# DETEKSI API PADA VIDEO DENGAN GAUSSIAN MIXTURE MODEL UNTUK DETEKSI GERAKAN DAN SEGMENTASI WARNA API DALAM RUANG WARNA YCBCR

*(Fire Detection in Video with Gaussian Mixture Model for Motion Detection and Segmentation of Fire Color in YCbCr Color Space)*

Ristirianto Adi*, I Gede Pasek Suta Wijaya
Program Studi Teknik Informatika, Fakultas Teknik, Universitas Mataram
Jl. Majapahit 62, Mataram, Lombok NTB, INDONESIA
*Email: ristiriantoadi@gmail.com, gpsutawijaya@unram.ac.id*

***Abstract***

*Fire is a disaster that can endanger lives and cause property loss. The solution to detect fire that is commonly used today is to use a sensor. Fire sensors can be used together with surveillance cameras (CCTV) which are now being installed in many office buildings. This study tries to build a model for detecting fire in video with a digital image processing approach using the Gaussian Mixture Model for motion detection and fire color segmentation in the YCbCr color space. The model is then tested with metrics for accuracy, precision, recall, and processing speed. The dataset used is in the form of videos with small, medium, large fire sizes, and videos that only have fire-like objects. The test results show that the algorithm is able to detect fire when the size of the fire is not too small or the position of the fire is close enough to the camera. For videos with a resolution of 800x600 and a framerate of 30 fps, it can achieve 66.89% accuracy, 73.77% precision, and 66.66% recall. The performance during the day is relatively better than at night. Algorithm processing speed is too slow to be implemented in real-time.*

***Keywords:*** *Computer Vision, Color Segmentation , YCbCr, Gaussian Mixture Model, Fire*

**Penulis Korespondesi*

## 1. PENDAHULUAN

Api merupakan awal mula dari kebakaran, yang merupakan bencana alam yang dapat membahayakan jiwa dan menyebabkan kerugian harta-benda. Di Indonesia sendiri, kebakaran yang sering menjadi sorotan adalah kebakaran hutan, yang dianggap sebagai ancaman potensial bagi pembangunan berkelanjutan karena efeknya secara langsung bagi ekosistem, kontribusinya terhadap peningkatan emisi karbon, dan dampaknya bagi keanekaragaman hayati[1]. Kebakaran juga kerap terjadi di area padat penduduk, yang dipengaruhi oleh korslet listrik dan kebocoran gas di dapur[2].

Solusi untuk mendeteksi api yang umum digunakan saat ini adalah dengan menggunakan sensor, yang bekerja mendeteksi api dengan bergantung kepada karakteristik tertentu seperti asap, suhu, atau radiasi[3]. Sensor api dapat digunakan bersama-sama dengan kamera pengawas (CCTV) yang sekarang banyak dipasang di gedung-gedung perkantoran. Metode pendeteksian secara visual ini memiliki beberapa kelebihan: dapat menjangkau wilayah yang lebih luas, dan dapat langsung menunjukkan lokasi persis di mana api berada lewat gambar yang ditangkap oleh kamera. Selain itu, saat ini kamera pengawas umumnya hanya digunakan untuk menjaga keamanan dari pencurian, bukan pengawasan kebakaran[3].

Untuk memanfaatkan potensi CCTV, telah dilakukan beberapa penelitian mengenai deteksi api menggunakan pemrosesan citra. Beberapa penelitian menggunakan algoritma klasifikasi machine learning dalam melakukan deteksi api, antara lain metode Convolutional Neural Network (CNN), Artificial Neural Network (ANN), dan K-Nearest Neighbour (KNN). Fitur yang digunakan adalah warna dan tekstur. Beberapa penelitian juga mengikutsertakan deteksi gerak (motion detection) untuk meningkatkan akurasi.

Dua metode yang kiranya dapat memberikan hasil yang cukup baik dalam memenuhi syarat akurasi dan kecepatan komputasi adalah deteksi gerakan dan segmentasi warna. Algoritma deteksi gerakan yang banyak digunakan karena kecepatan komputasinya adalah background substraction berbasis Gaussian Mixture Model (GMM). Selain memiliki kecepatan

komputas yang tinggi, background substraction berbasis GMM juga robust terhadap noise sehingga tidak terlalu sensitif terhadap gerakan-gerakan yang kecil[4]. Penelitian sebelumnya telah menunjukkan bahwa ruang warna YCbCr dan HSV dapat memisahkan komponen luminance (kecerahan) dan chrominance (warna) sehingga efektif dalam mengidentifikasi piksel api. Akan tetapi, konversi RGB ke HSV biaya komputasinya lebih tinggi jika dibandingkan konversi RGB ke YCbCr sehingga menggunakan ruang warna YCbCr lebih menguntungkan[5].

Atas dasar tersebut, dalam penelitian ini akan dibangun sebuah model untuk mendeteksi api pada video dengan pendekatan pengolahan citra digital menggunakan GMM untuk deteksi gerakan dan segmentasi warna api pada ruang warna YCbCr.

## 2. TINJAUAN PUSTAKA

Paper [6] menggunakan Background Subtraction untuk menentukan objek yang bergerak (api dianggap tidak pernah statis) kemudian ekstraksi fitur dilakukan dengan Gray-Level Co-Occurrence Matrix (GLCM) untuk masukan ANN. Pengujian pada 15 buah video menunjukkan error rate sebesar 7 persen. Data video yang digunakan mengandung adegan kebakaran dengan menggunakan bahan bakar berlainan jenis dan jarak pengambilan gambar yang berlainan, selain itu terdapat juga adegan video berupa objek yang menyerupai api.

Beberapa penelitian telah mencoba menggunakan CNN. Paper [7] menggunakan CNN tanpa ekstraksi fitur untuk mendeteksi api dan asap, yang memperoleh akurasi klasifikasi sebesar 97,9 persen terhadap testing dataset sebanyak 1427 gambar api dan 1758 gambar asap. Paper [8] melatih CNN untuk mengenali api pada gambar. Peneliti membuat dua model classifier, untuk mengenali api pada gambar secara global, dan mengenali posisi api secara spesifik di dalam gambar. Akurasi mencapai 97 persen untuk training set dan 90 persen untuk testing set.

Referensi [9] dan [10] menggunakan ruang warna HSV dalam melakukan segmentasi warna api. Paper [9] menggunakan Adaptive GMM untuk mendeteksi gerakan dengan statistik warna api pada ruang warna HSV untuk mendeteksi kebakaran. Akurasi sistem terhadap video dengan latar siang hari sebesar 97,6 persen sedangkan pada video dengan latar malam hari sebesar 98,65 persen. Kesalahan terutama disebabkan oleh pantulan cahaya kebakaran pada dinding atau permukaan yang mengkilap. Hasil penelitian menunjukkan bahwa waktu komputasi cukup rendah sehingga bisa dilakukan secara *real-time*. Pada *paper* [10], segmentasi pada ruang warna HSV diterapkan pada robot pemadam api. Eksperimen menunjukkan bahwa algoritma yang digunakan tidak dapat mendeteksi api dikarenakan selalu salah mengidentifikasi lampu sebagai api. *Paper* [10] tidak menggunakan *background subtraction* untuk deteksi gerakan.

