taW

;/--n

**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]***

**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**

# 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 Desain dan Terapan Komputasi

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)

1. 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*

***[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*

***[Halaman ini sengaja dikosongkan]***

# KATA PENGANTAR



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

[LEMBAR PENGESAHAN v](#_Toc438162060)

[*Abstrak* vii](#_Toc438162061)

[*Abstract* ix](#_Toc438162062)

[KATA PENGANTAR xi](#_Toc438162063)

[DAFTAR ISI xiii](#_Toc438162064)

[DAFTAR GAMBAR xv](#_Toc438162065)

[DAFTAR TABEL xvii](#_Toc438162066)

[DAFTAR KODE SUMBER xix](#_Toc438162067)

[BAB I PENDAHULUAN 1](#_Toc438162068)

[1.1 Latar Belakang 1](#_Toc438162069)

[1.2 Rumusan Masalah 1](#_Toc438162070)

[1.3 Batasan Masalah 1](#_Toc438162071)

[1.4 Tujuan 1](#_Toc438162072)

[1.5 Manfaat 1](#_Toc438162073)

[1.6 Metodologi 1](#_Toc438162074)

[BAB II TINJAUAN PUSTAKA 3](#_Toc438162075)

[BAB III DESAIN PERANGKAT LUNAK 5](#_Toc438162076)

[3.1 Data 5](#_Toc438162077)

[3.1.1 Data Masukkan 5](#_Toc438162078)

[3.1.2 Data Keluaran 6](#_Toc438162079)

[3.2 Desain Sistem Secara Umum 7](#_Toc438162080)

[3.3 Preprocessing 8](#_Toc438162081)

[3.3.1 Reduksi Reduksi Size Frame 8](#_Toc438162082)

[3.3.2 Deteksi Gerak 9](#_Toc438162083)

[3.3.3 Probabilitas Warna Pixel 10](#_Toc438162084)

[3.3.4 Region Growing 11](#_Toc438162085)

[3.3.5 Luasan Region Api 12](#_Toc438162086)

[3.4 Verifikasi 13](#_Toc438162087)

[3.4.1 Ekstraksi Fitur dengan Wavelet 14](#_Toc438162088)

[3.4.2 Klasifikasi 15](#_Toc438162089)

[BAB IV IMPLEMENTASI 17](#_Toc438162090)

[4.1 Lingkungan Implementasi 17](#_Toc438162091)

[4.2 Implementasi 17](#_Toc438162092)

[4.2.1 Implementasi Tahap Reduksi Size Frame 17](#_Toc438162093)

[4.2.2 Implementasi Tahap Deteksi Gerak 18](#_Toc438162094)

[4.2.3 Implementasi Tahap Probabilitas Warna Pixel 19](#_Toc438162095)

[4.2.4 Implementasi Tahap Region Growing 23](#_Toc438162096)

[4.2.5 Implementasi Tahap Luasan Region Api 25](#_Toc438162097)

[4.2.6 Implementasi Tahap Ekstraksi Fitur dengan Wavelet 26](#_Toc438162098)

[4.2.7 Implementasi Tahap Klasifikasi 27](#_Toc438162099)

[BAB V UJI COBA DAN EVALUASI 30](#_Toc438162100)

[5.1 Lingkungan Uji Coba 30](#_Toc438162101)

[5.1.1 Data Uji Coba 30](#_Toc438162102)

[5.1.2 Skenario Uji Coba 31](#_Toc438162103)

[5.1.3 Skenario Uji Coba 1 31](#_Toc438162104)

[5.1.4 Skenario Uji Coba 2 32](#_Toc438162105)

[BAB VI KESIMPULAN DAN SARAN 35](#_Toc438162106)

[6.1 Kesimpulan 35](#_Toc438162107)

[6.2 Saran 35](#_Toc438162108)

[DAFTAR ACUAN 37](#_Toc438162109)

[LAMPIRAN A 39](#_Toc438162110)

[BIODATA PENULIS 41](#_Toc438162111)

# DAFTAR GAMBAR

[Gambar 3.1.1 Contoh Data Masukkan 6](#_Toc438162112)

[Gambar 3.1.1 Contoh Data Keluaran 6](#_Toc438162113)

[Gambar 3.2.1 Diagram alir rancangan perangkat lunak secara umum 8](#_Toc438162114)

[Gambar 3.3.1 Reduksi *Size Frame* 9](#_Toc438162115)

[Gambar 3.3.2 Contoh Deteksi Gerak 10](#_Toc438162116)

[Gambar 3.3.3 *Pseudocode* Probabilitas Warna Pixel 11](#_Toc438162117)

[Gambar 3.3.4 *Pseudocode Regon Growing* 12](#_Toc438162118)

[Gambar 3.3.5 Diagram Alir Variasi Warna Pixel 13](#_Toc438162119)

[Gambar 3.4.1 Diagram alir Ekstraksi Fitur 15](#_Toc438162120)

[Gambar 5.1.1 Contoh Video Kejadian 31](#_Toc438162121)

[Gambar 6.2.1 Hasil uji coba 15 kali pada permasalahan CTSTRING 40](#_Toc438162122)

