IMPLEMENTASI ALGORITMA PENDETEKSI API BERDASAR KOMPOSISI WARNA CITRA DIGITAL PADA *QUADCOPTER* YANG BERGERAK OTOMATIS

IMPLEMENTATION FIRE DETECTION ALGORITHM BASED ON ITS COLOR COMPOSITION IN AN AUTONOMOUS QUADCOPTER

M. Fadhil Abdullah[1], Inung Wijayanto, S.T., M.T.[2], Angga Rusdinar, ST.,MT.,PhD.[3]
[1,2]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[3]Prodi S1 Teknik Elektro, Fakultas Teknik Elektro, Universitas Telkom
[1]mfadhilabdullah@outlook.com

Abstrak

Kontes Robot Terbang Indonesia (KRTI) pada tahun 2015 mengusung tema "Menuju Kemandirian Teknologi Wahana Terbang Tanpa Awak". Maka, pada tahun 2015, dilombakan beberapa kategori lomba baru, salah satu diantaranya ialah wahana *Vertical Take-off Landing* (VTOL) yang dapat mendeteksi dan memadamkan api dengan atau tanpa air secara otomatis. Tapi untuk mendapatkan akurasi yang baik tentu dibutuhkan kamera thermal yang harganya sangat mahal. Kemudian, sensor termal efektif tetapi hanya mampu mendeteksi pada ketinggian dan jarak tertentu.

Pada tugas akhir ini, akan diimplementasikan algoritma pendeteksi api berdasar komposisi warnanya pada sebuah *Quadcopter* yang dapat terbang secara otomatis. menggunakan kamera webcam dan sebuah mikrokomputer tambahan, khusus untuk memproses pendeteksian api yang dapat terintegrasi dengan *flight controller*.

Dalam Tugas Akhir ini berharap agar implementasi pendeteksi pada *Quadcopter* berdasar komposisi warnanya dapat menghasilkan harga yang lebih murah dengan kualitas deteksi yang baik. Selain itu, penulis berharap jarak untuk dapat mendeteksi bisa menjadi lebih jauh dan tinggi sehingga dapat diimplementasikan juga pada sebuah pesawat tanpa awak. Serta penulis berharap, *Quadcopter* ini dapat direalisasikan dalam keadaan riil seperti kasus kebakaran hutan.

Kata Kunci: *Quadcopter*, Api, Deteksi, *Otomatis*

Abstact

Kontes Robot Terbang Indonesia (KRTI) in 2015 propose a theme "Towards Stand Alone Unmanned Aoutonomous Vehichel Technology". So, in 2015, compete several new categories, one of them is a Vertical Take-off Landing (VTOL) Plane that can detect and extinguish fire automatically. But to get best accuracy will require a termal camera which is very expensive. And, termal sensor is effective but it cannot detect in long range and high altitude.

Thus, in this final project will be implemented the fire detection algorithm based on its color composition on a autonomous Quadcopter. This Quadcopter will use a webcam camera and additional microprocessor, especially for processing fire detection that can be integrated with flight controller.

The writer hopes this fire detection Quadcopter can be implemented in next event of KRTI and in the real cases such as, house fire or forest fire. Beside that the writer also hopes this final project will makes good accuracy and low cost hardware in fire detection

Keywords: Quadcopter, Fire detection, Autonomous

1. Pendahuluan

Masalah yang dilirik oleh panitia KRTI tahun 2015 ialah terkait dengan kebakaran hutan yang sering terjadi di Negara Indonesia. Hutan merupakan aset kekayaan penting bagi bangsa Indonesia. Dikutip dari WWF Indonesia, Indonesia memiliki hamparan hutan yang luas, dengan luas sebesar 99,6 juta hektar atau 52,3% luas wilayah

Indonesia [1]. Area hutan yang sangat luas itu menjadi salah satu paru-paru dunia yang sangat penting peranannya bagi kehidupan makhluk hidup di Bumi.

Selama ini, upaya pemadaman api pada kebakaran hutan menggunakan strategi *water bombing*, yakni dengan menggunakan pesawat berawak atau helicopter yang dapat membawa muatan air kemudian menjatuhkannya didaerah terbakar. Pesawat berawak memiliki risiko yang sangat besar bagi pilot, ko-pilot, teknisi, serta penumpangnya.