Deteksi api juga telah dilakukan dengan ekstraksi tekstur. Paper [11] menggunakan metode Three-frame Differencing untuk mendeteksi gerakan, ekstraksi tekstur dengan gabungan GLCM dan LBP-three orthogonal, dan pengklasifikasian menggunakan KNN. Hasil pengujian keseluruhan sistem diatas dapat diambil rata-rata akurasi untuk video berisi objek api sebesar 76.70% dan untuk video berisi objek non-api sebesar 65.18%. Akurasi terkecil terjadi pada video dengan api yang sudah mulai padam.

## 3. METODE PENELITIAN

### 3.1. Tahapan Penelitian

Penelitian dilakukan dalam beberapa tahap secara sistematis, dimulai dari pengumpulan data hingga pembuatan laporan. Tahapan pertama yaitu pengumpulan dataset yang akan digunakan sebagai data pengujian. Tahapan kedua adalah studi literatur yang dilakukan dengan mengumpulkan referensi-referensi yang berkaitan dengan penelitian yang akan dilakukan. Selanjutnya adalah perancangan model yang terdiri atas segmentasi warna dan GMM. Selanjutnya model akan diuji terhadap dataset yang telah dikumpulkan. Hasil pengujian akan dianalisa dan diambil kesimpulan mengenai performa model. Terakhir adalah proses pendokumentasian dalam bentuk penulisan laporan.

### 3.2. Persiapan *Dataset*

Sebanyak 60 video berdurasi 10-20 detik dikumpulkan dari berbagai sumber di internet. Sumber yang dimaksud seperti Youtube, dan berbagai repository dataset yang dibagikan kepada publik. Video terdiri atas video yang terdapat api, dan video yang tidak terdapat api, masing-masing sebanyak 30 video.

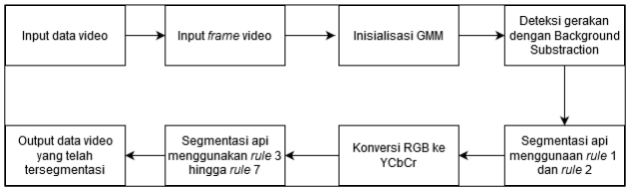
Video yang berasal dari Youtube didapat dengan kata kunci “api”, “kebakaran”, “kebakaran cctv”, “fire”, “fire cctv”, dan “fire surveillance”. Repository dataset yang digunakan adalah [30] dan [31]. Data video yang terdiri atas video yang terdapat api dan tidak terdapat api, masing-masing berjumlah 30, dibagi lagi ke dalam video yang berlatar siang dan malam hari. Lebih lanjut lagi, data video api diperinci menurut skalanya, yaitu api kecil, api sedang, dan api besar.

Video api yang berukuran kecil dikarakterisasikan oleh ukuran api yang relatif kecil dibandingkan ukuran api di video lain dalam dataset, namun api masih jelas terlihat oleh mata manusia. Video api yang berukuran sedang dikarakterisasikan oleh ukuran api yang tidak besar, namun juga tidak kecil. Objek yang terbakar bervariasi mulai dari sesemakan hingga barang elektronik. Api berukuran sedang terlihat jelas oleh mata manusia. Video api yang berukuran besar dikarakterisasikan oleh ukuran api yang relatif besar dibandingkan objek api di video lain dalam dataset. Video api berukuran besar termasuk kebakaran hutan, kebakaran pom bensin, dan ledakan kendaraan.


Gambar 1. Api siang dan malam hari

### 3.3. Perancangan Algoritma

Secara umum, model deteksi api menggunakan GMM dan segmentasi warna api terdiri dari pembacaan frame video, inisialisasi model untuk GMM, deteksi gerakan dengan background subtraction berbasis GMM, konversi RGB ke YCbCr, dan segmentasi api berdasarkan warna seperti yang ditunjukkan pada Gambar 2.




Gambar 2. Model deteksi api

### 3.4. Ruang Warna YCbCr

YCbCr dan RGB merupakan dua ruang warna yang paling banyak digunakan di dalam merepresentasikan citra digital[12]. YCbCr merepresentasikan citra ke dalam satu komponen kecerahan (luminance) dan dua komponen warna (kromatik), sedangkan RGB merepresentasikan citra ke dalam tiga komponen Red, Green, dan Blue[12]. Pada YCbCr, luminance disimpan di dalam komponen Y, dan warna disimpan di dalam komponen Cb dan Cr[12].

Ruang warna RGB sangat sensitif terhadap perbedaan intensitas pencahayaan, sehingga ruang warna lain dibutuhkan untuk meningkatkan efektifitas segmentasi warna[12]. YCbCr adalah salah satu ruang warna yang dapat digunakan sebagai alternatif. Beberapa alasan menggunakan ruang warna YCbCr antara lain: komponen Y independen dari komponen warna sehingga bisa mengurangi dampak perbedaan intensitas pencahayaan[12], dan konversi antara RGB dan YCbCr bersifat linear sehingga biaya komputasinya rendah[13]. Adapun konversi dari RGB ke YCbCr dapat dilakukan dengan Persamaan (1)[14]

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{bmatrix} . \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \quad (1)$$

### 3.5 Klasifikasi Piksel Api Berdasarkan Model Warna

Klasifikasi piksel api dilakukan menggunakan suatu model warna yang berbasis rules, didasarkan pada paper [14]. Dalam paper tersebut, digunakan tujuh buah rules yang menggabungkan informasi dari ruang warna RGB dan YCbCr. Ketujuh rules tersebut akan dijelaskan sebagai berikut:

a. *Rule* 1
*Paper* [14] mendapati bahwa dalam piksel yang merupakan api, intensitas komponen R lebih tinggi dari intensitas G, dan intensitas G lebih tinggi dari intensitas B. *Rule* 1 didefinisikan dalam Persamaan (2).

$$R_1(x,y) = \begin{cases} 1, \ jika \ R(x,y) > \\ G(x,y) > B(x,y) \\ 0 \end{cases} \quad (2)$$

b. *Rule* 2
*Paper* [14] memperoleh nilai *threshold* dalam ruang warna RGB yang mendeskripsikan piksel yang merupakan api. Berdasarkan analisis histogram yang telah dilakukan oleh [14], didapatkan aturan berikut yang didefinisikan dalam Persamaan (3).

$$R_2(x,y) = \begin{cases} 1, \ jika \ R(x,y) > 190 \cap \\ G(x,y) > 100 \cap \\ B(x,y) < 140 \\ 0 \end{cases} \quad (3)$$

c. *Rule* 3 dan *Rule* 4
Untuk *rule* 3 dan selanjutnya, klasifikasi piksel dilakukan di dalam ruang warna YCbCr, oleh karena itu diperlukan konversi dari RGB ke YCbCr menggunakan Persamaan (1). Setelah dilakukan konversi, maka pada piksel dapat diterapkan Persamaan (4) dan Persamaan (5).