# DAFTAR TABEL

[Tabel 4.1.1 Lingkungan Perancangan Perangkat Lunak 17](#_Toc438162336)

[Tabel 5.1.1 Lingkungan Perancangan Perangkat Lunak 32](#_Toc438162337)

[Tabel 6.2.1 Hasil uji coba 15 kali pada permasalahan CTSTRING 39](#_Toc438162338)

***[Halaman ini sengaja dikosongkan]***

# DAFTAR KODE SUMBER

[Kode Sumber 4.2.1 Implementasi Tahap Reduksi *Size Frame* 18](#_Toc438162125)

[Kode Sumber 4.2.2 Penggunaan Fungsi pyrDown() 18](#_Toc438162126)

[Kode Sumber 4.2.3 Implementasi Tahap Deteksi Gerak 19](#_Toc438162127)

[Kode Sumber 4.2.4 Implementasi Penyimpanan Pixel 19](#_Toc438162128)

[Kode Sumber 4.2.5 Implementasi Menghitung Nilai Standar Deviasi dan Mean Setiap *Channel* 20](#_Toc438162129)

[Kode Sumber 4.2.6 *Generate list* pixel api 21](#_Toc438162130)

[Kode Sumber 4.2.6 memanggil fungsi getGaussianProbability(). Fungsi getGaussianProbability() digunakan untuk menghitung nilai probabilitas tiap *channel* pixel. Fungsi tersebut mengimplementasikan rumus dari probabilitas distribusi gaussian. Implementasi fungsi getGaussianProbability() dapat dilihat pada Kode Sumber 4.2.7. 21](#_Toc438162131)

[Kode Sumber 4.2.7 Fungsi Menghitung Nilai Probabilitas Distribusi Gaussian 22](#_Toc438162132)

[Kode Sumber 4.2.8 Mendapatkan Threshold 22](#_Toc438162133)

[Kode Sumber 4.2.9 Membaca *List* Pixel Api 22](#_Toc438162134)

[Pada Kode Sumber 4.2.9, *list* pixel api dimasukkan kedalam *array* tiga dimensi. *Array* tiga dimensi tersebut bernilai True atau False, dimana jika 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 pixel yang didapatkan pada proses deteksi gerak. Implementasi proses pengecekan kandidat pixel dapat dilihat pada Kode Sumber 4.2.10 22](#_Toc438162135)

[Kode Sumber 4.2.10 Implementasi Tahap Probabilitas Warna Pixel 23](#_Toc438162136)

[Kode Sumber 4.2.11 Implementasi Tahap *Region Growing* 24](#_Toc438162137)

[Kode Sumber 4.2.12 Implementasi Tahap *Flood Fill* 24](#_Toc438162138)

[Kode Sumber 4.2.13 Implementasi Tahap *Clock Wise* 25](#_Toc438162139)

[Kode Sumber 4.2.14 Implementasi Tahap Variasi Warna *Region* 26](#_Toc438162140)

[Kode Sumber 4.2.15 Implementasi Memasukkan Nilai Wavelet Kedalam *List* 26](#_Toc438162141)

[Kode Sumber 4.2.16 Pemanggilan Fungsi Wavelet 27](#_Toc438162142)

[Kode Sumber 4.2.17 *Training* Klasifikasi 27](#_Toc438162143)

[Kode Sumber 4.2.18 Implementasi Tahap Klasifikasi 28](#_Toc438162144)

***[Halaman ini sengaja dikosongkan]***

# PENDAHULUAN

## Latar Belakang

## Rumusan Masalah

## Batasan Masalah

## Tujuan

## Manfaat

## Metodologi

***[Halaman ini sengaja dikosongkan]***

# TINJAUAN PUSTAKA

***[Halaman ini sengaja dikosongkan]***

# DESAIN PERANGKAT LUNAK

Pada bab ini akan 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 pebelajaran 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, probabilitas warna pixel, luasan *region* api dan klasifikasi.

