Deteksi Api Berbasis Sensor Visual Menggunakan Metode Support Vector Machines

Hamdi Ahmadi Muzakkiy, Handayani Tjandrasa, dan Chastine Fatichah
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia
e-mail: handatj@its.ac.id

***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 karena 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*, ekstraksi fitur *wavelet* dan klasifikasi piksel menggunakan *support vector machines*. Hasil dari deteksi bentuk akan digunakan dalam proses penentuan api.

Dataset yang digunakan dalam proses uji coba berisi enam puluh tujuh video dengan panjang video enam sampai enambelas detik yang diambil dari berbagai sumber. Performa terbaik yang dihasilkan adalah *true positif* sebesar 96.32%, *false positif* sebesar 1.46% dan *missing rate* sebesar 2.23%.

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

I. PENDAHULUAN

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 asap, suhu, atau radiasi [1].

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 dilakukan preprocessing, dan terakhir 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*.

II. DASAR TEORI

A. Gaussian Mixture Models

Gaussian mixture models adalah metode deteksi gerak. Diberikan nilai piksel sebagai x_N dan bilangan K, dimana K adalah bilangan gaussian (antara 3 hingga 5). Setiap distribusi K memiliki nilai w, μ, dan σ. 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.

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

Dimana nilai T adalah nilai minimum dari *background model*. Selanjutnya dilakukan *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 [2].

$$w_k^{N+1} = (1-\alpha)w_k^N + \alpha p(w_k|x_{n+1}) \tag{2}$$

$$\mu_k^{N+1} = (1-\alpha)\mu_k^N + \rho x_{n+1} \tag{3}$$

$$\sigma_k^{N+1} = (1-\alpha)\sigma_k^N + \rho(x_{N+1}-\mu_k^{N+1})(x_{N+1}-\mu_k^{N+1})^T \tag{4}$$

$$\rho = \alpha\,\eta\left(X_{N+1}, \mu_k^N, cov_k^{N+1}\right) \tag{5}$$

$$p(w_k|x_{n+1}) = \begin{cases} 1, & w_k \text{ adalah komponen gaussian yang match pertama kali} \\ 0, & \text{selain itu} \end{cases} \tag{6}$$

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.

B. Gaussian Pyramid

Gaussian pyramid digunakan untuk melakukan reduksi resolusi citra. Citra yang tereduksi resolusinya akan berkurang menjadi setengah dari resolusi awal. *Gaussian pyramid* dilakukan dengan dua operasi, yaitu *smoothing* dan *down sampling*. *Smoothing* dilakukan dengan menggunakan filter 5x5. Persamaan *Gaussian Pyramid* dapat dilihat pada persamaan berikut [3].

$$g_1(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) \cdot g_0(2i+m, 2j+n) \quad (7)$$

Dimana w adalah filter 5x5, g_0 adalah citra masukan dan g_1 adalah citra keluaran.

C. 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 gaussian dapat dilihat pada persamaan berikut [4].

$$p = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (8)$$

Nilai probabilitas warna piksel didapatkan dengan mengalikan probabilitas setiap *channel*. Persamaan probabilitas warna piksel dapat dilihat pada persamaan berikut.

$$p(I(x,y)) = \prod_{i \in \{R,G,B\}} p_i(I_i(x,y)) \quad (9)$$

D. 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* [5].

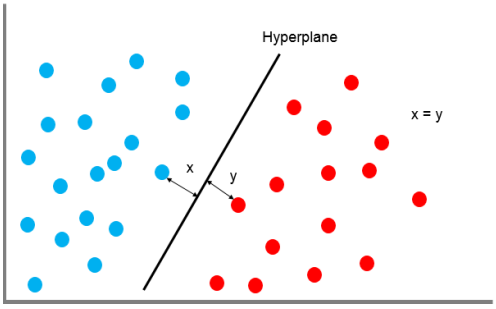
E. Daubachies 4 Wavelet

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 [6]. Pada citra, transformasi *wavelet* adalah transformasi yang melakukan filter terhadap suatu masukan. Filter yang digunakan dalam *wavelet* adalah *high pass filter* dan *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.

