Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Smoke and Fire Detection using 2D and 3D Multi-Sensor Fusion

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Informatik

eingereicht von

Herbert Moldan
Matrikelnummer 0728098

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Privatdoz. Dipl.-Ing. Dr.techn. Martin Kampel
Mitwirkung: Mag. Dipl.-Ing. Rainer Planinc

Wien, 11.08.2014

(Unterschrift Verfasser)

(Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Smoke and Fire Detection using 2D and 3D Multi-Sensor Fusion

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Informatics

by

Herbert Moldan
Registration Number 0728098

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Privatdoz. Dipl.-Ing. Dr.techn. Martin Kampel
Assistance: Mag. Dipl.-Ing. Rainer Planinc

Vienna, 11.08.2014

(Signature of Author)

(Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Herbert Moldan
Am Kalten Gang 15, 2483 Ebreichsdorf

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)	(Unterschrift Verfasser)

Acknowledgements

First and foremost, I would like to thank Tina for enduring all my moods over the last couple years

Also, I would like to take this opportunity to thank Mike for all his unending support on this way we have walked together

In addition, I would also like to thank my family, especially my Mum and Dad, for their perpetual endorsement

And to all of my friends, in particular Alex and Flo, without you I would have graduated two years earlier

Abstract

The outbreak of an uncontrolled fire is one of the most endangerments known to human beings. Recent approaches for contactless fire- and smoke detection rely widely on evaluating the content of video-streams, provided by for example video-cameras monitoring a scene. Different features for fire and smoke are calculated using information extracted from the video-stream. The final decision - if smoke or fire is represented in a video frame or not - is then made by merging the result of each single feature to an overall decision. In this thesis a novel approach to detect fire and a novel approach to detect smoke by using two different sensors is proposed. The first sensor is a RGB camera, providing a colored video-stream, while the second sensor is a 3D-sensor which calculates the distance between the sensor itself an objects located in a scene. Hardware, like the Microsoft Kinect or the ASUS Xtion Pro, already combine both sensors. Results show that fire itself leads to significant changes in the depth-image provided by the 3D sensor, while the presence of smoke leads to changes in the color-stream only. Therefore, a method is presented, where the information of the RGB color-sensor is continuously analyzed if specific features like the color of moving pixels and their statistical analysis indicate a presence of fire in a scene. By examining the result-values extracted by the different features, the area where a possible fire-incident is located is calculated. In addition, the information provided by the 3D depth-sensor is used to identify possible fire regions. If a potential fire-region is recognized by the color-sensor, the implemented method for the depth-sensor verifies the potential fire-area (detected by the RGB sensor only) regarding its plausibility. This combined approach, by fusing RGB and depth information, leads to a decrease of the *false-positive rate* (fire is falsely detected in a frame) of 97.46%, compared to the detection rates by using the information of the color-sensor only.

Due to the fact that smoke does not lead to a significant change in the stream provided by the depth-sensor, for smoke-detection a method using the RGB sensor only is implemented. The results show that fire-detection using e.g. a ASUS Xtion Pro enhances the detection of real-fire events significantly, due to the fact that *false-positives* are effectively reduced. Even a recorded fire replayed on a monitor is correctly dismissed by this method. However, due to the fact that only the color-stream is used for smoke detection (without the additional information provided by the depth-sensor), and the video-stream captured by the ASUS Xtion Pro is rather in low-quality - compared to e.g. videos recorded by smartphones - for the detection of smoke other hardware (like a high-quality webcam) is more useful.

Kurzfassung

Der unkontrollierte Ausbruch eines Feuers stellt eine große Gefahr für jeden Menschen dar. In jüngster Zeit wurden viele Ansätze entwickelt um Feuer und Rauch kontaktlos zu erkennen - zumeist realisiert durch statische Kameras welche einen entsprechenden Bereich überwachen. Hierzu werden typische Merkmale von Feuer aus den Video-Daten extrahiert, um danach - anhand der erhalten Werte - zu bestimmen ob ein Feuer erkannt worden ist oder nicht. In dieser Diplomarbeit wird ein neuartiger Ansatz zur Feuer- und Raucherkennung vorgestellt. Hierzu werden die Daten von zwei Sensortypen kombiniert verwendet: eine Kamera welche ein Farbvideo erfasst und ein 3D Tiefensensor welcher die Entfernungen von Objekten zum Sensor bestimmt. Beide Sensoren sind bereits in Geräten wie der Microsoft Kinect oder einer ASUS Xtion Pro vorhanden. Tests zeigen, dass Feuer zu einer sichtbaren Veränderung im Tiefenbild führt, während Rauch keine signifikante Änderung hervorruft und deshalb nur im Video-Stream erkennbar bleibt. Im Zuge der Diplomarbeit wird eine Methode vorgestellt, wo die Informationen aus dem Videostream verwendet werden um Feuer an Hand unterschiedlicher Merkmale in Videos zu detektieren und zu lokalisieren. Zusätzlich werden die Daten des Tiefensensors verwendet um potentielle Feuer-Bereiche zu errechnen. Wird nun ein mögliches Feuer durch die RGB Kamera entdeckt, wird die Information des Tiefensensors dazu verwendet um den Bereich auf seine Plausibilität hin zu überprüfen. Dieser kombinierte Ansatz vermindert die erkannten *Falsch-Positiven* (Feuer wurde erkannt obwohl kein wirkliches Feuer in der Szene vorhanden ist) um $97,46\%$, verglichen mit den Entscheidungen die alleine durch die Auswertung der Informationen des Farbsensors getroffen wurden. Die Analyse von Testvideos und die anschließende Auswertung der Ergebnisse zeigt, dass das kombinierte Verwenden von beiden Sensoren die Erkennungsrate von echtem Feuer deutlich verbessert und die *Falsch-Positiven* effektiv reduziert. Selbst Aufnahmen von Feuer, welche über einen Bildschirm abgespielt wurden, wurden als *Korrekt-Negativ* erkannt. Eine z.B. ASUS Xtion Pro eignet sich jedoch nicht für die Raucherkennung, da der Farbsensor Videos nur in relativ schlechter Qualität aufzeichnen kann.

Contents

1 Introduction 1
1.1 Motivation 1
1.2 Problem Statement 2
1.3 Aim of the Work 2
1.4 Methodological Approach and Own Contribution 3
1.5 Structure of the Work 3

2 State of the Art 5
2.1 Literature Studies 5
  2.1.1 Moving Pixel Detection 5
  2.1.2 Features for Fire Detection in Videos 9
  2.1.3 Fire Feature Extraction using Time-of-Flight Cameras 22
  2.1.4 Features for Smoke Detection in Videos 26
  2.1.5 Decision Fusion 32
2.2 Related Work 34
  2.2.1 Real-Time Fire and Flame Detection 34
  2.2.2 Fire Detection in Newscast Videos 35
  2.2.3 Fire Detection through Time-of-Flight Imaging 36
  2.2.4 Early Smoke Detection Algorithm 39
  2.2.5 Vision Based Smoke Detection Algorithm 42
  2.2.6 Forest Fire Smoke Detection in Video 42
  2.2.7 Summary 44

3 Methodology 47
3.1 Hardware Requirements 47
  3.1.1 Image Registration 48
  3.1.2 Time Difference between RGB and Depth Frames 50
  3.1.3 Fire Color Range 53
  3.1.4 Smoke Color Range 53
  3.1.5 Depth Camera and Fire 56
  3.1.6 Flickering Depth Image 56
  3.1.7 Depth Camera and Smoke 58
  3.1.8 Depth Camera and Sunlight 58

3.1.9 Framework 59
3.2 Methods 60
3.2.1 Otsus Method 60
3.2.2 Morphological Operations 61
3.2.3 Integral Images 61
3.3 Analysis of Movement Detection 61
3.3.1 Frame Differencing 64
3.3.2 Background Subtraction 64
3.4 Analysis of Fire-Features 66
3.4.1 Color Range 66
3.4.2 Color Histogram 67
3.4.3 Comparison of Color Range and Color Histogram 67
3.4.4 Surface Coarseness 67
3.4.5 Skewness 69
3.4.6 Comparison between Surface Coarseness and Skewness 70
3.4.7 Randomness of Area Size 71
3.4.8 Fire Growth Rate 72
3.5 Implemented Method for Fire Detection 74
3.5.1 RGB Camera 75
3.5.2 Depth Camera 79
3.5.3 Decision made by Sensor Fusion 84
3.5.4 Final Decision 85
3.6 Implemented Method for Smoke Detection 86
3.6.1 Motion Estimation 86
3.6.2 Color Restriction 87
3.6.3 Movement and Color 87
3.6.4 Average Grayscale Value 88
3.6.5 Regional Grayscale Difference 89
3.6.6 Decreased Amount of Edges 90
3.6.7 Decision Fusion per Frame 90
3.6.8 Final Decision 91

4 Evaluation **93**
4.1 Experimental Results 93
4.1.1 Precision, Recall and F1-Score 93
4.1.2 Detecting Fire 94
4.1.3 Detecting Smoke 101
4.2 Comparison with Related Work 104
4.2.1 Fire Detection 104
4.2.2 Smoke Detection 104

5 Summary and Future Work **107**
5.1 Summary 107
5.1.1 Fire-Detection 107

5.1.2 Smoke-Detection 108
5.1.3 Conclusion 108
5.2 Discussion of Open Issues 108

Bibliography **111**

CHAPTER 1

Introduction

1.1 Motivation

An uncontrolled fire is one of the greatest endangerments known to every human being [10]. Fire is so dangerous that - even in a place completely surrounded by water (like on a ship) - it is generally the most dangerous thread to the life and safety of the people that reside in that area[1]. Even in a case where no one was injured, the damage to property can be big enough for someone to lose his livelihood [21]. Most houses are not equipped with smoke detectors, because they e.g. do not fit into the design of the room or the owner thinks the chance of a fire incident is fairly small. Moreover, back fitting an existing building could be very cost-expensive or complex when you try to install wired detectors [25] [44]. Nevertheless, if a house is equipped with fire and smoke detectors, the alarm is limited to the range where the alarm-sound can be heard. If no-one is within the range of audibility, a fire is not detected until flames or smoke is noticed by e.g. a neighbor. If the smoke-alarm system is linked to e.g. the nearest fire department, a false alarm could also lead to a penalty payment. So it would be a great advantage, to receive a notification that a fire detector was triggered, to get an efficient *human-in-the-loop* validation [12] to make it possible to monitor the scene before calling the fire brigade. Also, in an *Ambient Assisted Living* environment, where elderly people is given the possibility to stay at their own house instead of going into a retirement home, it would be a great advantage if a caregiver could supervise a fire-alarm event to distinguish between a cooking attempt that went wrong and a real fire event [36]. The sooner a fire is detected the better the chances of saving life and property [17]. Fire detection by *Computer Vision* allows the contactless and fast detection of fire and smoke events [52].