$$R_3(x,y) = \begin{cases} 1, \ jika \ Y(x,y) \geq Cb(x,y) \\ 0 \end{cases} \quad (4)$$

$$R_4(x,y) = \begin{cases} 1, \ jika \ Cr(x,y) \geq Cb(x,y) \\ 0 \end{cases} \quad (5)$$

d. *Rule* 5
Area yang terdapat api pada umumnya merupakan area yang memiliki tingkat kecerahan paling tinggi di dalam suatu gambar. Berdasarkan analisis yang telah dilakukan oleh *paper* [14], didapati bahwa pada suatu piksel yang merupakan api, intensitas Y lebih tinggi dari Ymean, intensitas Cb lebih tinggi dari Cbmean, dan intensitas Cr lebih tinggi dari Crmean. Hasil pengamatan ini dirangkum di dalam Persamaan (6).

$$R_5(x,y) = \begin{cases} 1, \ jika\ Y(x,y) \geq Y_{mean} \cap \\ Cb(x,y) \leq Cb_{mean} \cap \\ Cr(x,y) \geq Cr_{mean} \\ 0 \end{cases} \quad (6)$$

e. *Rule* 6
*Paper* [14] mendapati bahwa terdapat perbedaan yang signfikan antara intensitas Cb dan Cr pada piksel api. Piksel api mengandung intensitas Cb yang rendah, sedangkan intensitas Cr tinggi. Pengamatan ini dirangkum dalam Persamaan (7).

$$R_6(x,y) = \begin{cases} 1, \ jika\ |Cb(x,y) - Cr(x,y)| \geq 70 \\ 0 \end{cases} \quad (7)$$

f. *Rule* 7
*Paper* [14] menemukan nilai *threshold* untuk komponen Cb dan Cr dari piksel api melalui analisis histogram. Hasil analisis ini dirangkum dalam Persamaan (8).

$$R_7(x,y) = \begin{cases} 1, \ jika\ Cb(x,y) \leq 120 \cap \\ Cr(x,y) \geq 150 \\ 0 \end{cases} \quad (8)$$

### 3.6 Background Subtraction dengan Gaussian Mixture Model

Background Subtraction merupakan pendekatan yang umum digunakan untuk mendeteksi objek yang bergerak di dalam video dengan kamera yang statis (stationary). Ide dari pendekatan ini adalah mengidentifikasi objek yang bergerak dengan cara mencari perbedaan antara frame saat-ini (current frame) dan frame referensi, atau yang biasa disebut "background model". Background model merupakan representasi dari video yang tidak mengandung objek yang bergerak, dan harus selalu diperbaharui untuk beradaptasi dengan perubahan pada background([15].

Salah satu metode background subtraction adalah dengan menggunakan GMM. Metode ini memodelkan time series yang terdiri dari nilai-nilai piksel ke dalam GMM. Pada waktu t, diketahui nilai-nilai (history) suatu piksel pada posisi {x0,y0}, sebagai Persamaan (9).

$$\{X\_1, \dots X\_t\} = \{I(x\_0, y\_0, i) : 1 \leq i \leq t\} \quad (9)$$

Data time series dapat digunakan untuk memperkirakan probabilitas dari munculnya suatu piksel. Dengan kata lain, probabilitas kemunculan suatu piksel dapat diperkirakan dengan membuat model GMM terhadap data time series.

Probabilitas kemunculan suatu piksel didefinisikan dengan Persamaan (10).

$$\{P(X_t) = \sum_{i=1}^{K} w_{i,t} * \eta(X_t, \mu_{i,t}, \sum_{i,t}) \quad (10)$$

dimana K adalah jumlah distribusi, ωi,t merupakan estimasi bobot dari komponen Gaussian ke-i pada waktu t, ∑i,t adalah matriks kovarians dari komponen Gaussian ke-i pada waktu t, dan η adalah probability density function (pdf), dengan Persamaan (11).

$$\eta(X_t, \mu, \sum) = \frac{1}{(2\pi)^{\frac{n}{2}} |\sum|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t - \mu_t)^T \Sigma^{-1} (X_t - \mu_t)} \quad (11)$$

Setiap piksel baru Xt dihitung jaraknya dari masing-masing komponen GMM. Apabila jaraknya di bawah 2.5 Standar Deviasi (SD) dari suatu komponen, maka nilai dari suatu piksel dianggap match dengan komponen tersebut. Jika tidak ada satupun dari K komponen *match* dengan suatu piksel, maka komponen dengan bobot terendah dibuang dan digantikan dengan komponen yang nilai rata-ratanya merupakan nilai piksel baru, dengan varian tinggi dan bobot awal rendah.

Bobot untuk komponen k pada waktu t, ωk,t, diperbaharui dengan Persamaan (12).

$$w_{k,t} = (1 - \alpha) w_{k,t-1} + \alpha(M_{k,t}) \quad (12)$$

Nilai varian dan rata-rata dari komponen yang tidak match tidak berubah. Untuk komponen yang match, nilai rata-rata dan varian diperbaharui dengan Persamaan (13) dan Persamaan (14).

$$\mu_t = (1 - \rho)\mu_{t-1} + \rho X_t \quad (13)$$

$$\sigma_t^2 = (1 - \rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T (X_t - \mu_t) \quad (14)$$

dimana

$$\rho = \alpha N(X_t | \mu_k, \sigma_k)$$

Komponen GMM kemudian diurutkan berdasarkan nilai ω/σ dari besar ke kecil. B komponen pertama dipilih sebagai *background mode*l berdasarkan Persamaan (15).

$$B = argmin_b \left( \sum_{k=1}^{b} \omega_k > T \right) \quad (15)$$

Untuk menentukan apakah suatu piksel merupakan *background* atau *foreground*, dilihat apakah *piksel* tersebut *match* dengan salah satu B komponen yang dipilih. Apabila iya, maka piksel

dikategorikan sebagai *background*. Apabila tidak, maka piksel dikategorikan sebagai *foreground*[4].