## Data

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

### Data Masukkan

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



Gambar 3.1.1 Contoh Data Masukkan

### Data Keluaran

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



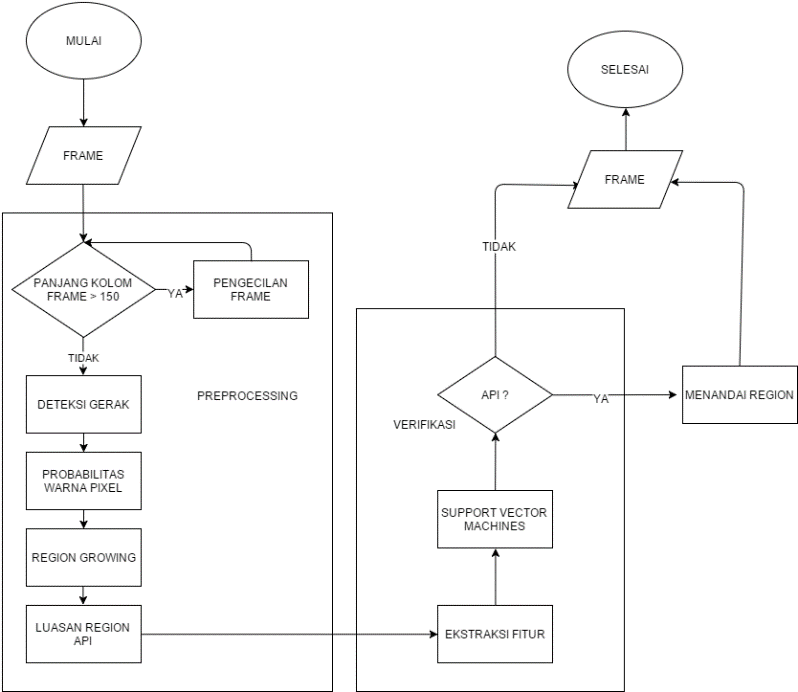
Gambar 3.1.1 Contoh Data Keluaran

## Desain Sistem Secara Umum

Rancangan perangkat lunak deteksi api berbasis sensor visual menggunakan *support vector machines* dimulai dengan membaca masukan berupa file video. Setiap *frame* akan dilakukan *preprocessing* sebelum dilakukan verifikasi api menggunakan *support vector machines*. Tahap pertama *preprocessing* adalah mengubah *size frame* yang diproses. Ukuran *frame* diubah hingga panjang kolom *frame* lebih kecil dari 150 pixel. Tahap berikutnya adalah malakukan deteksi gerak *frame*, hasil dari deteksi gerak adalah pixel-pixel bergerak terhadap *frame* sebelumnya. Setelah mendapatkan pixel-pixel bergerak, tahap selanjutnya adalah menentukan probabilitas warna tiap pixel menggunakan distribusi probabilitas gaussian. Warna api yang didefinisikan pada sistem ini adalah warna yang memiliki *range* dariwarna kuning hingga merah. Hasil keluaran dari metode probabilitas warna pixel adalah pixel-pixel yang masuk kedalam kandidat pixel api.

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

Setelah melalui tahap *preprocessing*, pixel-pixel yang termasuk kandidat api akan di verifikasi menggunakan metode *support vector machines*. Fitur didapatkan dari nilai konstanta wavelet setiap pixel statis dengan sepuluh *frame* yang berurutan. Hasil akhir yang dikeluarkan adalah adanya penanda pada *frame* yang diproses jika *frame* tersebut mengandung pixel api. Data dibagi menjadi data pembelajaran dan data uji sehingga dapat diperoleh nilai *true positif*, *false positif*, dan *missing rate* dari hasil klasifikasi. Diagram alir desain umum perangkat lunak ditunjukkan pada Gambar 3.2.1 berikut ini.



Gambar 3.2.1 Diagram alir rancangan perangkat lunak secara umum

## Preprocessing

Setiap pixel pada *Frame* yang diproses tidak langsung dilakukan verifikasi untuk menentukan apakah pixel tersebut pixel api atau bukan. Tahap awal yang dilakukan adalah *preprocessing. Preprocessing* bertujuan untuk menghilangkan pixel-pixel yang tidak memiliki karakteristik pixel api. *Preprocessing* dilakukan melalui lima tahap yaitu *sizing frame*, deteksi gerak, probabilitas warna pixel, *region growing,* luasan *region* api.

### Reduksi Reduksi Size Frame

Tahap reduksi *size frame* adalah tahap dimana *size* dari *frame* direduksi. Kualitas *frame* masukkan direduksi dengan tujuan mempercepat proses yang pendeteksian api. Gambar 3.3.1 adalah diagram alir dari proses reduksi *size frame*.

F:\KULIAH\TA\bukuTA\source\resizing.png

Gambar 3.3.1 Reduksi *Size Frame*

### Deteksi Gerak

Didalam proses ini dilakukan proses mendeteksi gerak pixel. Untuk mendapatkan kandidat pixel yang bergerak, dilakukan dengan mengurangi *frame* yang diproses saat ini dengan *frame* sebelumnya. Dari pengurangan *frame* tersebut didapatkan pixel-pixel yang bergerak. Ilustrasi deteksi gerak ditunjukkan pada Gambar 3.3.2.



Gambar 3.3.2 Contoh Deteksi Gerak

Setelah mendapatkan pixel yang bergerak, kumpulan pixel tersebut disimpan untuk diproses di tahap berikutnya, yakni probabilitas warna pixel.

### Probabilitas Warna Pixel

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

Sebelum melakukan perhitungan probabilitas pada setiap pixel 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. Perhitungan probabilitas pixel 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 pixel tersebut. Jika nilai probabilitas pixel tersebut melebihi *threshold*, maka pixel tersebut dimasukkan kedalam kandidat pixel api. *Pseudocode* fungsi ini dapat dilihat pada Gambar 3.3.3

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7. | for i=1 to length(CandidatePixel)  R,G,B = CandidatePixel[i].RGBvalue  If probability(R)\*probability(G)\* probability(B) > threshold  add CandidatePixel[i] to firePixel  else  add CandidatePixel[i] to nonFirePixel  return firePixel |

Gambar 3.3.3 *Pseudocode* Probabilitas Warna Pixel

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

### Region Growing

Kandidat pixel api yang didapatkan pada tahap probabilitas warna pixel dilakukan *region growing* untuk mendapatkan *region* dari setiap pixel. Masukkan dari tahap ini adalah *list* kandidat pixel api dan *frame* yang di proses. Setiap pixel akan dilakukan *region growing* dengan cara melakukan pengecekan terhadap delapan tetangga pixel tersebut. Untuk menentukan apakah pixel 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 pixel tersebut dianggap satu *region*. *Pseudocode* fungsi ini dapat dilihat pada Gambar 3.3.4.