---

[1] Ten Years of Cruise Ship Fires, `http://www.cruiselawnews.com/2010/03/articles/fires-1/ten-years-of-cruise-ship-fires-has-the-cruise-industry-learned-anything/`, Accessed: 26.05.2014

1.2 Problem Statement

A fire detector is implemented as either a smoke sensor, a gas sensor or a temperature sensor [49]. The detection of smoke is the most common approach, because the sensor itself is relatively low-cost, but provides a good detection rate. A disadvantage is that the number of triggered false alarms is high. The temperature sensor is more reliable than the smoke detector, but with a slower response time [49]. Another problem is that the fire detectors need to be placed on areas, where they are able to measure the temperature/gas concentration and the visibility changes in their environment as soon as possible [50]. Due to the fact that warm air, smoke or gas tends to ascend, fire detectors are placed on the ceiling of rooms [31]. Although for safety concerns this would represent the best location, regarding aesthetics it is not. As a matter of fact, these detectors can only detect an fire-incident, if the e.g. smoke or heat directly reaches the sensor itself [44]. To equip a house with fire-detectors that just trigger an (local) acoustic alarm, there is also a power supply needed, which means either a visible cable to the detector, or you need to force open your walls and/or ceiling to provide flush-mounted power supply. One solution could be the use of a battery-operated detector, but they have the great disadvantage that the alarm (usually acoustic) can only be heard if someone is near the room where the alarm went off. A communication from the sensors to an centralized system is useful e.g. as demonstrated in [25]. These Systems are capable of processing the information where a fire took place to calculate alternative escape-routes on safe paths only. But also a communication from the centralized system (that receives the alarm) to the detector itself is useful, e.g. to reset the alarm to see if the fire event is detected again to decrease the likelihood of a false detection. Another problem of battery-operated systems is that the lifetime of batteries are limited and therefore have to be regularly replaced. In [4], a wireless gas and fire detection system is introduced that informs its host station about incidents via wireless ZigBee communication. However, a Li-Ion battery is defined as a backup power-supply - its main-power-supply is realized by a solar panel with super-capacitors, which is ill-suited for indoor use. [44] describes a battery-operated, wireless and long-lasting sensor for indoor use like museums, ancient buildings or temples. But this method also measures the gas concentration (CO), the temperature and the smoke intensity and has therefore be placed on strategic positions that are usually good visible for the owners or visitors of such buildings. Another advantage of early fire detection through video analysis is that its relatively low cost when e.g. using CCTV cameras that are already installed for surveillance purposes [27]. In addition, large, open areas - such as tunnels or forests - can be monitored by vision-based fire detection, as shown in in [39] where an automated forest-fire detection is introduced.

1.3 Aim of the Work

In this work, the ability for the combination of a 2D sensor (RGB camera) with a 3D sensor (depth sensor calculating the distance to objects in a scene via the *Structured Light Method*) to detect fire and smoke incidents indoors is proposed and evaluated. Since fire detection using RGB cameras is already well-established (e.g. [12], [27], [39],...), the focus of this work is the detection of fire and smoke using a 3D sensor and the fusion of 2D and 3D information. In addition, the effect that such events have on the sensor used and possibilities how this information can be used to detect smoke and fire incidents is shown. The outcome is a reliable, contactless indoor smoke and fire detector that enables a detection in real-time by using a low-cost sensor.

1.4 Methodological Approach and Own Contribution

First, existing methods and the features used are described, showing the common ground that the methods have as well as how different the approaches can be when extracting the same features. In addition, the behavior of selected features is analyzed by using example videos showing *real fire* and *fire-like* events (like someone moving through the scene while wearing an orange shirt). Also, the effect that recorded fire has on e.g different motion-estimation techniques or on calculated statistical data like the variance of the pixel-color is analyzed further. Due to the fact that smoke and fire detection using a 3D sensor like e.g. an ASUS Xtion Pro is a novel approach, the impact of smoke and fire incidents onto the outcome of the 3D sensor is analyzed to gain information if or how the detection of the depth of a scene is influenced by such events. Furthermore, characteristics of the sensor used - like the image synchronization between the sensors - is analyzed after extracting statistical data such as the difference between the timestamps of frames recorded by both sensors or the behavior of each sensor when assigning a number to a frame. For specific features (like the fire color) the corresponding values when using this 3D sensor are extracted. Therefore, a software was developed to save all frames provided by the RGB and the depth-sensor to analyze them regarding the recorded fire-color, for example. Moreover, the behavior of the sensor used is analyzed further when recording a *non-changing* (and therefore static) scene, to receive information of the reliability and consistency of the estimated depth. In the end, this gained information is used to develop a reliable software to detect smoke and fire incidents in real time using the ASUS Xtion Pro. In addition, the performance of the software is analyzed using different example events like *real fire* events or events with a *fire-like* appearance, such as a recorded fire replayed on a monitor. It is further examined how the several detection rates are influenced by using the proposed multi-sensor fusion, compared to the rates achieved by using the information provided by the RGB-sensor only.

1.5 Structure of the Work

Due to the fact that smoke does not lead to a significant change in the 3D depth sensor, this work mainly focuses on the detection of fire, using a multi-sensor fusion of a 2D RGB and a 3D depth sensor. Section 2 of this work is called 'State of the Art' and provides an overview of existing approaches and the features used. Additionally, selected methods are described in detail with a subsequent highlighting of commonalities and the contrasting of differences when e.g. extracting a specific feature. In Section 3, called 'Methodology', the quality of the sensor and the characteristic of selected features are analyzed. In addition, this section introduces the smoke and fire detecting method implemented for this work and its different features used. Section 4, namely 'Evaluation', explains the performance and the behavior of smoke and fire detection using the ASUS Xtion Pro as well as possible problems. In addition, the proposed method is compared to other methods used. Finally, the performance achieved by both methods and the impact on the detection rates using the *Multi-Sensor Fusion* is described. The last Section, called 'Summary and Future Work' explains issues not covered by this software/work and reflects on parts that could be solved in a different way.

CHAPTER 2

State of the Art

2.1 Literature Studies

The basic approach of fire detection using static cameras is firstly restricting the areas in a video frame to e.g. pixels that are considered as *moving* with specific *color* values. For the remaining pixels different features like their *changing behavior* or their *color distribution* are calculated. There are basically two different types of features: *Static features* [21] can be used on each frame individually - they do not need information from preceding frames. The second class is called *Dynamic features* [21] - these features use preceding frames to calculate corresponding values, so they can not be used for static fire representations like pictures, for example. Every feature that needs at least two different frames to generate the decision if fire is detected is considered to be a dynamic feature (therefore, it can be concluded that dynamic features need *time* as an input dimension). After that, the outcome of these features is merged to the decision, if fire/smoke is detected in a video frame or not [16] [21] [45]. Figure 2.1 represents this basic workflow of fire detection in video-streams. In this chapter, firstly ways to detect motion in videos are described. Secondly, different features for fire and smoke detection are explained. After that, the workflow of already existing methods for fire and smoke detection (using these features) are shown.

2.1.1 Moving Pixel Detection

Both, fire and smoke cause motion in videos. Therefore, most of the features described in this part can be used to detect smoke and/or fire. However, the motion detection algorithm do not distinguish between motion caused from real fire or smoke, and motion provoked for example by people that move through the scene. A precondition for all presented motion-estimation techniques is the use of *static cameras* [3].

Background Subtraction

Background subtraction is a possibility to estimate moving objects inside a video stream. It is divided in two different approaches, first the non-adaptive method, where a once calculated

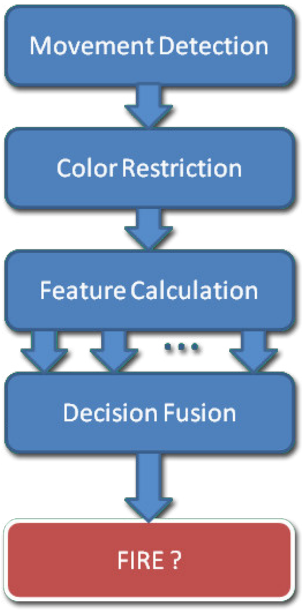


Figure 2.1: Basic workflow of fire detection in video-streams using static cameras.

background image is used to identify differences to subsequent frames [3]. Second, the adaptive method that frequently updates its estimated background, which helps to reduce its sensitivity to noise [3]. However, one requirement of the basic background subtraction algorithms is that the camera is stationary. In [39], a RGB video stream is first converted to the YUV color space, where the channel Y represents the luminance part of the video. The conversion from RGB to YUV is made through three formulas with three multiplications and additions:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.3 & 0.59 & 0.11 \\ -0.15 & -0.29 & 0.44 \\ 0.51 & -0.52 & -0.095 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}, \tag{2.1}$$

so for example the Y channel (brightness) is calculated by

$$Y = 0.3R + 0.59G + 0.11B, \tag{2.2}$$

where R, G and B are the values of the associated red, green and blue channel respectively. An advantage from switching to the YUV space is that the color channels for chrominance (U, V) are separated from the channel for the brightness Y, so it is possible to process each characteristic separately - reducing the amount of calculation needed [46]. To detect motion, first of all a background image has to be estimated. To do this, the first frame is converted to a Y image (see Equation 2.2) and then set as the background image $imBG_{(t,x,y)}$. After that, every consecutive frame is read and saved as the current image, namely $imIn_{(t,x,y)}$. $imIn_{(t,x,y)}$ is also converted from the RBG to the YUV color space. Now the absolute difference between the image and the estimated background is calculated by

$$imDF_{t,x,y} = |imBG_{(t,x,y)} - imIn_{(t,x,y)}|, \tag{2.3}$$

helping to recognize motion in the scene by comparing the calculated difference-values from the difference image $imDF_{t,x,y}$ (calculated for each pixel) with a chosen threshold λ_Y. However, it is also possible to use the mean [8] or median [33] value of the first n frames to calculate the background image. This method is used to reduce the sensibility to noise, for example a false movement detection caused by as a single brightness peak leading to high brightness values for different pixels. The for example mean value for each pixel is calculated by

$$Mean_{(x,y)} = \frac{1}{N}\sum_{i=1}^{N} I_{(x,y,t-i)}, \tag{2.4}$$

where N stands for the number of frames used and $I_{(x,y)}$ is the pixel of the Image I, at position x, y at the time t.

Frame Differencing

As the name *frame differencing* suggests, this method estimates moving pixels through building the difference between the current $frame_t$ and its preceding frame $frame_{t-1}$. The main difference to the approach described in Section 'Background Subtraction' is that the frames are compared among themselves, and not with a generated, static background image. Therefore, slow changes, as for example illumination changes due sunlight or clouds moving, do not have the same high impact on the difference picture as when using a background-image, because the recognized differences are only relative to the previous frame, and not absolute as when comparing pixels with the values stored at the beginning of recording [26].

Combined Motion Estimation

The approach suggested in [27] combines the frame-differencing and the background-image methods. Here, the pixel values of two considered consecutive frames are linearly combined after the decision - if a pixel $p_{(x,y)}$ is moving or not - is made. So, a hybrid background image B_t is produced, where each pixel $I_{t_{(x,y)}}$ represents the intensity value of a pixel in the current frame. A new hybrid background image is generated by

$$B_{t+1_{(x,y)}} = \begin{cases} \text{if} & |B_{t_{(x,y)}} - I_{t_{(x,y)}}| < \lambda_{FD1} \\ & \alpha B_{t_{(x,y)}} + (1-\alpha) I_{t_{(x,y)}} \\ \text{else} & \\ & I_{t_{(x,y)}} \end{cases}, \tag{2.5}$$

where the *if* part represents a pixel detected as non-moving, and the *else* part is executed if the pixel is considered moving. α is a positive real number between 0 and 1. In practice a value close to 1 is chosen [16]. So, if the difference between the two intensity values $B_{t_{(x,y)}}$ and $I_{t_{(x,y)}}$ is smaller than a threshold λ_{FD1}, the new value for the background pixel is the linear combination of the intensity value of the current pixel and the value for the last valid background pixel. If the difference is bigger than the threshold, this pixel is considered as *moving*. In this case, the value for the next background image B_{t+1} is just the intensity value of the current frame $I_{t_{(x,y)}}$. On the beginning, the background image B_{t+1} is set to the first frame I_0.

In [16], the used method differs from that one used in [27]. In [16], in a first step moving regions are determined through direct comparison of two consecutive frames. In the second step, the pixel value is compared to the estimated background image to identify pixel that differ significant from the background. So, first the formula to estimate the background (Equation: 2.5) is adapted to

$$B_{t+1_{(x,y)}} = \begin{cases} \text{if} & |I_{t+1_{(x,y)}} - I_{t_{(x,y)}}| \leq \lambda_{FD2} \\ & \alpha B_{t_{(x,y)}} + (1-\alpha) I_{t_{(x,y)}} \\ \text{else} & \\ & B_{t_{(x,y)}} \end{cases} . \tag{2.6}$$

The *if* branch represents a non-moving and the *else* part represents a pixel detected as moving. So, a pixel is considered as *moving*, if $|I_{t+1_{(x,y)}} - I_{t_{(x,y)}}| > \lambda_{FD2}$ is true, where $I_{(x,y)}$ represents the brightness value of a pixel. The second possibility (that a pixel is considered as moving) is that its brightness value differs significantly from the estimated background image - in other words, all pixel that fulfill the equation $|I_{t_{(x,y)}} - B_{t_{(x,y)}}| > \lambda_{FD2}$, where $B_{t_{(x,y)}}$ represents the background-value estimated through Equation 2.6.