## 4. HASIL DAN PEMBAHASAN

### 4.1. Skenario Pengujian

Tahapan pengujian dimaksudkan untuk menjadi bahan evaluasi akan kinerja model ketika diterapkan terhadap sebuah *dataset*. Adapun yang menjadi tolak ukur kinerja model pada penelitian ini adalah akurasi, presisi, *recall*, dan kecepatan komputasi. Pada tahap ini dicoba untuk menemukan parameter model yang terbaik. Model terdiri atas segmentasi warna dalam ruang RGB dan YCbCr, serta deteksi gerak menggunakan GMM. Dikarenakan *rules* dalam segmentasi warna telah menggunakan *threshold* yang optimal berdasarkan penelitian [14], maka pada penelitian ini sebagian besar pengujian dilakukan terhadap parameter-parameter yang terdapat pada GMM. Parameter-parameter tersebut adalah jumlah distribusi di dalam GMM (K) dan porsi data yang diasumsikan sebagai *background* (T). Selain itu, pengujian juga dilakukan terhadap resolusi dan *framerate* dari video. Resolusi diuji untuk melihat pengaruhnya terhadap kecepatan pemrosesan pada video. Performa dari model diukur dengan metrik akurasi, presisi, dan *recall* dengan persamaan (16), (17), dan (18).

$$Akurasi = \frac{TP+TN}{(\mathrm{TP+FP+FN+TN})} * 100\ \% \quad (16)$$

$$Presisi = \frac{TP}{(\mathrm{TP+FP})} * 100\ \% \quad (17)$$

$$Recall = \frac{TP}{(\mathrm{TP + FN})} * 100\ \% \quad (18)$$

*True Positive* adalah jumlah prediksi positif terhadap *frame* api (prediksi positif yang benar). *True Negative* adalah jumlah prediksi negatif terhadap *frame* bukan api (prediksi negatif yang benar). *False positive* adalah jumlah prediksi positif terhadap *frame* bukan api (prediksi positif yang salah). *False negative* adalah jumlah prediksi negatif terhadap *frame* api (prediksi negatif yang salah). Suatu *frame* diprediksi sebagai *frame* api (positif) apabila lebih dari sepuluh piksel diprediksi sebagai piksel api. Suatu *frame* diprediksi sebagai *frame* bukan api (negatif) apabila kurang dari atau sama dengan sepuluh piksel diprediksi sebagai piksel api[14].

### 4.2. Pengujian RGB

Pengujian dilakukan terhadap data api dan bukan api yang berjumlah enam puluh, kemudian hasil pengujian dibagi ke dalam latar siang dan malam hari. Hasil pengujian dapat dilihat dalam Tabel I.

TABEL I. HASIL PENGUJIAN RGB

| | Akurasi(%) | Presisi(%) | Recall(%) |
|---|---|---|---|
| Siang | 63.80 | 63.86 | 85.30 |
| Malam | 51.91 | 52.78 | 67.47 |
| Rata-rata | 57.86 | 58.32 | 76.38 |

Pengamatan terhadap hasil pengujian ruang warna RGB menunjukkan bahwa dalam mendeteksi api, ruang warna ini banyak menghasilkan *false positive*. Objek-objek yang umumnya menjadi sumber *false positive* antara lain lampu kendaraan, lampu jalan, pantulan sinar matahari, tanah, objek-objek berwarna kuning seperti papan jalan, juga kulit manusia. Beberapa contoh false positive dapat dilihat pada Gambar 3.

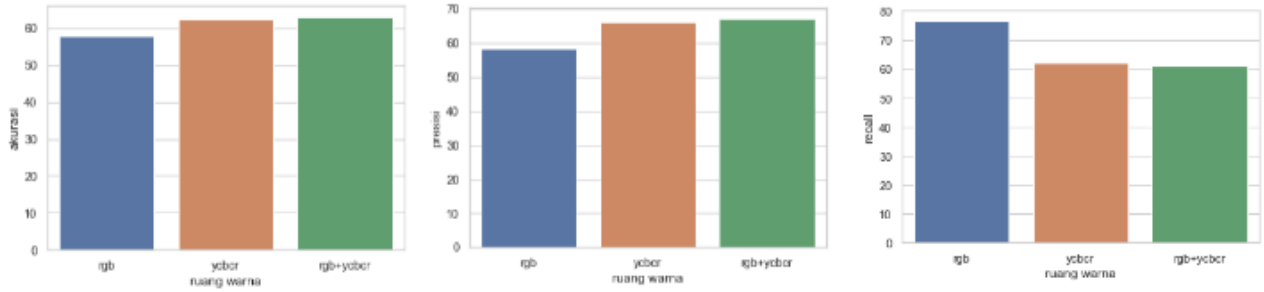

Gambar 3. Contoh *false positive* yang dihasilkan RGB:

*Rules* RGB berhasil mencapai *recall* 86 persen pada siang hari, namun *recall* pada malam hari hanya 67 persen. Rendahnya nilai *recall* pada malam hari ini disebabkan oleh api yang sangat terang, kontras dengan objek-objek di sekitarnya, sehingga hanya tertangkap sebaga objek berwarna putih oleh kamera Pada siang hari, *false negative* umumnya disebabkan oleh skala api yang terlalu kecil. Beberapa contoh *false negative* dapat dilihat pada Gambar 4.


Gambar 4. Contoh *false negative* yang dihasilkan RGB

### 4.3. Pengujian YCbCr

Pengujian dilakukan terhadap 60 data api dan bukan api, kemudian dibagi ke dalam latar siang dan malam hari. Hasil pengujian dapat dilihat dalam Tabel II.

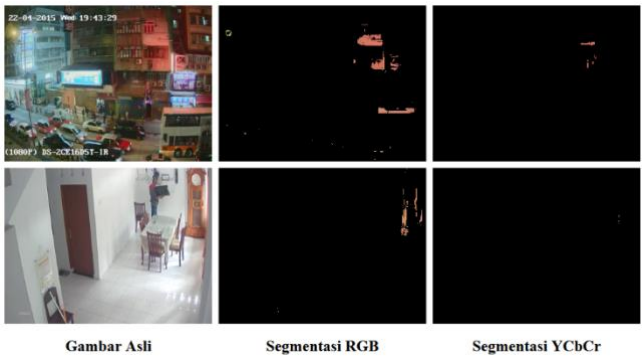



Gambar 5. Perbandingan akurasi, presisi, dan *recall* RGB, YCbCr, dan RGB+YCbCr

TABEL II. HASIL PENGUJIAN YCBCR

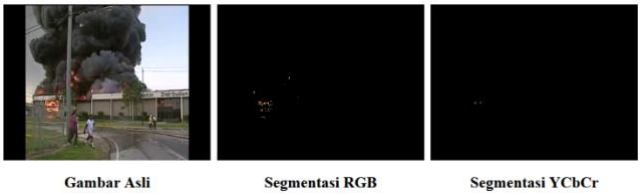
| | Akurasi(%) | Presisi(%) | Recall(%) |
|---|---|---|---|
| Siang | 67.17 | 71.82 | 70.60 |
| Malam | 57.84 | 60.42 | 53.80 |
| Rata-rata | 62.50 | 66.12 | 62.20 |

Hasil pengujian menunjukkan peningkatan nilai presisi, ini berarti bahwa *rules* YCbCr lebih baik dalam menghindari *false positive* dibanding *rules* RGB. Beberapa objek seperti lampu jalan dan pantulan cahaya matahari pada tembok tidak lagi terdeteksi sebagai api. Contoh *frame* yang terdeteksi *false positive* pada ruang warna RGB dan terdeteksi *true negative* pada ruang warna YCbCr dapat dilihat pada Gambar 6.




Gambar 6. Perbedaan RGB dan YCbCr.