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

Gambar 3.3.4 *Pseudocode Regon Growing*

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

### Luasan Region Api

Luasan *region* api adalah metode untuk melakukan *filter* terhadap *region* api. Masukkan dari tahap ini adalah *region* yang telah diperoleh dari metode *region growing*, kandaidat pixel api yang didapatkan dari metode probabilitas warna pixel. Dilakukan proses perhitungan luasansetiap *region*. Jika luasan *region* melebihi *threshold*, maka kandidat pixel api yang ada pada *region* tersebut masuk kedalam kandidat pixel api. Gambar 3.3.6 adalah diagram alir dari proses luasan *region* api.

F:\KULIAH\TA\bukuTA\source\variasi warna region.png

Gambar 3.3.5 Diagram Alir Variasi Warna Pixel

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

## Verifikasi

Pada subbab ini akan dijelaskan mengenai proses verifikasi pixel sehingga menghasilkan keluaran yang diharapkan seperti yang sudah dijelaskan sebelumnya. Setelah melalui tahap *preprocessing*, pixel-pixel yang masuk kedalam kandidat pixel 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 pixel tersebut masuk kedalam pixel api atau bukan.

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

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 pixelnya. Setelah dilakukan normalisasi, dilakukan pengurutan nilai fitur secara *ascending*. Gambar 3.4.1 adalah diagram alir dari proses ekstraksi fitur.

F:\KULIAH\TA\bukuTA\source\ekstraksi fitur.png

Gambar 3.4.1 Diagram alir Ekstraksi Fitur

### Klasifikasi

Klasifikasi dilakukan dengan mengolah data fitur yang sudah didapatkan dari proses ekstraksi fitur. Metode klasifikasi yang digunakan adalah *support vector machines* menggunakan kernel RBF. Hasil akhir dari proses ini adalah pixel-pixel yang masuk kedalam pixel api.

***[Halaman ini sengaja dikosongkan]***

# 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.

## Lingkungan Implementasi

Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam implementasi ini ditampilkan pada tabel xx.

|  |  |
| --- | --- |
| 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

## Implementasi

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

### Implementasi Tahap Reduksi Size Frame

Subbab 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.  2.  3.  4. | while (len(currentFrame)>150):  if (len(currentFrame)<=300):  currentFrame2 = copy.copy(currentFrame)  currentFrame = ImageProcessing.getDownSize(currentFrame) |

Kode Sumber 4.2.1 Implementasi Tahap Reduksi *Size Frame*

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

Kode Sumber 4.2.2 Penggunaan Fungsi pyrDown()

Kode Sumber 4.2.1 melakukan reduksi hinggal pixel kolom masukkan lebih kecil dari 150 pixel dengan memanggil fungsi getDownSize().

### Implementasi Tahap Deteksi Gerak

Subbab ini membahas implementasi tahap deteksi gerak pixel. 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.  2.  3. | def getMovingForeGround(self, image):  self.BckgrSbsMOG = cv2.BackgroundSubtractorMOG()  return self.BckgrSbsMOG.apply(image,learningRate = 0.00) |

Kode Sumber 4.2.3 Implementasi Tahap Deteksi Gerak

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

|  |  |
| --- | --- |
| 1.  2.  3. | def getMovingCandidatePixel(self, moving\_frame):  listY,listX = np.where( moving\_frame == 255 )  return np.vstack((listY,listX)) |

Kode Sumber 4.2.4 Implementasi Penyimpanan Pixel

Fungsi getMovingCandidatePixel() mengembalikan hasil berupa *list* indeks pixel yang bergerak. Hasil keluaran dari keseluruhan proses deteksi gerak adalah *list* yang berisi indeks pixel y dan x.