Movement Detection using Motion Vectors

Motion information is automatically generated when a video is compressed with a MPEG-x codec [48]. The extracted motion-information is, as a vector, stored separately inside the MPEG data format. These vectors are accessible as *read-only*, even if the video file is not fully decoded [48]. There are different methods to detect motion inside a video - however, they all have in common that they produce an encoded motion vector MV as a result. This represents the motion of a so called *Macro-Block* (MB), the dimension of the movement of such a MB is defined as

$$MV_{MB} = \sqrt{(MV_x)^2 + (MV_y)^2}, \tag{2.7}$$

where MV_x and MV_y represent the x- and y-values of the motion vector of this MB, and x, y defines which MB is currently processed. To determine if this region is considered as *moving*, it is compared to a chosen threshold λ_{MB}, leading to the *Moving Region Decision* (MRD)

$$MRD = \begin{cases} 1, & \text{if } (MV_{MB} > \lambda_{MB}) \\ 0, & \text{otherwise.} \end{cases} \tag{2.8}$$

This leads to the knowledge, which regions of this image (or frame) are considered as moving regions [23]. Figure 2.2 shows a frame captured from a video. The bottom image shows the image after processing, where the detected moving macro blocks have been colored in blue. In this case, the motion estimation for calculating the motion vectors for the macro blocks inside the video is executed through the *Simple and Efficient Three-Step Search* (SETSS) [24].

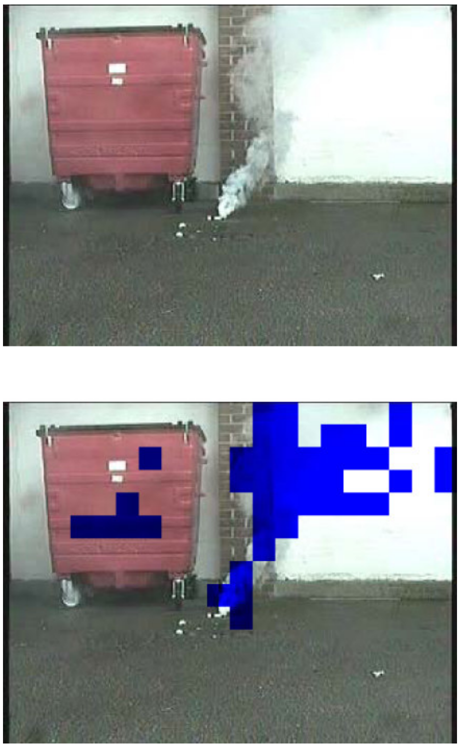


Figure 2.2: The original frame on top, below the processed frame with all blocks considered as *moving* colored in blue [23].

2.1.2 Features for Fire Detection in Videos

Fire detection through video surveillance leads to a big advantage: the detection speed of classical detectors, such as gas, heat or smoke detectors, depends on their distance between the fire and the position of the sensor. An approach based on video processing leads to a very fast response in case of a fire event [7]. The basic idea is to copy the detection ability and speed of the human eye [7]. In this section, a selection of features is presented which are used to identify fire and smoke regions.

Colorhistogram

As mentioned in [5], color is the most powerful single feature to detect fire regions in images or videos. The typical flame seen in nature belongs to the *red-yellow* color range, and is called a *hydrocarbon flame*. Koerich Borges and Ebroul Izquierdo [5] also observed a special feature regarding a fire region: if the values of the red, green and blue parts of a fire pixel is analyzed, the value of the red channel is bigger than the value of the green channel. In addition, the value of the green channel is bigger than the value of the blue part of that pixel. Figure 2.3 shows this typical characteristic of fire pixels. In [40], the criteria $R > G$ and $G > B$, which leads to a

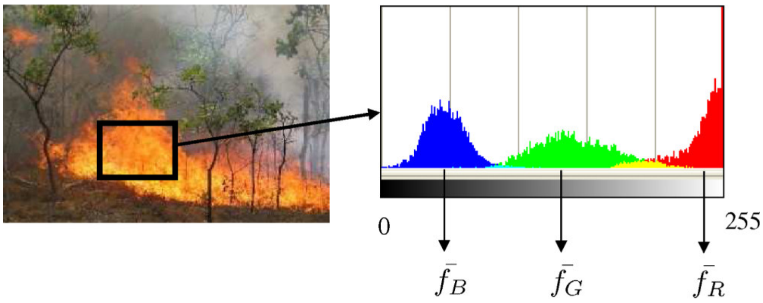


Figure 2.3: Histogram for the red, green and blue channels of the fire region inside the black square [5].

combined condition of $R > G > B$, are further restricted with an threshold λ_R, where the value for the red channel has to bigger than λ_R to detect fire. The last refinement is that the saturation value of a pixel has to be over a threshold λ_{Sat}, leading to the combined fire detection

$$FireColored_{(x,y)} = (R > G > B) \wedge (R > \lambda_R) \wedge (Sat > \lambda_{Sat}) \tag{2.9}$$

where, R, G, B are the values for the red/green/blue channel, respectively. λ_R is the threshold for the red channel, Sat is the saturation value of the pixel and λ_{Sat} is the threshold for the saturation. If Equation 2.9 is true, a observed pixel is considered to be a fire pixel [40].

Color Range

Due to the unique color range of fire, it plays a important role in fire detection in videos. Video-shots taken are mostly represented in the RGB color space, where every pixel is a three dimensional vector of the three color parts represented in this part of the video, namely *Red* (R), *Green* (G) and *Blue* (B). The human characteristic of seeing colors is better represented due the HSV/HSI color model, where the pixel is represented by its values of *Hue* (H), *Saturation* (S), *Value* (V) / *Intensity* (I). Hue represents the color that is perceived, for example red, orange, and so on. Saturation is the value of how much white is mixed with the color, where a smaller number means a less amount of white. Value measures the brightness of that particular color. In [7], ten different frames from five video shots containing fire were taken. After analyzing all 50 frames, the range of each color part (R, G, B) was identified (see Table 2.1 for details). If only RGB values are used, the results of the generated threshold image is quite satisfying. However, the image presented in Figure 2.4 does not contain any *non-fire* objects with a color near the color range of a fire (for example an orange-colored object). A conversion from RGB to the

Table 2.1: RGB color range of fire [7]

	Red	**Green**	**Blue**
Range	190 to 255	113 to 255	0 to 80

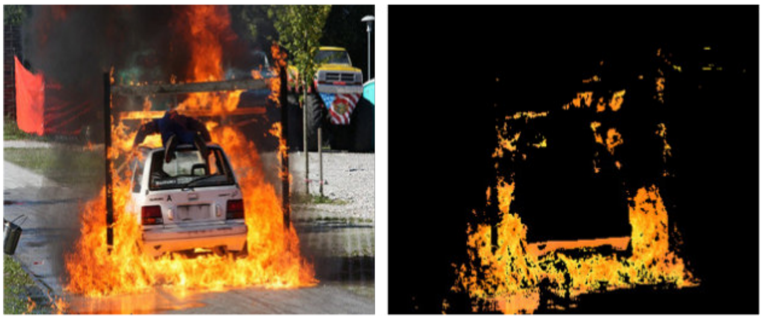


Figure 2.4: The image of a burning car before and after thresholding with RGB values [7].

HSV/HSI color space is performed as follows: Before, all RGB values have to be normalized to a range between 0 and 1 [7]. The formulas for the conversion are,

$$H = \begin{cases} \Theta & \text{if } B \le G \\ 360 - \Theta & \text{if } B > G \end{cases} \tag{2.10}$$

where,

$$\Theta = \cos^{-1}\left(\frac{\frac{1}{2}((R-G)+(R-B))}{((R-G)^2+(R-B)(G-B))^{\frac{1}{2}}} \right) \tag{2.11}$$

$$S = 1 - \frac{3}{(R+G+B)}[min(R,G,B] \tag{2.12}$$

$$I = \frac{1}{3}(R+G+B) \tag{2.13}$$

$$V = max(R,G,B) \tag{2.14}$$

After calculating the values with the Equations (2.10), (2.11), (2.12), (2.13) and (2.14), all values have to be rescaled to a range from 0 to 255. Table 2.2 shows the corresponding HSV values to the measured RGB values from Table 2.1.

Table 2.2: HSV color range of fire [7]

	Hue	**Saturation**	**Value**
Range	0 to 43	128 to 255	190 to 255

Color Based Detection Metric

As presented in [5], another method to decide if the color of a pixel is a possible fire candidate is using the Gaussian distribution of the three color channels. A given pixel $f(m,n)$, is defined as

$$f(m,n) = \begin{bmatrix} f_R(m,n) \\ f_G(m,n) \\ f_B(m,n) \end{bmatrix}, \tag{2.15}$$

where f_R, f_G and f_B are the values of the three color channels red, green and blue. To determine the values of $\bar{f_R}$, $\bar{f_G}$ and $\bar{f_B}$ the mean value of the channels extracted from pixels inside the fire region is used. After the standard deviation is calculated a Gaussian Model is applied for every channel, so every color channel $f_{Color} \sim N(\mu_{\bar{f}_{Color}}, \sigma^2_{\bar{f}_{Color}})$. The basic idea is, to define a function that represents the probability that an observed region $f_{observed}$ is a part of a fire region. So three different values D_{C_R}, D_{C_G} and D_{C_B} are calculated by equation

$$D_{Color} = \frac{p_{\bar{f}_{Color}}(\bar{f}_{Color_{observed}})}{p_{\bar{f}_{Color}}(\mu_{\bar{f}_{Color}})}. \tag{2.16}$$

This function has its max, when the average color value of the observed region is the same value as defined before, namely when $\bar{f}_{Color_{observed}} = \mu_{\bar{f}_{Color}}$. In this particular case the value of D_{Color} would be 1. Using the three values calculated in Equation 2.16, the detection metric is defined as

$$\begin{aligned} D_C = D_{C_R} + D_{C_G} + D_{C_B} - (D_{C_R}D_{C_G} + D_{C_R}D_{C_B} + \\ + D_{C_G}D_{C_B}) + D_{C_R}D_{C_B}D_{C_G}. \end{aligned} \tag{2.17}$$

As mentioned in [5], D_C provides meaningful results. To get an image that shows fire regions only, a binary function is defined as

$$FireImage(m,n) = \begin{cases} 0, & \text{if } D_c(m,n) < \lambda_C \\ 1, & \text{otherwise} \end{cases}, \tag{2.18}$$

where λ_C is a chosen threshold level. The false-positive rate (assumes that there is a fire but in fact there is none) was, using this method only, 9.37%, the false-negative rate (assumes that there is no fire but in fact there is one) performed better with 0.033% [5].

Color by Gaussian Mixture Model

In [16], the color restriction is executed differently. A Gaussian Mixture Model (GMM) using ten different Gaussian distributions is used to evaluate whether the color value of a given pixel fits the requirements for fire or not. The values for the GMM are calculated by analyzing training images containing fire events. If the color value of this pixel lies inside the double standard deviation of at least one of the Gaussian distributions it is assumed as fire-colored [16]. Figure 2.5 shows these restrictions illustrated by a 3D space, with each color value (R,G,B) representing a different dimension.

