ISSN: 2320-1363

OPTICAL FLOW ESTIMATION BASED SEGMENTATION AND FLAME DETECTION ON VIDEOS USING ARTIFICIAL NEURAL NETWORK

V.Shanmuga Priya *1, Prof. C. Prakash Narayanan*2

Research Scholar, Dept of Computer Science, PSV College of Engineering & Technology, Elathagiri, TN,India

Assistant professor, Dept. of Computer Science, PSV College of Engineering & Technology, Elathagiri, TN, India

ABSTRACT

This paper presents an automatic system for fire detection in video sequences. There are several previous methods to detect fire, however, all except two use spectroscopy or particle sensors. The two that use visual information suffer from the inability to cope with a moving camera or a moving scene. One of these is not able to work on general data, such as movie sequences. The other is too simplistic and unrestrictive in determining what is considered fire; so that it can be used reliably only in aircraft dry bays. We propose a system that uses color and motion information computed from video sequences to locate fire. This is done by first using an approach that is based upon creating a Gaussian-smoothed color histogram to detect the fire-colored pixels, and then using a temporal variation of pixels to determine which of these pixels are actually fire pixels. Next, some spurious fire pixels are automatically removed using an erode operation, and some missing fire pixels are found using region growing method. Unlike the two previous vision-based methods for fire detection, our method is applicable to more areas because of its insensitivity to camera motion. Two specific applications not possible with previous algorithms are the recognition of fire in the presence of global camera motion or scene motion and the recognition of fire in movies for possible use in an automatic rating system. We show that our method works in a variety of conditions, and that it can automatically determine when it has insufficient information.

INTRODUCTION

Visual fire detection has the potential to be useful in conditions in which conventional methods cannot be used – especially in the recognition of fire in movies. This could be useful in categorizing movies according to the level of violence. A vision-based approach also serves to supplement current methods. Particle sampling, temperature sampling, and air transparency testing are simple methods used most frequently today for fire detection (e.g. Cleary, 1999; Davis, 1999). Unfortunately, these methods require a close proximity to the fire. In addition, these methods are not always reliable, as they do not always detect the combustion itself.

Most detect smoke, which could be produced in other ways.

Existing methods of visual fire detection rely almost exclusively upon spectral analysis using rare and usually costly spectroscopy equipment. This limits fire detection to those individuals who can afford the high prices of the expensive sensors that are necessary to implement these methods. In addition, these approaches are still vulnerable to false alarms caused by objects that are the same color as fire, especially the sun.

Another method used in the system reported in Plumb, 1996 makes use of specialized point-based thermal sensors which change intensity based upon temperature. A black and white camera is used to observe these intensity changes at the various locations. Using the heat-transfer flow model gained from these sensors, a computer solves for the location, size and intensity of the problem using the appropriately named inverse problem solution. Though this would be more precise than our method in finding the center of the blaze, it requires sensors that our method does not. In addition, the exact position of these sensors must be calibrated for this algorithm to be effective.

The method described in this paper employs only color video input, does not require a stationary camera, and is designed to detect fire in nearly any environment, with a minimum camera speed of about 30 frames per second. In addition, it may be implemented more effectively through the use of other imagery, if it is available, besides imagery in the visible spectrum, because the training method can use all available color information.

In our method a color predicate is built using the method presented in sections 2.1 and 2.2. Based upon both the color properties, and the temporal variation of a small subset of images (section 3), a label is assigned to each pixel location indicating if it is a fire pixel (section 4). Based upon some conditions also presented in section 4, we can determine if this test will be reliable. The reason this is an effective combination is explained in section 5. If the test to find fire has been successful, an erode operation (section 6) is performed to remove spurious fire pixels. A region-growing algorithm designed to find fire regions not initially found follows this (section 7). An overall summary of the steps of this fire-finding algorithm is given in section 8. The results presented in section 9 show the effectiveness of this algorithm. Future work and conclusions follow in sections 10 and 11, respectively.