Untuk itu, dirancang sebuah *Quadcopter* yang dapat mendeteksi dan memadamkan api secara *autonomous*. *Quadcopter* tersebut dapat mengirim notifikasi juga ke *ground control station*. *Quadcopter* ini diharapkan tidak hanya dapat digunakan saat kompetisi robot terbang indonesia, namun dapat menjadi sistema deteksi dan pemadaman api pada kasus kebakaran hutan dikemudian hari.

2. Sistem Perangkat Keras

Pada seksi ini akan dijelaskan sistem perangkat keras yang digunakkan untuk mendeteksi api dengan quacopter secara autonomous. Pada gambar 1 menampilkan desain, material dan juga komponen-komponen yang digunakan oleh quadcopter.

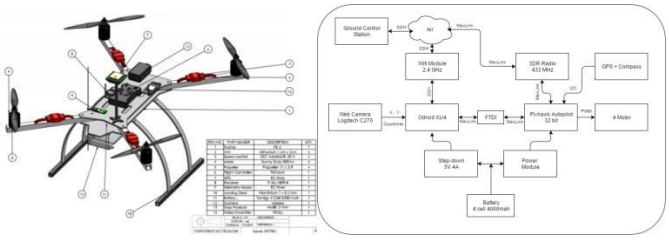

Fig. 1. Desain dan diagram blok *quadcopter*

Untuk menghitung proses sinyal digital pada deteksi api dalam sebuah video saat quadcopter sedang terbang, kita menambahkan sebuah mikrokomputer, yakni Odroid XU4[2]. Mikrokomputer ini tidak hanya akan menghitung algoritma untuk deteeksi api pada video tetapi juga mengirim perintah-perintah ke flight controller, Pixhawk. Perintah-perintah ini yang akan membuat quadcopter bergerak. Komunikasi antar Odroid XU4 dan Pixhawk menggunakan protokol Mavlink. Protokol ini sangat ringat dan memang digunakan untuk mengembangkan Micro Aerial Vehicle.

3. Deteksi Api

Pada seksi ini, kita akan mendiskusikan detail-detail dari algoritma pendeteksi api. Gambar 2 menampilkan flowchart yang diajukan sebagai algoritma untuk mendeteksi api pada quadcopter yang bergerak otomatis.

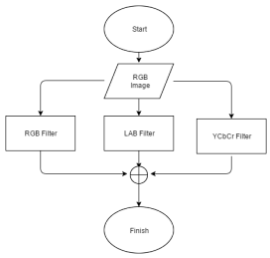

Fig. 2. *Flowchart* algoritma deteksi api

3.1. Model Warna RGB

Agar mendapatkan informasi warna merah dalam sebuah frame video, karena warna dari sebuah api umumnya mendekati kepada warna merah, maka kita menggunakan ruang warna RGB. Lalu, pada Node et al. Disebutkan bahwa mereka telah menemukan korelasi yang linier antara perbandingan warna hijau dan merah dengan distribusi suhu. Berdasarkan fakta itu, suhu dari api dapat diperkirakan dengan menggunakan ruang warna RGB. Dalam nilai-

nilai warna RGB, fakta ini berkorespondensi pada relasi antara kanal warna merah, hijau, dan biru: R>G>B[8]. R harus lebih besar daripada yang kanal yang lain karena kanal warna R mendominasi pada sebuah gambar api. Hal ini juga menyatakan kondisi lain untuk nilai R agar lebih besar dari nilai threshold RTh. Berdasarkan tes yang dilakukan Rth harus lebih besar dari nilai rata-rata R. Berikut aturan-aturan dalam raung warna RGB[3].

( ) { ( ) (1)

( ) { −∑ ( ) (2)

( ) { ( ) ( ) ( ) (3)

( ) { ( ) (( )) (4)

( ) { ( ) (( )) (5)

( ) { ( ) (( )) (6)

3.2 Model Warna LAB

Ruang warna LAB adalah usaha untuk menyeragamkan skala dari diagram chromaticity yang selaras dengan presepsi mata manusia. Ruang warna lab adalah ruang warna dengan L sebagai kecerahan (hitam-putih) serta a dan b sebagai dimensi warna berlawanan berdasarkan koordinat kompresi non-linier. Konversi RGB ke warna LAB [4].