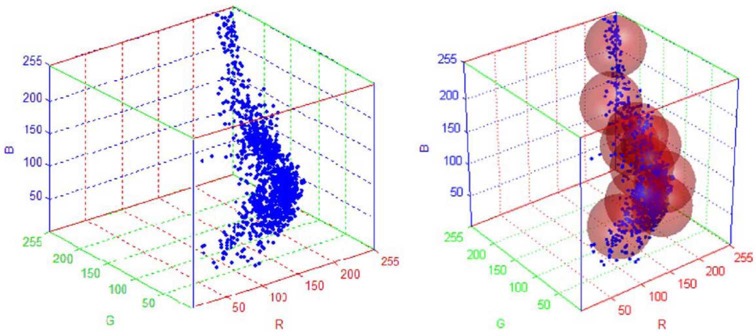


Figure 2.5: Left: Pixel values extracted from sample images containing fire. Right: Valid pixel values considered as *fire-colored* are shown as spheres in the 3D space, represented by one dimension for each color R, G and B [16]

Skewness

The skewness is a measurement degree for distributions [28]. If the distribution itself is perfectly symmetric about its mean value, the skewness factor is 0 [28]. If the center of the distribution is shifted to the left, the skewness value is positive, if it is shifted to the right it is negative [5]. Figure 2.6 illustrates this effect. As mentioned in Section 'Colorhistogram' and illustrated in Figure 2.3, a fire region has high values for the green channel and even higher values for the red channel. If analyzing the red channel of a fire region alone, it can be observed that this channel has a high degree of saturation [5]. Figure 2.7 shows the histogram of the red color-channel, calculated from a fire region. A distribution, as shown in Figure 2.7, leads to a skewness with a negative value. The skewness γ_R is calculated by

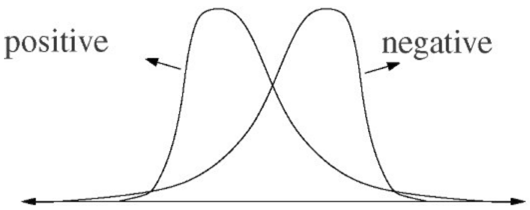


Figure 2.6: Distribution with a positive and a negative skewness [5]

$$\gamma_R = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s}\right)^3, \tag{2.19}$$

where x_i represents the red value of each pixel inside the observed region and $\bar{x}$ presents the mean value of the red channel of all pixels. The value s is the calculated standard deviation [5].

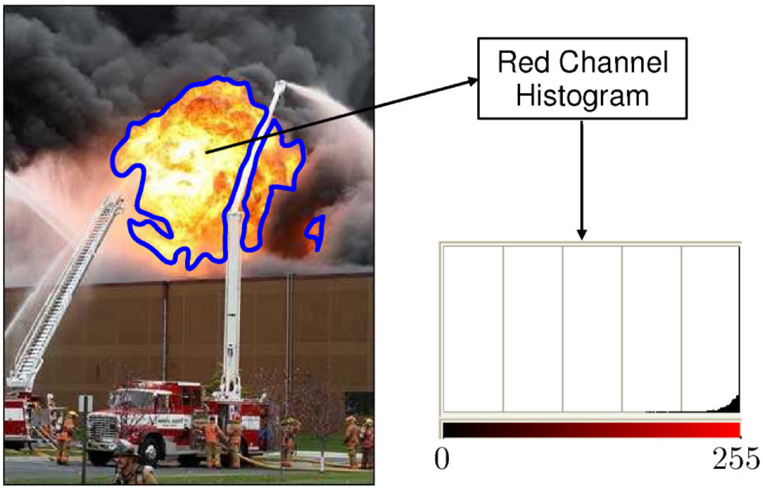


Figure 2.7: Histogram of the fire region inside the blue border. This leads to a high amount of red pixels with the value 255, which is called the *saturation effect* [5].

An observed region is a potential fire region, if

$$\gamma_{R_i} < \lambda_{\gamma_R}, \tag{2.20}$$

where λ_{γ_R} is a chosen threshold. Experiments in [5] shows that fire regions usually have a skewness value $\lambda_{\gamma_R} < -1$.

Surface Coarseness

Another feature that can distinguish for example between a fire or a person wearing a fire-colored shirt is called *surface coarseness* [5]. Inside fire regions, there is a huge amount of pixels within the fire color range, but their color values are widely spread, while in typical false alarm regions the color does not vary that much [5]. Although, there a several tools to recognize texture, fire itself has a huge randomness in its behavior and can therefore vary a lot in its appearance [5]. So in [5] this coarseness in the region is measured by the standard-deviation σ of the color values of pixel inside its border. As described in [28], the standard-deviation is calculated by

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}, \tag{2.21}$$

where x describes a pixel color and $\bar{x}$ presents the mean color value of all pixels within the region. If $\sigma > \lambda_\sigma$ is true, it is assumed that the observed area is a fire area, where λ_σ represents a chosen threshold. Figure 2.8 shows the principle for a detected region using a threshold $\lambda_\sigma = 50$.

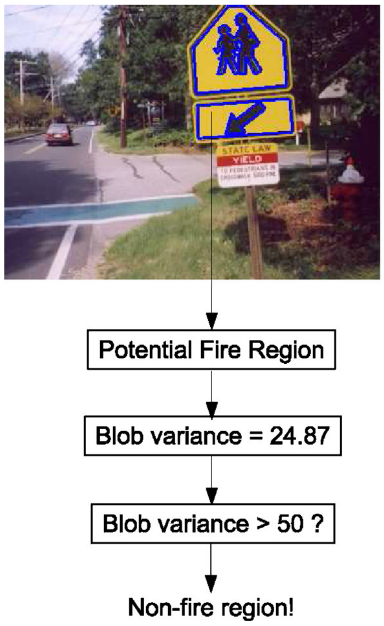


Figure 2.8: Determination if a detected region is a potential fire region through analysis of its variance σ [5].

Temporal Wavelet Analysis

If the video capture rate is high enough (at least 20 frames per second), fire can be estimated through continuously measuring pixels inside fire colored regions by analyzing their history of change. Therefore, a filter bank with two stages is used. A pixel $x_{n_{(x,y)}}$, represented by its frame number n and its position (x, y) is fed into the filter bank, using either the luminance value Y from the YUV color model (see Section 'Background Subtraction' for details), or the R value from the RGB model. The filter bank consist of half-band high pass and half-band low pass filters as shown in Figure 2.9. The outcome of this filter are two wavelet (sub-)signals, namely $d_{n_{(x,y)}}$ and $e_{n_{(x,y)}}$. If the values of an input pixel $x_{n_{(x,y)}}$ do not alternate much, the values for both wavelet sub-signals are equal or very close to zero. However, if there is a high frequency activity they get non-zero values. So, the zero crossings rate of $d_{n_{(x,y)}}$ and $e_{n_{(x,y)}}$ can help do discriminate between a pixel that represents a real fire pixel and a pixel from an ordinary, fire colored object moving through the scene. Due to the down-sampling behavior caused by each number of wavelet stages, for example when using the filter-bank shown in Figure 2.9, only oscillations between $\frac{1}{8}$ and $\frac{1}{2}$ of the original frame-rate can be detected, which means that - if the input video is captured at a frame rate of $10Hz$ - only fluctuations between $1.25Hz$ and $5Hz$ are detected [16].
Figure 2.10 shows the behavior of $d_{n_{(x,y)}}$ and $e_{n_{(x,y)}}$ when the input pixel $x_{n_{(x,y)}}$ is part of a real flame (using a video from [16], a pixel at the position $(111, 34)$ is evaluated, which was part of a real flame in the frames *n = 1, 2, 3, 19, 23, 24, 41* and part of the background in *n = 12, ..., 17, 20,*

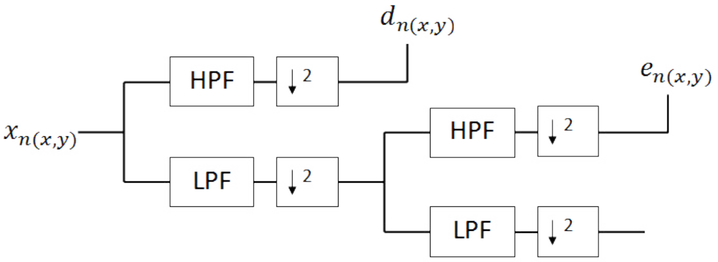


Figure 2.9: Two stage filter bank, using high-pass filters (HPF) and low-pass filters (LPF), producing the two wavelet (sub-)signals $d_{n_{(x,y)}}$ and $e_{n_{(x,y)}}$ [16].

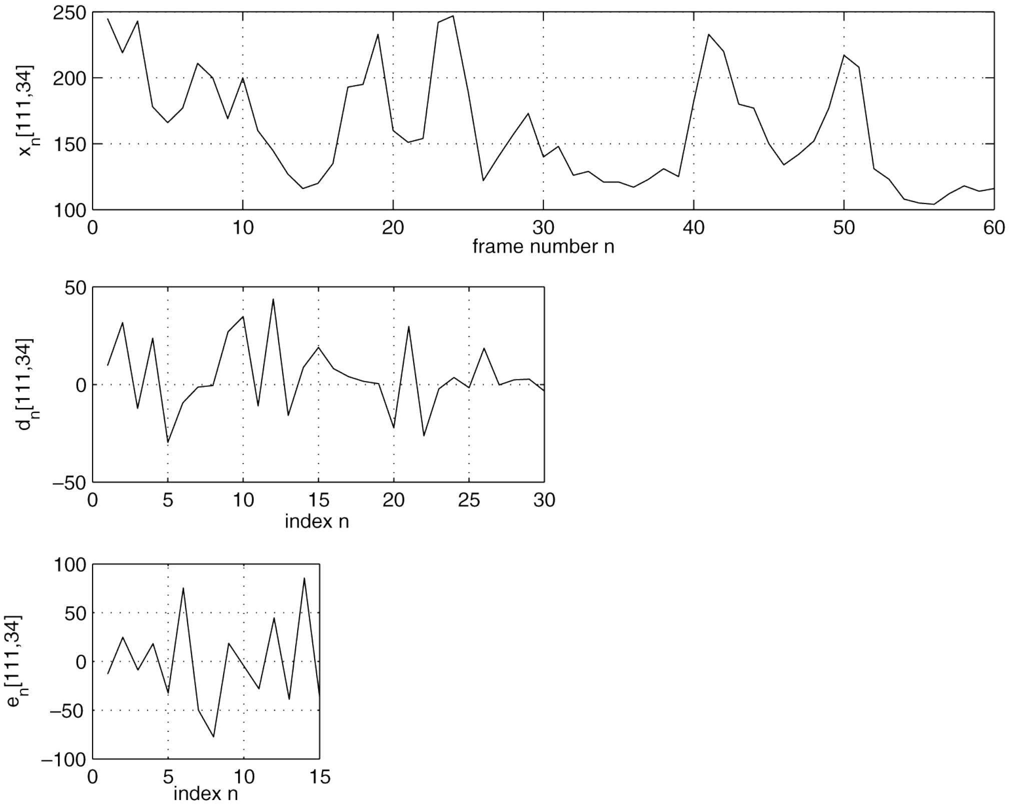


Figure 2.10: Outcome of the filter bank proposed in Figure 2.9. On top, the values of the red channel for pixel $x_{n_{(111,34)}}$ can be seen. This pixel is part of a flame for the frames *n = 1, 2, 3, 19, 23, 24, 41*, and for *n = 12, ..., 17, 20, 21, 26, 27, 31, ..., 39, 45, 52, ..., 60* it is part of the background. The oscillating behavior of $d_{n_{(111,34)}}$ and $e_{n_{(111,34)}}$ can be clearly seen. Please note that each wavelet sub-signal is down-sampled by half, respectively [16].

