisor	**: Dr. Chastine Fatichah, S.Kom, M.Kom**

Abstract

Fire is one of the disasters that often occur. The cause of frequent occurrence of fires are due to human negligence and short-circuits. Fire not only damaged buildings even cause many casualties. Currently many of fire detector uses heat sensors, ion, and infrared. However, the use of this alarm system will not work until the particles reach the sensor. Therefore, there has to be a fire detection system that can detect fires quickly.

In this final project is implemented fire-detection software uses motion detection, color detection using color probabilities, region growing, and shape detection using wavelet features with classification method of support vector machines. The Results from shape detection will be used in the process of determining the fire.

The dataset used in the testing process contains fifty-three videos with a flame length of six to sixteen second video taken from various sources. The resulting performance is best at 92.96% true positive, 0.54% false negative and missing rate of 6.50%

Keywords: Motion Detection, Color Detection, Probability, Wavelet, Support Vector Machines.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Puji syukur penulis kehadirat Tuhan YME karena berkat rahmat dan karunia-NYA penulis dapat menyelesaikan Tugas Akhir yang berjudul

DETEKSI API BERBASIS SENSOR VISUAL MENGGUNAKAN METODE SUPPORT VECTOR MACHINES

Tugas Akhir ini merupakan salah satu syarat untuk memperoleh gelar Sarjana Komputer di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Penulis ingin menyampaikan terima kasih yang sebesar-besarnya atas dukungan dan semangat yang diberikan dan membantu penulis baik secara langsung ataupun tidak dalam menyelesaikan Tugas Akhir ini. Penulis ingin mengucapkan terima kasih kepada

1. Tuhan YME karena berkat rahmat dan karunianya penulis berhasil menyelesaikan Tugas Akhir dengan baik.
2. Kedua orang tua, dan keluarga penulis, terima kasih atas doa dan bantuan moral dan material selama penulis belajar di Teknik Informatika ITS.
3. Ibu Dr. Eng. Nanik Suciati, S.Kom., M.Kom., selaku ketua jurusan Teknik Informatika ITS
4. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku Koordinator Tugas Akhir di Teknik Informatika ITS.
5. Ibu Prof. Ir. Handayani Tjandrasa, M.Sc.,Ph.D selaku Dosen Pembimbing I Tugas Akhir yang telah memberikan bimbingan dan dukungan selama penulis menyelesaikan Tugas Akhir.

6. Ibu Dr. Chastine Fatichah, S.Kom, M.Kom selaku pembimbing II Tugas Akhir yang telah memberikan banyak waktu untuk berdiskusi dan memberi semangat dan motivasi kepada penulis untuk menyelesaikan Tugas Akhir.
7. Bapak dan Ibu Dosen di Jurusan Teknik Informatika yang telah memberikan ilmu selama penulis kuliah di Teknik Informatika
8. Seluruh Staf dan karyawan Teknik Informatika yang telah memberikan bantuan selama penulis kuliah di Teknik Informatika.
9. Rekan-rekan di laboratorium Pemrograman yang telah bersedia dan betah dengan adanya penulis di lab selama pengerjaan Tugas Akhir.

Penulis Mohon maaf apabila terdapat kekurangan dalam penulisan Tugas Akhir ini. Kritik dan saran penulis harapkan untuk perbaikan dan pembelajaran di kemudian hari. Semoga Tugas Akhir ini dapat memberikan Manfaat yang sebesar besarnya.

Surabaya, Desember 2015

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini dibahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metedologi, dan sistematika laporan tugas akhir. Diharapkan dari penjelasan dalam bab ini gambaran Tugas Akhir secara umum dapat dipahami.

1.1 Latar Belakang

Banyaknya kamera pengawas yang digunakan pada bangunan-bangunan saat ini tidak terlalu optimal digunakan, hal ini dikarenakan masih banyaknya kamera pengawas yang masih diawasi oleh operator. Oleh karena itu, pengaplikasian pemrosesan gambar sangat penting untuk mempermudah dalam pendeteksian suatu objek. Salah satu deteksi objek yang saat ini penting untuk digunakan adalah deteksi api, banyak dari sistem yang sekarang digunakan dalam mendeteksi api adalah penggunaan sensor panas, ion, atau infrared yang bergantung pada karakteristik tertentu seperti api, asap, suhu, atau radiasi.

Penggunaan deteksi api berdasarkan sensor visual memberikan banyak keuntungan. Pertama, peralatan yang digunakan relatif murah seperti sistem yang berbasis CDC (*Charge Coupled Device*) *cameras*, yang mana sudah banyak dipasang ditempat umum. Kedua, kecepatan untuk mendeteksi lebih cepat karena kamera tidak menunggu asap atau panas menyebar. Ketiga,petugas dapat mengkonfirmasi keberadaan api tanpa mengunjungi lokasi kejadian.

Dari masalah yang ada, tujuan dari usulan tugas akhir ini yaitu, membuat sistem deteksi api menggunakan rekaman video. Data yang akan digunakan adalah data rekaman video, dalam prosesnya sistem akan memproses gambar setiap *frame* dengan jumlah frame yang telah ditentukan. Setiap *frame* akan dilakukan *preprocessing*, dan terakhir akan dilakukan verifikasi. Metode yang digunakan dalam tugas akhir ini adalah

1) Reduksi *size frame*; 2) Deteksi gerak; 3) Deteksi warna piksel; 4) *Region growing*; 5) Perhitungan luasan *region*; 6) Ekstraksi Fitur; 7) Klasifikasi menggunakan *support vector machines*.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir dapat dipaparkan sebagai berikut.

a) Bagaimana melakukan deteksi warna piksel.
b) Bagaimana melakukan *region growing* pada setiap kandidat piksel.
c) Bagaimana melakukan ekstraksi fitur pada setiap kandidat piksel.
d) Bagaimana melakukan klasifikasi untuk verisikasi piksel api.

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir memiliki beberapa batasan, yakni sebagai berikut.

1. Implementasi menggunakan bahasa pemrograman Python berbasis desktop.
2. Jumlah piksel objek api yang dideteksi lebih besar dari 1% dari luas piksel *frame*.
3. Data yang digunakan adalah data video dengan panjang video 6 – 16 detik.
4. Data video memiliki ukuran 240 x 320 piksel dengan *channel* R,G,B.
5. Warna api yang didefinisikan adalah *range* warna kuning hingga merah.
6. Pergerakan dari kamera tidak terlalu besar.
7. Pantulan dari objek api termasuk kedalam objek api.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah merancang dan membangun perangkat lunak deteksi api menggunakan data video secara *real time*.

1.5 Manfaat

Pengerjaan tugas akhir ini dilakukan dengan harapan bisa memberikan kontribusi pada sistem keamanan kebakaran dan mempercepat deteksi kebakaran.

1.6 Metodologi

Metodologi yang dipakai pada pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan Proposal Tugas Akhir
   Tahap awal yang dilakukan dalam pengerjaan Tugas Akhir ini adalah penyusunan proposal Tugas Akhir. Di dalam proposal diajukan suatu gagasan pembuatan perangkat lunak untuk melakukan deteksi api menggunakan data video.
2. Studi Literatur
   Pada tahap ini dilakukan pencarian, pengumpulan, penyaringan, pemahaman, dan pembelajaran literatur yang berhubungan dengan reduksi *size frame*, deteksi gerak, deteksi warna piksel, *region growing*, *wavelet*, dan *support vector machines*. Literatur yang digunakan meliputi: buku referensi, jurnal, dan dokumentasi internet
3. Implementasi dan pembuatan perangkat lunak
   Pada tahap ini dilakukan impelementasi perangkat lunak sesuai dengan rancangan perangkat lunak yang dibuat.
4. Uji coba dan Evaluasi
   Pada tahap ini dilakukan uji coba terhadap perangkat lunak yang telah dibuat untuk mengetahui kemampuan algoritma yang dipakai, mengamati kinerja sistem, serta mengidentifikasi kendala yang mungkin timbul. Parameter yang diujicobakan adalah parameter *threshold* pada deteksi warna piksel, dan nilai konstanta C dan kernel pada klasifikasi *support vector machines*.
5. Penyusunan Laporan Tugas Akhir
   Pada tahap ini dilakukan penyusunan laporan pengerjaan Tugas Akhir yang berisi dasar teori, dokumentasi dari

perangkat lunak, dan hasil yang diperoleh selama pengerjaan Tugas Akhir.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Pada bab ini dibahas mengenai teori-teori dan metodelogi yang menjadi dasar dari pembuatan Tugas Akhir.

2.1 Adaptive Background Mixture Models

Adaptive background mixture models adalah metode deteksi gerak. Didefinisikan piksel pada *frame* yang sedang di proses sebagai *X* dengan nilai sebelumnya dari *frame* 1 sampai *frame* t. Model dapat dilihat pada persamaan berikut.

$$\{X_1, X_1, \ldots, X_i, \ldots, X_t\}\, 1 \leq i \leq t \tag{2.1.1}$$

Probabilitas piksel X_t didefinisikan sebagai berikut.

$$P(X_t) = \sum_{i=1}^{K} w_{i,t} * \eta(X_t, \mu_{i,t}, cov_{i,t}) \tag{2.1.2}$$

Dimana K adalah bilangan gaussian (antara 3 hingga 5), cov adalah *covariance* yang didapatkan dari $variance \,.\, matriks\ identitas$, w adalah *weight*, η adalah *gaussian probability destiny function*. *Gaussian probability destiny function* dapat dilihat pada persamaan berikut.

$$\eta(x, \mu, cov) = \frac{1}{(2\pi)^{\frac{n}{2}} |cov|^{\frac{1}{2}}}\, e^{-\frac{1}{2}(x-\mu)^2 cov^{-1}(x-\mu)} \tag{2.1.3}$$

Distribusi K dirutkan secara *descending* berdasarkan nilai w_k/σ_k , nilai distribusi B akan digunakan sebagai model *background*. Nilai B didapatkan menggunakan persamaan berikut.

$$B = argmin(\sum_{j=1}^{b} w_j > T) \quad \textbf{(2.1.4)}$$

Dimana nilai T adalah nilai minimum dari *background model*. Selanjutnya melakukan *background subtraction* dengan menghitung nilai piksel dengan distribusi B. Jika nilai piksel lebih dari 2.5 dari nilai standar deviasi distribusi tersebut (*match*), maka komponen distribusi tersebut dilakukan *update* dan piksel tersebut dianggap sebagai *foreground*. Untuk *update* nilai komponen gaussian dilakukan dengan persamaan berikut [1].

$$w_k^{N+1} = (1-\alpha)w_k^N + \alpha p(w_k|x_{n+1}) \quad \textbf{(2.1.5)}$$

$$\mu_k^{N+1} = (1-\alpha)\mu_k^N + \rho x_{n+1} \quad \textbf{(2.1.6)}$$

$$\sigma_k^{N+1} = (1-\alpha)\sigma_k^N + \rho(x_{N+1} - \mu_k^{N+1})(x_{N+1} - \mu_k^{N+1})^T \quad \textbf{(2.1.7)}$$

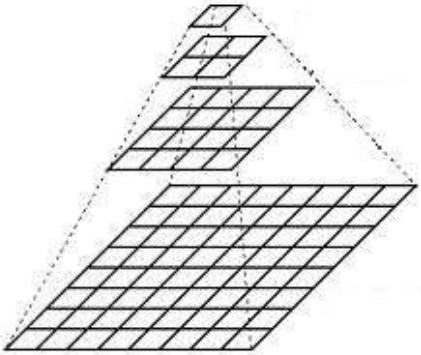
$$\rho = \alpha\, \eta(X_{N+1}, \mu_k^N, cov_k^{N+1}) \quad \textbf{(2.1.8)}$$

$$p(w_k|x_{n+1}) = \begin{cases} 1, & w_k \text{ adalah komponen} \\ \text{gaussian yang match pertama kali} & \\ 0, & \text{selain itu} \end{cases} \quad \textbf{(2.1.9)}$$

Jika dari semua distribusi piksel yang dicek tidak memenuhi syarat (*unmatch*) maka komponen gaussian yang terakhir akan dilakukan *update*. *Update* dilakukan dengan mengubah rata-rata (μ) dengan nilai piksel yang sedang dicek dan mengubah nilai variasi (σ^2) dengan nilai tinggi dan nilai *weight* (w) dengan nilai yang rendah.

2.2 Gaussian Pyramid

Gaussian pyramid digunakan untuk melakukan reduksi resolusi citra. Citra yang tereduksi resolusinya akan berkurang menjadi setengah dari resolusi awal. Hal ini dilakukan untuk mempercepat proses perhitungan.



Gambar 2.2.1 Ilustrasi Gaussian Pyramid

Gaussian pyramid dilakukan dengan dua operasi, yaitu *smoothing* dan *down sampling*. *Smoothing* dilakukan dengan menggunakan filter 5x5. *Smoothing* dilakukan dengan menggunakan persamaan berikut.

$$g_1 = w * g_0 \quad \textbf{(2.2.1)}$$

Dimana w adalah filter 5x5. Detail persamaan setiap piksel dilakukan menggunakan persamaan berikut.

$$g_1(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) \, . \, g_0\,(i+m, j+n) \quad \textbf{(2.2.2)}$$

Setelah mendapatkan citra yang telah di *smoothing*, langkah selanjutnya adalah melakukan *down sampling*. *Down sampling* dilakukan untuk mengubah resolusi citra asli menjadi resolusi yang lebih kecil. *Down sampling* dilakukan dengan persamaan berikut.

$$g_2(i,j) = g_1\,(2i, 2j) \quad \textbf{(2.2.3)}$$

Untuk melakukan reduksi, dilakukan menggunakan dua proses tersebut, yaitu *smoothing* dan *down sampling*. Perhitungan dapat dilakukan dengan menggabungkan ke dua persamaan tersebut menjadi [2].

$$g_1(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n)\,.\; g_0\,(2i+m, 2j+n) \quad \textbf{(2.2.4)}$$

2.3 Probabilitas Distribusi Gaussian

Probabilitas distribusi gaussian adalah sebuah metode untuk menghitung probabilitas dari suatu data. Probabilitas dilakukan dengan menghitung nilai rata-rata dan standar deviasi dari suatu data. Persamaan umum probabilitas distribusi gausian dapat dilihat pada persamaan berikut [3].

$$p = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad \textbf{(2.3.1)}$$

Pada kasus tugas akhir ini, probabilitas gaussian digunakan untuk mendapatkan probabilitas warna piksel. Setiap piksel dihitung nilai probabilitas R,G,B. Setelah dilakukan perhitungan probabilitas R,G,B selanjunya probabilitas tersebut dikalikan sehingga mendapatkan nilai probabilitas piksel. Perhitungan probabilitas setiap piksel dapat dilihat pada persamaan berikut.