### Implementasi Tahap Probabilitas Warna Pixel

Probabilitas warna pixel dilakukan dengan pengecekan warna setiap pixel pada *list* pixelyang dihasilkan melalui proses deteksi gerak. Masukkan dari tahap ini adalah *list* pixel yang telah didapatkan pada tahap deteksi gerak dan *frame* dengan channel R,G,B. Sebelum menghitung probabilitas warna pixel, terlebih dahulu dilakukan perhitungan mencari nilai rata-rata dan standar deviasi untuk nilai pixel 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.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15.  16.  17.  18.  19.  20.  21.  22. | def getStdDevAndMean(self,path):  list\_file = File.readFolder(self,path)  R = []  G = []  B = []  for x in list\_file:  image = ImageProcessing.readImage(self,path+'/'+x)  r = np.array(image[:,:,2]).ravel()  g = np.array(image[:,:,1]).ravel()  b = np.array(image[:,:,0]).ravel()  R += (r.tolist())  G += (g.tolist())  B += (b.tolist())  mean = []  mean.append(np.average(B))  mean.append(np.average(G))  mean.append(np.average(R))  standard\_deviasi = []  standard\_deviasi.append(np.std(B))  standard\_deviasi.append(np.std(G))  standard\_deviasi.append(np.std(R))  return standard\_deviasi, mean |

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

Setelah didapatkan nilai rata-rata dan standar deviasi setiap *channel*, proses penentuan kandidat api menggunakan metode probabilitas warna pixel dilakukan. Setap pixel dihitung nilai probabilitas *channel* R,G,B. Selanjutnya, menghitung nilai probabilitas pixel dengan mengalikan nilai probabilitas R,G,B. Proses perhitungan probabilitas pixel memakan waktu yang cukup lama jika menghitung setiap kemungkinan kandidat pixel api. Pada implementasi, nilai probabilitas tiap pixel yang masuk kedalam probabilitas warna api dimasukkan kedalam file. Hal ini dilakukan karena proses perhitungan yang lama jika menghitungan probabilitas setiap pixel. Isi dari file adalah *list* kemungkinan pixel-pixel yang masuk dalam probabilitas warna api. Proses *generate* *list* pixel dapat dilihat pada Kode Sumber 4.2.6

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10. | def getListColorPixel  (self,list\_standard\_deviasi, list\_mean):  res = []  threshold = self.getThreshold()  for B in range(0,256):  for G in range(B,256):  for R in xrange(G,256):  if 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:  res.append([B,G,R])  return res |

Kode Sumber 4.2.6 *Generate list* pixel api

Kode Sumber 4.2.6 memanggil fungsi getGaussianProbability(). Fungsi getGaussianProbability() digunakan untuk menghitung nilai probabilitas tiap *channel* pixel. Fungsi tersebut mengimplementasikan rumus dari probabilitas distribusi gaussian. Implementasi fungsi getGaussianProbability() dapat dilihat pada Kode Sumber 4.2.7.

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10 | def getGaussianProbability(self, data, standard\_deviasi, mean):  data = float(data)  standard\_deviasi = float(standard\_deviasi)  mean = float(mean)  result = pow((data-mean),2)  div = 2\*pow(standard\_deviasi,2)  exp = np.exp(-result/div)  result = standard\_deviasi\*np.sqrt(2\*np.pi)  result = 1/result  return result\* exp |

Kode Sumber 4.2.7 Fungsi Menghitung Nilai Probabilitas Distribusi Gaussian

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

Kode Sumber 4.2.8 Mendapatkan Threshold

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* pixel yang sudah di *generate* sebelumnya dan menyimpan nilai pixel tersebut *array*. Proses pembacaan file bisa dilihat pada Kode Sumber 4.2.9

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

Kode Sumber 4.2.9 Membaca *List* Pixel Api

Pada Kode Sumber 4.2.9, *list* pixel api dimasukkan kedalam *array* tiga dimensi. *Array* tiga dimensi tersebut bernilai True atau False, dimana jika 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 pixel yang didapatkan pada proses deteksi gerak. Implementasi proses pengecekan kandidat pixel dapat dilihat pada Kode Sumber 4.2.10

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12. | def getColorCandidatePixel(self, list\_standard\_deviasi, list\_mean):  true\_pixel = []  false\_pixel = []  for x in range(0,len(list\_candidate[0])):  data = image[list\_candidate[0][x]] [list\_candidate[1][x]]  B,G,R = data[0],data[1],data[2]  if color\_dataset[B][G][R] == True:  true\_pixel.append([list\_candidate[0][x], list\_candidate[1][x]])  else : false\_pixel.append([list\_candidate[0][x], list\_candidate[1][x]])  return true\_pixel,false\_pixel |

Kode Sumber 4.2.10 Implementasi Tahap Probabilitas Warna Pixel

### Implementasi Tahap Region Growing

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

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15.  16. | def getRegionGrowing(self, list\_candidate, images, color\_dataset,counter):  gray\_image = ImageProcessing.getRGBtoGray(self,images)  is\_visit = gray\_image\*0  result\_image = copy.copy(gray\_image)  region\_number = 0  for x in list\_candidate:  coor\_y = x[0]  coor\_x = x[1]  if is\_visit[coor\_y][coor\_x] == 0:  stack = []  region\_number+=1  stack.append([coor\_y,coor\_x])  is\_visit[coor\_y][coor\_x] = region\_number  result\_image, is\_visit = self.doFloodFill( gray\_image, result\_image, is\_visit, stack, region\_number, color\_dataset, images)  return is\_visit |