F. Support Vector Machines

Support vector machines adalah metode klasifikasi yang mengklasifikasikan dua kelas, yaitu kelas +1 dan -1. Pada metode klasifikasi *support vector machines*, 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 1.



Gambar 1. Ilustrasi *Support Vector Machines*

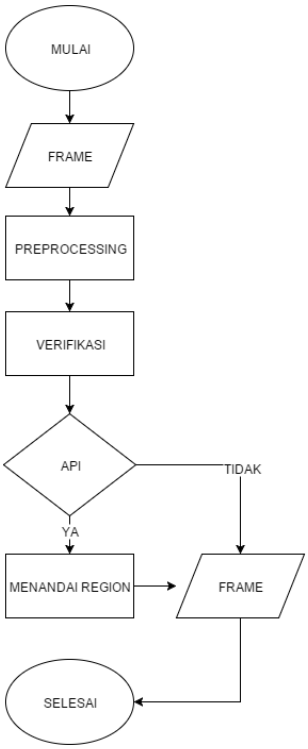
Fungsi *hyperplane* dibuat dengan persamaan berikut.

$$w \cdot x + b = 0 \quad (10)$$

III. PERANCANGAN PERANGKAT LUNAK

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 2.

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



Gambar 2. Diagram Alir Rancangan Perangkat Lunak Secara Umum

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.

A. Preprocessing

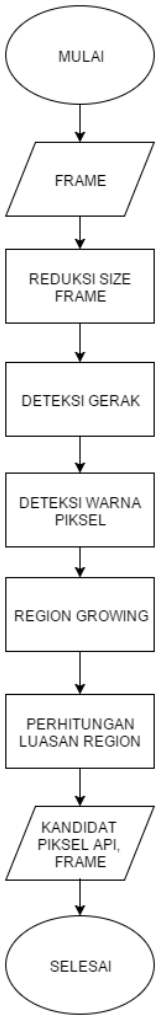
Preprocessing bertujuan untuk menghilangkan piksel-piksel yang tidak memiliki karakteristik piksel api. *Preprocessing* dilakukan melalui lima tahap yaitu reduksi *size frame*, deteksi gerak, deteksi warna piksel, *region growing*, perhitungan luasan *region*. Diagram alir tahap *preprocessing* ditunjukkan pada Gambar 3.

B. Verifikasi

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 4.

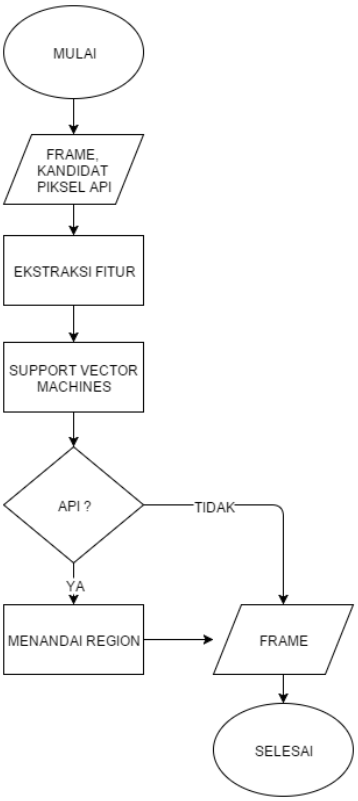
IV. IMPLEMENTASI

Perangkat lunak ini dibangun dengan menggunakan perangkat keras berprosessor Intel® Core™ i3-2350M CPU @ 2.30GHz