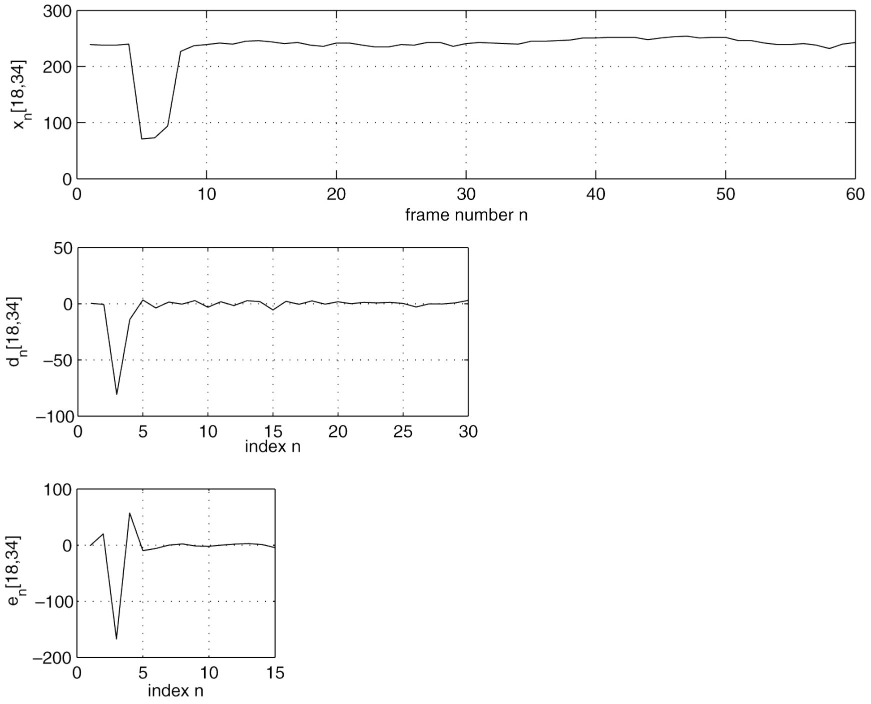


Figure 2.11: Temporal history of pixel $x_{n_{(18,34)}}$, which is part of a fire-colored object moving through the scene in frames *n* = *4, 5, 6, 7, 8*. For every other frame it is part of the background. It can be recognized that, after the transition from *background to object* and the subsequent transition between *object and background* no oscillating behavior appears [16].

21, 26, 27, 31, ..., 39, 45, 52, ..., 60). Because of the down-sampling by half, it can be recognized that the initial 60 frames are respectively down-sampled by 50%, leading to 30 frames for d_n and 15 frames for e_n. This leads to the fact that the value for the sub-signal $d_{5_{(111,34)}}$, for example, belongs to the same pixel located in two frames, namely $x_{10_{(111,34)}}$ and $x_{11_{(111,34)}}$. This means that the value for e.g. $e_{4_{(111,34)}}$ corresponds to the pixel values of four frames, namely $x_{12_{(111,34)}}$, $x_{13_{(111,34)}}$, $x_{14_{(111,34)}}$ and $x_{15_{(111,34)}}$ [16]. For a better comparison, Figure 2.11 shows the same procedure, but in this case only a fire colored object moves through the scene. In this case, the pixel $x_{n_{(18,34)}}$ is evaluated, which is part of the (white) background for the frames *n = 1, 2, 3*, and part of a fire colored object for *n = 4, 5, 6, 7, 8*. After that, it is part of the background again. It can be noticed that the transition between object and background (and vice versa) leads in each case to a single peak in the two wavelet sub-signals. After both peaks occurred, nearly no oscillating behavior can be recognized. Those small variations, mainly caused through noise, can be eliminated by using a threshold [16].

Spatial Wavelet Analysis for Fire

A fire colored object differs from fire due the lesser amount of spatial variation in color [16]. Figure 2.12 shows such a fire colored object on the left side, showing that the color located within the rectangle does not show much variation. On the other hand, at the left side of Figure 2.13, the significant amount of spatial variations inside the marked area - containing real fire - is visible. The difference in the spatial variation, comparing the spatial wavelet analysis shown on the right side in Figure 2.12 and Figure 2.13, is a discriminant for real fire [16]. The wavelet sub-image itself contains edge information about the computed scene, because edges in the original image are causing local extrema in the wavelet images. So, the wavelet image low-

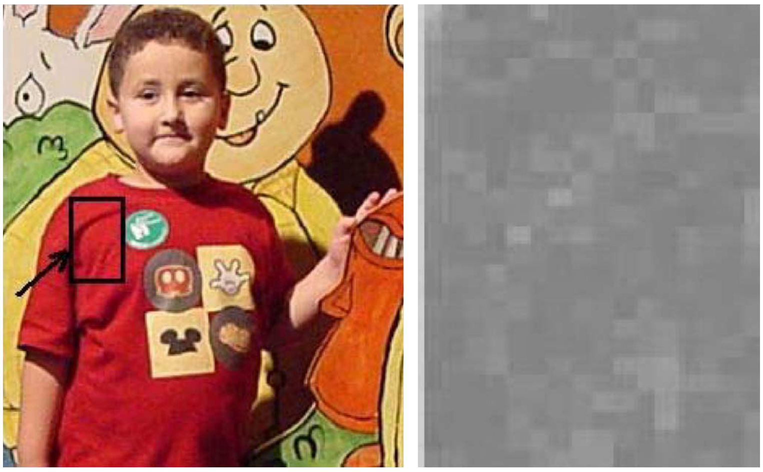


Figure 2.12: Left: a child with a fire colored shirt. Right: the wavelet image containing the accumulated, absolute values of the low-high, high-low and high-high wavelet transformations $|LH_{(x,y)}| + |HL_{(x,y)}| + |HH_{(x,y)}|$ from pixels inside the rectangle [16].

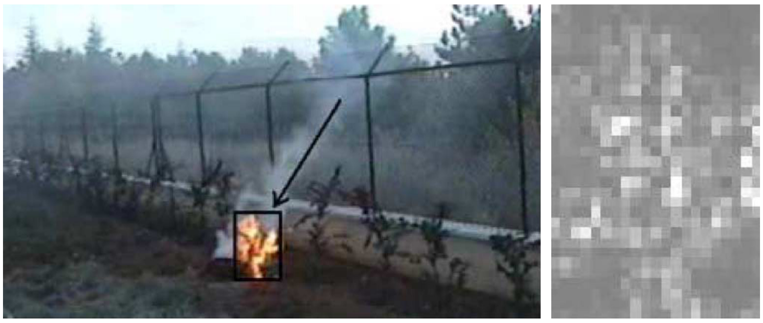


Figure 2.13: Left: an image of a real fire. Right: the wavelet image containing the accumulated, absolute values of the low-high, high-low and high-high wavelet transformations $|LH_{(x,y)}| + |HL_{(x,y)}| + |HH_{(x,y)}|$ from pixels inside the rectangle [16].

high (LH) is using a low-pass filter for the horizontal scan and a high-pass filter for the vertical scan. Due to the fact that a high pass-filter shows the high-frequency parts of an image (caused by edges, for example), the image after the LH pass contains the horizontal edges of the original image. Furthermore, a high-low pass (HL) contains the vertical edges from the input-image. In conclusion, the high-high pass (HH) contains all diagonal edges [13]. High-pass and low-pass are dividing the frequencies in half, resulting in scaled images with 50% height and 50% width from the original image.

In the case of detecting fire, it is sufficient enough to use the red channel of the RGB image. To measure the amount of disorder, a decision parameter E_{fire} is used. The absolute values of the LH, HL and HH pass are added to compute the resulting image. The decision parameter is calculated by

$$E_{fire} = \frac{1}{M \times N} \sum_{x,y} |LH_{(x,y)}|^2 + |HL_{(x,y)}|^2 + |HH_{(x,y)}|^2, \tag{2.22}$$

where $LH_{(x,y)}$ is the low-high sub-image, $HL_{(x,y)}$ is the high-low sub-image and $HH_{(x,y)}$ is the high-high sub-image of the wavelet transform. $M \times N$ calculates the number of considered pixel inside the region.

After that, the decision parameter is compared to a threshold $\lambda_{Wavelet_{fire}}$. If $E_{fire} > \lambda_{Wavelet_{fire}}$ holds *true*, the spatial disorder is high enough to be considered as a possible fire region [16]. It is noteworthy that the wavelet analysis is also used in Section 'Spatial Wavelet Analysis for Smoke'.

Randomness of Area Size

When comparing a fire area from a video at the time-stamp t and the same fire area from the same video at the time-stamp $t + 1$ it can be recognized that the size of the area changes [5]. Wrongly detected fire areas usually do not trigger as large changing areas as a real fire does. Therefore, this feature can help to distinguish between a true-positive (fire correctly detected) and a false-positive (fire detected but there is no real fire in the scene). The degree of change can be measured by

$$\Delta A_i = \frac{|A_i - A_{i-1}|}{A_i}. \tag{2.23}$$

The result is a normalized value for the amount of change that happened between *frame(i-1)*, where an possible fire area with the size A_{i-1} (counting the fire-pixel candidates) is detected, and *frame(i)* where an area with the size A_i is identified as a fire region. If the degree of change ΔA is above the decision threshold λ_A, the area is marked as a potential fire region [5]. Figure 2.14 shows the change of the fire area between *frame(i-1)* and *frame(i)*. In this case, the size changes from $A_{i-1} = 1692$ pixels to $A_i = 2473$ pixels, which results in a $\Delta A = 0.316$.

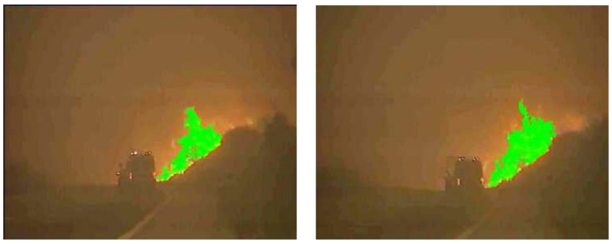


Figure 2.14: The difference of the area size between *frame(i-1)* and *frame(i)*, leading to a change from $A_{i-1} = 1692$ pixels to $A_i = 2473$ pixels [5].

Boundary Roughness

Because the shape of fire changes randomly [5], another feature of interest is the roughness or randomness of the shape of a fire region. Therefore, in [5] a roughness descriptor B_R is defined as the ratio between the perimeter of the potential fire region and the smallest convex hull surrounding this area. If B_R exceeds a decision threshold the examined region is considered to be a fire region. Figure 2.15 shows a fire region marked in green with its smallest surrounding convex hull.

Fire Growth Rate

As mentioned in [45], the growing rate of fire is dominated by the airflow around it and its fuel type. Generally, the size of the fire region changes randomly but with an overall growing behavior over time. Let m_i be the amount of detected fire pixels for the current frame, and m_{i+1} the quantity of fire pixels from the next frame. If the statement $m_{i+1} > m_i$ is true for more than g times (in an interval t) the detected region can be considered as *growing* and therefore as a real fire region. The parameters g and t are evaluated by statistical data provided by experiments.

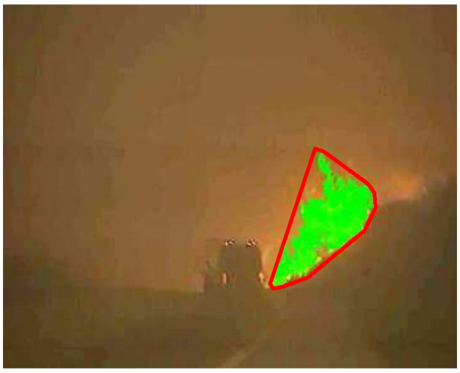


Figure 2.15: Due to the random shape of fire, a shape descriptor is calculated by using the ratio between the perimeter of the potential fire region and its smallest convex hull surrounding this area [5].

Spatial Distribution of Fire

Fire in *human recorded* videos (for example newscast-videos) is usually the most significant part that want to be captured in a shot. Therefore, the fire region itself is usually close to the center of the video-frame when using handheld cameras. In [5], fire regions were manually segmented from 120 frames. After that, a 3D representation s was generated showing the distribution of the fire color pixel by equation $s = \sum_{i=1}^{120} frame_i$. Figure 2.16 shows the generated 3D image. It can be observed that the probability for a fire (in videos recorded manually) is higher if the detected fire area is in the center of the video-frame. Therefore, the position of potential fire-candidate pixels is used as a feature. However, this statement holds not true for videos from for example surveillance cameras as mentioned in [5]. Therefore, this feature is not useful for an automatic detection of an fire incident through static cameras.