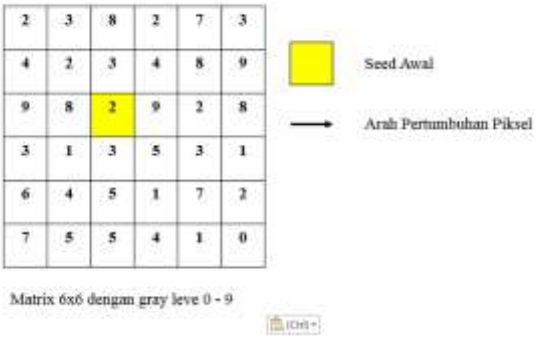
$$p(I(x,y)) = \prod_{i \in \{R,G,B\}} p_i(I_i(x,y)) \quad \textbf{(2.3.2)}$$

2.4 Region Growing

Region Growing adalah metode untuk melakukan segmentasi citra. Pendekatan dasar adalah degan memulai titik

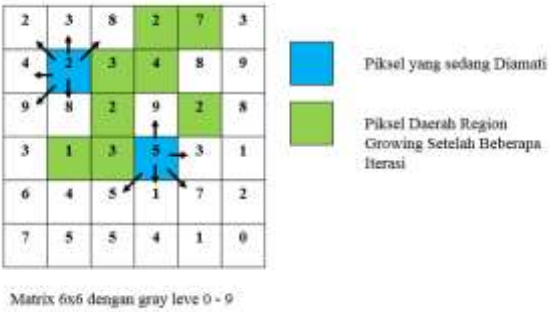
yang sudah diinisialisasi (*seed*) dan dari titik tersebut dilakukan menumbuhkan daerah dengan cara menambahkan tetangga piksel dari *seed* yang mempunyai kesamaan dengan *seed* [4].

Untuk metode ini, dibutuhkan aturan yang mengatur mekanisme tumbuhnya *seed* dan suatu aturan lain yang menguji kehomogenan dari *region* setelah satu tahap tumbuh selesai.pertumbuhan *region* dimulai dari seed awal dengan menambahkan tetangga piksel (menggunakan 8-tetangga) yang serupa untuk menumbuhkan *region.* Ketika suatu pertumbuhan *region* selesai, langkah selanjutnya adalah memilih *seed* baru dan melakukan *region growing* kembali. Proses tersebut dilakukan hingga semua piksel berhasil dikelompokkan dalam beberapa *region*.



Matrix 6x6 dengan gray leve 0 - 9

Gambar 2.4.1 *Seed* Awal *Region Growing*



Gambar 2.4.2 Piksel yang diamati dan *Region*

2.5 Daubachies D4 Wavelet

Wavelet adalah fungsi matematika yang membagi data menjadi beberapa komponen frekuensi yang berbeda-beda dan menganalisis setiap komponen tersebut dengan menggunakan resolusi yang sesuai dengan skalanya. Transformasi *wavelet* mendekomposisi signal kedalam *frequency bands* dengan memproyeksikan signal kedalam set fungsi dasar [5]. Pada citra, transformasi *wavelet* adalah transformasi yang melakukan filter terhadap suatu masukkan. Filter yang digunakan dalam *wavelet* adalah *high pass filter* dan *low pass filter*. Pada daubachies D4, terdapat dua koefisien yang digunakan untuk melakukan *filter*, yaitu *scaling function coefficient* dan *wavelet function coefficient*. Dimana *scaling function coeficient* adalah koefisien yang digunakan dalam melakukan *low pass filter*, sedangkan *wavelet function coeficient* digunakan dalam melakukan *high pass filter* . Berikut persamaan *scaling function coeficient.*

$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}} \qquad \textbf{(2.5.1)}$$

$$h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}$$

$$h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}$$

$$h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$$

Wavelet function coeficient dapat dilihat pada persamaan berikut.

$$g_0 = \frac{1-\sqrt{3}}{4\sqrt{2}} \quad \textbf{(2.5.2)}$$

$$g_1 = \frac{-3+\sqrt{3}}{4\sqrt{2}}$$

$$g_2 = \frac{1+\sqrt{3}}{4\sqrt{2}}$$

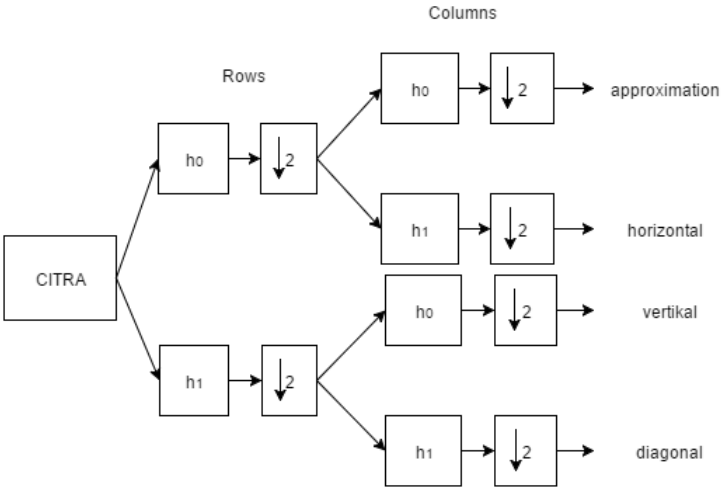
$$g_3 = \frac{-1-\sqrt{3}}{4\sqrt{2}}$$

Tahapan dari proses *filter* adalah melakukan *filter* terhadap baris citra terlebih dahulu. Dilakukan *high pass filter* dan *low pass filter*, dimana hasil dari proses ini adalah dua citra, yaitu citra *high pass* dan citra *low pass*. Ilustrasi tahap *filter* dapat dilihat pada Gambar 2.5.1

$$\begin{matrix} h_0 & h_1 & h_2 & h_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & h_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_0 & h_1 & h_2 & h_3 \end{matrix} \cdot \begin{matrix} s_1 \\ s_0 \\ s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_3 \\ s_2 \end{matrix}$$

Gambar 2.5.1 Ilustrasi *Low Pass Filter*

Citra yang dilakukan *filter* akan ter-reduksi menjadi setengah. Dilanjutkan dengan melakukan *filter* terhadap kolom citra, dilakukan *high pass filter* dan *low pass filter*. Hasil dari proses ini adalah empat citra dengan satu citra approximasi, dan tiga citra detail. Citra detail yang didapat adalah detail horizontal, vertikal dan diagonal. Gambar 2.5.2 Adalah lustrasi dari proses *filter* yang dilakukan.



Gambar 2.5.2 Ilustrasi *Filter* yang dilakukan

Ilustrasi citra yang dikeluarkan pada proses ini dapat dilihat pada Gambar 2.5.4.

Gambar 2.5.3 Citra Masukkan

Gambar 2.5.4 Citra Keluaran Hasil *Wavelet*

2.6 Normalisasi Min-Max

Normalisasi adalah sebuah proses untuk mengubah suatu data ke dalam rentang nilai tertentu. Tujuannya adalah untuk menghindari persebaran data yang terlalu jauh sehingga

sebuah variabel tidak mendominasi terhadap variabel lain. Salah satu jenis normalisasi adalah normalisasi skala. Pada normalisasi, skala rentang yang umum digunakan yaitu 0 hingga 1. Rumus umum skala adalah sebagai berikut .

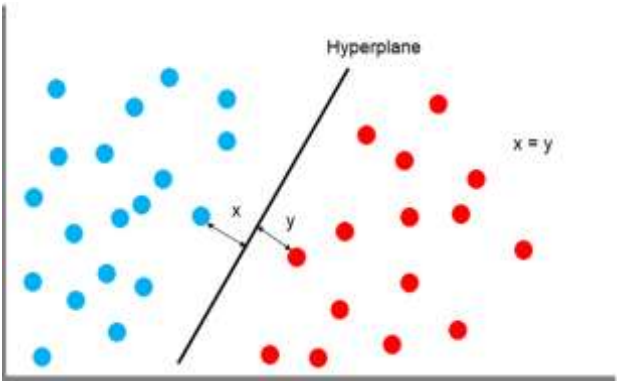
$$Y_{new} = \frac{(Y - Y_{min})(Y_{newmax} - Y_{newmin})}{Y_{max} - Y_{min}} + Y_{newmin} \quad \textbf{(2.6.1)}$$

Variabel Y adalah nilai yang akan dinormalisasi. Variabel Y_{min} dan Y_{max} adalah nilai minimum dan maximum pada nilai-nilai atribut dimana Y berada. Variabel Y_{newmin} dan Y_{newmax} adalah nilai minimum dan maximum yang diinginkan. Apabila rentang yang digunakan adalah 0 hingga 1 maka rumus diatas menjadi seperti berikut [6].

$$Y_{new} = \frac{(Y - Y_{min})}{Y_{max} - Y_{min}} \quad \textbf{(2.6.2)}$$

2.7 Support Vector Machines

Support vector machines adalah metode klasifikasi yang mengklasifikasikan dua kelas, yaitu kelas +1 dan -1. Pada metode klasifikasi *support vector machines*, akan dibentuk suatu *hyperplane*. *Hyperplane* adalah garis pemisah yang memisahkan dua kelas yang berbeda. Dalam metode *support vector machines* dikenal istilah *margin*. *Margin* adalah jarak antara kelas +1 dengan kelas -1 yang paling dekat, pada *support vector machines* dicari *margin* terpanjang antara dua kelas tersebut. Ilustrasi *support vector machines* dapat dilihat pada Gambar 2.7.1.



Gambar 2.7.1 Ilustrasi *Support Vector Machines*

Diberikan masukkan berupa data belajar $(\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \ldots, \boldsymbol{x_n})$ dan masing-masing kelas dianotasikan $\boldsymbol{y_i} \in \{-\mathbf{1}, +\mathbf{1}\}$ untuk i = 1,2,3,…,l , dimana l adalah banyaknya data. Fungsi *hyperplane* dibuat dengan persamaan berikut.

$$w \,.\, x + b = 0 \qquad \textbf{(2.7.1)}$$

Dalam mencari nilai w dan b yang optimal, dilakukan dengan persamaan berikut.

$$\min \frac{1}{2} \, ||w||^2 + C \sum_{i=1}^{l} t_i \qquad \textbf{(2.7.2)}$$

Persamaan diatas mempunyai variabel C, dimana variabel tersebut adalah konstanta nilai pinalti dari kesalahan klasifikasi. Pencarian nilai w dan b dilakukan dengan batasan yang ditulis menggnakan persamaan berikut.

$$y_i(wx_i + b) + \, t_i \, \geq 1 \qquad \textbf{(2.7.3)}$$

Persamaan (2.7.2) dilakukan untuk mencari nilai w dan b yang yang optimum. Fungsi tujuan Persamaan (2.7.2)

berbentuk kuadrat. Untuk menyelesaikannya, bentuk tersebut ditransformasi kedalam bentuk *dual space*. Persamaan *dual space* dapat ditulis menggunakan persamaan berikut.

$$\max \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j {x_i}^T x_j \quad \textbf{(2.7.4)}$$

Dengan batasan sebagai berikut.

$$\alpha_i \geq 0 \, , \sum_{i=1}^{l} \alpha_i y_i = 0 \quad \textbf{(2.7.5)}$$

Untuk mencari nilai α_i yang optimum digunakan *quadratic programming*. Setelah mendapatkan nilai α_i, persamaan *hyperplane* dilakukan dengan persamaan berikut.

$$f = w^T z + b = \sum_{i=1}^{s} \alpha_i y_i \, {x_i}^T z + b \quad \textbf{(2.7.6)}$$

Dimana *z* adalah masukan data masukkan. Pada banyak kasus, data yang diklasifikasikan tidak bisa langsung dipisahkan dengan garis yang linear. Oleh karena itu, digunakan metode kernel untuk mengatasi permasalahan tersebut. Dengan metode kernel, suatu data x di *input space* dimapping ke fitur *space* F dengan dimensi yang lebih tinggi. Salah satu kernel yang biasa dipakai adalah kernel RBF, *Polynomial*, *Linear*. Persamaan kernel RBF, *Polynomial*, *Linear* berurutan dapat dilihat pada persamaan berikut.

$$k(x, y) = \exp(\frac{-||x - y||^2}{2\sigma^2}) \quad \textbf{(2.7.7)}$$

$$k(x, y) = (x^T y + 1)^p \quad \textbf{(2.7.8)}$$

$$k(x, y) = x^T y \quad \textbf{(2.7.9)}$$

Penggunaan fungsi kernel mengubah persamaan *training*. Persamaan tersebut menjadi.

$$\max \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j k(x_i, x_j) \quad \textbf{(2.7.10)}$$

Persamaan *hyperplane* diubah menjadi persamaan berikut.

$$f = \sum_{i=1}^{s} \alpha_i y_i \, k(x_i, z) + b \quad \textbf{(2.7.11)}$$

[Halaman ini sengaja dikosongkan]

BAB III
DESAIN PERANGKAT LUNAK

Pada bab ini dijelaskan mengenai rancangan sistem perangkat lunak yang akan dibuat. Perancangan yang dijelaskan meliputi data dan proses. Data yang dimaksud adalah data yang akan diolah dalam perangkat lunak baik digunakan sebagai pembelajaran maupun pengujian sehingga tujuan Tugas Akhir ini bisa tercapai. Proses yaitu tahap-tahap yang ada dalam sistem sebagai pengolah data meliputi reduksi *size frame*, deteksi gerak, deteksi warna piksel, *region growing*, perhitungan luasan *region* api, ekstraksi fitur dan klasifikasi.

3.1 Data

Pada sub bab ini akan dijelaskan mengenai data yang digunakan sebagai masukan perangkat lunak untuk selanjutnya diolah dan dilakukan pengujian sehingga menghasilkan data keluaran yang diharapkan.

3.1.1 Data Masukkan

Data masukkan adalah data yang digunakan sebagai masukan dari sistem. Data yang digunakan adalah data video yang memiliki frekuensi minimal 20 Hz, kualitas video yang digunakan adalah 240x320 piksel.

Gambar 3.1.1 Contoh Data Masukkan

3.1.2 Data Pembelajaran

Data pembelajaran digunakan sebagai data belajar klasifikasi. Data yang digunakan adalah data video yang dibagi dalam dua jenis yaitu video berisi objek api dan video berisi objek bukan api.