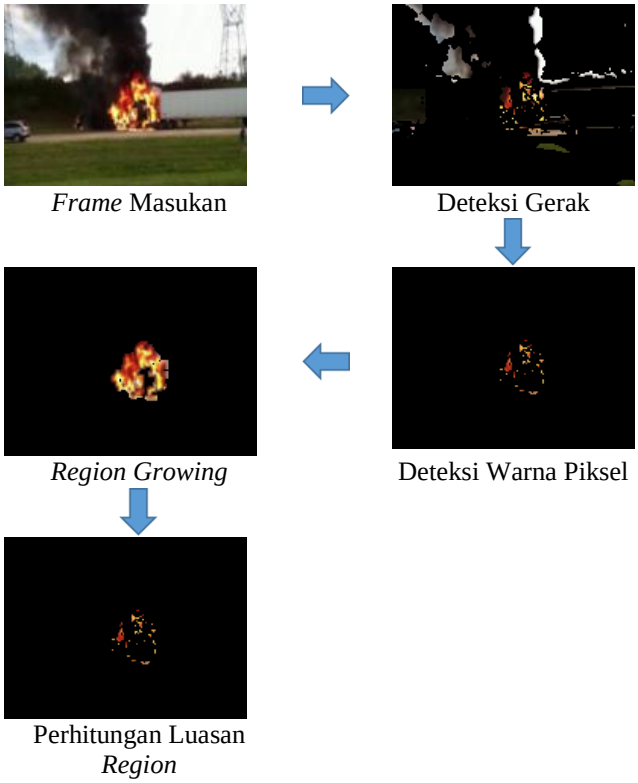


Gambar 3. Diagram Alir *Preprocessing*

2.30GHz Memori 4.00 GB. Sedangkan perangkat lunak yang digunakan antara lain sistem operasi Microsoft Windows 8 64-bit Pro dan perangkat pengembang PyCharm. Data masukan sistem adalah data video dengan ukuran piksel 240x320. Video akan diproses setiap *frame*. Pada *preprocessing*, dilakukan reduksi *size frame* menggunakan *gaussian pyramid*. Hasil dari reduksi *size frame* adalah *frame frame* dengan ukuran piksel 120x160. Selanjutnya adalah deteksi gerak menggunakan *gaussian mixture model*. Pada tahap ini didapatkan piksel-piksel yang bergerak. Koordinat piksel-piksel yang bergerak disimpan kedalam *array*. Setelah mendapatkan piksel-piksel yang bergerak, selanjutnya dilakukan deteksi warna piksel. Piksel-piksel yang bergerak dilakukan pengecekan warna menggunakan probabilitas distribusi gaussian. Setiap piksel dilakukan perhitungan probabilitas piksel tersebut dengan probabilitas nilai *channel red*, *green*, *blue*. *List* kombinasi *channel* yang masuk kedalam piksel api disimpan kedalam *array* untuk mempercepat peroses pengecekan. Setiap piksel yang lolos tahap deteksi warna dimasukkan kedalam *array* dengan menyimpan koordinat piksel tersebut. Setelah mendapatkan piksel-piksel yang masuk kedalam kandidat piksel api, dilakukan *region growing* untuk mendapatkan *region* dari setiap piksel. Pada *region growing*, tingkat homogen suatu piksel ditentukan apakah piksel tetangga masuk kedalam piksel api. Hasil keluaran dari tahap *region growing*



Gambar 4. Diagram Alir Verifikasi



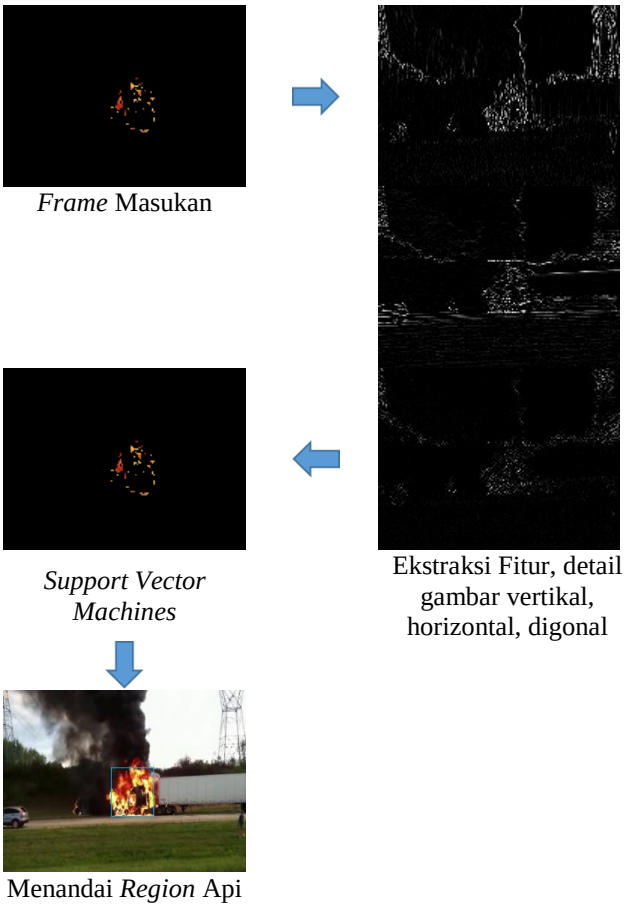
Gambar 5. Ilustrasi Tahap *Preprocessing*