Combined Features

All features can be combined or executed consecutively in a pipeline. A simple form could be

- Generate a binary image of each frame through the pre-condition *'Color'* as described in Section 'Color Range'
- Refine this binary image through using the restrictions known from Section 'Skewness'
- Refine the outcome through 'Randomness of Area Size'
- Finally refine that outcome with the restrictions described in Section 'Surface Coarseness'

Borges and Izquierdo [5] are proposing a consecutive execution of features with hard decision rule for each value V $< \lambda_{feature}$ [5], where possible fire candidates are estimated for each feature. The final fire region is the intersecting set of a frame thresholded by all features. The

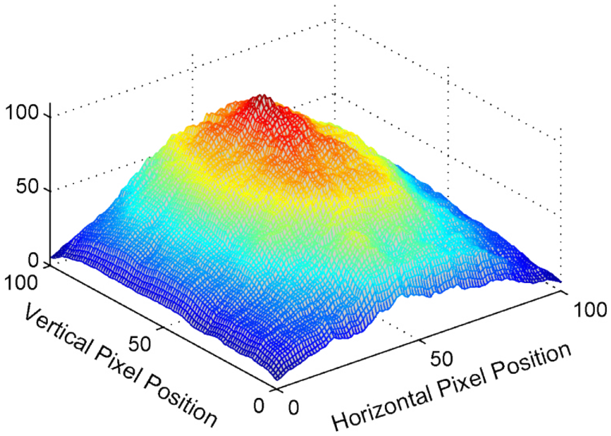


Figure 2.16: A 3D representation of the spatial distribution of fire in *human recorded* videos such as used for newscast. It can be clearly seen that the fire itself is usually the most important part of the video an therefore placed near the center of the video [5].

pipelining method leads to a speed advantage, because only potential fire pixels estimated from the previous feature are used for the next stage (instead of defining them with each feature again and again for each pixel). A disadvantage is that it is sufficient that only one feature has a false-negative detection to dismiss this part as a fire region. However, Borges and Izquierda are using a Bayes classifier in the end in [5].

2.1.3 Fire Feature Extraction using Time-of-Flight Cameras

In this section, *Time of Flight* (TOF) cameras are introduced to allow a deeper understanding of the features already used by Verstockt and De Potter in [29] for fire-detection using a TOF 3D sensor. A TOF camera tries to estimate a depth map of a monitored scene using *Infrared Light* (IR). It is noticeable that the working principle is different than the *structured light method* - where a defined light pattern is projected by the sensor and the depth is calculated by detecting parts of the pattern with additional triangulation - used by sensors like the *Microsoft Kinect* or the *ASUS Xtion Pro*. In this section, the working-principle of TOF cameras is described. After that, a method for detecting fire is examined. The TOF camera uses its IR-light to illuminate the scene, so the camera itself is surrounded by IR-*Light-Emitting Diodes* (LEDs) as shown in Figure 2.17. Based on frequency modulation, all IR-LEDs are flashing. The distance d to an object in front of the camera is calculated to through the time difference Δt and the speed of the used signal, in this case the speed of light $c \approx 3 * 10^8 m/s$, leading to equation $d = c * \Delta t/2$. This principle can also be seen in Figure 2.17. The second image (besides the depth-image) a TOF camera generates is the so called *amplitude image*. It represents the strength of the received IR signal. The third image is a common RGB image of the scene [29].

Some of the advantages of TOF cameras are [29]:

- Due to the fact that TOF cameras are using their own invisible IR light, they are not sen-

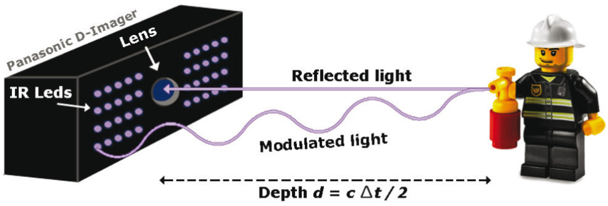


Figure 2.17: The working principle of a time-of-flight camera. Infrared light is emitted by LEDs surrounding the camera. The time difference between sending the IR light and receiving its reflection is measured. The distance (depth d) (to the object that had reflected the light) is calculated by $d = c * \Delta t/2$ [29].

sitive to shadows or light changing conditions. This helps a lot when using TOF cameras for example for motion detection [29]

- There is not much post-processing needed, so TOF cameras can be used for real time applications [29]

- The depth map can help do separate objects that overlap or occlude each other [29]

- Low price compared to other IR sensors [29]

But there are disadvantages, too [29]:

- Low resolution. An average TOF camera has a resolution of about $176 \times 144\, pixels$. However, it is quite certain that the resolution will increase the next years, as it has happened in digital imaging [29]

- Measurement artifacts: An object far away from the camera can not be illuminated enough by the IR LEDs, so it leads to poor measuring results [29]

- Fast motion in a scene can also lead to measuring artifacts [29]

- If an object is far away, artifacts can occur, because they lie out of the *non-ambiguity* range of the sensor leading to a *out-of-phase* signal an therefore to unreliable depth maps [29]

- TOF cameras have a high power consumption, because they use active illumination [29]

However, fire itself emits infrared light, which leads to a lot of measuring artifacts. Usually, this influence is greater for the depth map than for the amplitude image. So if a TOF camera provides both, depth map and amplitude image, it is recommended to use the amplitude image only, as mentioned in [29].

Time-of-Flight Detectors - Amplitude Image

The *Time of Flight* (TOF) camera emits infrared (IR) light and measures the time it takes to receive the reflection of it again to calculate a corresponding depth map. The amplitude image (also called *intensity image*) on the other hand, does not represent the elapsed time - it represents the intensity of the IR light received. The basic idea is that the intensity of the received IR light decreases with the distance of the object that had reflected it. Figure 2.18 shows such an amplitude image and the resulting ranges calculated from it.

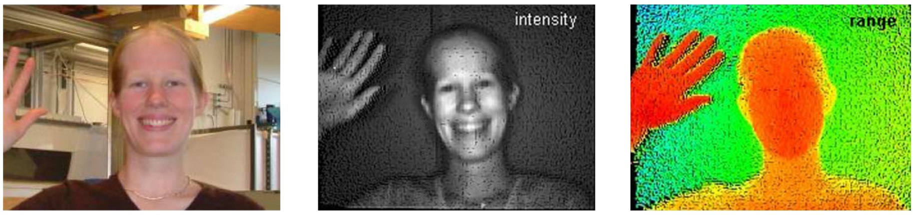


Figure 2.18: Left: a RGB frame, Middle: an amplitude image from the similar scene, showing the intensity of the received infrared light, Right: the calculated depth for the scene [32].

Amplitude Disorder Detection

Fire itself emits IR light and causes therefore artifacts in the amplitude image [29]. Those artifacts show a high amount of disorder, as shown in Figure 2.19. To detect this disorder, the so called *Accumulated Amplitude Difference* AFD_n^{amp} [29] is calculated by equation

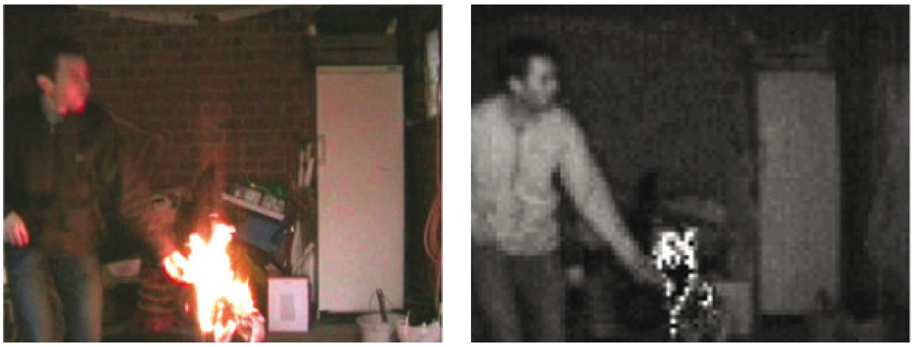


Figure 2.19: Fire that causes a high amount of disorder in the corresponding amplitude/intensity image [29].

$$AFD_n^{amp} = \lfloor |F_n^{amp} - F_{n+1}^{amp}| + |F_n^{amp} - F_{n-1}^{amp}| \rceil \tag{2.24}$$

It is a measurement for the difference between the amplitude image of the current frame F_n^{amp}, and the amplitude image of the previous frame F_{n+1}^{amp} and the next amplitude frame F_{n+1}^{amp} respectively. Through rounding the two absolute differences, this method is able to distinguish between (fast moving) fire and (slow moving) non-fire objects [29]. However, a problem arises with slow moving objects close to the TOF sensors, because a high amount of disorder is received at the boundaries of these objects [29].

Discrete Wavelet Transformation of Amplitude Images

The discrete wavelet transform helps to identify areas with high contrast (e.g. boundaries), through scanning an image horizontal, vertical and diagonal. This conversion to the wavelet-domain can be used to eliminate the problem with the disorder generated by ordinary (non-fire) objects, as mentioned in Section 'Amplitude Disorder Detection'. Figure 2.20 shows the high values caused by fire in the discrete wavelet transformation of the amplitude frame. This high values in each wavelet transformed sub-image are unique for regions with fire. Ordinary moving objects close to the TOF sensor do not generate such a behavior. Hence, fire is detected, if

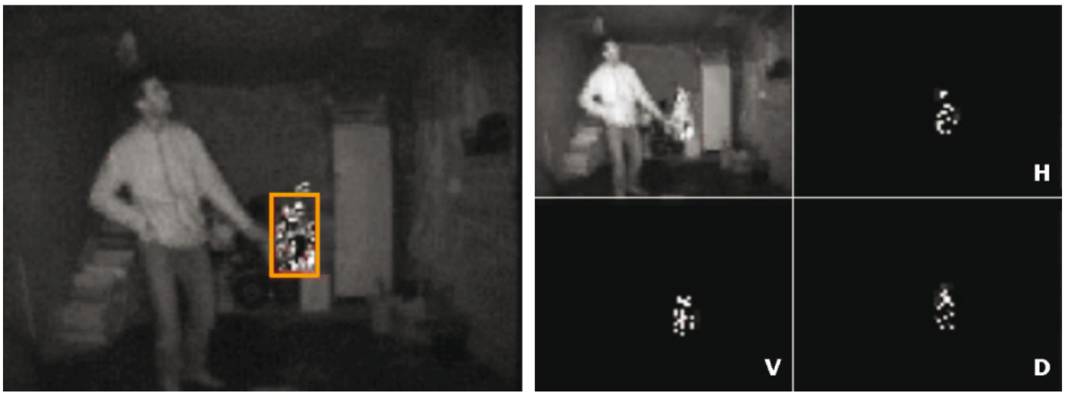


Figure 2.20: A captured amplitude image frame. The discrete wavelet transformation shows a high amount of disorder in the horizontal, vertical and diagonal analyzed sub-images, respectively [29].

$$DWT_R = \begin{cases} 1, & \text{if } max(H_R) \times max(V_R) \times max(D_R) = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{2.25}$$

where H_R, V_R and D_R are the values for the horizontal, vertical and diagonal discrete wavelet transformations of Region R and is set to e.g. 1 if there are high values represented in all three discrete wavelet transformations. DWT_R generates a binary image where flame candidate pixel have the value $true$ [29]. In [29] it is mentioned that the depth map of a TOF camera is no reliable source for fire detection. The decision to use the depth-map for the method proposed in this thesis, regardless the previous results provided by [29], was made, because peripheral devices like the ASUS Xtion Pro use a different method to estimate the depth-map, namely the

Structured Light Imaging. However, this method provides useful information for e.g. a multi-sensor fusion.