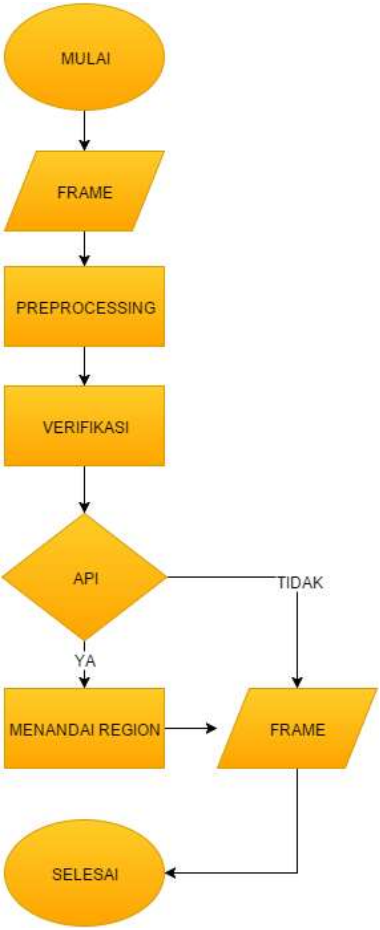
3.1.3 Data Keluaran

Data masukkan akan diputar ulang dan diproses setiap *frame* menggunakan metode reduksi *frame*, deteksi gerak, deteksi warna piksel, *region growing*, perhitungan luasan *region* api, ekstraksi fitur, dan klasifikasi. Dari proses tersebut diharapkan keluaran berupa tanda pada tiap frame jika memiliki piksel api. Tanda yang dimaksud adalah tanda berwarna biru yang menghubungkan empat nilai titik koordinat ekstrim piksel api.

Gambar 3.1.2 Contoh Data Keluaran

3.2 Desain Sistem Secara Umum

Rancangan perangkat lunak deteksi api berbasis sensor visual menggunakan *support vector machines* dimulai dengan membaca masukan berupa file video. Proses deteksi api terdiri dari dua proses besar, yaitu *preprocessing* dan verifikasi. Diagram alir desain umum perangkat lunak ditunjukkan pada Gambar 3.2.1



Gambar 3.2.1 Diagram alir rancangan perangkat lunak secara umum

Setiap *frame* akan dilakukan *preprocessing* sebelum dilakukan verifikasi menggunakan *support vector machines*. Tahap pertama *preprocessing* adalah mengubah *size frame* yang diproses. Ukuran *frame* diubah menjadi dua kali lebih kecil dari ukuran *frame* yang diproses. Tahap berikutnya adalah malakukan deteksi gerak *frame*, hasil dari deteksi gerak adalah piksel-piksel bergerak terhadap *frame* sebelumnya. Setelah

mendapatkan piksel-piksel bergerak, tahap selanjutnya adalah deteksi warna setiap piksel menggunakan probabilitas distribusi gaussian. Warna api yang didefinisikan pada sistem ini adalah warna yang memiliki *range* antara warna kuning hingga merah. Hasil keluaran dari metode deteksi warna piksel adalah piksel-piksel yang masuk kedalam kandidat piksel api.

Setelah mendapatkan kandidat piksel api, dilakukan metode *region growing* untuk mendapatkan *region* kandidat piksel api. Hasil dari *region growing* digunakan pada tahap selanjutnya yaitu perhitungan luasan *region* api. Pada metode perhitungan luasan *region* , jika luasan *region* melebihi *threshold*, maka kandidat piksel yang masuk pada *region* tersebut merupakan kandidat piksel api selanjutnya.

Setelah melalui tahap *preprocessing*, piksel-piksel yang termasuk kandidat api akan di verifikasi menggunakan metode *support vector machines*. Fitur didapatkan dari nilai konstanta *wavelet* setiap piksel statis dengan sepuluh *frame* yang berurutan. Hasil akhir yang dikeluarkan adalah adanya penanda pada *frame* yang diproses jika *frame* tersebut mengandung piksel api. Data dibagi menjadi data pembelajaran dan data uji sehingga dapat diperoleh nilai *true positif, false positif,* dan *missing rate* dari hasil klasifikasi.

3.3 Preprocessing

Setiap piksel pada *Frame* yang diproses tidak langsung dilakukan verifikasi untuk menentukan apakah piksel tersebut piksel api atau bukan. Tahap awal yang dilakukan adalah *preprocessing. Preprocessing* bertujuan untuk menghilangkan piksel-piksel yang tidak memiliki karakteristik piksel api. *Preprocessing* dilakukan melalui lima tahap yaitu reduksi *size frame*, deteksi gerak, deteksi warna piksel, *region growing*, perhitungan luasan *region* api. Diagram alir tahap *preprocessing* ditunjukkan pada Gambar 3.3.1

Gambar 3.3.1 Diagram Alir *Preprocessing*

3.3.1 Reduksi Size Frame

Tahap reduksi *size frame* adalah tahap dimana *size* dari *frame* direduksi. Kualitas *frame* masukkan direduksi dengan tujuan mempercepat proses pendeteksian api. Gambar 3.3.2 adalah diagram alir dari proses reduksi *size frame*.

Gambar 3.3.2 Digram Alir Reduksi *Size Frame*

Masukkan dari tahap ini adalah *frame* yang sedang diproses dalam *channel* R,G,B. Proses reduksi dilakukan menggunakan metode *gaussian pyramid*. Hasil keluaran dari tahap ini adalah *frame* yang telah tereduksi kolom dan barisnya menjadi setengah dari *frame* awal.

3.3.2 Deteksi Gerak

Didalam proses ini dilakukan proses deteksi gerak piksel. Masukkan dari proses ini adalah *frame* yang sedang di proses. Untuk mendapatkan kandidat piksel yang bergerak, dilakukan dengan metode *adaptive background mixture model*. Dari metode tersebut didapatkan piksel-piksel yang bergerak. Ilustrasi deteksi gerak ditunjukkan pada Gambar 3.3.3.

Gambar 3.3.3 Contoh Deteksi Gerak

Setelah mendapatkan piksel yang bergerak, kumpulan piksel tersebut disimpan untuk diproses pada tahap berikutnya yakni deteksi warna piksel.

3.3.3 Deteksi Warna Piksel

Penentuan warna piksel yang termasuk kandidat piksel api menggunakan metode distribusi probabilitas gaussian. Masukkan dari tahap ini adalah *frame* dengan *channel* R,G,B yang sedang di proses dan *list* piksel bergerak yang didapatkan pada tahap deteksi gerak. Setiap piksel bergerak dihitung nilai perkalian probabilitas R,G,B dan ditentukan apakah piksel tersebut masuk kedalam kandidat piksel api atau tidak.

Sebelum melakukan perhitungan probabilitas pada setiap piksel yang bergerak, dilakukan proses mendapatkan nilai rata-rata dan standar deviasi *channel* R,G,B. Nilai rata-rata dan standar deviasi *channel* R,G,B didapatkan dari data yang

diambil dari memproses *dataset* gambar api sebanyak sebelas gambar.

Gambar 3.3.4 Contoh *Dataset* Gambar Api

Perhitungan probabilitas piksel dilakukan dengan mencari probabilitas setiap nilai R,G,B menggunakan probabilitas distribusi gaussian. Selanjutnya nilai probabilitas R,G,B dikalikan untuk mendapatkan nilai probabilitas dari piksel tersebut. Jika nilai probabilitas piksel tersebut melebihi *threshold*, maka piksel tersebut dimasukkan kedalam kandidat piksel api. *Pseudocode* fungsi ini dapat dilihat pada Gambar 3.3.5

```
1.  for i=1 to length(CandidatePiksel)
2.    R,G,B = CandidatePiksel[i].RGBvalue
3.    If          probability(R)*probability(G)*
    probability(B) > threshold
4.       add CandidatePiksel[i] to firePiksel
5.    else
6.       add CandidatePiksel[i] to nonFirePiksel
7.  return firePiksel
```

Gambar 3.3.5 *Pseudocode* Deteksi Warna Piksel

Keluaran dari tahap ini adalah *list* kandidat piksel api yang akan diproses selanjutnya di tahap *region growing* dan luasan *region* api.

3.3.4 Region Growing

Kandidat piksel api yang didapatkan pada tahap deteksi warna piksel dilakukan *region growing* untuk mendapatkan *region* dari setiap piksel. Masukkan dari tahap ini adalah *list* kandidat piksel api dan *frame* yang diproses. Setiap piksel akan dilakukan *region growing* dengan cara melakukan pengecekan terhadap delapan tetangga piksel tersebut. Untuk menentukan

apakah piksel tetangga tersebut termasuk *region* api atau tidak, dilakukan dengan cara menentukan nilai probabilitas warna R,G,B menggunakan distribusi probabilitas gaussian. Jika nilai probabilitas tersebut melebihi *threshold*, maka piksel tersebut dianggap satu *region*. *Pseudocode* fungsi ini dapat dilihat pada Gambar 3.3.6.

```
1.  regionCounter = 0
2.  region.size = image.size
3.  clockWise = clockWise()
4.  for i=1 to length(CandidatePiksel)
5.    push    CandidatePiksel[i]    to    stack
6.  regionStack
7.    if CandidatePiksel[i].is_visit == False
8.       increment regionCounter
9.    while regionStack is not empty
10.     pop regionStack and assign to temporary
11.     R,G,B = temporary.RGBvalue
        if
    probability(R)*probability(G)*probability(B)
12. > threshold and temporary.is_visit == False
13.      temporary.is_visit = True
14.      region[temporary] = regionCounter
15.      for j to clockWise
           push  temporary.index  -  j  to  stack
16. region Stack
    return region
```

Gambar 3.3.6 *Pseudocode Regon Growing*

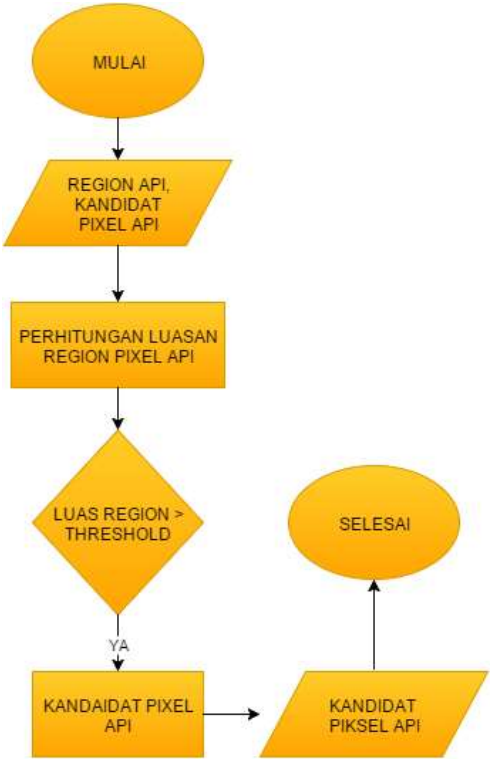
Keluaran dari tahap *region growing* adalah *region* dengan nilai yang berbeda tiap *region*. Nilai tersebut membedakan antara satu *region* dengan *region* lainnya.

3.3.5 Perhitungan Luasan Region Api

Perhitungan luasan *region* api adalah metode untuk melakukan *filter* terhadap *region* api. Masukkan dari tahap ini adalah *region* yang telah diperoleh dari metode *region growing*, kandidat piksel api yang didapatkan dari metode deteksi warna piksel. Pada tahap ini dilakukan proses perhitungan luasan

setiap *region*. Jika luasan *region* melebihi *threshold*, maka kandidat piksel api yang ada pada *region* tersebut masuk kedalam kandidat piksel api.

Gambar 3.3.7 adalah diagram alir dari proses luasan *region* api.

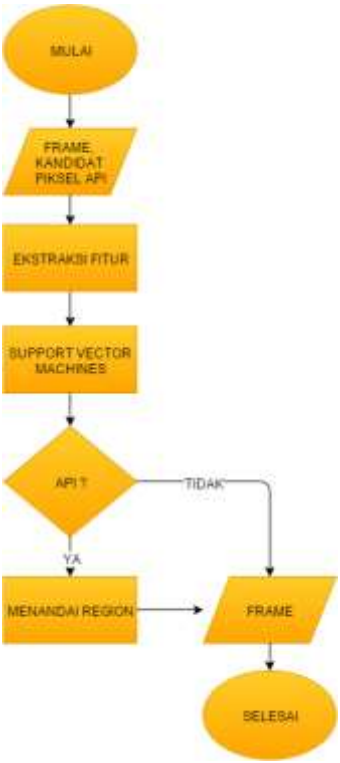


Gambar 3.3.7 Diagram Alir Luasan *Region* Api

Keluaran dari proses ini adalah *list* piksel yang termasuk kedalam kandidat piksel api.

3.4 Verifikasi

Pada sub bab ini akan dijelaskan mengenai proses verifikasi piksel sehingga menghasilkan keluaran yang diharapkan seperti yang sudah dijelaskan sebelumnya. Setelah melalui tahap *preprocessing*, piksel-piksel yang masuk kedalam kandidat piksel api dilakukan tahap verifikasi menggunakan metode klasifikasi *support vector machines*. Sebelum dilakukan klasifikasi, terdapat proses untuk mendapatkan fitur sebagai data masukan klasifikasi. Pencarian fitur dilakukan dengan mengubah gambar spasial kedalam domain *wavelet*. Setelah mendapatkan fitur yang dicari, dilakukan klasifikasi untuk menentukan apakah piksel tersebut masuk kedalam piksel api atau bukan. Diagram alir tahap *preprocessing* ditunjukkan pada Gambar 3.4.1



Gambar 3.4.1 Diagram Alir Verifikasi

3.4.1 Ekstraksi Fitur dengan Wavelet

Data masukkan yang digunakan pada ekstraksi fitur adalah data *frame* sebanyak sepuluh *frame* secara berurutan. *Frame* diambil dari *frame* yang sedang diproses berserta sembilan *frame* sebelumnya. Ekstraksi fitur dilakukan dengan mengubah *frame* kedalam domain *wavelet*. *Frame* yang diubah kedalam domain *wavelet* ini berupa sepuluh *frame* yang berurutan. Tiap *frame* akan menghasilkan empat sub *frame* baru, yaitu sub *frame low-low* (LL), *low-high* (LH), *high-low* (HL), dan *high-high* (HH). Untuk ekstraksi fitur, sub *frame*

yang digunakan adalah sub *frame* LL, LH, HH. *Wavelet* yang digunakan adalah *wavelet* daubechies D4 . Tiap kandidat piksel api dicari nilai piksel pada sub *frame wavelet* yang telah dilakukan. Setiap piksel mendapatkan tiga nilai untuk setiap *framenya*, yaitu nilai LL, LH, HH. Dari ketiga nilai tersebut akan dihitung menggunakan persamaan :

$$M_n(x,y) = |LH_n(x,y)|^2 + |HL_n(x,y)|^2 + |HH_n(x,y)|^2 \qquad \textbf{(3.4.1)}$$

Total nilai yang didapatkan adalah sepuluh buah nilai fitur. Sebelum masuk tahap klasifikasi, nilai fitur yang didapatkan dinormalisasi menggunakan normalisasi min-max. Nilai min-max didapatkan dengan melakukan perhitungan nilai kuadrat sub *frame* LH, HL, HH. Selanjutnya nilai LH, HL, dan HH dikalkulasi menjadi satu, dicari nilai minimum dan maximum dari hasil perhitungan tersebut. Dari proses ini akan dilakukan terhadap seluruh *frame*, dimana nilai min-max yang didapatkan ada sepuluh buah. Normalisasi dilakukan dengan menyesuaikan nilai min-max terhadap *frame* yang dicari nilai *wavelet* setiap pikselnya. Setelah dilakukan normalisasi, dilakukan pengurutan nilai fitur secara *ascending*.