adalah *region-region* dengan nilai pembeda setiap *region*. Tahap akhir *preprocessing* adalah perhitungan luasan *region*. Perhitungan luasan *region* dilakukan menggunakan keluaran dari *region growing* untuk mendapatkan luasan *region* dan kandidat piksel api. Jika *region* masuk lebih besar dari *threshold*, maka seluruh kandidat piksel api *region* dimasukkan kedalam *array*.

Setela melalui tahap *preprocessing*, setiap piksel dilakukan verifikasi. Setiap piksel dihitung nilai *magnitude* hasil transformasi *wavelet detail* horizontal, vertikal, dan diagonal. *Wavelet* yang digunakan adalah *daubachies* 4. Persamaan perhitungan *magnitude* dapat dilakukan menggunakan persamaan berikut.

$$M_n(x,y) = |LH_n(x,y)|^2 + |HL_n(x,y)|^2 + |HH_n(x,y)|^2 \quad (11)$$

Banyaknya *frame* untuk mendapatkan fitur adalah sepuluh buah *frame*. Dimana *frame* yang digunakan adalah *frame* yang sedang diproses dan sembilan *frame* sebelumnya secara berurutan. Setiap fitur dilakukan normalisasi min-max, dimana nilai min dan max didapatkan dari nilai *magnitude* ter-rendah dan ter-tinggi setiap *frame*. Setelah dilakukan normalisasi, dilakukan *sotring ascending* terhadap fitur. Hasil fitur yang telah di-*sorting* dilakukan klasifikasi menggunakan *support vector machines*. Setelah semua piksel dilakukan klasifikasi, jika terdapat piksel api dilakukan penandaan menggunakan titik ekstrim dari piksel-piksel api. Gambar 5 dan Gambar 6 adalah ilustrasi tahap *preprocessing* dan verifikasi.



Gambar 6. Ilustrasi Tahap *Verifikasi*

V. UJI COBA

Terdapat enam skenario dalam uji coba perangakat lunak ini. setiap skenario menghitung nilai *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. Data yang diuji sebanyak 67 data dengan jumlah data video api sebanyak 34 dan video bukan api sebanyak 33.

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.

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.

Sekenario uji 3 dilakukan uji coba variasi kernel klasifikasi. Variasi kernel yang digunakan yaitu *polynomial 2*, *polynomial 3*, dan RBF. Untuk parameter *threshold* warna piksel diberikan nilai 5 x 10^{-9} dan nilai C diberikan nilai 5.

Sekenario uji 4 dilakukan uji coba variasi besarnya *region*. Variasi *region* yang digunakan yaitu 1%, 5%, 10%. Untuk parameter *threshold* warna piksel diberikan nilai 5 x 10^{-9} , nilai C diberikan nilai 5 dan menggunakan kernel RBF.

Sekenario uji 5 dilakukan uji coba variasi *size frame*. Variasi *size frame* yang digunakan yaitu 240 x 320, 120 x 160, 60 x 80. Untuk parameter *threshold* warna piksel diberikan nilai 5 x 10^{-9} , nilai C diberikan nilai 5 dan menggunakan kernel RBF.

Skenario uji coba 6, dilakukan Perbandingan hasil *true positif*, *false positif*, dan *missing rate* dengan menghilangkan tahap *region growing* dan perhitungan luasan *region*. Hasil uji coba dapat dilihat pada Tabel **1**. Hasil Uji Coba 1Tabel **1**.