Meskipun lebih baik dalam menghindari *false positive*, di sisi lain *rules* YCbCr lebih banyak menghasilkan *false negative*. Beberapa contoh objek api yang disegmentasi dengan benar oleh RGB namun salah disegmentasi oleh YCbCr dapat dilihat pada Gambar 7.




Gambar 7. Perbedaan RGB dan YCbCr dalam mensegmentasi objek api

### 4.4. Pengujian RGB dan YCbCr

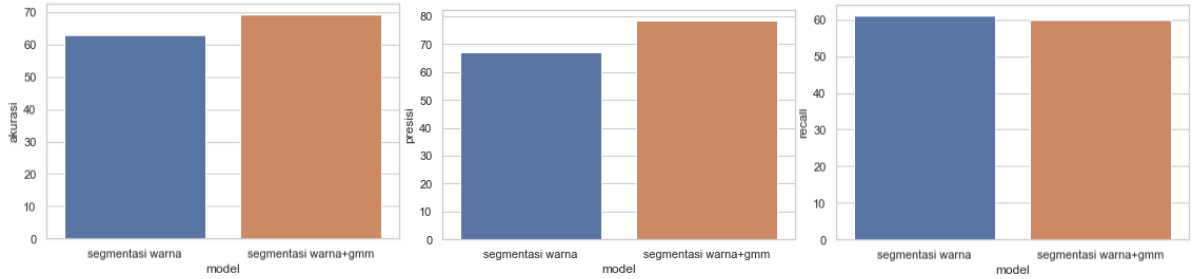
Segmentasi warna dengan Rules RGB dan YCbCr menggabungkan *rule* pertama hingga *rule* ketujuh. Pengujian dilakukan terhadap 60 data api dan bukan api, kemudian dibagi ke dalam latar siang dan malam hari. Hasil pengujian dapat dilihat dalam Tabel III.

TABEL III. HASIL PENGUJIAN RGB DAN YCBCR

| | Akurasi(%) | Presisi(%) | Recall(%) |
|---|---|---|---|
| Siang | 67.68 | 72.83 | 69.84 |
| Malam | 58.16 | 61.25 | 52.25 |
| Rata-rata | 62.92 | 67.04 | 61.05 |

Tabel III menunjukkan nilai akurasi, presisi, dan *recall* dari segmentasi di ruang warna YCbCr dan RGB. Hasil pengujian menunjukkan bahwa secara umum performa model pada siang hari lebih tinggi daripada malam hari. Ini konsisten dengan dua ruang warna sebelumnya.

Dapat dilihat dari Gambar 5, presisi mengalami peningkatan sedangkan *recall* mengalami penurunan. Perbedaan performa terbesar terjadi antara ruang warna rgb dan ycbcr, sedangkan antara ycbcr dan rgb-ycbcr perbedaannya tidak terlalu signifikan.




Gambar 8. Perbandingan akurasi, presisi, dan *recall* tanpa GMM dan dengan GMM

### 4.5. Pengujian GMM

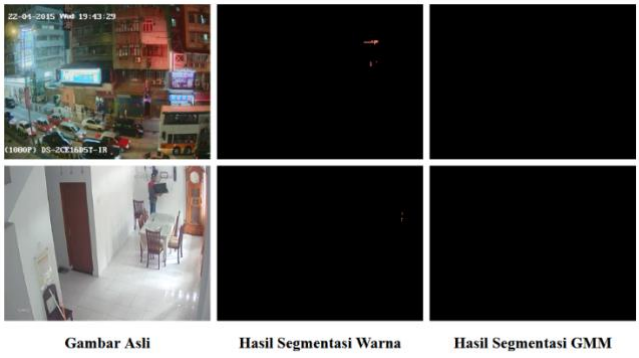
Pengujian dilakukan terhadap 60 data api dan bukan api, kemudian dibagi ke dalam latar siang dan malam hari. Parameter yang digunakan pada GMM adalah *Threshold* (T) = 0.7, *learning rate*(α)=0.05, dan jumlah distribusi (K) = 3.

TABEL IV. HASIL PENGUJIAN GMM

| | Akurasi(%) | Presisi(%) | Recall(%) |
|---|---|---|---|
| Siang | 71.06 | 78.43 | 68.49 |
| Malam | 67.50 | 78.36 | 51.42 |
| Rata-rata | 69.28 | 78.39 | 59.96 |

Hasil pengujian menunjukkan bahwa performa model lebih baik pada siang hari daripada malam hari. Ini masih konsisten dengan pengujian tanpa GMM. Perbandingan nilai akurasi, presisi, dan *recall* dari segmentasi warna tanpa GMM dan dengan GMM dapat dilihat pada Gambar 8.

Penambahan GMM menyebabkan model mampu menghindari *false positive* yang sebelumnya masih terjadi. Objek-objek non-api yang berhasil disegmentasi dengan benar antara lain lampu jalan dan pantulan cahaya matahari. Contoh objek non-api yang salah disegmentasi tanpa menggunakan GMM dan yang disegmentasi dengan benar setelah menggunakan GMM dapat dilihat pada Gambar 9.




Gambar 9. Perbedaan GMM dalam mensegmentasi objek bukan api

Penambahan GMM juga menyebabkan menurunnya *recall*. Beberapa objek api diidentifikasi sebagai bukan api karena dianggap *background* (objek statis) oleh GMM. Contoh dapat dilihat di Gambar 10.


Gambar 10. Perbedaan GMM dalam mensegmentasi objek api

### 4.6. Pengaruh nilai K terhadap Akurasi, Presisi, dan Recall

Pengujian dilakukan terhadap 60 dataset, yang dibagi menjadi siang dan malam hari. Video berukuran 320x240 dengan jumlah fps 10 Model menggunakan *rules* warna yang lengkap (RGB dan YCbCr), serta GMM dengan parameter T=0.7, α = 0.05, dan nilai K berturut-turut untuk tiap pengujian adalah 3, 4, dan 5. Hasil pengujian dapat dilihat pada Tabel V.

TABEL V. HASIL PENGUJIAN K=3, K=4, DAN K=5

| | | Akurasi (%) | Presisi (%) | Recall (%) |
|---|---|---|---|---|
| K=3 | Siang | 71.06 | 78.43 | 68.49 |
| | Malam | 67.50 | 78.36 | 51.42 |
| | Rata-rata | 69.28 | 78.39 | 59.96 |
| K=4 | Siang | 71.95 | 79.86 | 68.49 |
| | Malam | 68.95 | 82.01 | 51.28 |
| | Rata-rata | 70.45 | 80.93 | 59.89 |
| K=5 | Siang | 71.71 | 79.76 | 68.07 |
| | Malam | 69.00 | 82.14 | 51.28 |
| | Rata-rata | 70.36 | 80.95 | 59.67 |

Nilai presisi mengalami peningkatan. Jumlah distribusi menentukan berapa banyak piksel yang dapat dimodelkan oleh GMM di dalam gambar. Semakin banyak jumlah distribusi, maka jumlah objek yang dapat dimodelkan oleh GMM semakin banyak. Hasil pengujian konsisten menunjukkan bahwa presisi yang dapat dicapai model semakin tinggi apabila jumlah distribusi yang digunakan semakin banyak.