Gambar 3.4.2 adalah diagram alir dari proses ekstraksi fitur.

Gambar 3.4.2 Diagram Alir Ekstraksi Fitur

Fitur akhir yang didapatkan adalah fitur dengan dimensi sebesar sepuluh dimensi yang sudah terurut. Selanjutnya akan masuk kedalam tahap klasifikasi.

3.4.2 Klasifikasi

Klasifikasi dilakukan dengan mengolah data fitur yang sudah didapatkan dari proses ekstraksi fitur. Metode klasifikasi yang digunakan adalah *support vector machines*. Masukkan dari tahap klasifikasi adalah nilai fitur *wavelet* sebanyak sepuluh dimensi yang sudah dijelaskan sebelumnya . Hasil akhir dari proses ini adalah piksel-piksel yang masuk kedalam piksel api.

3.5 Menandai Region Api

Penanda *region* api dilakukan untuk menunjukkan bagian yang terdeteksi api. Masukkan dari proses ini adalah piksel-piksel yang lolos tahap verifikasi dan *frame* yang sedang di proses. *Pseudocode* fungsi ini dapat dilihat pada Gambar 3.3.6.

```
1.  Min_x, Max_x = min(X), max(X)
2.  Min_y, Max_y = min(Y), max(Y)
3.  For i = Min_y to Max_y:
4.     For j = Min_x to Max_x :
5.        Mark(Image[i][j])
6.  Return Image
```

Gambar 3.5.1 *Pseudocode* fungsi menandai *region*

Hasil yang dikeluarkan pada proses ini adalah *frame* yang memiliki tanda jika terdapat objek api.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Implementasi kode program dilakukan dengan menggunakan bahasa Python.

4.1 Lingkungan Implementasi

Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam implementasi ini ditampilkan pada Tabel 4.1.1.

Perangkat	Spesifikasi
Perangkat keras	Prosesor: Intel® Core™ i3-2350M CPU @ 2.30GHz 2.30GHz Memori: 4.00 GB
Perangkat lunak	Sistem Operasi: Microsoft Windows 8 64-bit Pro Perangkat Pengembang: PyCharm Perangkat Pembantu: Microsoft Excel 2013

Tabel 4.1.1 Lingkungan Perancangan Perangkat Lunak

4.2 Implementasi

Sub bab implementasi ini menjelaskan tentang implementasi proses yang sudah dijelaskan pada bab desain perangkat lunak.

4.2.1 Implementasi Tahap Reduksi Size Frame

Sub bab ini membahas implementasi tahap reduksi *size frame*. Pada tahap ini data masukan berupa *frame* dan data keluaran yang dihasilkan pada tahap ini adalah *frame* yang telah

tereduksi kualitasnya. Implementasi dilakukan dengan menggunakan fungsi yang sudah disediakan oleh OpenCV yaitu *pyrDown()* dan di tunjukkan oleh Kode Sumber 4.2.1

```
1.  currentFrame2 = copy.copy(currentFrame)
2.  currentFrame =
    ImageProcessing.getDownSize(currentFrame)
```

Kode Sumber 4.2.1 Implementasi Tahap Reduksi *Size Frame*

```
1.  def getDownSize(self,image):
2.          return cv2.pyrDown(image)
```

Kode Sumber 4.2.2 Penggunaan Fungsi pyrDown()

Kode Sumber 4.2.1 melakukan reduksi dengan memanggil fungsi *getDownSize()* dam disimpan kedalam variabel *currentFrame*. Variabel *currentFrame2* akan digunakan sebagai masukan dari ekstraksi fitur dan *frame* keluaran yang ditampilkan, sedangkan proses lainnya akan menggunakan variabel *currentFrame*.

4.2.2 Implementasi Tahap Deteksi Gerak

Sub bab ini membahas implementasi tahap deteksi gerak piksel. Masukkan dari tahap ini adalah *frame* yang sedang diproses. Implementasi dilakukan menggunakan fungsi yang sudah disediakan oleh OpenCV yaitu *BackgroundSubtractorMOG()*. Implementasi ditunjukkan oleh Kode Sumber 4.2.3

```
1.  def getMovingForeGround(self, image):
2.     self.BckgrSbsMOG =
    cv2.BackgroundSubtractorMOG()
3.     return
    self.BckgrSbsMOG.apply(image,learningRate =
    0.0005)
```

Kode Sumber 4.2.3 Implementasi Tahap Deteksi Gerak

Hasil yang dikembalikan oleh fungsi getMovingForeGround() adalah gambar dengan nilai tiap

pikselnya antara 0 atau 255 dengan. Dimana nilai 255 adalah nilai piksel yang bergerak terhadap piksel pada *frame* sebelumnya. Hasil dari fungsi *getMovingForeGround()* tidak secara langsung diproses kedalam tahap selanjutnya. Dari hasil *frame* ini akan diambil indeks piksel yang bergerak, implementasi ditunjukkan oleh Kode Sumber 4.2.4

```
1.  def getMovingCandidatePiksel(self,
    moving_frame):
2.     listY,listX = np.where( moving_frame ==
    255 )
3.     return np.vstack((listY,listX))
```

Kode Sumber 4.2.4 Implementasi Penyimpanan Piksel

Fungsi *getMovingCandidatePiksel()* melakukan iterasi untuk mendapatkan piksel-piksel yang bernilai 255. Fungsi ini mengembalikan hasil berupa *list* indeks piksel yang bergerak. Hasil keluaran dari keseluruhan proses deteksi gerak adalah *list* yang berisi indeks piksel y dan x.

4.2.3 Implementasi Tahap Deteksi Warna Piksel

Deteksi warna piksel dilakukan dengan pengecekan warna setiap piksel pada *list* piksel yang dihasilkan melalui proses deteksi gerak. Masukkan dari tahap ini adalah *list* piksel yang telah didapatkan pada tahap deteksi gerak dan *frame* dengan *channel* R,G,B. Sebelum menghitung probabilitas warna piksel, terlebih dahulu dilakukan perhitungan mencari nilai rata-rata dan standar deviasi untuk nilai piksel R,G,B. Sebelas gambar api disimpan dan dilakukan proses perhitungan nilai rata-rata dan nilai standar deviasi untuk setiap *channel*. Pada implementasi sistem, proses menghitung nilai rata-rata dan standar deviasi setiap *channel* dapat dilihat pada Kode Sumber 4.2.5

```
1.  def getStdDevAndMean(self,path):
2.     list_file = File.readFolder(self,path)
3.     R = []
4.     G = []
```

```
5.      B = []
6.      for x in list_file:
7.          image =
    ImageProcessing.readImage(self,path+'/'+x)
8.          r = np.array(image[:,:,2]).ravel()
9.          g = np.array(image[:,:,1]).ravel()
10.         b = np.array(image[:,:,0]).ravel()
11.         R += (r.tolist())
12.         G += (g.tolist())
13.         B += (b.tolist())
14.     mean = []
15.     mean.append(np.average(B))
16.     mean.append(np.average(G))
17.     mean.append(np.average(R))
18.     standard_deviasi = []
19.     standard_deviasi.append(np.std(B))
20.     standard_deviasi.append(np.std(G))
21.     standard_deviasi.append(np.std(R))
22.     return standard_deviasi, mean
```

Kode Sumber 4.2.5 Implementasi Menghitung Nilai Standar Deviasi dan Rata-Rata Setiap *Channel*

Pada Kode Sumber 4.2.5 setiap gambar *dataset* dibaca dan dilakukan pemisahan setiap *channel*. Selanjutnya, dilakukan perhitungan rata-rata dan perhitungan standar deviasi setiap *channel*. Setelah didapatkan nilai rata-rata dan standar deviasi setiap *channel*, proses penentuan kandidat api menggunakan metode probabilitas warna piksel dilakukan. Setap piksel dihitung nilai probabilitas *channel* R,G,B. Selanjutnya, menghitung nilai probabilitas piksel dengan mengalikan nilai probabilitas R,G,B. Proses perhitungan probabilitas piksel memakan waktu yang cukup lama jika menghitung setiap kemungkinan kandidat piksel api yang lolos pada tahap deteksi gerak. Pada implementasi, nilai probabilitas tiap piksel yang masuk kedalam probabilitas warna api dimasukkan kedalam file. Nilai yang disimpan adalah nilai R,G,B piksel yang termasuk kedalam warna piksel api. Hal ini dilakukan karena proses perhitungan yang lama jika menghitungan probabilitas setiap piksel. Isi dari file adalah *list* kemungkinan piksel-piksel yang masuk dalam probabilitas

warna api. Proses *generate list* piksel api dapat dilihat pada Kode Sumber 4.2.6

```
1.  def getListColorPiksel
    (self,list_standard_deviasi, list_mean):
2.     res = []
3.     threshold = self.getThreshold()
4.     for B in range(0,256):
5.        for G in range(B,256):
6.           for R in xrange(G,256):
7.              if
8.  Data.getGaussianProbability(self,B,
    list_standard_deviasi[0], list_mean[0])*
    Data.getGaussianProbability(self,G,
    list_standard_deviasi[1], list_mean[1])*
    Data.getGaussianProbability(self,R,
    list_standard_deviasi[2], list_mean[2]) >
    threshold:
9.                 res.append([B,G,R])
10. return res
```

Kode Sumber 4.2.6 *Generate list* piksel api

Kode Sumber 4.2.6 melakukan iterasi pengecekan piksel. Iterasi tidak dilakukan sebanyak kombinasi nilai R,G,B. hal ini dikarenakan nilai *channel* R dari piksel api lebih besar dari nilai *channel* G dan nilai *channel* G lebih besar dari nilai *channel* B [7]. Pada Kode Sumber 4.2.6, dilakukan pemanggilan fungsi *getGaussianProbability()*. Fungsi *getGaussianProbability()* digunakan untuk menghitung nilai probabilitas tiap *channel* piksel. Fungsi tersebut mengimplementasikan rumus dari probabilitas distribusi gaussian. Implementasi fungsi *getGaussianProbability()* dapat dilihat pada Kode Sumber 4.2.7.

```
1.  def getGaussianProbability(self, data,
    standard_deviasi, mean):
2.     data = float(data)
3.
```

```
    standard_deviasi =
4.  float(standard_deviasi)
5.     mean = float(mean)
6.     result = pow((data-mean),2)
7.     div = 2*pow(standard_deviasi,2)
8.     exp = np.exp(-result/div)
       result =
9.  standard_deviasi*np.sqrt(2*np.pi)
10     result = 1/result
       return result* exp
```

Kode Sumber 4.2.7 Fungsi Menghitung Nilai Probabilitas Distribusi Gaussian

```
1.  def getThreshold(self):
2.     return 5*pow(10,-9)
```

Kode Sumber 4.2.8 Mendapatkan Threshold

Kode Sumber 4.2.7 melakukan perhitungan probabilitas piksel api setiap *channel*. Nilai *threshold* pada Kode Sumber 4.2.8 didapatkan dari hasil analisa yang telah dilakukan. Ketika program dijalankan, program akan membaca file yang berisi *list* piksel yang sudah di *generate* sebelumnya dan menyimpan nilai piksel tersebut *array*. Proses pembacaan file bisa dilihat pada Kode Sumber 4.2.9

```
1.  def getFireArray(self,path):
2.     lists = [[[False for k in xrange(256)]
    for j in xrange(256)]for i in xrange(256)]
3.     data = open(path,'r')
4.     for x in data:
5.        color = (x.split('\n')[0]).split(' ')
6.        lists[int(color[0])]
    [int(color[1])][int(color[2])] = True
7.     return lists
```

Kode Sumber 4.2.9 Membaca *List* Piksel Api

Pada Kode Sumber 4.2.9 dilakukan peroses pembuatan *array* tiga dimensi, dengan *default* *False* pada nilai indeks. Pada *line* enam, dilakukan perubahan nilai sesuai indeks yang ada pada *list* dengan mengubah nilai *array* tiga dimensi tersebut menjadi *True*. Nilai indeks dimensi *array* tersebut mewakilkan

indeks R,G,B. Setelah didapatkan *array* dengan indeks yang mewakili nilai R,G,B dan hasilnya, akan dilakukan pengecekan terhadap kandidat piksel yang didapatkan pada proses deteksi gerak. Implementasi proses pengecekan kandidat piksel dapat dilihat pada Kode Sumber 4.2.10

```
1.  def getColorCandidatePiksel(self,
2.  list_standard_deviasi, list_mean):
3.     true_piksel = []
4.     false_piksel = []
5.     for x in range(0,len(list_candidate[0])):
6.        data = image[list_candidate[0][x]]
    [list_candidate[1][x]]
7.        B,G,R = data[0],data[1],data[2]
8.        if color_dataset[B][G][R] == True:
9.  true_piksel.append([list_candidate[0][x],
    list_candidate[1][x]])
10.       else :
11. false_piksel.append([list_candidate[0][x],
    list_candidate[1][x]])
12.    return true_piksel,false_piksel
```

Kode Sumber 4.2.10 Implementasi Tahap Deteksi Warna Piksel

Hasil yang dikeluarkan pada tahap ini adalah *list* nilai indeks piksel yang masuk kedalam piksel api.