Matching techniques

Block-correlation

Block-based motion estimation algorithms consider a block of pixels in one image and search for the corresponding block in the next image. Some correlation measure is usually used for matching. These methods are also referred to as area-based, correlation-like, or template matching methods **Error! Reference source not found.**. Block-based methods require a huge amount of computations since all possible motions within some search window must be evaluated (Figure 1).

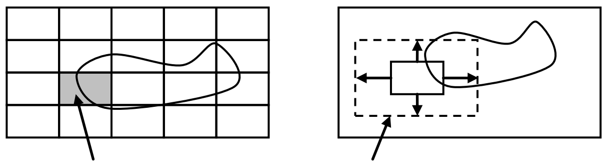


Figure 1. Block-matching.

The block-size will affect the resolution of the estimated motion field. A small block-size provides a detailed motion field but is also more vulnerable to false motion estimates since the correlation measure might provide false matches due to noise. A large block-size gives a more robust but less detailed motion estimate. However, the total number of computations does not depend on the block-size.

It is possible to reduce the amount of computations by using hierarchical search patterns. A rough estimate is determined with larger blocks at low resolution levels. Estimates are then fine-tuned at higher resolution levels with smaller blocks.

Block matching algorithms differ in search strategy and matching criteria. A search may be performed over the whole image or within a small window. The most common matching criteria are the sum of squared difference (SSD)

$$SSD\ u,v\ = \sum_{x,y} \ I_2\ \ x+u, y+v\ \ -I_1\ \ x,y\ \ ^{2}$$

or the sum of absolute difference (SAD)

$$SAD\ u,v\ = \sum_{x,y} \left| I_2\ \ x+u, y+v\ \ -I_1\ \ x,y\ \right|$$

The displacement estimate is the vector (*u*,*v*) that minimizes the SSD or SAD criterion. It is assumed that all pixels belonging to one block have a single displacement vector, which is a special case of the local smoothness constraint (same as for Lucas and Kanade's method). Minimizing the SSD or SAD criteria can be seen as imposing the optical flow constraint on the entire block.

SUPPORT VECTOR MACHINE BASED CLASSIFICATION

The basic idea behind the SVM classification technique is to identify the class of the input test vectors. This is a supervised learning algorithm, where the training vectors are used to train the system to map these training vectors in a space with clear gaps between them using some standard kernel functions and the input test vectors are mapped on to the same space to predict the possible class [24], [25].

Given some training data D, a set of *n* points of the form

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p,\ y_i \in \{-1, 1\}\}_{i=1}^{n}$$

where the y_i is either belonging to the class 1 or class −1, indicating the class to which the point $\mathbf{X_i}$ belongs. Each $\mathbf{X_i}$ is a *p*-dimensional real vector. Here it is needed to find the maximum-margin hyperplane that divides the points having y_i=1 from those having y_i= - 1. So any hyperplane can be written as the set of points $\mathbf{x}$ satisfying

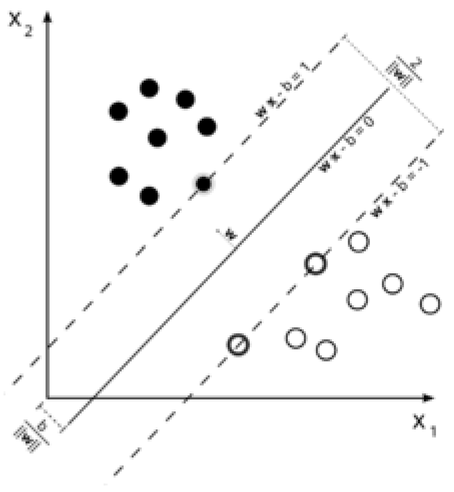


Fig 2. SVM Scenario

Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.w.x – b = 0 where · denotes the dot product and $\mathbf{w}$ is the normal vector to the hyperplane. The parameter b/||w|| determines the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$.