Peningkatan nilai presisi diikuti dengan penurunan nilai *recall*. Peningkatan presisi yang perbedaannya paling signifikan terjadi antara K=3 dan K=4, sedangkan antara K=4 dan K=5 hampir tidak terjadi peningkatan presisi. Di sisi lain, *recall* terus mengalami penurunan pada K=3, K=4, dan K=5. K=4 dianggap sebagai jumlah distribusi yang paling baik, di mana nilai tersebut memaksimalkan peningkatan presisi dan meminimalkan penurunan *recall*. Selain itu, akurasi tertinggi juga terjadi pada K=4.

### 4.7. Pengaruh nilai T terhadap Akurasi, Presisi, dan Recall

Pengujian dilakukan terhadap 60 dataset, yang dibagi menjadi siang dan malam hari. Video berukuran 320x240 dengan jumlah fps 10. Model menggunakan *rules* warna lengkap (RGB dan YCbCr), serta GMM dengan parameter K = 4 dan α = 0.05. Nilai T yang digunakan untuk masing-masing pengujian adalah 0.7, 0.8, dan 0.9. Hasil pengujian dapat dilihat pada Tabel VI.

TABEL VI. HASIL PENGUJIAN T=0.7, T=0.8, DAN T=0.9

| | | Akurasi (%) | Presisi (%) | Recall (%) |
|---|---|---|---|---|
| T=0.7 | Siang | 71.71 | 79.76 | 68.07 |
| | Malam | 69.00 | 82.14 | 51.28 |
| | Rata-rata | 70.36 | 80.95 | 59.67 |
| T=0.8 | Siang | 71.74 | 79.80 | 68.07 |
| | Malam | 69.13 | 82.46 | 51.28 |
| | Rata-rata | 70.43 | 81.13 | 59.67 |
| T=0.9 | Siang | 71.74 | 79.83 | 68.03 |
| | Malam | 69.33 | 83.13 | 51.13 |
| | Rata-rata | 70.53 | 81.48 | 59.58 |

. Akurasi nampak tidak mengalami peningkatan atau penurunan yang berarti. Nilai T nampak meningkatkan presisi dan mengurangi *recall,* akan tetapi efeknya tidak sebesar faktor K. Nilai akurasi dan presisi terbaik didapatkan pada T 0.9, namun peningkatannya tidak signifikan. Di sisi lain, nilai *recall* pada T 0.9 mengalami penurunan menjadi 59.58%. Pada T 0.8, nilai *recall* tidak mengalami perubahan. Nilai *recall* merupakan metrik yang cukup penting mengingat *cost* dari *false negative* adalah ancaman korban jiwa. Konsekuensi ini jelas lebih penting dibandingkan waktu dan sumber daya yang terbuang apabila terjadi *false positive*. Oleh sebab itu, selain mengoptimalkan nilai presisi dan akurasi, perlu juga untuk mencapai nilai *recall* setinggi mungkin. Pada pengujian pengaruh T, nilai yang dianggap terbaik adalah 0.8, karena memaksimalkan presisi dan *recall*.

### 4.8. Pengaruh FPS terhadap Akurasi, Presisi, dan Recall

Pengujian dilakukan terhadap 60 dataset, yang dibagi menjadi siang dan malam hari. Video berukuran 320x240 dengan jumlah fps dibagi tiga, yaitu 10, 20, dan 30. Model menggunakan *rules* warna lengkap (RGB dan YCbCr), serta GMM dengan parameter K = 4, α = 0.05, dan T=0.8. Hasil pengujian dapat dilihat pada Tabel VII.

TABEL VII. HASIL PENGUJIAN FPS 10, 20, DAN 30

| | | Akurasi (%) | Presisi (%) | Recall (%) |
|---|---|---|---|---|
| 10 fps | Siang | 71.74 | 79.80 | 68.07 |
| | Malam | 69.13 | 82.46 | 51.28 |
| | Rata-rata | 70.43 | 81.13 | 59.67 |
| 20 fps | Siang | 71.15 | 80.13 | 66.38 |
| | Malam | 69.78 | 85.36 | 50.14 |
| | Rata-rata | 70.47 | 82.75 | 58.26 |
| 30 fps | Siang | 72.48 | 85.71 | 67.62 |
| | Malam | 69.42 | 87.5 | 52.06 |
| | Rata-rata | 70.95 | 86.60 | 59.84 |

Akurasi mengalami peningkatan. Peningkatan dari akurasi yang dipengaruhi oleh fps tidak terlalu signifikan. Presisi mengalami peningkatan dari 81.13% menjadi 82.75%, dan menjadi 86.60%. Meningkatnya nilai presisi menunjukkan bahwa semakin tinggi nilai fps, maka performa GMM juga semakin baik. Recall mengalami penurunan dari fps 10 ke 20, yaitu 59.67% menjadi 58.26%. Namun kemudian *recall* mengalami kenaikan menjadi 59.84%. Secara umum dapat dikatakan kenaikan nilai fps tidak terlalu memengaruhi *recall*, di mana nilai *recall* tetap pada kisaran 59%.

### 4.9. Pengaruh Resolusi terhadap Akurasi, Presisi, dan Recall

Pengujian dilakukan terhadap 60 dataset, yang dibagi menjadi siang dan malam hari. Video memiliki fps 10 dengan ukuran resolusi dibagi tiga, yaitu 320x240, 640x480, dan 800x600. Model menggunakan *rules* warna lengkap (RGB dan YCbCr), serta GMM dengan parameter K = 3, α = 0.05, dan T=0.7. Hasil pengujian dapat dilihat pada Tabel VIII.

TABEL VIII. HASIL PENGUJIAN RESOLUSI 320x240,640x480, 800x600

| | | Akurasi (%) | Presisi (%) | Recall (%) |
|---|---|---|---|---|
| 320x240 | Siang | 71.06 | 78.43 | 68.49 |
| | Malam | 67.50 | 78.36 | 51.42 |
| | Rata- | 69.28 | 78.39 | 59.96 |
| 640x480 | Siang | 70.28 | 75.30 | 71.87 |
| | Malam | 61.55 | 64.95 | 55.88 |
| | Rata- | 65.91 | 70.13 | 63.88 |
| 800x600 | Siang | 70.76 | 75.08 | 73.56 |
| | Malam | 59.54 | 61.88 | 56.90 |
| | Rata- | 65.15 | 68.48 | 65.23 |

Akurasi dan presisi model mengalamii penurunan. Penurunan terbesar terjadi antara resolusi 320x240 dan 640x480. Penurunan akurasi disebabkan karena semakin banyak *false positive* yang terjadi pada dataset bukan api. Penurunan terbesar terjadi antara resolusi 320x240 dan 640x480. Penurunan presisi menunjukkan bahwa semakin besar resolusi, maka semakin banyak *false positive* yang terjadi. Peningkatan *recall* menunjukkan bahwa semakin besar resolusi, maka semakin sedikit *false negative* yang muncul.