Multi-Sensor Fusion

When using multiple sensors, e.g. a RGB sensor and a depth-sensor, and therefore usually multiple different approaches to detect fire, it is necessary that these sensors have to be aligned and (if necessary) correctly rectified. Supposed, fire has been detected with two different sensors in two different ways respectively, for example Sensor 1 called $Fire_n^{RGB}$ for the fire detection using a RGB camera and calculating corresponding features, and Sensor 2 called $Fire_n^{AMP}$ for the fire detection through an amplitude image approach. Each binary image consists of pixel considered as *fire* marked by 1, and pixel considered as *no-fire* represented by 0 values. The fusion of the aligned (!) images takes place through using a logical *AND* operator, which leads in this case to equation

$$Fire_n^{ALL} = Fire_n^{RGB} \wedge Fire_n^{AMP}, \tag{2.26}$$

where $Fire_n^{ALL}$ represents the final output frame of detected fire pixel [29]. Figure 2.21 shows such an overlap detection by combining the binary-images provided by two different sensors.

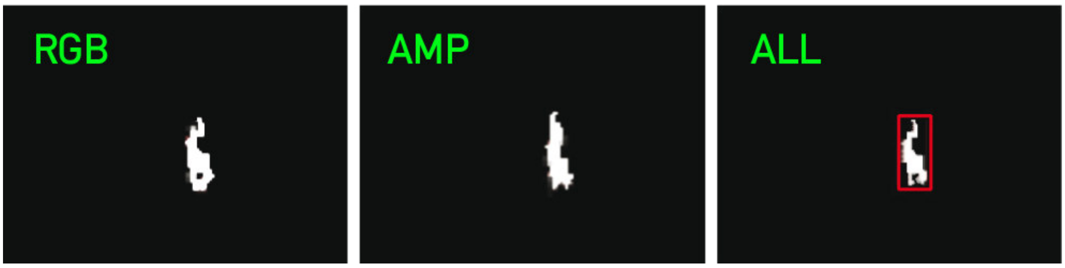


Figure 2.21: Left, the output (candidate region) of the fire detection algorithm using a RGB camera. The illustration in the middle visualizes the binary image, resulting from analyzing the amplitude image provided by the detection algorithm of *sensor 2*. The right side shows the resulting frame, in this case the combination by a logical AND operator [29].

2.1.4 Features for Smoke Detection in Videos

Smoke can be used as an early detector, for indoor purposes or for large area outdoor purposes, like in a forest fire detection through video analysis, as shown in [42]. In this section, different smoke features are described.

Smoke Color

Smoke is composed of small particles from the completely burned charcoal on the one hand and not completely burned dust on the other [21]. The appearance of the smoke depends on different factors, like the amount of oxygen in the air, the heat of the fire, and so on. The smoke color ranges from white-blueish to white when the smoke temperature itself is low. The color range is more black-grayish to black, when the temperature increases until the material catches fire. Therefore, two color ranges are used as a smoke-feature [21]. Figure 2.22 shows the two different types of smoke - light and dark smoke.

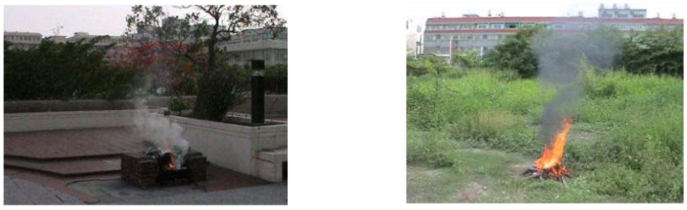


Figure 2.22: Two possible colors of smoke - light gray smoke (left) and dark-gray smoke (right) [21].

Static Analysis - Chroma

The most common smoke shows a wide range of grayish colors, as shown in Figure 2.22. It is mostly light gray or dark gray [21]. In the RGB color space, a gray color means an equal amount of each color, which means that the values of all three color channels (red, green and blue) have to be equal. If the values of each color channel would be perfectly equal that would result in HSI values with $H = 0$ and $S = 0$, so only resulting in different values for I (see Equations (2.10), (2.11), (2.12) and (2.13) for details). In fact, smoke colors extracted from videos are not exactly equal in all three color channels, with an allowed small difference ϵ [21]. So smoke is recognizable through the I channel only, given different thresholds for light or dark gray smoke. As mentioned above, the distribution in the color channels is just nearly perfect, and also a small variation in only one of the three color-channels is sufficient to lead to big changes in the H or the I channel. This means that three rules are necessary to define if a pixel is either smoke colored or not - one rule for each channel. The first rule defines that the three color values are nearly equally distributed, which leads to the constraint that $R \pm \alpha = G \pm \alpha = B \pm \alpha$, where α defines a range of tolerance [45]. The second restriction is that the intensity-value I (from the HSI color space) has to be within two ranges, either inside the *light gray* range or the *dark gray* range as defined in [21]. This leads to two more decision functions $\lambda_{LightGray1} \leq I \leq \lambda_{LightGray2}$ and $\lambda_{DarkGray1} \leq I \leq \lambda_{DarkGray2}$, where $\lambda_{LightGray1}$ and $\lambda_{LDarkGray1}$ are the lower thresholds for the two intensity ranges and $\lambda_{LightGray2}$ and $\lambda_{LDarkGray2}$ define both upper boundaries. In [21], values for the tolerance α and all four thresholds are estimated through evaluating the statistical data provided by their experiments, leading to a value $\alpha = [15; 20]$ and values for the dark gray range from $31\%(=\lambda_{DarkGray1})$ to $59\%(=\lambda_{DarkGray2})$ and a light gray range from $59\%(=\lambda_{LightGray1})$ to $86\%(=\lambda_{LightGray2})$, finally leading to the two intensity ranges as shown in Figure 2.23.

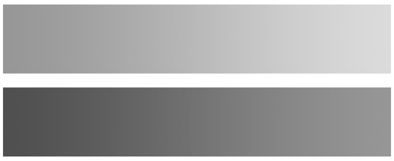


Figure 2.23: The two HSI intensity ranges for light and dark smoke as estimated in [21].

Disorder Measurement

The movement of smoke is called *smoke dynamics* [45]. The main factor for the movement of smoke is simply controlled by the condition of airflows nearby [45]. So the shape of smoke is often changing rapidly over time. The disorder of growth can be used, to distinguish between smoke and smoke colored objects. This disorder can be described as the change of the pixel quantity through time. Using two (through color constrains restricted) smoke images, the difference between those two consecutive frames can be used to measure the degree of change. So if the value

$$\Delta S = \frac{|SD_{t-1} - SD_t|}{SD_t} \tag{2.27}$$

is over a threshold λ_{SD} it is considered to be real smoke. SD_t is calculated by the amount of smoke colored pixels from frame t and frame $t-1$, leading to $SD_t = S_t(x,y) - S_{t-1}(x,y)$. It is noticeable that $S_t(x,y)$ and $S_{t-1}(x,y)$ are the smoke images itself, while SD_t and SD_{t-1} are the particular measurements of the differences between two frames. The value for λ_{SD} has to be estimated through experimental results [45].

Adapted Disorder Measurement

The in Section 'Disorder Measurement' described feature is further improved in [21]. Smoke changes its shape in a nearly random way, so the detection of e.g. a specific shape would not make sense. Hence, a improved disorder measurement in comparison to [45] is used. Instead of measuring the quantity of smoke pixels the authors of [21] measure the enlargement of the area for an extracted smoke region. The proposed decision rule is

$$\Delta S_2 = \frac{\sum_1^n CIR_n}{|SMP|}, \tag{2.28}$$

where CIR_n defines the circumference of the detected smoke region n, and $|SMP|$ is the cardinality of smoke pixels detected (the amount of smoke pixels detected). The value of ΔS_2 can be compared to a disorder threshold λ_{SD2}. If $\Delta S_2 \leq \lambda_{SD2} = true$, then it is considered as a real smoke. However, the variable λ_{SD2} depends on various criteria and should also be estimated though statistical data from experiments [21].

Smoke Growth Rate

The growth rate of smoke is usually larger than the growth rate of the fire itself [21]. If an amount of n_i potential smoke pixels is calculated for the frame t and n_{i+1} potential smoke pixels are counted in frame $t + 1$, a simple form (to see if the smoke-region is growing) is provided by counting and comparing these quantities for h times. So if

$$n_{t+1} > n_t + \alpha, \tag{2.29}$$

where α is a chosen tolerance, is satisfied for h times, a growing behavior of the detected region can be assumed. The value of h can be estimated through experimental results [45].

Adapted Smoke Growth Rate

A simple measurement for the growth rate is proposed in Section 'Smoke Growth Rate' by Equation 2.29. In [21], further improvements to the decision rules and equations are made and the increment of the smoke area is measured by

$$\Delta A_i = \frac{dA}{dt} = \frac{A_{i+k} - A_i}{t_{i+k} - t_i}, \tag{2.30}$$

where A_i is the size of the smoke area at time t_i. If this feature is used in computer vision, the frame number can be used instead of the timestamps t_i and the amount of pixel can be used instead of the area A_i. So Equation 2.30 can be rewritten as

$$\Delta A_i = \frac{dP}{dt} = \frac{P_{i+k} - P_i}{(i+k) - i}, \tag{2.31}$$

where P_i is the cardinality of smoke colored pixel for frame i and therefore ΔA measures the ratio of change for the time interval between frame i and frame $i + k$. To get meaningful results, the average smoke rate is calculated through

$$\Delta \bar{A} = \frac{1}{n} \sum_{i=1}^{n} \Delta A_i, \tag{2.32}$$

where n is the number of considered growth rates. In [21] it is mentioned that exceeding the threshold only once is an insufficient constraint to give reliable results. Therefore, the smoke detection rule $\lambda_{A_{low}} < \Delta \bar{A} < \lambda_{A_{high}}$ has to be true for N_d times to be regarded as real smoke, where $\lambda_{A_{low}}$ and $\lambda_{A_{high}}$ are the low-bound and high-bound thresholds of the growth rate.

Upward Characteristics

In most cases, smoke is continuously ascending through the hot airflow beneath it [23]. Using a motion vector, which is automatically calculated in for example MPEG-x compression (for details the reader is referred to [24]), the direction in which this considered *Macro Block* (MB) is moving can be identified. This direction is calculated through

$$\varphi_{MB} = \arctan\left(\frac{MV_y}{MV_x}\right), \tag{2.33}$$

where MV_x and MV_y define the horizontal and vertical parts of the movement. If the calculated angle φ_{MB} is above a decision threshold $\lambda_{\varphi_{MB}}$, the motion of this block is considered as *upward* and therefore marked as a possible smoke region [23].

Spatial Wavelet Analysis for Smoke

Smoke, contrary to fire, does not generate high spatial disorder inside its region. So, due to the fact that smoke continuously covers areas in the background - and therefore smooths out for example edges [13], the spatial wavelet analysis for smoke measures the decreasing amount of disorder (other than proposed in Section 'Spatial Wavelet Analysis for Fire', where only a high amount of disorder is used as an indicator for fire). Figure 2.24 shows a frame with its corresponding sub-images, each generated by a single-level wavelet transformation. Please note that the contrast of the three sub-images from high-low pass (HL), low-high pass (LH) and high-high pass (HH) have been artificially enhanced to allow a better visual distinction.

All three sub-images (HL, LH and HH) are combined in a single sub-image, using the equation $w_{n(x,y)} = |LH_{n(x,y)}|^2 + |HL_{n(x,y)}|^2 + |HH_{n(x,y)}|^2$ to generate a single value for each pixel for the composite image. In this case, sub-images are computed either from the luminance (Y) image of the video, or considering all three color-channels red, green and blue in the RGB color space (in contrast to the detection of fire, where only the red color-channel is used). After that, the generated sub-image w_n, which contains the detected high frequency information, is divided into blocks of the size K_1, K_2. The energy $e_{n(l_1,l_2)}$ of each block is calculated by

$$e_{n(l_1,l_2)} = \sum_{(x,y)\in R_i} w_n(x + l_1K_1, y + l_2K_2), \tag{2.34}$$

where R_i represents all pixels inside the block of the wavelet sub-image within its size (K_1, K_2). In [13], the block size 8×8 is considered as sufficient.