4.2.4 Implementasi Tahap Region Growing

Sub bab ini membahas implementasi tahap *region growing* yang menggunakan kandidat piksel api pada tahap deteksi warna api sebagai masukkan. *Region growing* dilakukan menggunakan *list* kandidat piksel api sebagai masukan awal dari piksel yang dicari *region*nya. Titik piksel kandidat api dilakukan pengecekan probabilitas warna api terhadap delapan tetangga piksel tersebut. Jika piksel tetangga termasuk piksel api, maka piksel tersebut akan ditandai sebagai region dari piksel awalan tersebut. *Region* akan diberi nomor sesuai dengan titik piksel awal dari *region* tersebut. Implementasi *region growing* dapat dilihat pada Kode Sumber 4.2.11

```
1.  def getRegionGrowing(self, list_candidate,
    images, color_dataset,counter):
2.     gray_image =
    ImageProcessing.getRGBtoGray(self,images)
3.     is_visit = gray_image*0
4.     result_image = copy.copy(gray_image)
5.     region_number = 0
6.     for x in list_candidate:
7.        coor_y = x[0]
8.        coor_x = x[1]
9.        if is_visit[coor_y][coor_x] == 0:
10.          stack = []
11.          region_number+=1
12.          stack.append([coor_y,coor_x])
13.          is_visit[coor_y][coor_x] =
    region_number
             result_image, is_visit =
14. self.doFloodFill( gray_image, result_image,
    is_visit, stack, region_number,
15. color_dataset, images)
16.    return is_visit
```

Kode Sumber 4.2.11 Implementasi Tahap *Region Growing*

Kode Sumber 4.2.11 melakukan inisialisasi *region* dengan nilai 0 pada variabel *is_visit*. Selanjutnya melakukan iterasi sebanyak kandidat piksel api yang masuk kedalam tahap ini. Setiap kandidat piksel api dilakukan pengecekan, apakah piksel tersebut sudah dilakukan *region growing* atau belum. Jika belum (nilai variabel indeks yang sedang dicek bernilai 0) , maka koordinat dari titik tersebut akan dijadikan sebagai *seed* untuk dimasukkan kedalam *stack* dan dilakukan *growing*. Implementasi *growing* dapat dilihat pada Kode Sumber 4.2.12.

```
1.  def growing(self,
    gray_image ,result_image ,is_visit, stack,
    region_number, color_dataset,
    original_image):
2.     clocks = Data.getClockwise(self)
3.     while len(stack) != 0:
4.        coory,coorx = stack[0]
5.        stack.pop(0)
6.        result_image[coory][coorx] = 255
```

```
7.        data = original_image[coory][coorx]
8.        B,G,R = data[0],data[1],data[2]
9.        for x in clocks:
10.          try :
11.             if
12.  is_visit[coory+x[0]][coorx+x[1]] == 0 and
     color_dataset[B][G][R] == True :
                    is_visit[coory+x[0]]
13.  [coorx+x[1]] = region_number
14.                 stack.append([coory+x[0],
15.  coorx+x[1]]
             except :
16.             pass
17.     return result_image,is_visit
```

Kode Sumber 4.2.12 Implementasi *Growing*

Kode Sumber 4.2.12 melakukan pegecekan terhadap *stack* piksel yang akan dicek. Jika stack masih memiliki nilai, maka nilai tersebut akan digunakan sebagai titik untuk melakukan cek terhadap tetangga piksel. Jika tetangga piksel belum mempunyai *region* dan masuk kedalam *region* tersebut, maka nilai dari variabel *is_visit* diganti sesuai dengan nilai *region* dan koordinat piksel tersebut dimasukkan kedalam *stack* untuk melakukan pengecekan tetangga selanjutnya. Pada fungsi *growing()* melakukan pemenggilan fungsi *getClockWise()*, dimana fungsi ini akan mengembalikan *array* yang akan digunakan untuk iterasi pengecekan delapan tetangga piksel. Implementasi *getClockWise()* dapat dilihat pada Kode Sumber 4.2.12.

```
1.  def getClockwise(self):
2.     clocks = []
3.     clocks.append([-1,-1])
4.     clocks.append([-1,0])
5.     clocks.append([-1,1])
6.     clocks.append([0,-1])
7.     clocks.append([0,1])
8.     clocks.append([1,-1])
9.     clocks.append([1,0])
10.    clocks.append([1,1])
```

```
11.     return clocks
```

Kode Sumber 4.2.13 Implementasi Tahap *Clock Wise*

Iterasi yang dilakukan pada *getClockWise()* akan melakukan pengecekan delapan tetangga searah jarum jam. Hasil keluaran dair ptoses ini adalah *region* yang mempunyai nomor *region* yang berbeda antar *region*.

4.2.5 Implementasi Tahap Luasan Region Api

Sub bab ini membahas implementasi tahap luasan *region* api. Masukkan dari tahap ini adalah kandidat piksel api yang telah didapatkan dari tahap probabilitas warna api, dan *region*. Pada tahap luasan *region* api dihitung banyaknya piksel yang ada pada *region* tersebut. Jika luasan piksel *region* tersebut melebihi batas, maka kandidat piksel yang ada pada *region* tersebut masuk sebagai kandidat piksel api . Implementasi mendapatkan nilai variasi warna region dapat dilihat pada Kode Sumber 4.2.14

```
1.  def getFilterSizeRegion
    (self,list_candidate,region):
2.      true_piksel = []
3.      false_piksel = []
4.      list_region = np.unique(region)
5.      threshold = dict()
6.      for x in range(1,len(list_region)):
7.          lists = np.where(region == x)
8.          if len(lists[0]) > 1*len(region)*
    len(region[0])/100:
9.              threshold[x] = True
10.         else :
11.             threshold[x] = False
12.     for x in list_candidate:
13.         coor_y = x[0]
14.         coor_x = x[1]
15.     if threshold[region[coor_y][coor_x]] ==
    True:
16.         true_piksel.append([coor_y,coor_x])
17.     else :
```

```
18.             false_piksel.append([coor_y,coor_x])
19.         return true_piksel,false_piksel
```

Kode Sumber 4.2.14 Implementasi Tahap Variasi Warna *Region*

Setiap *region* akan dilakukan iterasi dan dilakukan pengecekan. Jika luasan dari *region* tersebut memenuhi syarat, maka seluruh piksel kandidat api masuk kedala proses berikutnya. Hasil keluaran dari fungsi ini adalah *list* piksel yang yang masuk kedalam kandidat piksel api.

4.2.6 Implementasi Tahap Ekstraksi Fitur dengan Wavelet

Sub bab ini membahas implementasi tahap ekstraksi fitur. Ekstraksi fitur dilakukan dengan mengubah *frame* kedalam domain *wavelet*. Digunakan sepuluh buah *frame* yang berurutan untuk diubah kedalam domain *wavelet*. Pada implementasi setiap *frame* di ubah kedalam domain *wavelet* dan disimpan kedalam *list*. Implementasi mengubah dan manyimpan *frame* domain *wavelet* dapat di lihat pada Kode Sumber 4.2.15

```
1.  LL,(HL,LH,HH) =
    wv.toWavelet(copy.copy(grayImage))
2.  list_wavelet.append([HL,LH,HH])
```

Kode Sumber 4.2.15 Implementasi Memasukkan Nilai *Wavelet* Kedalam *List*

Pada Kode Sumber 4.2.16 ,fungsi *toWavelet()* digunakan untuk mengubah *frame* kedalam domain *wavelet*. Implementasi mengubah *frame* kedalam domain *wavelet* dapat dilihat pada Kode Sumber 4.2.16

```
1.  def toWavelet(image):
2.      return pywt.dwt2(image,'db2')
```

Kode Sumber 4.2.16 Pemanggilan Fungsi *Wavelet*

4.2.7 Implementasi Tahap Klasifikasi

Sub bab ini membahas implementasi tahap klasifikasi. Pada tahap ini sistem diberi masukkan kandidat piksel api dan sepuluh buah *frame* domain *wavelet*, dimana masing-masing *frame* berisi tiga buah sub *frame* seperti penjelasan sub bab 3.4.1 . Sebelum melakukan klasifikasi, data training terlebih dahulu diproses sebagai data *training* untuk klasifikasi. Implementasi *training* klasifikasi dapat dilihat pada Kode Sumber 4.2.17

```
1.  def getClassifier(datatraining):
2.      x,y = readDataSet(datatraining)
3.      clf = svm.SVC(kernel = 'rbf',C = 3.5)
        clf.fit(x,y)
        return clf
```

Kode Sumber 4.2.17 *Training* Klasifikasi

Setiap piksel pada kandidat piksel api dilakukan perhitungan nilai fitur seperti yang sudah dijelaskan pada sub bab 3.4.1. Implementasi klasifikasi dapat dilihat pada Kode Sumber 4.2.18

```
1.  def doClassification(classifier, list,
2.  wavelet):
3.      truePiksel = []
4.      falsePiksel = []
5.      listMax = []
6.      listMin = []
7.      cpyWavelet = np.int_(copy.copy(wavelet))
8.      cpyWavelet = np.power(cpyWavelet,2)
9.      for x in cpyWavelet:
10.        lists = np.add(np.add(x[0],x[1]) ,
    x[2])
11.        listMax.append(np.max(lists))
12.        listMin.append(np.min(lists))
13.     for x in list:
14.        data = []
15.        cnt= 0
16.        for y in wavelet:
17.
```

```
            res = pow(y[0][x[0]][x[1]],2) +
pow(y[1][x[0]][x[1]],2) +
pow(y[2][x[0]][x[1]],2)
            res = (float(res)-
float(listMin[cnt]))/(float(listMax[cnt])-
float(listMin[cnt]))
            res = float('%.2f' % res)
            cnt+=1
            data.append(res)
         data = np.sort(data)
         classes = classifier.predict(data)
         if classes == 'Api':
            truePiksel.append([x[0],x[1]])
          else :
            falsePiksel.append([x[0],x[1]])
      return truePiksel,falsePiksel
```

Kode Sumber 4.2.18 Implementasi Tahap Klasifikasi

Pada Kode Sumber 4.2.18 dilakukan iterasi sebanyak kandidat piksel api yang lolos ketahap verifikasi. Setiap piksel akan dihitung nilai fitur *wavelet*. Dilakukan normalisasi nilai fitur yang dilakukan pada *line* 18. Jika hasil klasifikasi suatu fitur adalah api, maka nilai indeks dari piksel tersebut akan dimasukkan kedalam *list*. Hasil yang dikeluarkan dari proses ini adalah *list* piksel api yang lolos tahap verifikasi.

4.2.8 Implementasi Tahap Menandai Region Api

Sub bab ini membahas implementasi menandai *region* api. Piksel api yang sudah lolos tahap verifikasi selenjutnya diproses sebagai data keluaran yang ditampilkan. Masukkan dari tahap ini adalah kandidat piksel api dan *frame* dari variabel currentFrame2. Karena perbedaan ukuran antara indeks piksel api, dilakukan normalisasi indeks. Dilakukan penyesuaian indeks-indeks piksel dengan *frame* keluaran. Implementasi *training* klasifikasi dapat dilihat pada Kode Sumber 4.2.19.

```
1.  def markingFire(self, list_fire, image,
    constanta):
2.     if len(list_fire) == 0:
```

```
3.          return image
4.      list = np.array(list_fire)
5.      min_y,max_y =
    min(list[:,0]),max(list[:,0])
6.      min_x,max_x =
    min(list[:,1]),max(list[:,1])
7.      distance_y = int((max_y-
    min_y)/2)*constanta
8.      distance_x = int((max_x-
    min_x)/2)*constanta
9.      center_point =
    [int((max_y+min_y)*constanta/2) ,
    int((max_x+min_x)*constanta/2)]
10.     min_y,min_x,max_y,max_x =
    center_point[0]-distance_y, center_point[1]
    - distance_x, center_point[0] + distance_y,
    center_point[1] +distance_x
11.     for y in range(min_y,max_y+1):
12.         image[y][min_x] = [255,191,0]
13.         image[y][max_x] = [255,191,0]
14.     for x in range(min_x,max_x+1):
15.         image[min_y][x] = [255,191,0]
16.         image[max_y][x] = [255,191,0]
17.     return image
```

Kode Sumber 4.2.19 Implementasi Tahap Menandai Region Api

Hasil keluaran dari Kode Sumber 4.2.19 adalah *frame* dengan tanda persegi jika terdapat piksel api pada *frame* yang diproses.

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan hasil uji coba dan evaluasi program yang telah selesai diimplementasi.

5.1 Lingkungan Uji Coba

Lingkungan uji coba yang akan digunakan adalah,

1. Perangkat Keras
   Prosesor Intel® Core™ i3-2350M CPU @ 2.30GHz 2.30GHz RAM 4 GB
   Sistem Operasi 64-bit
2. Perangkat Lunak
   Sistem Operasi Microsoft Windows 8 64-bit Pro
   Perangkat Pengembang PyCharm

5.2 Data Uji Coba

Data yang digunakan untuk uji coba implementasi deteksi api berbasis sensor visual menggunakan metode support vector machines adalah potongan video yang didapatkan dari berbagai sumber. Kualitas video yang digunakan adalah video dengan *size* 240x320 piksel dan memiliki *channel* R,G,B. Data video yang digunakan diambil dari beberapa kejadian. Data video yang digunakan meiputi dua buah jenis video. Video dengan objek api dan video dengan objek bukan api. Jumlah video yang di uji berjumlah lima puluh sembilan video dengan jumlah video api sejumlah dua puluh enam dan video bukan api berjumlah tiga puluh tiga.


Movie 1


Movie 2


Movie 3


Movie 4


Movie 5 Movie 6 Movie 7 Movie 8

Movie 9 Movie 10 Movie 11 Movie 12


Movie 13 Movie 14 Movie 15 Movie 16

Gambar 5.2.1 Contoh Video Kejadian

5.3 Alur Uji Coba

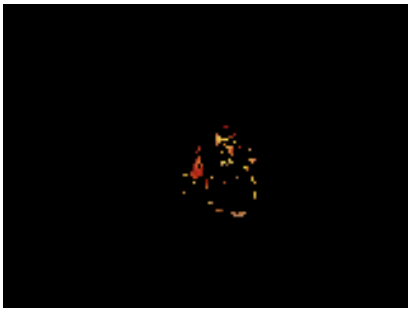
Pada sub bab ini akan dijelaskan mengenai alur kerja dari sistem deteksi api. Dimulai dari *preprocessing* hingga verifikasi.

5.3.1 Preprocessing

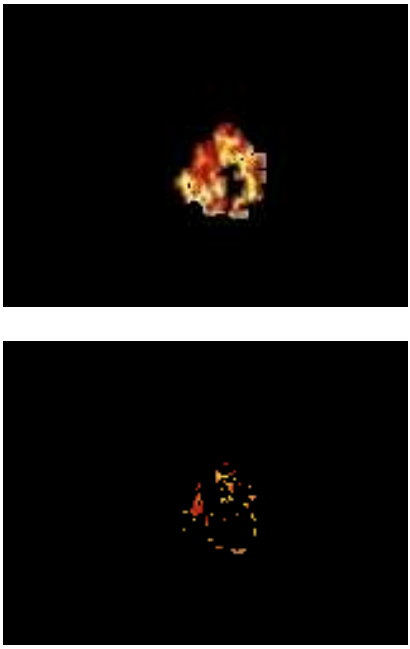
Tahap *preprocessing* akan dijelaskan bagaimana alur setiap *frame* masuk hingga menghasilkan kandidat api yang selanjutnya akan di proses pada tahap verifikasi. Ilustrasi tahap *proprocessing* dapat dilihat pada Gambar 5.3.1.