Kita tahu bahwa jarak dari warna api dapat didefinisikan sebagai interval antara warna merah dan kuning. Karena itu, warna api secara general mendekati merah dan memiliki kecerahan yang tinggi. Kita akan menggunakan sifat-sifat ini untuk mendefinisikan pengukuran untuk mendeteksi keberadaan api dalam piksel sebuah gambar. Celik et al. Telah membuat aturan-aturan untuk mendefinisikan wilayah api pada piksel dengan ruang warna lab. Formula tersebut sebagai berikut[5].

( ) { ( ) (7)

( ) { ( ) (8)

( ) { ( ) (9)

( ) { ( ) ( ) (10)

3.3 Model Warna YCbCr

Ruang warna YCbCr dipilih karena kemampuannya untuk memisahkan informasi kecerahan dari warna kromatik. Y adalah luminance (hitam-putih atau warna akromatik)[6]. Sedangkan Cb ialah kromatik biru dan Cr ialah kromatik merah.

Kita tahu bahwa warna api umumnya memiliki warna yang paling cerah diantara wilayah lainnya dalam sebuah gambar. Hal ini mengindikasikan bahwa luminance dan kromatik merah harus memiliki nilai yang lebih besar daripada nilai rata-ratanya. Sementara itu, kromatik biru dari sebuah piksel pada frame harus kurang dari rata-rata kromatik biru. Berikut formula untuk mendeteksi api pada ruang warna YCbCr[7].

( ) { ( ) ( ) (11)

( ) { ( ) ( ) (12)

( ) { ( ) (13)

( ) { ( ) (14)

( ) { ( ) (15)

4. Observasi dan Prediski State

Pada seksi ini, kita akan mendiskusikan ruang state, model-model observasi dan model pergerakkan. Seperti dijelaskan sebelumnya, Odroid akan menghitung posisi dari wilayah api pada frame kamera (berdasarkan frame lokal) dan menerima posisi global dari quadcopter. Selanjutnya, akan mengirim balik state ke flight controller untuk bergerak menuju lokasi api yang terdeteksi dalam frame global. Gambar 2 mengilustrasikan transformasi matriks posisi pada quadcopter.

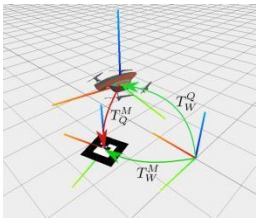

Fig. 3. Tansformasi matriks pada *quadcopter* pendeteksi

Umumnya koordinat x dan y suatu gambar bermula pada pojok kiri atas. Sehingga, koordinat x dan y ini perlu diubah ke sumbu asalnya menjadi pada bagian tengah dari frame tersebut. Berikut persamaan untuk menggeser titik asal pada frame.

(16)

( ) (17)

Kemudian, untuk mendapatkan kecepatan-x (v_x) dan kecepatan-y (v_y) serta syarat-syarat untuk kecepatan-z (v_z). Kecepatan-x (v_x) ialah kecepatan maju dan mundur, yang bila bernilai negatif berarti *quadcopter* tersebut akan mundur dan sebaliknya. Kecepatan-y (v_y) ialah kecepatan dalam pergerakan kanan dan kiri, bila bernilai negatif maka *quadcopter* akan bergerak ke kiri, dan sebaliknya. Sedangkan kecepatan-z (v_z) ialah kecepatan pergerakan *quadcopter* dalam atas-bawah, bila bernilai negatif akan bergerak ke atas, dan sebaliknya.

(18)

(19)

(( )) (20)

(( )) (21)

(–) (22)

√ (23)

(24)

√ (25)

(26)

( ) (27)

( ) (28)

5. Pengujian dan Analisis Performansi Sistem

Pengujian sistem *autonomous quadcopter*, dilakukan pengujian mula-mula dengan simulator. Kemudian, pengujian riilnya menggunakan marker berupa lingkaran-lingkaran hitam seperti *bullseye* untuk panahan. Tujuan mengguanakan marker ini ialah karena karakter yang dideteksinya lebih jelas daripada karakter api sehingga akan memudahkan *quadcopter* untuk bergerak sesuai target.

5.1 Pengujian dan Analisis Model Api

Model api ini diuji pada video sekuensial yang berbeda-beda. Video ini dijadikan video simulasi, Video-video ini memiliki beragam kondisi lingkungan seperti cuaca, siang-malam, dan didalam-diluar ruangan. Vide video tersebut juga tidak hanya mengandung gambar yang memiliki api, tetapi juga yang tidak terdapat api dan objek yang mirip warna api. Tabel dibawah ini berisi hasil pengujian dari video simulasi.