### 4.10. Pengaruh Resolusi terhadap Waktu Pemrosesan

Model menggunakan *rules* warna lengkap (RGB dan YCbCr), serta GMM dengan parameter K = 4 dan α = 0.05, dan T=0.8. Pengujian dilakukan terhadap video 10 fps dengan resolusi 320x240, 640x480, dan 800x600. Spesifikasi perangkat keras yang digunakan adalah prosesor Intel® Core™ i5 dan RAM 4GB. Perangkat lunak yang digunakan oleh Windows 10 dan Python 3. Hasil pengujian dapat dilihat pada Tabel IX.

TABEL IX. HASIL PENGUJIAN RESOLUSI TERHADAP WAKTU PEMROSESAN

| Resolusi | Waktu (detik) | FPS |
|---|---|---|
| 320x240 | 1.79 | 0.57 |
| 640x480 | 6.87 | 0.15 |
| 800x600 | 10.48 | 0.1 |

Waktu pemrosesan yang dimaksudkan adalah waktu pemrosesan pada setiap frame video. Hasil pengujian menunjukkan bahwa algoritma bekerja cukup lambat. Ini berarti algoritma belum dapat diterapkan untuk memproses video secara *real time.*

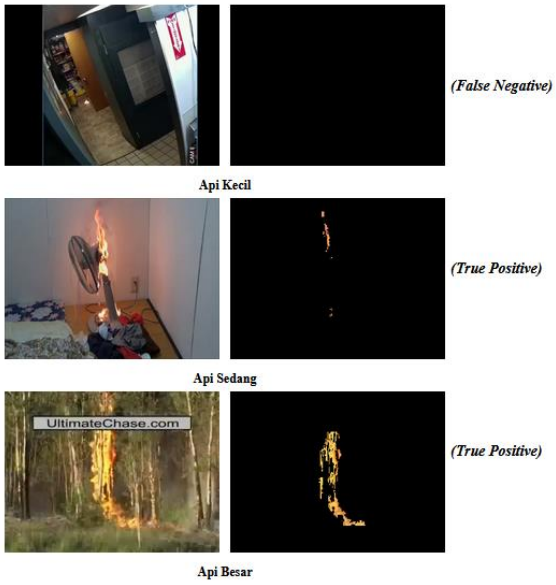
### 4.11. Pengaruh Skala Api terhadap Recall

Pengujian dilakukan terhadap 30 dataset api yang dibagi menjadi api kecil, sedang, dan besar. Model menggunakan *rules* warna lengkap (RGB dan YCbCr) serta GMM dengan parameter K=4 dan α = 0.05, dan T=0.8. Pengujian dilakukan terhadap video 10 fps dengan resolusi 320x240. Hasil pengujian dapat dilihat pada Tabel X.

TABEL X. HASIL PENGUJIAN MODEL TERHADAP SKALA API

| Skala | Recall(%) |
|---|---|
| Api Kecil | 29.24 |
| Api Sedang | 58.29 |
| Api Besar | 83.07 |

Hasil pengujian menunjukkan bahwa semakin besar skala api, maka nilai *recall* semakin tinggi. Ini berarti bahwa algoritma bekerja lebih baik terhadap api yang sudah berukuran besar. Algoritma cenderung kesulitan mensegmentasi api yang berukuran kecil sebab api hampir tidak terlihat dan warnanya cenderung terlihat putih di kamera. Contoh hasil pengujian terhadap skala api dapat dilihat pada Gambar 11.




Gambar 11. Contoh hasil segmentasi berdasarkan skala

### 4.12. Pengujian Model dengan Parameter Terbaik dan Spesifikasi Video Tertinggi

Pengujian ini dilakukan untuk mengetahui performa model apabila menggunakan parameter terbaik yang telah ditentukan melalui hasil pengujian, dan dengan spesifikasi video tertinggi. Parameter model yang dimaksud adalah *rules* warna lengkap (RGB dan YCbCr) serta GMM dengan parameter K=4, α = 0.05, dan T=0.8. Pengujian dilakukan terhadap video 30 fps dengan resolusi 800x600. Dataset terdiri atas 60 video, yang dibagi menjadi siang dan malam hari. Hasil pengujian dapat dilihat pada Tabel XI.

TABEL XI. HASIL PENGUJIAN MODEL DENGAN PARAMETER TERBAIK DAN SPESIFIKASI VIDEO TERTINGGII

| | Akurasi(%) | Presisi(%) | Recall(%) |
|---|---|---|---|
| Siang | 73.39 | 81.14 | 75.28 |
| Malam | 60.39 | 66.39 | 58.04 |
| Rata-rata | 66.89 | 73.77 | 66.66 |

Hasil pengujian menunjukkan bahwa performa model pada malam hari relatif lebih buruk dari siang hari. Faktor penyebab rendahnya performa model pada malam hari, khususnya untuk *recall*, adalah api yang terlalu terang (dikarenakan sekitarnya gelap) sehingga hanya tampak sebagai cahaya putih di kamera, sedangkan model segmentasi warna menganggap api berwarna kuning hingga oranye. Skala api juga sebagian besar berukuran kecil pada malam hari, dan model belum dapat mendeteksi api secara baik apabila skalanya terlalu kecil.

Hasil yang didapatkan dalam penelitian ini tidak sebaik yang didapatkan oleh paper [14]. Pada paper tersebut, deteksi api dilakukan terhadap gambar dalam ruang warna YCbCr. *Rules* pada penelitian didasarkan pada paper tersebut. Hasil pengujian yang dilakukan terhadap 400 gambar memperoleh nilai *recall* 99 persen, akurasi 95%, presisi 87%. Performa model pada paper tersebut jauh lebih tinggi dibandingkan hasil pengujian penelitian ini. Hal ini boleh jadi disebabkan karena paper [14] hanya menggunakan data gambar kebakaran hutan, di mana skal apinya sudah termasuk besar. Selain itu, sebagian besar gambar nampaknya diambil pada siang hari. Seperti yang ditunjukkan oleh penelitian ini, api skala besar pada siang hari dapat disegmentai dengan cukup baik. Objek-objek false positive seperti pantulan cahaya matahari judapat dihindari oleh *rules* YCbCr. Hasil ini konsisten dengan yang didapatkan oleh paper [14].

Paper [9] juga melakukan deteksi berbasis segmentasi warna dan *background subtraction*. Ruang warna yang digunakan adalah HSV. Paper ini mengujicoba algoritmanya terhadap 30 video yang dibagi ke dalam siang dan malam hari. Paper [9] mendapatkan rata-rata recall yang sangat tinggi, yaitu 97,96 persen pada siang hari dan 98,65 persen pada

malam hari. Akan tetapi tidak ada pengujian terhadap video bukan api sehingga nilai *false positive* atau presisi belum terwakili.