Frame Masukkan

Deteksi Gerak



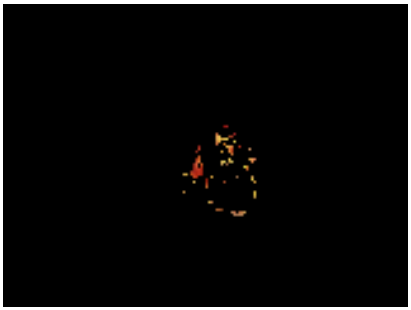
Deteksi Warna Piksel



Gambar 5.3.1 Tahap *Preprocessing*

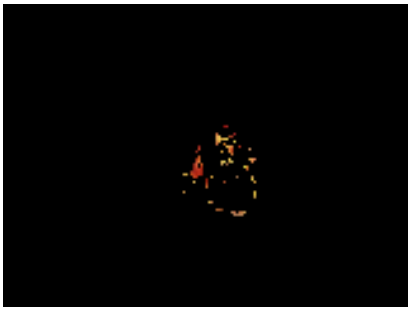
5.3.2 Verifikasi

Tahap verifikasi akan dijelaskan bagaimana alur verifikasi dilakukan. Masukkan dari tahap ini adalah hasil akhir dari tahap *preprocessing*. Hasil akhir dari proses verifikasi adalah *region* yang masuk kedalam objek api. Ilustrasi proses verifikasi dapat dilihat pada Gambar 5.3.2



Frame Masukkan dari proses *preprocessing*

Ekstraksi Fitur, detail gambar vertikal, horizontal, digonal



Support Vector Machines

Menandai *Region* Api

Gambar 5.3.2 Tahap Verifikasi

5.4 Skenario Uji Coba

Pada sub bab ini akan dijelaskan mengenai skenario uji coba yang telah dilakukan. Telah dilakukan beberapa skenario uji coba, diantaranya yaitu:

1. Perbandingan hasil akurasi berdasarkan variasi nilai *threshold* pada deteksi warna api. *Threshold* yang akan diuji yaitu 10^{-7}, 10^{-8}, 5 x 10^{-9}, 10^{-9}.
2. Perbandingan hasil akurasi berdasarkan variasi nilai C (*penalty error term*) pada klasifikasi dengan kernel tetap yaitu rbf. Nilai C yang akan diuji yaitu 1, 3.5, 5,dan 7.
3. Perbandingan hasi akurasi berdasarkan variasi kernel yang digunakan pada klasifikasi. Kernel yang akan diuji yaitu *polynomial*, dan RBF.

5.4.1 Skenario Uji Coba 1

Sekenario uji coba 1 adalah perhitungan *true positif, false positif,* dan *missing rate*. Dimana *true positif* adalah kondisi suatu *frame* mengandung gambar api dan terdeteksi api atau *frame* tidak mengandung api dan tidak terdeteksi api. *False positif* adalah kondisi dimana *frame* tidak mengandung gambar api, namun terdeteksi api dan *missing rate* adalah keadaan dimana suatu *frame* yang mempunyai gambar api namun tidak terdeteksi api. Pada sekenario uji coba 1 dilakukan uji coba pada tahap probabilitas warna api dengan mengubah nilai *threshold* probabilitas piksel api. Nilai *threshold* yang diuji yaitu 10^{-7}, 10^{-8}, 5 x 10^{-9}, 10^{-9}. Untuk parameter nilai C pada uji coba 1 diberikan nilai 5 menggunakan kernel RBF.

Threshold	*True Positif*	*False Positif*	*Missing Rate*
10^{-7}	73.96	0.27	25.77
10^{-8}	90.33	0.38	9.29
$5x10^{-9}$	92.87	0.55	6.58
10^{-9}	93.97	3.25	2.78

Tabel 5.4.1 Hasil Uji Coba 1

Dari hasil uji yang dilakukan, makin kecil nilai *threshold* yang digunakan, hasil dari *true positif* akan semakin besar. Begitu juga untuk *false positif*, dimana makin kecil nilai *threshold* makin besar nilai *false positif*. Hal ini dikarenakan piksel yang dianggap piksel api sudah melewati batas warna kuning hingga merah. Dari uji coba tersebut didapatkan nilai *threshold* $5x10^{-9}$ sebagai nilai terbaik, karena *False positif* yang dihasilkan tidak terlalu besar dan *True Positif* bernilai besar. Hasil uji coba 1 lebih lengkap terdapat pada lampiran.

5.4.2 Skenario Uji Coba 2

Sekenario uji coba 2 dilakukan dengan menghitung nilai *true positif, false positif,* dan *missing rate*. Pada sekenario uji 2 dilakukan uji coba variasi nilai C pada klasifikasi, dimana

nilai C adalah nilai *penalty error term*. Nilai C yang diuji yaitu 1, 3.5, 5, dan 7. Untuk parameter *threshold* warna piksel diberikan nilai 5 x 10^{-9} menggunakan kernel RBF sebagai klasifikasi.

C	*True Positif*	*False Positif*	*Missing Rate*
1	91.85	0.51	7.65
3.5	92.71	0.54	6.74
5	92.87	0.55	6.58
7	92.96	0.54	6.50

Tabel 5.4.2 Hasil Uji Coba 2

Hasil uji coba tahap 2 hasil terbaik didapatkan ketika nilai C = 7, dimana memiliki nilai *true positif* yang lebih tinggi dari yang lainnya.

5.4.3 Skenario Uji Coba 3

Sekenario uji coba 3 dilakukan dengan menghitung nilai *true positif, false positif,* dan *missing rate*. Pada sekenario uji 3 dilakukan uji coba variasi kernel klasifikasi. Variasi kernel yang digunakan yaitu *polynomial*, dan RBF. Untuk parameter *threshold* warna piksel diberikan nilai 5 x 10^{-9} dan nilai C diberikan nilai 5.

Kernel	*True Positif*	*False Positif*	*Missing Rate*
Polynomial	61.56	0.00	38.44
RBF	92.87	0.55	6.58

Tabel 5.4.3 Hasil Uji Coba 2

Hasil uji coba tahap 3 didapatkan nilai *true positif* terbaik didapatkan dengan menggunakan kernel RBF.

5.5 Analisis Hasil Uji Coba

Dari hasil skenario uji coba yang telah dilakukan, beberapa parameter memberikan pengaruh terhadap hasil

deteksi. Parameter yang digunakan antara lain nilai *threshold* pada deteksi warna api dan nilai C pada klasifikasi. Uji coba dilakukan dengan membandingkan nilai *true positif* , *false pisitif*, dan *missing rate*.

Dari uji coba yang dilakukan, parameter-parameter tersebut menghasilkan presentase terbaik ketika *threshold* yang digunakan 5×10^{-9} dan C yang digunakan sebesar 7 dan menggunakan kernel RBF.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan mengenai kesimpulan dari proses dan uji coba dari program dan saran untuk pengembangan dari program itu sendiri.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan, dapat diambil kesimpulan sebagai berikut :

1. Metode reduksi *size frame* mengurangi proses komputasi sehingga deteksi api bisa lebih cepat dilakukan.
2. Deteksi warna menyaring piksel-piksel yang tidak masuk kedalam *range* warna api dapat menyaring piksel-piksel yang tidak termasuk piksel api dengan baik.
3. Metode perhitungan luasan *region* api dapat menghilangkan *noise* atau objek-objek kecil yang mempengaruhi hasil dari klasifikasi.
4. Penggunaan kernel pada
5. Hasil terbaik pada uji coba adalah menggunakan nilai *threshold* = 5 x 10^{-9} dan nilai C = 7. Menghasilkan nilai *true positif* sebesar 92.96, *false positif* sebesar 0.54 dan *missing rate* sebesar 6.50

6.2 Saran

Saran yang diberikan untuk pengembangan perangkat lunak ini adalah :

1. Proses deteksi api dan klasifikasi perlu dikembangkan, tidak melakukan proses verifikasi setiap piksel.

DAFTAR ACUAN

[1] R. B. P. KaewTraKulPong, "An Improved Adaptive Background Mixture Model for Real- time Tracking with Shadow Detection," Kluwer Academic Publishers, 2001.

[2] E. H. A. PETER J. BURT, "The Laplacian Pyramid as a Compact Image Code," *IEEE* , Vols. COM-31, pp. 522-540, 1983.

[3] I. S. T. Maria Isabel Ribeiro, "Gaussian Probability Density Functions: Properties and Error Characterization," 2004.

[4] R. E. W. Refael C. Gonzalez, Digital Image Processing third edition, p. 785.

[5] R. S. F. D. R. S. Lee A. Barford, "An Introduction to Wavelets," 1992.

[6] B. K. HYERAN BYUN, "ROBUST FACE DETECTION AND TRACKING FOR REAL-LIFE APPLICATIONS," *International Journal of Pattern Recognition*, vol. 17, pp. 1035-1055, 2003.

[7] S. T. Punam Patel, "Flame Detection using Image Processing Techniques," *International Journal of Computer Applications*, vol. 58, pp. 13-16, 2012.

[Halaman ini sengaja dikosongkan]

LAMPIRAN A

Tabel 6.2.1 Hasil Uji Coba Manggunakan Parameter *Threshold* = 10^{-7}, C = 5 dan kernel RBF

No	Nama File	True Positif	False Positif	Missing Rate
1	api-bakar_sampah.avi	4.69	0.00	95.31
2	api-boneka_dora.avi	6.37	0.00	93.63
3	api-kayu.avi	57.35	0.00	42.65
4	api-kayu2.avi	58.82	0.00	41.18
5	api-kebakaran_hutan.avi	63.64	0.00	36.36
6	api-kebakaran_hutan2.avi	64.65	0.00	35.35
7	api-kebakaran_hutan3.avi	74.75	0.00	25.25
8	api-kebakaran_hutan4.avi	0.00	0.00	100.00
9	api-kebakaran_ladang.avi	93.49	0.00	6.51
10	api-kebakaran_mobil.avi	94.12	0.00	5.88
11	api-kebakaran_mobil2.avi	89.22	0.00	10.78
12	api-kebakaran_tol.avi	59.31	0.00	40.69
13	api-kebakaran-truck.avi	50.00	0.00	50.00
14	api-kebakaran-truck2.avi	62.25	0.00	37.75
15	api-kebakaran-truck3.avi	16.18	0.00	83.82
16	api-kertas.avi	0.00	0.00	100.00
17	api-kertas2.avi	4.32	0.00	95.68
18	api-miniatur_mainan.avi	9.47	0.00	90.53
19	api-mobil_mainan.avi	0.00	0.00	100.00
20	api-orang_terjun.avi	46.08	0.00	53.92

21	api-pesawat_mainan.avi	100.00	0.00	0.00
22	api-pesawat_mainan2.avi	26.63	0.00	73.37
23	api-ruang_tamu.avi	0.00	0.00	100.00
24	api-rumah_mainan.avi	98.53	0.00	1.47
25	api-senter.avi	0.00	0.00	100.00
26	api-senter2.avi	0.00	0.00	100.00
27	non_api-anak_kecil.avi	100.00	0.00	0.00
28	non_api-anak_kecil2.avi	100.00	0.00	0.00
29	non_api-anak_kecil3.avi	100.00	0.00	0.00
30	non_api-bertemu.avi	100.00	0.00	0.00
31	non_api-jaket_merah.avi	100.00	0.00	0.00
32	non_api-jalan_malam.avi	100.00	0.00	0.00
33	non_api-jalan_malam2.avi	100.00	0.00	0.00
34	non_api-jalan_raya.avi	100.00	0.00	0.00
35	non_api-jalan_raya2.avi	100.00	0.00	0.00
36	non_api-jalan_raya3.avi	100.00	0.00	0.00
37	non_api-kantor.avi	100.00	0.00	0.00
38	non_api-kecelakaan.avi	100.00	0.00	0.00
39	non_api-kecelakaan2.avi	100.00	0.00	0.00
40	non_api-kecelakaan3.avi	100.00	0.00	0.00
41	non_api-kerusuhan.avi	100.00	0.00	0.00
42	non_api-kerusuhan2.avi	100.00	0.00	0.00
43	non_api-kerusuhan3.avi	100.00	0.00	0.00
44	non_api-las_vegas.avi	100.00	0.00	0.00
45	non_api-manuju_mobil.avi	100.00	0.00	0.00
46	non_api-manuju_mobil2.avi	100.00	0.00	0.00

47	non_api-manuju_mobil3.avi	100.00	0.00	0.00
48	non_api-parkiran.avi	100.00	0.00	0.00
49	non_api-parkiran2.avi	100.00	0.00	0.00
50	non_api-parkiran3.avi	100.00	0.00	0.00
51	non_api-pencuri.avi	100.00	0.00	0.00
52	non_api-pencuri2.avi	100.00	0.00	0.00
53	non_api-penembakan.avi	100.00	0.00	0.00
54	non_api-serbet.avi	83.98	16.02	0.00
55	non_api-tas.avi	100.00	0.00	0.00
56	non_api-tas2.avi	100.00	0.00	0.00
57	non_api-tas3.avi	100.00	0.00	0.00
58	non_api-televisi.avi	100.00	0.00	0.00
59	non_api-televisi2.avi	100.00	0.00	0.00

Tabel 6.2.2 Hasil Uji Coba Manggunakan Parameter *Threshold* = 10^{-8}, C = 5 dan kernel RBF

No	Nama File	True Positif	False Positif	Missing Rate
1	api-bakar_sampah.avi	73.44	0.00	26.56
2	api-boneka_dora.avi	57.84	0.00	42.16
3	api-kayu.avi	84.80	0.00	15.20
4	api-kayu2.avi	88.73	0.00	11.27
5	api-kebakaran_hutan.avi	70.71	0.00	29.29
6	api-kebakaran_hutan2.avi	85.86	0.00	14.14
7	api-kebakaran_hutan3.avi	100.00	0.00	0.00
8	api-kebakaran_hutan4.avi	84.13	0.00	15.87
9	api-kebakaran_ladang.avi	100.00	0.00	0.00

