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*, 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 Masukan

Data masukan 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 240 x 320 piksel. Ilustrasi data masukan dapat dilihat pada Gambar 3.1.

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. Data yang digunakan sebagai data pembelajaran adalah empat data video bukan api dan enam data video api.

Gambar 3.1 Contoh Data Masukan

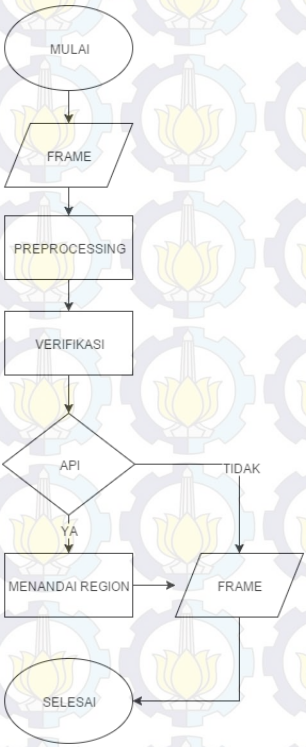
3.1.3 Data Keluaran

Data masukan akan diputar ulang dan diproses setiap *frame* menggunakan metode reduksi *frame*, deteksi gerak, deteksi warna piksel, *region growing*, perhitungan luasan *region*, 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. Ilustrasi data keluaran dapat dilihat pada Gambar 3.2.

Gambar 3.2 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 3.3.



Gambar 3.3 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 emapt kali lebih kecil dari ukuran *frame* yang diproses. Tahap berikutnya adalah malakukan deteksi gerak *frame*, hasil dari deteksi gerak adalah piksel-piksel bergerak. 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*. 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.

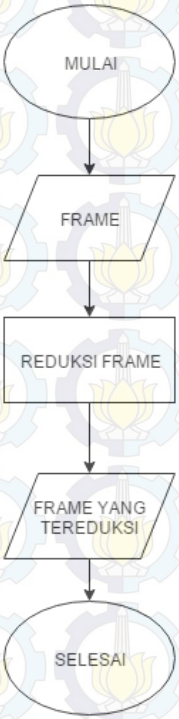
3.3 Preprocessing

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*. Diagram alir tahap *preprocessing* ditunjukkan pada Gambar 3.5.

3.3.1 Reduksi Size Frame

Tahap reduksi *size frame* adalah tahap dimana *size* dari *frame* direduksi. Kualitas *frame* masukan direduksi dengan tujuan mempercepat proses pendeteksian api. Diagram alir proses reduksi *size frame* dapat dilihat pada Gambar 3.4.



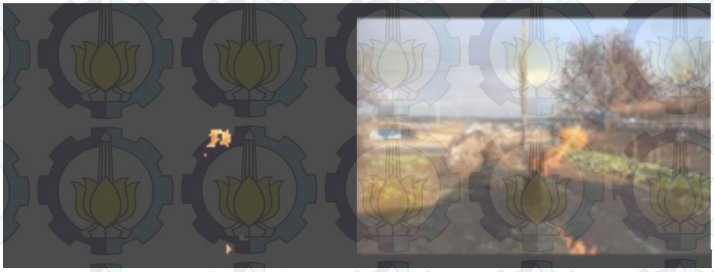
Gambar 3.4 Digram Alir Reduksi *Size Frame*

Gambar 3.5 Diagram Alir *Preprocessing*

Masukan 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 seperempat dari *frame* awal.

3.3.2 Deteksi Gerak

Didalam proses ini dilakukan proses deteksi gerak piksel. Masukan dari proses ini adalah *frame* yang sedang di proses. Untuk mendapatkan kandidat piksel yang bergerak, dilakukan dengan metode *gaussian mixture models*. Dari metode tersebut didapatkan piksel-piksel yang bergerak. Ilustrasi deteksi gerak ditunjukkan pada Gambar 3.6.



Gambar 3.6 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. Masukan 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. Ilustrasi *dataset* gambar api dapat dilihat pada Gambar 3.7.

Gambar 3.7 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 dimasukan kedalam kandidat piksel api. *Pseudocode* fungsi dapat dilihat pada Gambar 3.8.

```
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 3.8 *Pseudocode* Deteksi Warna Piksel

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

3.3.4 Region Growing

Kandidat piksel api yang didapatkan pada tahap deteksi warna piksel dilakukan *region growing* untuk mendapatkan *region* dari setiap piksel. Masukan 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 dapat dilihat pada Gambar 3.9.

```
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 3.9 *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.

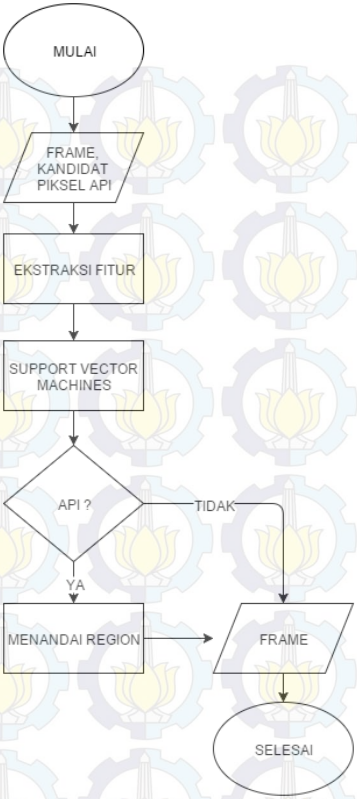
3.3.5 Perhitungan Luasan Region

Perhitungan luasan *region* adalah metode untuk melakukan *filter* terhadap *region* api. Masukan 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 3.10 adalah diagram alir dari proses perhitungan luasan *region*. 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 3.11.

Gambar 3.10 Diagram Alir Perhitungan Luasan *Region*



Gambar 3.11 Diagram Alir Verifikasi

3.4.1 Ekstraksi Fitur dengan Wavelet

Data masukan 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* LH, HL, HH. *Wavelet* yang digunakan adalah *wavelet* daubechies 4 . Tiap kandidat piksel api dicari nilai piksel pada sub *frame wavelet* yang telah dilakukan. Setiap piksel mendapatkan tiga nilai untuk setiap *framenya*, yaitu nilai LH, HL, HH. Dari ketiga nilai tersebut akan dihitung menggunakan Persamaan (3.1).

$$M_n(x,y) = |LH_n(x,y)|^2 + |HL_n(x,y)|^2 + |HH_n(x,y)|^2 \quad (3.1)$$

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 berjumlah 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*. Diagram alir ekstraksi fitur dapat dilihat pada Gambar 3.12.

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*. Masukan 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.

Gambar 3.12 Diagram Alir Ekstraksi Fitur

3.5 Menandai Region Api

Penanda *region* api dilakukan untuk menunjukkan bagian yang terdeteksi api. Masukan dari proses ini adalah piksel-piksel yang lolos tahap verifikasi dan *frame* yang sedang di proses. *Pseudocode* fungsi ini dapat dilihat pada Gambar 3.13

```
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 3.13 *Pseudocode* fungsi menandai *region*

Hasil yang dikeluarkan pada proses ini adalah *frame* yang memiliki tanda jika terdapat objek api. Ilustrasi menandai *region* api dapat dilihat pada Gambar 3.14.

Gambar 3.14 Menandai *region* api