TABLE I. HASIL PENGUJIAN DETEKSI API DENGAN VIDEO SIMULASI

ROC TABLE												
Video Sequences 1	Total Frames	Total Fire Frames	Total Non-fire Frames	True Positive	False Positive	True Negative	False Negative	TPR (Sensitivity / Recall)	FNR	Precision	F	Accuracy
Fire 1	155	155	0	155	0	0	0	100%	0%	100%	100 %	100%
Fire 2	344	344	0	201	132	0	11	92.30%	0.07 %	60.36 %	72.63%	58.43 %
Fire 3	346	346	0	346	0	0	0	100%	0%	100%	100 %	100%
Fire 4	360	360	0	352	0	0	8	97.77%	0.02 %	100%	98.47%	97.77 %
Fire 5	558	558	0	332	226	0	0	100%	0%	59.49 %	74.21%	59.49 %
Fire 6	440	440	0	440	0	0	0	100%	0%	100%	100 %	100%
Total 1	2203	2203	0	1651	358	0	19					
Video Sequences 2	Total Frames	Total Non-fire Frames	Total Fire Frames	True Positive	False Positive	True Negative	False Negative	TPR	FNR	Precision	F	Accuracy
Non-fire 1	467	467	0	467	0	0	0	100%	0%	100%	100 %	100%
Non-fire 2	149	149	0	149	0	0	0	100%	0%	100%	100 %	100%
Non-fire 3	298	298	0	298	0	0	0	100%	0%	100%	100 %	100%
Non-fire 4	391	391	0	391	0	0	0	100%	0%	100%	100 %	100%
Non-fire 5	305	305	0	176	129	0	0	57.70%	0.42 %	100%	72.61%	57.70 %
Non-fire 6	373	373	0	334	39	0	0	89.54%	0.10 %	100%	94.17%	89.54 %
Total 2	1983	1983	0	1815	168	0	0					
							MEAN	94.775%	0.065%	93.32 %	92.67%	80.24 %

Dari tabel diatas, dapat diketahui jika akurasi rata-rata yang dihasilkan oleh hasil simulasi ialah 80.24%. Jika dibandingkan dengan rata-rata hasil pengujian video *realtime* pada tabel bawah, yakni sebesar 72.947%, maka dapat disimpulkan bahwa hasil video simulasi lebih baik dari hasil riil.

TABLE II. HASIL PENGUJIAN DETEKSI API DENGAN VIDEO *REALTIME*

Video Sequences	Total Frames	Total Fire Frames	Total Non-fire Frames	True Positive	False Positive	True Negative	False Negative	Accuracy
Video 1	2981	2312	669	1225	1087	488	184	57.46%
Video 2	3160	3160	0	1902	0	0	1258	58.43%
Video 3	6968	5419	1549	5419	0	728	721	60.06%
Video 4	4037	3125	921	3125	0	125	796	85.29%
Video 5	2351	2351	0	2117	0	0	246	80.60%
Video 6	3946	2234	1719	2234	0	328	1231	90.04%
Video 7	4756	3313	1443	2755	495	359	1102	64.91%
Video 8	2323	1975	346	1975	0	159	187	65.44%
Video 9	3867	3212	655	3081	131	655	0	96.61%
Video 10	2724	2251	473	2251	0	391	88	97.00%

Video 11	7226	0	7226	6319	0	0	905	87.48%
Video 12	11121	0	11121	9996	0	0	1125	89.85%
TOTAL	**52479**	**27040**	**25453**	**41174**	**626**	**2745**	**7659**	
							MEAN	**72.947%**

Pegujian berikutnya ialah membandingkan hasil keluaran dari masing, masing metode. Perbandingan ini dilihat dari jumlah piksel bernilai satu pada dalam dan luar dari *región of interest* yang sebelumnya ditentukan dari hasil logika AND ketiga metode (RGB, LAB, dan YCbCr).