If the training data are linearly separable, then two hyperplanes can be selected in such a way that they separate the data and there are no points between them, and then tried to maximize their distance. The region bounded by them is called "the margin". These hyperplanes can be described by the equations

$w.x - b = 1$

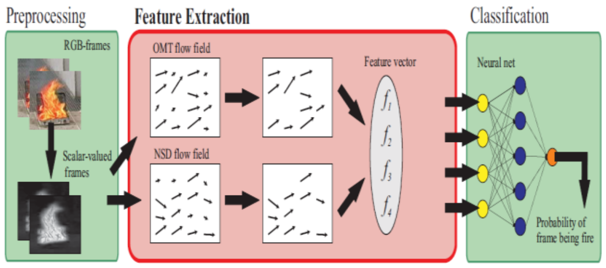
and

$w.x - b = -1$

At the testing phase, the data points Xi are separated using the following constraints

$w.x_i - b \geq 1$ for x_i of the first class or
$w.x_i - b \leq 1$ for x_i of the second class.

EXISTING SYSTEM



The very interesting dynamics of flames have motivated the use of motion estimators to distinguish fire from other types of motion. Two novel optical flow estimators, OMT and NSD, have been presented that overcome insufficiencies of classical optical flow models when applied to fire content. The obtained motion fields provide useful space on which to define motion features. These features reliably detect fire and reject non-fire motion, as demonstrated on a large dataset of real videos. Few false detections are observed in the presence of significant noise, partial occlusions, and rapid angle change. In an experiment using fire simulations, the discriminatory power of the selected features is demonstrated to separate fire motion from rigid motion. The controlled nature of this experiment allows for the quantitative evaluation of parameter changes. Key results are the need for a minimum spatial resolution, robustness to changes in the frame rate, and maximum allowable bounds on the additive noise level.

PROBLEMS IN EXISTING SYSTEM

1. The algorithm is very complex.
2. Application is restricted to flow calculation and consumes more time because of BPN
3. Unstablized video may give wrong and in accurate results in the existing works.

PROPOSED SYSTEM

2.1. Color Detection An often-used technique to identify fire employs models generated through color spectroscopy. We did not use this approach because models may ignore slight irregularities not considered for the type of burning material. Instead, our system is based upon training; using test data from which the fire has been isolated manually to create a color lookup table, usually known as a color predicate. This is accomplished using the algorithm described in Kjedlsen, 1996, which creates a thresholded Gaussian-smoothed color histogram. Note that this manual step is for training only, not for detection itself. It would be possible to create a fixed model of fire color, but our approach allows for increased accuracy if training sequences are available for specific kinds of fires, while if training sequences are not available, it allows for a generic fire look-up table (assuming the user can create a generic, all-purpose fire probability table). Under most circumstances, this method is scene-specific. This will only change the predicate if there are similar colors in both the background and the foreground. We do not consider this case of prime importance because fires are of higher intensity than most backgrounds, and the motion component of this algorithm further eliminates similarly colored backgrounds.

This algorithm for color lookup may be summarized by the following steps:

1) Create pairs of training images – each pair consists of a color image, and a Boolean mask, which specifies the locations at which the target object occurs. For every pixel in each image which represents a color that is being searched for, there should be a "1" in the corresponding location in the Boolean mask, and a "0" for every background location. From our tests, we found ten training images from five of our data sets to be sufficient to construct an effective color predicate. In order for this to be sufficient, it is necessary to ensure a variety of scenes. We used several shots from professional movies and one from a home-made video sequence. Sample masks and images are shown in figure 1.
2) Construct a color histogram as follows: for every pixel location in the image, if the value in the corresponding mask location is "1" then add a Gaussian distribution to the color histogram centered at the color value that corresponds to the color of the individual pixel. Otherwise, if the value in the corresponding mask location is "0," then subtract a smaller Gaussian distribution from the color histogram centered at the color value that corresponds to the color of the individual pixel. For our work, the positive examples used a Gaussian with σ=2, and the negative examples used a Gaussian with σ=1.