Kode Sumber 4.2.11 Implementasi Tahap *Region Growing*

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15.  16.  17. | def doFloodFill(self, gray\_image ,result\_image ,is\_visit, stack, region\_number, color\_dataset, original\_image):  clocks = Data.getClockwise(self)  while len(stack) != 0:  coory,coorx = stack[0]  stack.pop(0)  result\_image[coory][coorx] = 255  data = original\_image[coory][coorx]  B,G,R = data[0],data[1],data[2]  for x in clocks:  try :  if is\_visit[coory+x[0]][coorx+x[1]] == 0 and color\_dataset[B][G][R] == True :  is\_visit[coory+x[0]] [coorx+x[1]] = region\_number  stack.append([coory+x[0], coorx+x[1]]  except :  pass  return result\_image,is\_visit |

Kode Sumber 4.2.12 Implementasi Tahap *Flood Fill*

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11. | def getClockwise(self):  clocks = []  clocks.append([-1,-1])  clocks.append([-1,0])  clocks.append([-1,1])  clocks.append([0,-1])  clocks.append([0,1])  clocks.append([1,-1])  clocks.append([1,0])  clocks.append([1,1])  return clocks |

Kode Sumber 4.2.13 Implementasi Tahap *Clock Wise*

Fungsi getRegionGrowing() melakukan iterasi kandidat pixel yang didapatkan dari tahap probabilitas warna api. Fungsi getRegionGrowing() memanggil fungsi doFloodFill(), dimana fungsi doFloodFill() melakukan *region growing* terhadap satu pixel dengan nilai nomor *region* yang akan bertambah pada fungsi getRegionGrowing(). Jika pixel sudah mempunyai nomor *region*, maka pixel tersebut tidak akan dilakukan pengecekan lagi. Pengecekan dilakukan menggunakan fungsi getClockwise(), fungsi ini akan mengembalikan nilai array jarum jam . Keluaran dari tahap ini adalah *region* dengan nomor *region* sebagai pembeda antar *region*.

### Implementasi Tahap Luasan Region Api

Subbab ini membahas implementasi tahap variasi warna *region*. Masukkan dari tahap ini adalah kandidat pixel api yang telah didapatkan dari tahap probabilitas warna api, dan *region*. Pada tahap luasan *region* api dihitung banyaknya pixel yang ada pada *region* tersebut. Jika luasan pixel *region* tersebut melebihi batas, maka pixel yang ada pada *region* tersebut masuk sebagai kandidat pixel api . Implementasi mendapatkan nilai variasi warna regiondapat dilihat pada Kode Sumber 4.2.16

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15.  16.  17.  18.  19. | def getFilterSizeRegion (self,list\_candidate,region):  true\_pixel = []  false\_pixel = []  list\_region = np.unique(region)  threshold = dict()  for x in range(1,len(list\_region)):  lists = np.where(region == x)  if len(lists[0]) > 1\*len(region)\* len(region[0])/100:  threshold[x] = True  else :  threshold[x] = False  for x in list\_candidate:  coor\_y = x[0]  coor\_x = x[1]  if threshold[region[coor\_y][coor\_x]] == True:  true\_pixel.append([coor\_y,coor\_x])  else :  false\_pixel.append([coor\_y,coor\_x])  return true\_pixel,false\_pixel |

Kode Sumber 4.2.14 Implementasi Tahap Variasi Warna *Region*

### Implementasi Tahap Ekstraksi Fitur dengan Wavelet

Subbab 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.  2. | LL,(HL,LH,HH) = wv.toWavelet(copy.copy(grayImage))  list\_wavelet.append([HL,LH,HH]) |

Kode Sumber 4.2.15 Implementasi Memasukkan Nilai Wavelet Kedalam *List*

Pada Kode Sumber 4.2.20 ,fungsi toWavelet() digunakan untuk mengubah *frame* kedalam domain wavelet. Implementasi mengubah *frame* kedalam domain wavelet dapat dilihat pada Kode Sumber 4.2.16

|  |  |
| --- | --- |
| 1.  2. | def toWavelet(image):  return pywt.dwt2(image,'db4') |

Kode Sumber 4.2.16 Pemanggilan Fungsi Wavelet

### Implementasi Tahap Klasifikasi

Subbab ini membahas implementasi tahap klasifikasi. Pada tahap ini sistem diberi masukkan kandidat pixel api dan sepuluh buah *frame* domain wavelet, dimana masing-masing *frame* berisi tiga buah sub *frame* seperti penjelasan subbab 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.  2.  3. | def getClassifier(datatraining):  x,y = readDataSet(datatraining)  clf = svm.SVC(kernel = 'rbf',C = 3.5)  clf.fit(x,y)  return clf |

Kode Sumber 4.2.17 *Training* Klasifikasi

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

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15.  16.  17.  18.  19.  20.  21.  22.  23.  24.  25.  26.  27.  28. | def doClassification(classifier, list, wavelet):  truePixel = []  falsePixel = []  listMax = []  listMin = []  cpyWavelet = np.int\_(copy.copy(wavelet))  cpyWavelet = np.power(cpyWavelet,2)  for x in cpyWavelet:  lists = np.add(np.add(x[0],x[1]) , x[2])  listMax.append(np.max(lists))  listMin.append(np.min(lists))  for x in list:  data = []  cnt= 0  for y in wavelet:  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':  truePixel.append([x[0],x[1]])  else :  falsePixel.append([x[0],x[1]])  return truePixel,falsePixel |