Berdasarkan keenam uji coba skenario di atas, hasil terbaik didapatkan pada skenario 2 dengan variasi C = 7. *True positif* tertinggi didapatkan pada skenario 5 dengan variasi ukuran *frame* yang diproses sebesar 60x80 piksel, namun memiliki *false positif* yang lebih tinggi dibandingkan skenario 2 dengan variasi C = 7.

Tabel 1. Hasil Uji Coba 1

Skenario	Variasi	*True Positif (%)*	*False Positif (%)*	*Missing Rate (%)*
1	10^{-7}	77.58	0.37	22.05
	10^{-8}	94.58	1.22	4.19
	$5x10^{-9}$	96.32	1.46	2.23
	10^{-9}	91.95	7.85	0.20
2	1	96.30	1.43	2.27
	3.5	96.32	1.46	2.23
	5	96.32	1.46	2.23
	7	96.32	1.46	2.23
3	*Polynomial 2*	87.93	0.95	11.12
	Polynomial 3	65.78	0.05	34.17
	RBF	96.32	1.46	2.23
4	1%	96.32	1.46	2.23
	5%	62.92	0.47	36.61
	10%	53.99	0.41	45.60
5	240 x 320	92.96	0.79	6.25
	120 x 160	96.32	1.46	2.23
	60 x 80	96.40	3.22	0.37
6	Tanpa Menggunakan *region growing* dan perhitungan luasan *region*	88.62	11.38	0.00
	Menggunakan *region growing* dan perhitungan luasan *region*	96.32	1.46	2.23

VI. KESIMPULAN

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

1. Reduksi *size frame* mempercepat proses deteksi. Dari hasil skenario uji 5, reduksi *size frame* mempengaruhi hasil deteksi namun tidak terlalu besar.
2. Metode deteksi gerak menggunakan *gaussian mixture model* menyaring piksel-piksel pada *frame* dengan *thereshold* yang berbeda setiap piksel dan dapat beradaptasi dengan waktu. Sehingga piksel-piksel yang bergerak dapat disaring dengan baik.
3. Deteksi warna menyaring piksel-piksel yang tidak masuk kedalam *range* warna api menggunakan probabilitas distribusi gaussian menyaring warna piksel api dengan baik. *Threshold* terbaik didapatkan sebesar 5 x 10^{-9}, dapat dilihat pada skenario uji 5.
4. Metode perhitungan luasan *region* dapat menghilangkan *noise* dengan baik. Pada skenario uji 6, dapat disimpulkan bahwa penggunaan perhitungan luasan *region* meningkatkan hasil deteksi.
5. Penggunaan kernel pada klasifikasi mempengaruhi hasil dari verifikasi piksel, dapat dilihat pada skenario uji 3. Kernel terbaik pada pada uji coba 3 adalah RBF.
6. Hasil terbaik pada uji coba adalah menggunakan nilai *threshold* = 5 x 10^{-9} dan nilai C = 7. Menghasilkan nilai *true positif* sebesar 96.32, *false positif* sebesar 1.46 dan *missing rate* sebesar 2.23

UCAPAN TERIMA KASIH

Penulis menyampaikan ucapan terima kasih kepada Ibu Prof. Ir. Handayani Tjandrasa M.Sc., Ph.D dan Ibu Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. sebagai pembimbing penulis dalam mengerjakan penelitian.

DAFTAR PUSTAKA

[1] K.-H. C. J.-Y. N. Byoung Chul Ko, "Fire detection based on vision sensor and support vector machines," 200.

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

[3] E. H. A. PETER J. BURT, "The Laplacian Pyramid as a Compact Image Code," *IEEE* , Vol. %1 dari %2COM-31, pp. 522-540, 1983.

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

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

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

[7] "Images Pyramid Open CV," 2011-2014. [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/imgproc/pyramids/pyramids.html. [Diakses 6 1 2016].

[8] C.-J. L. Chih-Chung Chang, "A Library for Support Vector Machines," Taipei, Taiwan, 2001.