TABLE III. HASIL PENGUJIAN METODE FILTER DETEKSI API

Video Sequences	Mean of ROI Fire	Mean of ROI RGB	Mean of ROI LAB	Mean of ROI YCbCr	Mean of Outer RGB	Mean of Outer LAB	Mean of Outer YCbCr
Video 1	71	105.25	498.5	711	226	34470	5792.25
Video 2	109.9091	415	302.0909	411.0909	74.63636	86312.91	105265
Video 3	308.1818	672.4545	615.1818	1235.364	285.7273	77161.18	54806.55
Video 4	145.5455	330.9091	360.5455	578	53.72727	34388.82	50758.64
Video 5	416.2727	2316.727	1245.273	2977.455	3760	25336.45	99317.55
Video 6	46	135	154.6667	406	7.666667	37739.83	78784.17
Video 7	5002.25	13019.75	14010.75	16483.5	7496	43721.25	14689
Video 8	4802.75	12612.25	10945.25	14163.75	4	30782.75	11861
Video 9	7061.25	12786.25	10504.5	14600.75	44	40986.5	13062
Video 10	6543	7655.75	10370.5	10557.75	8.75	21188.5	15739.5
Video 11	89.221	122.21	156.312	583.131	9.343	331865.5	79406.26
Video 12	78	103.20	132.427	577.330	8.25	31097.25	78088.73
MEAN	**2056.115**	**4189.563**	**4108**	**5273.76**	**998.175**	**66254.25**	**50630.89**

5.2 Pengujian Jarak Objek Deteksi Terhadap Kamera

Pengujian jarak dilakukan dengan mengumpulkan nilai-nilai jarak yang dikeluarkan oleh sistem komputasi kemudian dibandingkan dengan nilai riil-nya. Pengumpulan data ini dilakukan sebanyak 14 kali, masing-masing pengumpulan sebanyak 7 untuk jarak riil 30 cm dan 60 cm. Berikut ialah table dari nilai yang didapatkan dan rata-rata jarak dari dua pengujian jarak.

TABLE IV. PENGUJIAN JARAK

Pengujian 60 cm		Pengujian 30 cm	
Jarak ke-i	Nilai	Jarak ke-i	Nilai
1	55.76306	1	23.43974
2	55.82796	2	23.99747
3	53.91457	3	35.35623
4	58.12825	4	35.42482
5	50.87971	5	34.63085
6	52.54422	6	23.35823
7	56.14698	7	37.11946

Rata-rata	54.73043	Rata-rata	30.46920

Berikut merupakan gambar-gambar aktual dari pengujian jarak deteksi target terhadap kamera. Pengujian ini, sebagaimana pada gambar, dilakukan dengan membandingkan hasil yang keluar pada terminal dengan jarak actual dari penggaris.

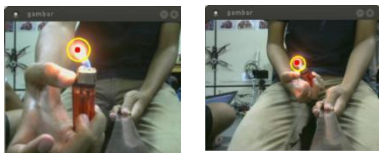

Fig. 4. Pengujian jarak deteksi api

5.3 Pengujian dan Analisis Performa Komunikasi SSH

Pengujian ini dilakukan untuk mengukur kemampuan komunikasi antara wahana terbang, dalam hal ini *quadcopter* dengan *ground station*. Performa komunikasi dengan protokol SSH ini diuji pada sisi *ground station* dengan sumber komunikasi dari *quadcopter* (Odroid XU4) dan *ground station* sebagai sisi penerima.

Seluruh komunikasi ditangkap oleh aplikasi bernama *Wireshark*. Kemudian, difilter untuk mendapatkan protokol yang diinginkan, yakni SSH. Lalu, diubah kedalam bentuk CSV, agar dapat dihitung rata-rata *delay* dengan *Microsoft Excel*. Dari hasil pengukuran tersebut didapat rata-rata *delay* sebesar 0.066901959.

5.4 Pengujian dan Analisis Performa Sistem Odroid dan Pixhawk

Hasil pengujian performansi Odroid dari sistem monitornya ditunjukkan seperti gambar x. Pada gambar tersebut dapat dilihat kinerja keseluruhan CPU yang terdapat pada Odroid, penggunaan memori, serta penggunaan network. Dari delapan CPU yang dimiliki Odroid, rata-rata hanya dua buah yang kinerjanya diatas 50%, dengan catatan satu CPU akan bekerja dengan kinerja 100% secara bergantian dengan CPU yang dengan periode tertentu.

Fig. 5. Sistem monitor dari Odroid XU4 pada saat seluruh sistem pendeteksi dijalankan

5.5 Pengujian dan Analisis Performa Sistem Kontrol Pada *Flight Controller*

Performa kontrol pada *quadcopter* dapat dilihat pada gambar x. Pada grafik tersebut terdapat parameter *roll*, *pitch*, *yaw*, *nav_roll*, *nav_pitch*, dan *nav_yaw*. Nav ialah navigasi, artinya ialah target attitude yang diinginkan oleh sistem. Sedangkan parameter tanpa „nav_" ialah kondisi aktual dari *roll*, *pitch*, dan *yaw*. Untuk melihat kinerja sistem kontrol yang baik pada *quadcopter*, parameter kondisi actual harus memiliki grafik yang berhimpit dengan grafik kondisi yang diinginkan.