Kode Sumber 4.2.18 Implementasi Tahap Klasifikasi

***[Halaman ini sengaja dikosongkan]***

# UJI COBA DAN EVALUASI

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

## 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

* 1. Perangkat Lunak

Sistem Operasi Microsoft Windows 8 64-bit Pro

Perangkat Pengembang PyCharm

### 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 pixel 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 tiga video dengan jumlah video api sejumlah dua puluh lima dan video bukan api berjumlah dua puluh delapan.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 5.1.1 Contoh Video Kejadian

### Skenario Uji Coba

Pada subbab 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 probabilitas 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 klasifikkasi dengan kernel tetap yaitu rbf. Nilai C yang akan diuji yaitu 1, 3.5, 5.

### Skenario Uji Coba 1

Sekenario uji coba 1 adalah perhitungan *true positif*, *false positif*, dan *missing rate*. Dimana *false 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 pixel api. Nilai *threshold* yang diuji yaitu 10-7, 10-8, 5 x 10-9, 10-9.

|  |  |  |  |
| --- | --- | --- | --- |
| *Threshold* | *True Positif* | *False Positif* | *Missing Rate* |
| 10-7 | 72.64 | 0.36 | 27.01 |
| 10-8 | 88.80 | 0.44 | 10.76 |
| 5x10-9 | 91.69 | 0.66 | 7.65 |
| 10-9 | 94.02 | 3.20 | 2.78 |

Tabel 5.1.1 Hasil Uji Coba 1

Dari hasil uji yang dilakukan, makin kecil nilai *threshold* yang digunakan, hasil dari *true positif* akan semakin besar. Begitu pula untuk *false positif* dimana makin kecil nilai *threshold* makin besar nilai *false positif*. Hal ini dikarenakan pixel yang dianggap pixel 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 di lampiran.

### 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

|  |  |  |  |
| --- | --- | --- | --- |
| C | *True Positif* | *False Positif* | *Missing Rate* |
| 1 | 90.25 | 0.63 | 9.12 |
| 3.5 | 91.69 | 0.66 | 7.65 |
| 5 | 91.89 | 0.66 | 7.45 |

Tabel 5.1.2 Hasil Uji Coba 2

Hasil uji coba tahap 2 didapatkan nilai *false positif* yang sama pada nilai C 3.5 dengan 5. Namun, nilai C = 5 memiliki nilai *true positif* yang lebih tinggi dari C = 3.5.

***[Halaman ini sengaja dikosongkan]***

# 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.

## Kesimpulan

## Saran

# DAFTAR ACUAN

|  |  |
| --- | --- |
| [1] | G. Navaro and M. Rafinot, Flexible Pattern Matching in Strings, Cambridge: Cambridge University Press, 2002. |
| [2] | J. Power, "Constructing a DFA from an NFA (Subset Construction)," National University of Ireland, 29 November 2002. [Online]. Available: http://www.cs.nuim.ie/~jpower/Courses/Previous/parsing/node9.html. [Accessed 25 April 2015]. |
| [3] | kuruma, "A tutorial on Fast Modulo Multiplication (Exponential Squaring)," 12 Agustus 2013. [Online]. Available: http://discuss.codechef.com/questions/20451/a-tutorial-on-fast-modulo-multiplication-exponential-squaring. [Accessed 25 April 2015]. |
| [4] | V. Jalan, "Count Strings," 9 September 2012. [Online]. Available: http://www.spoj.com/problems/CTSTRING/. [Accessed 10 Desember 2014]. |