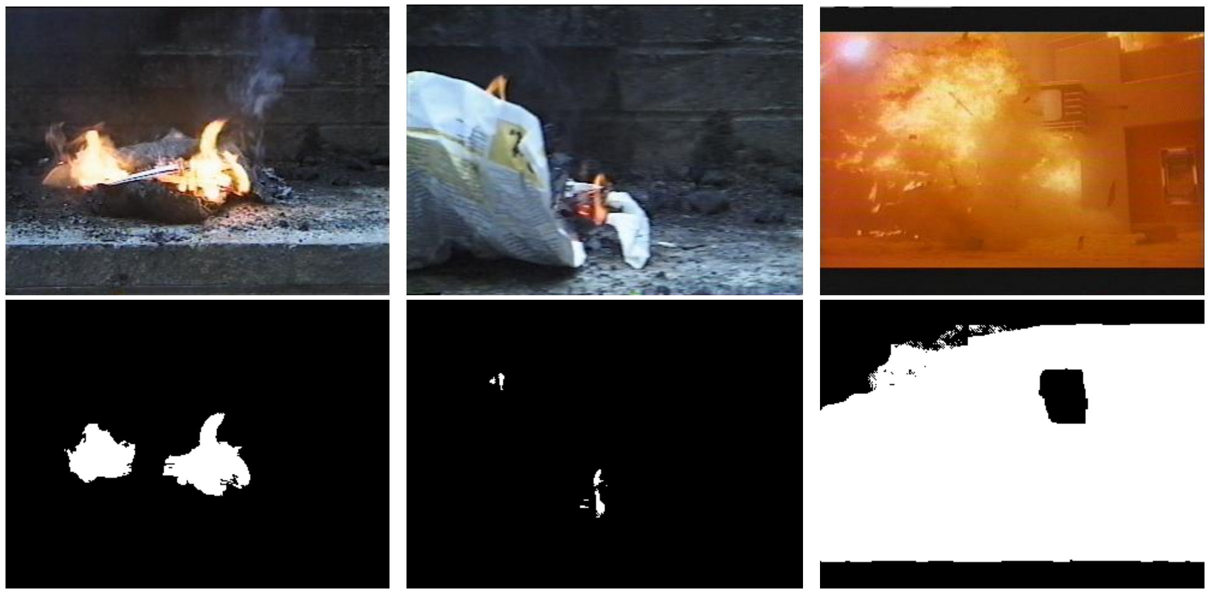


Figure 1: The first row shows the original images, while the second shows manually created fire masks. These were a few of the images used to learn the colors found in fire.

3) Threshold the Gaussian smoothed color histogram to the desired level, resulting in a function which we shall call *Colorlookup,* which, given an (R,G,B) triple, will return a Boolean value, indicating whether or not an input color is in the desired color region.

For our tests, we trained using the images shown above, along with three to eight images sampled from two other image sets. We have found that it is not as important to ensure a particular image, or particular quantity of images. Rather, it was crucial that we include a variety of colors, and use the highest quality recordings. For this reason, color predicates that we produced using our own video sequences, rather than those which include video exclusively from old VHS tapes, performed better than those that do not include our own footage.

2.2. Color in Video

Fire is gaseous, and as a result, in addition to becoming translucent, it may disperse enough to become undetectable, as in figure 2. This necessitates that we average the fire color estimate over small windows of time. A simple way to compute the probability that a pixel is fire-colored over a sequence is by averaging over time the probability that such a pixel is fire.

More precisely:

$$Colorprob(x,y) = \frac{\sum_{i=1}^{n} Colorlookup(P_i(x,y))}{n}$$

$$Color(x,y) = \begin{cases} 1 \text{ if } Colorprob(x,y) > k_1 \\ 0 \text{ if } Colorprob(x,y) \le k_1 \end{cases}$$

where *Colorlookup* is the Boolean color predicate produced by the algorithm in section 2.1, *n* is the number of images in a sequence subset, P_i is the i^{th} frame in a sequence subset. $P_i(x,y)$ is the (R,G,B) triple found at location *(x,y)* in the i^{th} image, and k_1 is an experimentally determined constant. From our experimentation, we have determined that choosing *n* to be between 3 and 7 is sufficient at 30 frames per second. *Colorprob* is a probability (between zero and one) indicating how often fire color occurs in the image subset in each pixel location, while *Color* is a predicate that indicates whether or not fire is present at all. From experimentation, we determined that fire must be detected at least 1/5 of the time by color to indicate the presence of fire. For this reason, we set

k_1 to 0.2.