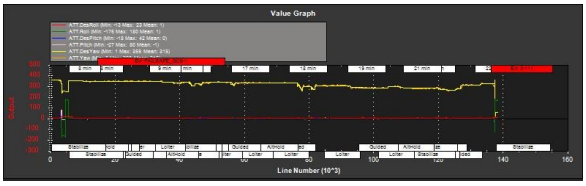

Fig. 6. Grafik sesor IMU dari *quadcopter* yang diuji

5.6 Pengujian dan Analisis Seluruh Sistem Secara Riil

Pengujian secara riil ini dimaksudkan untuk menguji kemampuan dari algoritma kontrol dan deteksi yang telah terintegrasi pada seluruh komponen *quadcopter* untuk mendeteksi objek secara aktual dan *real-time*. Pengujian dilakukan pada lapangan futsal, ditengah gedung E dan F di Universitas Telkom pada hari Sabtu, tanggal 23 April 2016 pada pukul 10.00 WIB.

Fig. 7. Pengujian Kontrol

6. Kesimpulan

Kesimpulan yang bisa didapatkan dari pengujian implementasi algoritma pendeteksi api berdasar komposisi warna citra digital dengan quadcopter yang bergerak otomatis ini dalah sebagai berikut:

1. Algoritma pendeteksi api dengan komposisi warna ini dapat bersifat adaptif, tidak perlu diubah-ubah nilai dari *threshold*-nya seperti deteksi warna pada umumnya.
2. Pada pengujian dari beberapa video yang memiliki variasi kondisi cuaca, ketajaman, kecerahan, dan terdapat warna mirip api, algoritma deteksi api dengan komposisi warna RGB, YCrCb, dan LAB dapat menghasilkan tingkat akurasi hingga 95.089% untuk mendeteksi api.
3. Performa kontrol dari *flight controller* sudah sangat baik, terbukti pada grafik antara nilai aktual dari *attitude quadcopter* terhadap nilai yang diinginan oleh sistem *flight controller* tersebut.
4. Algoritma kontrol untuk mendeteksi objek, dalam hal ini *bullseye* dan api telah berjalan dengan baik dan sesuai dengan yang diharapkan.
5. Performa Odroid sebagai pembuat *state* dan tempat komputasi algoritma sangat baik.
6. Rata-rata *Delay* komunikasi antara *quadcopter* dengan *ground station* dengan SSH sebesar 0,006 detik. Nilai ini sangat baik untuk komunikasi secara nyata
7. Rata-rata error GPS sekitar 3 meter
8. Pengujian secara riil mendapati tingkat akurasi sebesar 72.947%
9. Metode filter dengan RGB sangat efektif bila dibandingkan metode LAB dan YCbCr

Lampiran

[1] Kementrian Kehutanan, "Statistik Kehutanan Indonesia 2011," 2012. ISBN 979-606-073-6

[2] Hardkernel (2015) . Odroid XU4. Available at : http://www.hardkernel.com/main/products/prdt_info.php?g_code=G143452239825 [Accessed April 14, 2016]

[3] Pixhawk (2013). Available at: https://pixhawk.org/modules/pixhawk [Accesed April 14, 2016]

[4] Turgay Celik, "Fast and Efficient Method for Fire Detection Using Image Processing," ETRI Journal Volume 32, Number 6, December 2010

[5] T.Celik, H.Demirel, Huseyin Ozkaramanh. Fire Detection Using Statical Color Model in Video Sequences. PROC. Internat. Conf. on Acoustics, Speech, and Signal Processing, 2007. 213-216

[6] T.Celik, H.Demirel. Fire detection in video sequences using a generic color model. Fire Safety Journal. 2009

[7] T.Celik, Huseyin Ozkaramanh. Fire and Smoke Detection Without Sensors: Image processing Based Approach. 15th European Processing Conference (EUSIPCO 2007). 1794-1798

[8] S. Noda, K. Ueda (1994): Fire detection in tunnels using an image processing method, in Vehicle Navigation & Information Systems Conference Proceedings, pp. 57-62