10	api-kebakaran_mobil.avi	99.02	0.00	0.98
11	api-kebakaran_mobil2.avi	99.02	0.00	0.98
12	api-kebakaran_tol.avi	90.20	0.00	9.80
13	api-kebakaran-truck.avi	100.00	0.00	0.00
14	api-kebakaran-truck2.avi	100.00	0.00	0.00
15	api-kebakaran-truck3.avi	99.51	0.00	0.49
16	api-kertas.avi	60.49	0.00	39.51
17	api-kertas2.avi	72.84	0.00	27.16
18	api-miniatur_mainan.avi	76.92	0.00	23.08
19	api-mobil_mainan.avi	87.75	0.00	12.25
20	api-orang_terjun.avi	74.51	0.00	25.49
21	api-pesawat_mainan.avi	100.00	0.00	0.00
22	api-pesawat_mainan2.avi	97.63	0.00	2.37
23	api-ruang_tamu.avi	44.97	0.00	55.03
24	api-rumah_mainan.avi	100.00	0.00	0.00
25	api-senter.avi	1.96	0.00	98.04
26	api-senter2.avi	1.47	0.00	98.53
27	non_api-anak_kecil.avi	100.00	0.00	0.00
28	non_api-anak_kecil2.avi	100.00	0.00	0.00
29	non_api-anak_kecil3.avi	100.00	0.00	0.00
30	non_api-bertemu.avi	100.00	0.00	0.00
31	non_api-jaket_merah.avi	100.00	0.00	0.00
32	non_api-jalan_malam.avi	100.00	0.00	0.00
33	non_api-jalan_malam2.avi	100.00	0.00	0.00
34	non_api-jalan_raya.avi	100.00	0.00	0.00
35	non_api-jalan_raya2.avi	100.00	0.00	0.00

36	non_api-jalan_raya3.avi	100.00	0.00	0.00
37	non_api-kantor.avi	100.00	0.00	0.00
38	non_api-kecelakaan.avi	100.00	0.00	0.00
39	non_api-kecelakaan2.avi	100.00	0.00	0.00
40	non_api-kecelakaan3.avi	100.00	0.00	0.00
41	non_api-kerusuhan.avi	100.00	0.00	0.00
42	non_api-kerusuhan2.avi	100.00	0.00	0.00
43	non_api-kerusuhan3.avi	100.00	0.00	0.00
44	non_api-las_vegas.avi	100.00	0.00	0.00
45	non_api-manuju_mobil.avi	100.00	0.00	0.00
46	non_api-manuju_mobil2.avi	100.00	0.00	0.00
47	non_api-manuju_mobil3.avi	100.00	0.00	0.00
48	non_api-parkiran.avi	100.00	0.00	0.00
49	non_api-parkiran2.avi	100.00	0.00	0.00
50	non_api-parkiran3.avi	100.00	0.00	0.00
51	non_api-pencuri.avi	100.00	0.00	0.00
52	non_api-pencuri2.avi	99.41	0.59	0.00
53	non_api-penembakan.avi	100.00	0.00	0.00
54	non_api-serbet.avi	78.45	21.55	0.00
55	non_api-tas.avi	100.00	0.00	0.00
56	non_api-tas2.avi	100.00	0.00	0.00
57	non_api-tas3.avi	100.00	0.00	0.00
58	non_api-televisi.avi	100.00	0.00	0.00
59	non_api-televisi2.avi	100.00	0.00	0.00

Tabel 6.2.3 Hasil Uji Coba Manggunakan Parameter *Threshold* = 5x10^{-9}, C = 5 dan kernel RBF

No	Nama File	True Positif	False Positif	Missing Rate
1	api-bakar_sampah.avi	78.13	0.00	21.88
2	api-boneka_dora.avi	79.90	0.00	20.10
3	api-kayu.avi	85.29	0.00	14.71
4	api-kayu2.avi	88.73	0.00	11.27
5	api-kebakaran_hutan.avi	70.71	0.00	29.29
6	api-kebakaran_hutan2.avi	85.86	0.00	14.14
7	api-kebakaran_hutan3.avi	100.00	0.00	0.00
8	api-kebakaran_hutan4.avi	91.35	0.00	8.65
9	api-kebakaran_ladang.avi	100.00	0.00	0.00
10	api-kebakaran_mobil.avi	99.51	0.00	0.49
11	api-kebakaran_mobil2.avi	99.02	0.00	0.98
12	api-kebakaran_tol.avi	90.69	0.00	9.31
13	api-kebakaran-truck.avi	100.00	0.00	0.00
14	api-kebakaran-truck2.avi	100.00	0.00	0.00
15	api-kebakaran-truck3.avi	100.00	0.00	0.00
16	api-kertas.avi	67.28	0.00	32.72
17	api-kertas2.avi	98.15	0.00	1.85
18	api-miniatur_mainan.avi	89.35	0.00	10.65
19	api-mobil_mainan.avi	98.04	0.00	1.96
20	api-orang_terjun.avi	79.41	0.00	20.59
21	api-pesawat_mainan.avi	100.00	0.00	0.00

22	api-pesawat_mainan2.avi	100.00	0.00	0.00
23	api-ruang_tamu.avi	74.56	0.00	25.44
24	api-rumah_mainan.avi	100.00	0.00	0.00
25	api-senter.avi	28.92	0.00	71.08
26	api-senter2.avi	6.86	0.00	93.14
27	non_api-anak_kecil.avi	100.00	0.00	0.00
28	non_api-anak_kecil2.avi	100.00	0.00	0.00
29	non_api-anak_kecil3.avi	100.00	0.00	0.00
30	non_api-bertemu.avi	100.00	0.00	0.00
31	non_api-jaket_merah.avi	100.00	0.00	0.00
32	non_api-jalan_malam.avi	100.00	0.00	0.00
33	non_api-jalan_malam2.avi	100.00	0.00	0.00
34	non_api-jalan_raya.avi	100.00	0.00	0.00
35	non_api-jalan_raya2.avi	100.00	0.00	0.00
36	non_api-jalan_raya3.avi	100.00	0.00	0.00
37	non_api-kantor.avi	100.00	0.00	0.00
38	non_api-kecelakaan.avi	100.00	0.00	0.00
39	non_api-kecelakaan2.avi	100.00	0.00	0.00
40	non_api-kecelakaan3.avi	100.00	0.00	0.00
41	non_api-kerusuhan.avi	100.00	0.00	0.00
42	non_api-kerusuhan2.avi	100.00	0.00	0.00
43	non_api-kerusuhan3.avi	100.00	0.00	0.00
44	non_api-las_vegas.avi	100.00	0.00	0.00
45	non_api-manuju_mobil.avi	100.00	0.00	0.00
46	non_api-manuju_mobil2.avi	100.00	0.00	0.00

47	non_api-manuju_mobil3.avi	100.00	0.00	0.00
48	non_api-parkiran.avi	100.00	0.00	0.00
49	non_api-parkiran2.avi	100.00	0.00	0.00
50	non_api-parkiran3.avi	100.00	0.00	0.00
51	non_api-pencuri.avi	92.31	7.69	0.00
52	non_api-pencuri2.avi	99.41	0.59	0.00
53	non_api-penembakan.avi	100.00	0.00	0.00
54	non_api-serbet.avi	75.69	24.31	0.00
55	non_api-tas.avi	100.00	0.00	0.00
56	non_api-tas2.avi	100.00	0.00	0.00
57	non_api-tas3.avi	100.00	0.00	0.00
58	non_api-televisi.avi	100.00	0.00	0.00
59	non_api-televisi2.avi	100.00	0.00	0.00

Tabel 6.2.4 Hasil Uji Coba Manggunakan Parameter *Threshold* = 10^{-9}, C = 5 dan kernel RBF

No	Nama File	True Positif	False Positif	Missing Rate
1	api-bakar_sampah.avi	78.13	0.00	21.88
2	api-boneka_dora.avi	97.55	0.00	2.45
3	api-kayu.avi	85.29	0.00	14.71
4	api-kayu2.avi	89.22	0.00	10.78
5	api-kebakaran_hutan.avi	73.74	0.00	26.26
6	api-kebakaran_hutan2.avi	85.86	0.00	14.14
7	api-kebakaran_hutan3.avi	100.00	0.00	0.00
8	api-kebakaran_hutan4.avi	99.52	0.00	0.48
9	api-kebakaran_ladang.avi	100.00	0.00	0.00

10	api-kebakaran_mobil.avi	99.51	0.00	0.49
11	api-kebakaran_mobil2.avi	100.00	0.00	0.00
12	api-kebakaran_tol.avi	99.02	0.00	0.98
13	api-kebakaran-truck.avi	100.00	0.00	0.00
14	api-kebakaran-truck2.avi	100.00	0.00	0.00
15	api-kebakaran-truck3.avi	100.00	0.00	0.00
16	api-kertas.avi	95.06	0.00	4.94
17	api-kertas2.avi	100.00	0.00	0.00
18	api-miniatur_mainan.avi	99.41	0.00	0.59
19	api-mobil_mainan.avi	99.02	0.00	0.98
20	api-orang_terjun.avi	94.61	0.00	5.39
21	api-pesawat_mainan.avi	100.00	0.00	0.00
22	api-pesawat_mainan2.avi	100.00	0.00	0.00
23	api-ruang_tamu.avi	98.22	0.00	1.78
24	api-rumah_mainan.avi	100.00	0.00	0.00
25	api-senter.avi	91.18	0.00	8.82
26	api-senter2.avi	50.49	0.00	49.51
27	non_api-anak_kecil.avi	100.00	0.00	0.00
28	non_api-anak_kecil2.avi	100.00	0.00	0.00
29	non_api-anak_kecil3.avi	100.00	0.00	0.00
30	non_api-bertemu.avi	100.00	0.00	0.00
31	non_api-jaket_merah.avi	100.00	0.00	0.00
32	non_api-jalan_malam.avi	100.00	0.00	0.00
33	non_api-jalan_malam2.avi	100.00	0.00	0.00
34	non_api-jalan_raya.avi	100.00	0.00	0.00
35	non_api-jalan_raya2.avi	100.00	0.00	0.00

36	non_api-jalan_raya3.avi	100.00	0.00	0.00
37	non_api-kantor.avi	100.00	0.00	0.00
38	non_api-kecelakaan.avi	100.00	0.00	0.00
39	non_api-kecelakaan2.avi	100.00	0.00	0.00
40	non_api-kecelakaan3.avi	100.00	0.00	0.00
41	non_api-kerusuhan.avi	54.44	45.56	0.00
42	non_api-kerusuhan2.avi	98.22	1.78	0.00
43	non_api-kerusuhan3.avi	100.00	0.00	0.00
44	non_api-las_vegas.avi	98.77	1.23	0.00
45	non_api-manuju_mobil.avi	100.00	0.00	0.00
46	non_api-manuju_mobil2.avi	100.00	0.00	0.00
47	non_api-manuju_mobil3.avi	100.00	0.00	0.00
48	non_api-parkiran.avi	100.00	0.00	0.00
49	non_api-parkiran2.avi	100.00	0.00	0.00
50	non_api-parkiran3.avi	100.00	0.00	0.00
51	non_api-pencuri.avi	12.43	87.57	0.00
52	non_api-pencuri2.avi	98.82	1.18	0.00
53	non_api-penembakan.avi	97.06	2.94	0.00
54	non_api-serbet.avi	48.62	51.38	0.00
55	non_api-tas.avi	100.00	0.00	0.00
56	non_api-tas2.avi	100.00	0.00	0.00
57	non_api-tas3.avi	100.00	0.00	0.00
58	non_api-televisi.avi	100.00	0.00	0.00
59	non_api-televisi2.avi	100.00	0.00	0.00

Tabel 6.2.5 Hasil Uji Coba Manggunakan Parameter *Threshold* = 5x10^{-9}, C = 1 dan kernel RBF

No	Nama File	True Positif	False Positif	Missing Rate
1	api-bakar_sampah.avi	59.38	0.00	40.63
2	api-boneka_dora.avi	79.41	0.00	20.59
3	api-kayu.avi	81.37	0.00	18.63
4	api-kayu2.avi	84.80	0.00	15.20
5	api-kebakaran_hutan.avi	63.64	0.00	36.36
6	api-kebakaran_hutan2.avi	81.82	0.00	18.18
7	api-kebakaran_hutan3.avi	100.00	0.00	0.00
8	api-kebakaran_hutan4.avi	83.65	0.00	16.35
9	api-kebakaran_ladang.avi	100.00	0.00	0.00
10	api-kebakaran_mobil.avi	99.51	0.00	0.49
11	api-kebakaran_mobil2.avi	97.06	0.00	2.94
12	api-kebakaran_tol.avi	89.71	0.00	10.29
13	api-kebakaran-truck.avi	100.00	0.00	0.00
14	api-kebakaran-truck2.avi	100.00	0.00	0.00
15	api-kebakaran-truck3.avi	100.00	0.00	0.00
16	api-kertas.avi	67.28	0.00	32.72
17	api-kertas2.avi	98.15	0.00	1.85
18	api-miniatur_mainan.avi	89.35	0.00	10.65
19	api-mobil_mainan.avi	98.04	0.00	1.96
20	api-orang_terjun.avi	79.41	0.00	20.59
21	api-pesawat_mainan.avi	100.00	0.00	0.00

22	api-pesawat_mainan2.avi	100.00	0.00	0.00
23	api-ruang_tamu.avi	72.19	0.00	27.81
24	api-rumah_mainan.avi	100.00	0.00	0.00
25	api-senter.avi	17.16	0.00	82.84
26	api-senter2.avi	6.86	0.00	93.14
27	non_api-anak_kecil.avi	100.00	0.00	0.00
28	non_api-anak_kecil2.avi	100.00	0.00	0.00
29	non_api-anak_kecil3.avi	100.00	0.00	0.00
30	non_api-bertemu.avi	100.00	0.00	0.00
31	non_api-jaket_merah.avi	100.00	0.00	0.00
32	non_api-jalan_malam.avi	100.00	0.00	0.00
33	non_api-jalan_malam2.avi	100.00	0.00	0.00
34	non_api-jalan_raya.avi	100.00	0.00	0.00
35	non_api-jalan_raya2.avi	100.00	0.00	0.00
36	non_api-jalan_raya3.avi	100.00	0.00	0.00
37	non_api-kantor.avi	100.00	0.00	0.00
38	non_api-kecelakaan.avi	100.00	0.00	0.00
39	non_api-kecelakaan2.avi	100.00	0.00	0.00
40	non_api-kecelakaan3.avi	100.00	0.00	0.00
41	non_api-kerusuhan.avi	100.00	0.00	0.00
42	non_api-kerusuhan2.avi	100.00	0.00	0.00
43	non_api-kerusuhan3.avi	100.00	0.00	0.00
44	non_api-las_vegas.avi	100.00	0.00	0.00
45	non_api-manuju_mobil.avi	100.00	0.00	0.00
46	non_api-manuju_mobil2.avi	100.00	0.00	0.00