***[Halaman ini sengaja dikosongkan]***

# LAMPIRAN A

Tabel 6.2.1 Hasil Uji Coba Manggunakan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Nama File | True Positif | False Positif | Missing Rate |
| 1 | api\_cars.avi | 95.05 | 0.00 | 4.95 |
| 2 | api\_doll\_house2.avi | 100.00 | 0.00 | 0.00 |
| 3 | api\_dora.avi | 79.21 | 0.00 | 20.79 |
| 4 | api\_hutan\_besar1.avi | 55.67 | 0.00 | 44.33 |
| 5 | api\_kayu\_hutan\_besar1.avi | 84.54 | 0.00 | 15.46 |
| 6 | api\_kayu\_hutan1.avi | 100.00 | 0.00 | 0.00 |
| 7 | api\_kayu1.avi | 88.61 | 0.00 | 11.39 |
| 8 | api\_kayu2.avi | 97.03 | 0.00 | 2.97 |
| 9 | api\_kertas1.avi | 67.50 | 0.00 | 32.50 |
| 10 | api\_kertas2.avi | 97.50 | 0.00 | 2.50 |
| 11 | api\_ladang1.avi | 100.00 | 0.00 | 0.00 |
| 12 | api\_mainan1.avi | 100.00 | 0.00 | 0.00 |
| 13 | api\_mainan2.avi | 100.00 | 0.00 | 0.00 |
| 14 | api\_mobil1.avi | 97.03 | 0.00 | 2.97 |
| 15 | api\_mobil2.avi | 90.10 | 0.00 | 9.90 |
| 16 | api\_rc2.avi | 89.22 | 0.00 | 10.78 |
| 17 | api\_ruang\_tamu1.avi | 49.10 | 0.00 | 50.90 |
| 18 | api\_senter1.avi | 26.24 | 0.00 | 73.76 |
| 19 | api\_senter2.avi | 6.93 | 0.00 | 93.07 |
| 20 | api\_terjun.avi | 80.20 | 0.00 | 19.80 |
| 21 | api\_tol2.avi | 91.58 | 0.00 | 8.42 |
| 22 | api\_truck1.avi | 100.00 | 0.00 | 0.00 |
| 23 | api\_truck2.avi | 99.01 | 0.00 | 0.99 |
| 24 | api\_truck3.avi | 100.00 | 0.00 | 0.00 |
| 25 | non\_api\_anak\_kecil1.avi | 100.00 | 0.00 | 0.00 |
| 26 | non\_api\_anak\_kecil2.avi | 100.00 | 0.00 | 0.00 |
| 27 | non\_api\_anak\_kecil3.avi | 100.00 | 0.00 | 0.00 |
| 28 | non\_api\_bertemu.avi | 100.00 | 0.00 | 0.00 |
| 29 | non\_api\_hamdi2.avi | 72.63 | 27.37 | 0.00 |
| 30 | non\_api\_jaket\_merah.avi | 100.00 | 0.00 | 0.00 |
| 31 | non\_api\_jalan\_malam1.avi | 100.00 | 0.00 | 0.00 |
| 32 | non\_api\_jalan\_malam2.avi | 100.00 | 0.00 | 0.00 |
| 33 | non\_api\_jalan\_raya1.avi | 100.00 | 0.00 | 0.00 |
| 34 | non\_api\_jalan\_raya2.avi | 100.00 | 0.00 | 0.00 |
| 35 | non\_api\_jalan\_raya3.avi | 100.00 | 0.00 | 0.00 |
| 36 | non\_api\_ke\_mobil1.avi | 100.00 | 0.00 | 0.00 |
| 37 | non\_api\_ke\_mobil2.avi | 100.00 | 0.00 | 0.00 |
| 38 | non\_api\_ke\_mobil3.avi | 100.00 | 0.00 | 0.00 |
| 39 | non\_api\_kecelakaan1.avi | 100.00 | 0.00 | 0.00 |
| 40 | non\_api\_kecelakaan2.avi | 100.00 | 0.00 | 0.00 |
| 41 | non\_api\_kecelakaan3.avi | 100.00 | 0.00 | 0.00 |
| 42 | non\_api\_las\_vegas1.avi | 100.00 | 0.00 | 0.00 |
| 43 | non\_api\_parkiran1.avi | 100.00 | 0.00 | 0.00 |
| 44 | non\_api\_parkiran2.avi | 100.00 | 0.00 | 0.00 |
| 45 | non\_api\_parkiran3.avi | 100.00 | 0.00 | 0.00 |
| 46 | non\_api\_pencuri1.avi | 92.22 | 7.78 | 0.00 |
| 47 | non\_api\_pencuri3.avi | 100.00 | 0.00 | 0.00 |
| 48 | non\_api\_tauran1.avi | 100.00 | 0.00 | 0.00 |
| 49 | non\_api\_tauran2.avi | 100.00 | 0.00 | 0.00 |
| 50 | non\_api\_tauran3.avi | 100.00 | 0.00 | 0.00 |
| 51 | non\_api\_televisi1.avi | 100.00 | 0.00 | 0.00 |
| 52 | non\_api\_televisi2.avi | 100.00 | 0.00 | 0.00 |
| 53 | non\_api\_tembak3.avi | 100.00 | 0.00 | 0.00 |

# BIODATA PENULIS

Penulis memiliki nama lengkap Muhammad Yunus Bahari, lahir di Blitar pada 26 Mei 1993. Anak pertama dari 4 bersaudara. Penulis telah menempuh pendidikan formal pada jenjang TK sampai dengan S-1 di TK Aisyah Bustanul Athfal kab. Sumbawa Besar (1997-1999), SDN Kalipang 1 kab. Blitar (1999-2002), SDN Tlogo 1 kab. Blitar (2002-2005), SMPN 3 Blitar (2005-2008), SMAN 1 Blitar (2008-2011), Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya (2011-2015). Penulis pernah menjuarai kompetisi SNITCH I FTIf ITS dalam kategori pengembangan permainan. Semenjak tahun 2012 penulis telah aktif sebagai anggota dalam Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) ITS. Penulis pernah menjadi anggota (2012) dan ketua sub-divisi (2013) dalam National Logic Competition (NLC) Schematics ITS. Penulis juga pernah menjadi Koordinator Lab Algoritma dan Pemrograman (2014-2015) dan menjadi asisten mata kuliah Pemrograman Terstruktur (2012), Basis Data (2013), Pemrograman Framework .NET (2013) di Teknik Informatika ITS dan Basis Data PIKTI ITS (2013).