$$Diffs(x,y) = \frac{\sum_{i=2}^{n} \left| I(C_i(x,y)) - I(C_{i-1}(x,y)) \right|}{n-1}$$

3. Finding Temporal Variation

Color alone is not enough to identify fire. There are many things that share the same color as fire that are not fire, such as a desert sun and red leaves. The key to distinguishing between the fire and the fire-colored objects is the nature of their motion. Between consecutive frames (at 30 frames per second), fire moves significantly (see figure 3). The flames in fire dance around, so any particular pixel will only see fire for a fraction of the time.

In our approach, we employ temporal variation in conjunction with fire color to detect fire pixels. Temporal variation for each pixel, denoted by *Diffs*, is computed by finding the average of pixel-by-pixel absolute intensity difference between consecutive frames, for a set of images. However, this difference may be misleading, because the pixel intensity may also vary due to global motion in addition to fire flicker. Therefore, we also compute the pixel-by-pixel intensity difference for non-fire color pixels, denoted by *nonfireDiffs*, and subtract that quantity from the *Diffs* to remove the effect of global motion:

The highest possible temporal variation occurs in the case of flicker, that is, when a pixel is changing rapidly from one intensity value to another. This generally occurs only in the presence of fire. Motion of rigid bodies, in contrast, produces lower temporal variation. Therefore by first correcting for the temporal variation of non-fire pixels, it is possible to determine if fire-colored pixels actually represent fire. This is done as follows:

1) Deciding which pixels are fire candidates using *Color.*
2) Finding the average change in intensity of all non-fire candidate pixels
3) Subtracting this average value from the value in *Diffs* at each location.

Figure 4 shows the importance of the result of this step. The sun in the figure is fire colored, but because it does not move much throughout the course of the sequence, the ΔI for each pixel in the sequence is small, and thus indicative that no fire has been found. Figure 4 shows the importance of the result of this step. The sun in the figure is fire colored, but because it does not move much throughout the course of the sequence, the ΔI for each pixel in the sequence is small, and thus indicative that no fire has been found.

4. Finding Fire

Our test to find fire is directly dependent upon both color and temporal variation, that is a pixel should be a fire color and it should have significant temporal variation. This is best expressed by a simple conjunction:

$$Fire(x,y)=\begin{cases} 1 \text{ if } Color(x,y)=1 \text{ and } \Delta I(x,y)>k_2 \\ 0 \text{ otherwise} \end{cases}$$

where k_2 is an experimentally determined constant.

This is a binary measure of the temporal variation of the fire-colored pixels. There are several exceptions that indicate that merely computing the predicate *Fire* is not enough. The first of these occurs specifically in sunlight. Sunlight may reflect randomly, causing new light sources to appear and disappear in those reflecting regions. For that reason, there are often some pixels in an image containing the sun that have a temporal variation high enough to be recognized as fire. We put sequences, which contain a high number of fire-colored pixels, but which have a low number of fast moving fire-colored pixels into a "fire unlikely/undetectable" class. Specifically, we count the number of pixels in the image that are "1" in the predicate *Fire* (they have fire color and significant temporal variation) and compare it to total number of fire-colored pixels (i.e. those that are "1" in *Color*). If the number of fire colored pixels is less than some threshold, then we say that there is no fire in the sequence at all. For our tests, this threshold was 10 pixels. If the number of pixels detected as fire is greater than this threshold, but the ratio of pixels that are “1” in *Fire* to fire-colored pixels is low, then the sequence is placed into the “fire unlikely/undetectable” class. For our tests, if no more than one out of every thousand fire-colored pixels is found to be in the predicate *Fire*, then the sequence subset is put into the “fire unlikely/undetectable” class. There is one other case that contains fire that this method is unable to detect: if a sequence is recorded close enough to a fire, the fire may fully saturate the images with light, keeping the camera from observing changes or even colors other than white. Therefore, if contrast is very low and intensity is very high, as in figure 5, sequences are put into a “fire likely/undetectable” class.