47	non_api-manuju_mobil3.avi	100.00	0.00	0.00
48	non_api-parkiran.avi	100.00	0.00	0.00
49	non_api-parkiran2.avi	100.00	0.00	0.00
50	non_api-parkiran3.avi	100.00	0.00	0.00
51	non_api-pencuri.avi	92.31	7.69	0.00
52	non_api-pencuri2.avi	99.41	0.59	0.00
53	non_api-penembakan.avi	100.00	0.00	0.00
54	non_api-serbet.avi	78.45	21.55	0.00
55	non_api-tas.avi	100.00	0.00	0.00
56	non_api-tas2.avi	100.00	0.00	0.00
57	non_api-tas3.avi	100.00	0.00	0.00
58	non_api-televisi.avi	100.00	0.00	0.00
59	non_api-televisi2.avi	100.00	0.00	0.00

Tabel 6.2.6 Hasil Uji Coba Manggunakan Parameter *Threshold* = 5x10^{-9}, C = 3.5 dan kernel RBF

No	Nama File	True Positif	False Positif	Missing Rate
1	api-bakar_sampah.avi	75.00	0.00	25.00
2	api-boneka_dora.avi	79.90	0.00	20.10
3	api-kayu.avi	85.29	0.00	14.71
4	api-kayu2.avi	88.73	0.00	11.27
5	api-kebakaran_hutan.avi	68.69	0.00	31.31
6	api-kebakaran_hutan2.avi	85.86	0.00	14.14
7	api-kebakaran_hutan3.avi	100.00	0.00	0.00
8	api-kebakaran_hutan4.avi	90.87	0.00	9.13
9	api-kebakaran_ladang.avi	100.00	0.00	0.00

10	api-kebakaran_mobil.avi	99.51	0.00	0.49
11	api-kebakaran_mobil2.avi	98.53	0.00	1.47
12	api-kebakaran_tol.avi	90.69	0.00	9.31
13	api-kebakaran-truck.avi	100.00	0.00	0.00
14	api-kebakaran-truck2.avi	100.00	0.00	0.00
15	api-kebakaran-truck3.avi	100.00	0.00	0.00
16	api-kertas.avi	67.28	0.00	32.72
17	api-kertas2.avi	98.15	0.00	1.85
18	api-miniatur_mainan.avi	89.35	0.00	10.65
19	api-mobil_mainan.avi	98.04	0.00	1.96
20	api-orang_terjun.avi	79.41	0.00	20.59
21	api-pesawat_mainan.avi	100.00	0.00	0.00
22	api-pesawat_mainan2.avi	100.00	0.00	0.00
23	api-ruang_tamu.avi	74.56	0.00	25.44
24	api-rumah_mainan.avi	100.00	0.00	0.00
25	api-senter.avi	25.49	0.00	74.51
26	api-senter2.avi	6.86	0.00	93.14
27	non_api-anak_kecil.avi	100.00	0.00	0.00
28	non_api-anak_kecil2.avi	100.00	0.00	0.00
29	non_api-anak_kecil3.avi	100.00	0.00	0.00
30	non_api-bertemu.avi	100.00	0.00	0.00
31	non_api-jaket_merah.avi	100.00	0.00	0.00
32	non_api-jalan_malam.avi	100.00	0.00	0.00
33	non_api-jalan_malam2.avi	100.00	0.00	0.00
34	non_api-jalan_raya.avi	100.00	0.00	0.00
35	non_api-jalan_raya2.avi	100.00	0.00	0.00

36	non_api-jalan_raya3.avi	100.00	0.00	0.00
37	non_api-kantor.avi	100.00	0.00	0.00
38	non_api-kecelakaan.avi	100.00	0.00	0.00
39	non_api-kecelakaan2.avi	100.00	0.00	0.00
40	non_api-kecelakaan3.avi	100.00	0.00	0.00
41	non_api-kerusuhan.avi	100.00	0.00	0.00
42	non_api-kerusuhan2.avi	100.00	0.00	0.00
43	non_api-kerusuhan3.avi	100.00	0.00	0.00
44	non_api-las_vegas.avi	100.00	0.00	0.00
45	non_api-manuju_mobil.avi	100.00	0.00	0.00
46	non_api-manuju_mobil2.avi	100.00	0.00	0.00
47	non_api-manuju_mobil3.avi	100.00	0.00	0.00
48	non_api-parkiran.avi	100.00	0.00	0.00
49	non_api-parkiran2.avi	100.00	0.00	0.00
50	non_api-parkiran3.avi	100.00	0.00	0.00
51	non_api-pencuri.avi	92.31	7.69	0.00
52	non_api-pencuri2.avi	98.82	1.18	0.00
53	non_api-penembakan.avi	100.00	0.00	0.00
54	non_api-serbet.avi	76.80	23.20	0.00
55	non_api-tas.avi	100.00	0.00	0.00
56	non_api-tas2.avi	100.00	0.00	0.00
57	non_api-tas3.avi	100.00	0.00	0.00
58	non_api-televisi.avi	100.00	0.00	0.00
59	non_api-televisi2.avi	100.00	0.00	0.00

Tabel 6.2.7 Hasil Uji Coba Manggunakan Parameter *Threshold* = $5x10^{-9}$, C = 7 dan kernel RBF

No	Nama File	True Positif	False Positif	Missing Rate
1	api-bakar_sampah.avi	82.81	0.00	17.19
2	api-boneka_dora.avi	79.41	0.00	20.59
3	api-kayu.avi	85.29	0.00	14.71
4	api-kayu2.avi	89.71	0.00	10.29
5	api-kebakaran_hutan.avi	70.71	0.00	29.29
6	api-kebakaran_hutan2.avi	86.87	0.00	13.13
7	api-kebakaran_hutan3.avi	100.00	0.00	0.00
8	api-kebakaran_hutan4.avi	91.35	0.00	8.65
9	api-kebakaran_ladang.avi	100.00	0.00	0.00
10	api-kebakaran_mobil.avi	99.51	0.00	0.49
11	api-kebakaran_mobil2.avi	99.02	0.00	0.98
12	api-kebakaran_tol.avi	90.69	0.00	9.31
13	api-kebakaran-truck.avi	100.00	0.00	0.00
14	api-kebakaran-truck2.avi	100.00	0.00	0.00
15	api-kebakaran-truck3.avi	100.00	0.00	0.00
16	api-kertas.avi	67.28	0.00	32.72
17	api-kertas2.avi	98.15	0.00	1.85
18	api-miniatur_mainan.avi	89.35	0.00	10.65
19	api-mobil_mainan.avi	98.04	0.00	1.96
20	api-orang_terjun.avi	79.41	0.00	20.59
21	api-pesawat_mainan.avi	100.00	0.00	0.00

22	api-pesawat_mainan2.avi	100.00	0.00	0.00
23	api-ruang_tamu.avi	73.37	0.00	26.63
24	api-rumah_mainan.avi	100.00	0.00	0.00
25	api-senter.avi	28.92	0.00	71.08
26	api-senter2.avi	6.86	0.00	93.14
27	non_api-anak_kecil.avi	100.00	0.00	0.00
28	non_api-anak_kecil2.avi	100.00	0.00	0.00
29	non_api-anak_kecil3.avi	100.00	0.00	0.00
30	non_api-bertemu.avi	100.00	0.00	0.00
31	non_api-jaket_merah.avi	100.00	0.00	0.00
32	non_api-jalan_malam.avi	100.00	0.00	0.00
33	non_api-jalan_malam2.avi	100.00	0.00	0.00
34	non_api-jalan_raya.avi	100.00	0.00	0.00
35	non_api-jalan_raya2.avi	100.00	0.00	0.00
36	non_api-jalan_raya3.avi	100.00	0.00	0.00
37	non_api-kantor.avi	100.00	0.00	0.00
38	non_api-kecelakaan.avi	100.00	0.00	0.00
39	non_api-kecelakaan2.avi	100.00	0.00	0.00
40	non_api-kecelakaan3.avi	100.00	0.00	0.00
41	non_api-kerusuhan.avi	100.00	0.00	0.00
42	non_api-kerusuhan2.avi	100.00	0.00	0.00
43	non_api-kerusuhan3.avi	100.00	0.00	0.00
44	non_api-las_vegas.avi	100.00	0.00	0.00
45	non_api-manuju_mobil.avi	100.00	0.00	0.00
46	non_api-manuju_mobil2.avi	100.00	0.00	0.00

47	non_api-manuju_mobil3.avi	100.00	0.00	0.00
48	non_api-parkiran.avi	100.00	0.00	0.00
49	non_api-parkiran2.avi	100.00	0.00	0.00
50	non_api-parkiran3.avi	100.00	0.00	0.00
51	non_api-pencuri.avi	92.31	7.69	0.00
52	non_api-pencuri2.avi	99.41	0.59	0.00
53	non_api-penembakan.avi	100.00	0.00	0.00
54	non_api-serbet.avi	76.24	23.76	0.00
55	non_api-tas.avi	100.00	0.00	0.00
56	non_api-tas2.avi	100.00	0.00	0.00
57	non_api-tas3.avi	100.00	0.00	0.00
58	non_api-televisi.avi	100.00	0.00	0.00
59	non_api-televisi2.avi	100.00	0.00	0.00

Tabel 6.2.8 Hasil Uji Coba Manggunakan Parameter *Threshold* = $5x10^{-9}$, *C* = 5 dan kernel *Polynomial*

No	Nama File	True Positif	False Positif	Missing Rate
1	api-bakar_sampah.avi	0.00	0.00	100.00
2	api-boneka_dora.avi	0.49	0.00	99.51
3	api-kayu.avi	0.00	0.00	100.00
4	api-kayu2.avi	0.00	0.00	100.00
5	api-kebakaran_hutan.avi	0.00	0.00	100.00
6	api-kebakaran_hutan2.avi	22.22	0.00	77.78
7	api-kebakaran_hutan3.avi	0.00	0.00	100.00
8	api-kebakaran_hutan4.avi	0.00	0.00	100.00
9	api-kebakaran_ladang.avi	88.76	0.00	11.24

10	api-kebakaran_mobil.avi	9.31	0.00	90.69
11	api-kebakaran_mobil2.avi	0.00	0.00	100.00
12	api-kebakaran_tol.avi	0.00	0.00	100.00
13	api-kebakaran-truck.avi	33.33	0.00	66.67
14	api-kebakaran-truck2.avi	14.71	0.00	85.29
15	api-kebakaran-truck3.avi	27.45	0.00	72.55
16	api-kertas.avi	50.62	0.00	49.38
17	api-kertas2.avi	29.63	0.00	70.37
18	api-miniatur_mainan.avi	0.59	0.00	99.41
19	api-mobil_mainan.avi	5.88	0.00	94.12
20	api-orang_terjun.avi	1.47	0.00	98.53
21	api-pesawat_mainan.avi	19.44	0.00	80.56
22	api-pesawat_mainan2.avi	0.00	0.00	100.00
23	api-ruang_tamu.avi	0.00	0.00	100.00
24	api-rumah_mainan.avi	27.94	0.00	72.06
25	api-senter.avi	0.00	0.00	100.00
26	api-senter2.avi	0.00	0.00	100.00
27	non_api-anak_kecil.avi	100.00	0.00	0.00
28	non_api-anak_kecil2.avi	100.00	0.00	0.00
29	non_api-anak_kecil3.avi	100.00	0.00	0.00
30	non_api-bertemu.avi	100.00	0.00	0.00
31	non_api-jaket_merah.avi	100.00	0.00	0.00
32	non_api-jalan_malam.avi	100.00	0.00	0.00
33	non_api-jalan_malam2.avi	100.00	0.00	0.00
34	non_api-jalan_raya.avi	100.00	0.00	0.00
35	non_api-jalan_raya2.avi	100.00	0.00	0.00

36	non_api-jalan_raya3.avi	100.00	0.00	0.00
37	non_api-kantor.avi	100.00	0.00	0.00
38	non_api-kecelakaan.avi	100.00	0.00	0.00
39	non_api-kecelakaan2.avi	100.00	0.00	0.00
40	non_api-kecelakaan3.avi	100.00	0.00	0.00
41	non_api-kerusuhan.avi	100.00	0.00	0.00
42	non_api-kerusuhan2.avi	100.00	0.00	0.00
43	non_api-kerusuhan3.avi	100.00	0.00	0.00
44	non_api-las_vegas.avi	100.00	0.00	0.00
45	non_api-manuju_mobil.avi	100.00	0.00	0.00
46	non_api-manuju_mobil2.avi	100.00	0.00	0.00
47	non_api-manuju_mobil3.avi	100.00	0.00	0.00
48	non_api-parkiran.avi	100.00	0.00	0.00
49	non_api-parkiran2.avi	100.00	0.00	0.00
50	non_api-parkiran3.avi	100.00	0.00	0.00
51	non_api-pencuri.avi	100.00	0.00	0.00
52	non_api-pencuri2.avi	100.00	0.00	0.00
53	non_api-penembakan.avi	100.00	0.00	0.00
54	non_api-serbet.avi	100.00	0.00	0.00
55	non_api-tas.avi	100.00	0.00	0.00
56	non_api-tas2.avi	100.00	0.00	0.00
57	non_api-tas3.avi	100.00	0.00	0.00
58	non_api-televisi.avi	100.00	0.00	0.00
59	non_api-televisi2.avi	100.00	0.00	0.00

BIODATA PENULIS

Hamdi Ahmadi Muzakkiy atau biasa dipanggil Hamdi dilahirkan di Jakarta pada tanggal 15 April 1994 dan dibesarkan di Jakarta. Penulis adalah anak pertama dari tiga bersaudara.

Penulis menempuh pendidikan di SD kasih Ananda (1999-2006), SMP N 84 Jakarta (2006-2009), dan SMA N 75 Jakarta (2009-2012). Setelah lulus SMA penulis melanjutkan ke jenjang perkuliahan di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Bidang Studi yang diambil oleh penulis pada saat kuliah di Teknik Informatika ITS adalah Komputasi Cerdas dan Visualisasi.

Selama menempuh kuliah penulis aktif sebagai anggota Himpunan Mahasiswa Teknik Computer (HMTC) ITS. Penulis juga aktif dalam kegiatan kepanitiaan Schematics sebagai staff hubungan masyarakat (Humas) Schematics 2013 dan Wakil Ketua National Programming Contest (NPC) 2014. Selain itu penulis juga aktif menjadi administrator Lab pemrograman(LP) Teknik Informatika ITS.Penulis pernah menjadi asisten dosen dan praktikum untuk mata kuliah Pemrograman Terstruktur (2013), Algoritma dan Struktur Data (2013) , Basis Data (2014) dan Dasar Pemrograman (2015)

Penulis dapat dihubungi melalui alamat *email* hamdiahmadi1504@gmail.com.