Ruang warna HSV nampaknya mampu mensegmentasi api lebih baik dibandingkan ruang warna YCbCr. Akan tetapi paper [9] tidak melakukan pengujian terhadap api skala kecil. Sebagian besar api yang diuji telah berukuran besar. Dari hasil yang ditunjukkan, ruang HSV nampaknya mampu mensegmentasi api pada malam hari, termasuk untuk api yang tampak berwarna putih di kamera. Ini salah satu kekurangan YCbCr yang membuat banyak muncul prediksi *false negative.*

Gaussian Mixture Model yang digunakan paper [9] merupakan implementasi pada library OpenCV. Pengujian GMM pada paper [9] juga dapat menghindari false positive berupa lampu. Hasil tersebut konsisten dengan yang didapatkan oleh penelitian ini.

## 5. KESIMPULAN DAN SARAN

Pendeteksian api menggunakan informasi gerak dan warna pada citra dapat dilakukan, namun nilai *recall*-nya relatif rendah. Rules RGB dan YCbCr telah dapat mensegmentasi api dengan benar pada kondisi-kondisi tertentu, yaitu apabila api berukuran relatif besar atau dekat dengan kamera, dan api tampak berwarna oranye atau kuning. *Rules* RGB cenderung memberikan prediksi positif sehingga nilai *recall*-nya relatif tinggi, namun presisinya rendah. Di sisi lain, *rules* YCbCr cenderung memberikan prediksi negatif sehingga nilai *recall*-nya relatif rendah, namun presisinya tinggi. Segmentasi warna semakin optimal pada video dengan resolusi tinggi.

Metode deteksi gerak berbasis GMM dapat membantu mengurangi prediksi *false positive* terhadap objek-objek statis yang tampak menyerupai api seperti pantulan cahaya matahari, pantulan lampu jalan, dan tembok bata. Metode ini semakin optimal pada video dengan *framerate* tinggi.

Hasil terbaik didapatkan pada video dengan resolusi 800x600 dan *framerate* 30 fps, segmentasi warna dengan *rules* RGB dan YCbCr, serta GMM dengan parameter K=4, T=0.8, dan $\alpha$ = 0.05. Dengan parameter tersebut, rata-rata akurasi 66.89%, presisi 73.77%, dan recall 66.66%. Ini merupakan parameter terbaik berdasarkan hasil penelitian. Hasil yang didapatkan belum optimal dikarenakan model belum mampu mendeteksi api apabila ukurannya terlalu kecil, berada terlalu jauh dari kamera, atau dalam kondisi pencahayaan sedemikian rupa sehingga api tampak berwarna putih alih-alih oranye (model hanya mampu mendeteksi api secara efektif apabila api berukuran cukup besar dan tampak berwarna oranye dari sudut pandang kamera). Performa model secara umum lebih baik pada siang dibanding malam hari. Kecepatan komputasi dari algoritma yang telah diimplementasikan terlalu rendah untuk dapat diterapkan secara *real-time*.

Ruang warna selain RGB dan YCbCr dapat dicoba untuk menemukan *rules* alternatif yang dapat memaksimalkan nilai *recall*. Implementasi GMM di library OpenCV dapat dibandingkan dengan implementasi pada penelitian ini.

## DAFTAR PUSTAKA


[1] L. Tacconi, "Kebakaran hutan di Indonesia: penyebab, biaya dan implikasi kebijakan," *CIFOR Occas. Pap.*, vol. 38, no. 38, 2003, doi: 10.17528/cifor/001200.

[2] S. S. Dewi, D. Satria, E. Yusibani, and D. Sugiyanto, "Prototipe Sistem Informasi Monitoring Kebakaran Bangunan Berbasis Google Maps dan Modul GSM," *J. JTIK (Jurnal Teknol. Inf. dan Komunikasi)*, vol. 1, no. 1, p. 33, 2017, doi: 10.35870/jtik.v1i1.31.

[3] H. A. Muzakkiy, "Deteksi Api Berbasis Sensor Visual Menggunakan Metode Support Vector Machines," Institut Teknologi Sepuluh November, 2016.

[4] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2, pp. 246–252, 1999, doi: 10.1109/cvpr.1999.784637.

[5] K. B. Shaik, P. Ganesan, V. Kalist, B. S. Sathish, and J. M. M. Jenitha, "Comparative Study of Skin Color Detection and Segmentation in HSV and YCbCr Color Space," *Procedia Comput. Sci.*, vol. 57, pp. 41–48, 2015, doi: 10.1016/j.procs.2015.07.362.

[6] Andi, V. Suhartono, and R. A. Pramunendar, "Deteksi Api Menggunakan Background Substraction Dan Artificial Neural Network Untuk Real Time Monitoring," *J. Teknol. Inf.*, vol. 12, no. April, pp. 15–24, 2016.

[7] S. Frizzi, R. Kaabi, M. Bouchouicha, J. M. Ginoux, E. Moreau, and F. Fnaiech, "Convolutional neural network for video fire and smoke detection," *IECON 2016 - 42nd Annu. Conf. IEEE Ind. Electron. Soc.*, pp. 877–882, 2016, doi: 10.1109/IECON.2016.7793196.

[8] Q. Zhang, J. Xu, L. Xu, and H. Guo, "Deep Convolutional Neural Networks for Forest Fire Detection," *Int. Forum Manag. Educ. Inf. Technol. Appl.*, pp. 568–575, 2016, doi: 10.2991/ifmeita-16.2016.105.

[9] A. Prahara, "Deteksi Kebakaran pada Video Berbasis Pengolahan Citra dengan Dukungan GPU," *Semin. Nas. Apl. Teknol. Inf.*, p. 6, 2015.

[10] R. R. Suryadi, I. Wijayanto, A. Rusdinar, F. T. Elektro, and C. Detection, "Perancangan dan implementasi sistem pendeteksi api pada robot pemadam api dengan menggunakan sensor api dan kamera," *e-Proceeding Eng.*, vol. 4, no. 3, pp. 3611–3624, 2017.

[11] G. L. Khamdani, T. A. B. Wirayuda, and F. Sthevanie, "Sistem Deteksi Api Berbasis Visual menggunakan Metode Local Binary Patterns-Three Orthogonal Planes dan Grey-level Co-occurrence Matrix," 2015.

[12] C. Lin, "Face detection in complicated backgrounds and different illumination conditions by using YCbCr color space and neural network," *Pattern Recognit. Lett.*, vol. 28, no. 16, pp. 2190–2200, 2007, doi: 10.1016/j.patrec.2007.07.003.

[13] V. V K and S. C. Sajjan, "Fire Detection using YCbCr Color Model," *IJRSET*, vol. 2, no. 2, pp. 698–701, 2016.

[14] V. Vipin, "Image Processing Based Forest Fire Detection," *Int. J. Emerg. Technol. Adv. Eng.*, vol. 2, no. 2, pp. 87–95, 2012.

[15] M. Piccardi, "Background subtraction techniques: A review," *Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern.*, vol. 4, pp. 3099–3104, 2004, doi: 10.1109/ICSMC.2004.1400815.