Note that with respect to the fire detection task, it is possible that color and motion information could result in the same information so that knowing one is the same as knowing the other. In order to determine the correlation, we took a random sampling of 81,000 points from video data used in our experiments. For each point, we stored

1. The value of *Diffs*
2. The value of *Color*

We then computed ρ, the correlation coefficient:

$$\rho = \frac{\sum (x_i - \mu_x)(y_i - \mu_y)}{(n \cdot \sigma_x \sigma_y)}$$

where x_i is the i^{th} sample taken from *Color*, y_i is the i^{th} sample taken from *Diffs*, n is the size of the sample, μ_x and μ_y are the sample means of *Color* and *Diffs*, and σ_x and σ_y are the sample standard deviations taken from *Color* and *Diffs*, respectively. The correlation we measured by this method was .072, indicating that these two cues are independent.

Improving fire detection by using erosion and region growing:

One of the largest problems in the detection of fire is the reflection of fire upon the objects near the fire. However, barring surfaces with high reflectivity, such as mirrors, reflections tend to be incomplete. An erode operation can eliminate most of the reflection in an image. For our study, the following erode operation worked the best: examine the eight-neighbors of each pixel, remove all pixels from *Fire* that have less than five eight-neighbors, which are fire pixels.

The output from the erosion stage will contain only the most likely fire candidates; to have avoided false positives thus far, our conservative strategy will not have detected all of the fire in a sequence subset. Thus, this is not an accurate measure of the total quantity of fire in the sequence subset. For one thing, some of the fire in a sequence will not appear to be moving because it is right in the center of the fire. Hence, in order to find the rest of the flame, it is necessary to grow regions by examining color alone.

To find all fire pixels in a sequence, we apply the region-growing algorithm. We recursively look at all connected neighbors of each *FIRE* pixel and label them *FIRE* if they are of fire color. Here we relax the threshold for fire predicate, therefore pixels which were not detected as fire will be now be detected as fire if they are neighbors of strong fire pixels. This is essentially a hysteresis process, which is very similar to hysteresis process using low and high threshold in Canny edge detection. This process is repeated until there is no change in pixels labels. During every iteration, the threshold for fire color is increased gradually.

6. Complete Algorithm for Fire Detection

Here we summarize all the steps in our algorithm.

1. Manually select fire from images and create a color predicate using the algorithm in Kjedlsen, 1996 and summarized in section 2.1. Create a function that, given an (R,G,B) triple, returns a boolean. Call this *Colorlookup*.

$$Diffs(x,y)=\frac{\sum_{i=2}^{n}\left|I(C_i(x,y))-I(C_{i-1}(x,y))\right|}{n-1}$$

2. For *n* consecutive images, calculate *DIFFS, Colorprob,* and *Color:*

$$Colorprob(x,y)=\frac{\sum_{i=1}^{n}ColorLookup(P_i(x,y))}{n}$$

where *Colorlookup* is the predicate created in step #1.

3. Determine the net change of portions of the image that are not fire candidates based upon color, and compute from each value for the global difference, and subtract this from the resultant image to remove global motion.

First calculate:

$$Color(x,y)=\begin{cases}1\text{ if } Colorprob(x,y)>k_1\\0\text{ if } Colorprob(x,y)\le k_1\end{cases}$$

$$Nonfirediffs=\frac{\sum_{x,y,Color(x,y)=0}Diffs(x,y)}{\sum_{x,y,Color(x,y)=0}1}$$

and then calculate:

$$\Delta I(x,y)=Diffs(x,y)-Nonfirediffs$$

where the summation is over the *(x,y)* such that *Color(x,y)<k₁*, and k_1 - is an experimentally determined constant.

4. Create a fire Boolean image,

where k_2 is an experimentally determined constant.

5. Classify sequence as “fire likely/undetectable” if the average intensity is above some experimentally determined value, k_3.

6.a. Calculate the total number of 1’s in *Color*. Call this number *Numfire.*

b. Calculate the total number of 1’s in *Fire*. Call this number *Foundfire.*

c. Calculate *Foundfire/Numfire*. If this value is less than some experimentally determined constant, k_4 classify the sequence as “No fire.”

7. Examine the eight-neighbors (the eight adjacent pixels) of each pixel. Remove all pixels from *Fire* that have less than five eight-neighbors that are

CONCLUSION

This paper has presented a robust system for detecting fire in color video sequences. This

$$Fire(x,y) = \begin{cases} 1 \text{ if } color(x,y) = 1 \text{ and } \sigma(x,y) > k_2 \\ 0 \text{ otherwise} \end{cases}$$

algorithm employs information gained through both color and temporal variation to detect fire. We have shown a variety of conditions in which fire can be detected, and a way to determine when it cannot. Through these tests, this method has shown promise for detecting fire in real world situations, and in movies. It is also useful in forensic and fire capture for computer **graphics.**

REFERENCES

[1] M. Raffel, C. Willert, and J. Kompenhans, *Particle Image Velocimetry.* Berlin, Germany: Springer-Verlag, 2001.

[2] R. Adrian and J. Westerweel, *Particle Image Velocimetry. Cambridge,* U.K.: Cambridge Univ. Press, 2010.

[3] R. J. Adrian, “Twenty years of particle image velocimetry,” *Exp Fluids*, vol. 39, no. 2, pp. 159–169, Aug. 2005.

[4] B. Horn and B. Schunck, “Determining optical flow,” *Artif. Intell., vol.* 17, no. 1–3, pp. 185–203, Aug. 1981.

[5] P. Ruhnau, T. Kohlberger, H. Nobach, and C. Schnörr, “Variational optical flow estimation for particle image velocimetry,” *Exp. Fluids*, vol. 38, no. 1, pp. 21–32, Jan. 2005.

[6] P. Ruhnau, A. Stahl, and C. Schnörr, “Variational estimation of experimental fluid flows with physics-based spatio-temporal regularization,” *Meas. Sci. Technol., vol. 18, no. 3, pp. 755–763, Mar. 2007.*

[7] J. Yuan, C. Schnörr, and E. Mémin, “Discrete orthogonal decomposition and variational fluid flow estimation,” *J. Math. Imaging Vis., vol.* 28, no. 1, pp. 67–80, May 2007.

[8] T. Corpetti, D. Heitz, G. Arroyo, E. Mémin, and A. Santa-Cruz, “Fluid experimental flow estimation based on an optical-flow scheme,” *Exp. Fluids, vol. 40, no. 1, pp. 80–97, Jan. 2006.*

[9]Cleary, T., Grosshandler, W., 1999. Survey of fire detection technologies and system evaluation/certification methodologies and their suitability for aircraft cargo compartments, .US. Dept. of Commerce, Technology Administration, National Institute of Standards and Technology.

[10] Davis, W., Notarianni, K., 1999. Nasa Fire Detection Study. US Dept. of Commerce, Technology Administration, National Institute of Standards and Technology, <http://www.fire.nist.gov/bfrlpubs/fire96/PDF/f96001.pdf>.

[11]Healey, G., Slater, D., Lin, T., Drda, B., Goedeke, A.D., 1993. A system for Real-Time Fire Detection, IEEE Conf Computer Vision and Pattern Recognition, p 605-606

[12] Foo., S. Y., 1995 A rule-based machine vision system for fire detection in aircraft dry bays and engine compartments, Knowledge-Based Systems, vol 9 531-41.

[13] Plumb, O.A., Richards, R.F. 1996. Development of an Economical Video Based Fire Detection and Location System. US Dept. of Commerce, Technology Administration, National Institute of Standards and Technology <http://www.fire.nist.gov/bfrlpubs/fire96/PDF/f96005.pdf>

[14] Kjedlsen R, Kender, J., 1996. Finding Skin in Color Images, Face and Gesture Recognition, p 312-317.

[15] G. Hermosillo, C. Chefd'hotel, and O. Faugeras, "Variational methods for multimodal image matching," *Int. J. Comput. Vis., vol. 50, no. 3,* pp. 329–343, Dec. 2002.