At the beginning, all local energy values $e_{n(l_1,l_2)}$ are calculated to receive values for the background image, which can be compared to the subsequent frames later on. If the value of $e_{n(l_1,l_2)}$ decreases, compared to the estimated background value $e_{B(l_1,l_2)}$, the edges represented in this region do not appear as sharp as in the background image. This could mean that there is smoke covering those details in the $(l_1, l_2)^{th}$ block.

But if an local extrema in the sub-image appears or disappears instantaneous, this behavior could be due to the existence of a moving object in the scene. In case of smoke inside a block, the composite wavelet value $w_{n(x,y)}$ would decrease in a slowly manner. So for this case, estimating a slow decreasing behavior can be done with two thresholds, where condition $1 > T_1 > T_2 > 0$ must be true. To recognize smoke, the equation

$$1 > T_1 w_{B_n(x,y)} > w_n(x, y) > T_2 w_{B_n(x,y)} > 0 \tag{2.35}$$

holds true, where $w_{B_n(x,y)}$ is the composite wavelet value of the estimated background image and $w_n(x, y)$ is the composite image of the current frame [13].

Figure 2.25 shows a frame that includes smoke, with its corresponding sub-images, each generated by a single-level wavelet transformation. The decrease of the detected edges can be seen, by comparing these three wavelet-transformed sub-images with the wavelet-transformed sub-images from Figure 2.24.

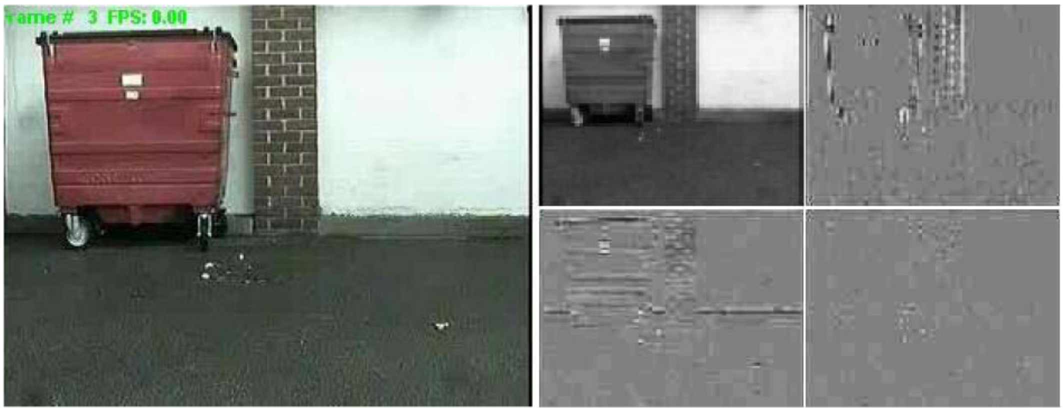


Figure 2.24: Left: the original frame. Right: the sub-images from the wavelet transformations, namely the high-low pass (HL), low-high pass (LH) and high-high pass (HH). (After applying a subsequent contrast enhancement) [13].

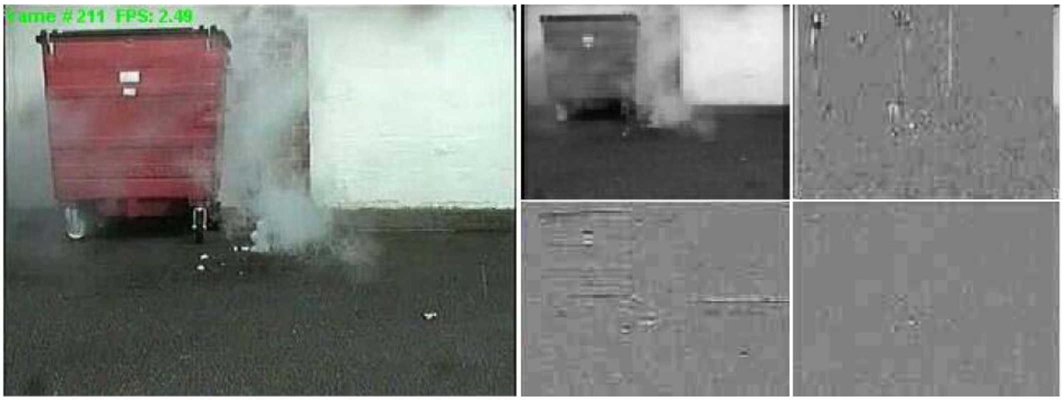


Figure 2.25: Left: the original frame. Right: the sub-images from the wavelet transformations, after a subsequent contrast enhancement. By comparing it to Figure 2.24, the decreasing variety of pixel-values in each sub-image can be recognized [13].

2.1.5 Decision Fusion

Decision Fusion is the part of fire and smoke detection called that merges the outcome of the fire and smoke features to the overall decision if an event (for example a fire) is considered as *detected* or *not detected*. In this section, different decision methods are described.

Decision Tree

A decision-tree covers all possibilities and events that can occur [9]. It is the logical representation of all possible decisions for a given choice. Therefore, decision-nodes are followed until a leave-node is reached. A simple query would lead to for example either *true* or *false* as an answer [9]. Figure 2.26 shows a simple one-sided decision tree, which evaluates if fire is represented by examining three different features.

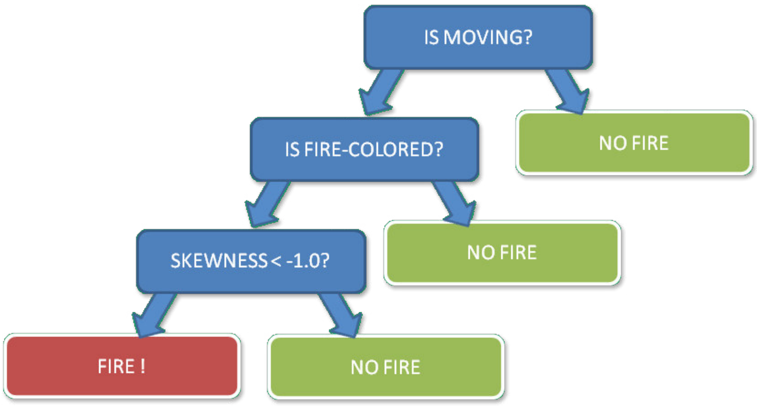


Figure 2.26: A simple decision tree that decides, if fire is represented by using the information provided by three different features. The left arrow of each decision represents *true* while the right arrow represents *false*.

Voting Based

Typical voting systems can be: *unanimity voting*, *majority voting* or *m-out-of-n voting* [16]. In an unanimity voting based system, every feature has to detect e.g. *fire* for a given pixel leading to $pixel_{(x,y,n)}^{feature} = 1$. If every feature recognizes fire at the same pixel, the vote for *fire or not* is unanimous and therefore a fire alarm is given. In majority voting based systems, more than half of the features have to detect fire for a given pixel to raise a fire alarm. In a m-out-of-n voting algorithm a choice is accepted if at least m-features (out of n used features) decide that a fire is detected. An variant of the m-out-of-n algorithm is the T-out-of-v voting method [16]. Here, the output is considered as accepted, if

$$H = \sum_i w_i v_i > T, \tag{2.36}$$

where every w_i is a user defined weight, every v_i is the result of the feature i and T is a user defined threshold. The results of each feature can have the binary values 0 and 1, set accordingly if a fire was detected or not. For example if one used feature is *moving*, a pixel $p_{(x,y)}$ is marked as 1 if a motion at this pixel was detected. The next feature could be the color. If the color of this pixel is within the fire range, as proposed in Section 'Color Range', the value for this feature and this pixel would also be 1, represented by $pixel^{Color}_{(x,y,n)} = 1$. The advantage of this method is that the influence of more unreliable features can be decreased. In a unanimity based voting system, just one single feature that does not detect fire leads to a false negative [16].

Bayes Classifier

A Bayes-Classifier can be used to separate two different classes [5]. Regarding smoke or fire detection, different features are combined to a vector. After that, the Bayes Classifier calculates the variance and the mean of each class. To achieve this, sample images (containing images representing fire - called *positives* - and images that do not contain fire - called *negatives*) are used to train the classifier accordingly [5]. Figure 2.27 shows how a trained decision function separates fire images from non-fire images. Here, the features *Boundary Roughness*, *Normalized Area Change* (Randomness of Area Size) and the *Variance* (Surface Coarseness) are used as features.

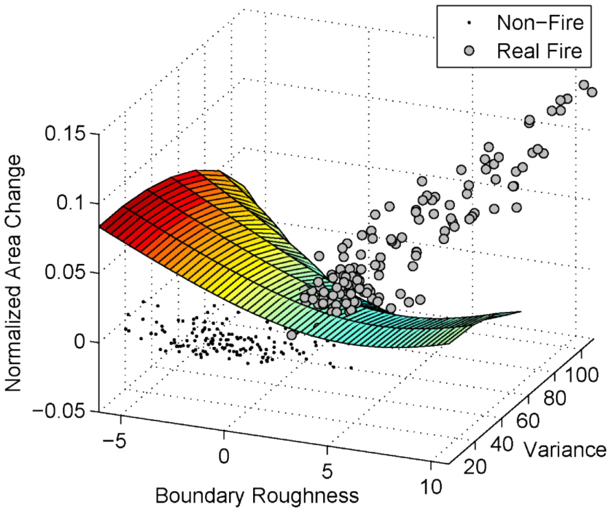


Figure 2.27: Illustration of a decision function to separate images containing fire from images that do not contain fire, calculated by using the three different features Boundary Roughness, Normalized Area Change (Randomness of Area Size) and the Variance (Surface Coarseness) [5].

2.2 Related Work

In this section, selected methods for fire and smoke detection are explained in detail, including the method using the *Time-of-Flight* (TOF) cameras for the detection of fire incidents. This shows how the presence of fire and smoke is detected by different approaches using the features explained before.

2.2.1 Real-Time Fire and Flame Detection

In [16], moving pixels are estimated, using the (adapted) motion estimation method described in Section 'Frame Differencing', where a pixel is considered as *moving*, either if its brightness differs from the brightness of the previous frame, or if its brightness differs from an estimated background (see Equation 2.6 for details). Regions of moving pixels are extracted using a connected component labeling algorithm, as proposed in for example [22]. After that, fire colored pixels are detected. However, in this case a Gaussian Mixture Model is used instead of static thresholds for each color-value. (For details see Section 'Color by Gaussian Mixture Model'). In addition, the temporal variation is calculated using the method described in Section 'Temporal Wavelet Analysis'. If the two estimated wavelet sub-signals show an oscillating behavior, this pixel is further processed. The fourth and last step - to recognize if fire is represented - is described in Section 'Spatial Wavelet Analysis for Fire'. Here, the spatial variation of the detected region is measured. If the pixel-value disorder is high enough, this region is considered as a potential fire region. Note that this last part is a 2D analysis using the height and width of the analyzed region as dimension, while the method described before (in Section 'Temporal Wavelet Analysis') is the analysis of each pixel separately over time. The final decision if a fire is detected is made by the T-out-of-v voting method, described in Section 'Voting Based'. Each described feature estimates for each pixel $p_{(x,y,n)}$ if there is a fire detected or not. This means for example for the third feature - called the zero crossings rate - a resulting value $pixel_{(x,y,n)}^{ZeroCrossings} = 1$, if the amount of zero crossings of both sub-signals $d_{n(x,y)}$ and $e_{n(x,y)}$ are above a threshold, and 0 otherwise. The proposed method was tested with 61 different videos, containing a total of 83.745 frames. In 19 of the scenes fire occurs. The method was able to detect fire in all scenes, leading to a scene-detection-rate of 1.0. Only 9 frames of one sequence have been wrongly detected as frames containing fire, leading to a false-positive rate of 0.001 [16]. Figure 2.28 shows the result extracted out of two frames from different scenes. On the top, each considered frame can be seen, on the bottom, the same frame is shown, with possible fire regions marked in green.

The proposed method provides good results, but works in real-time only when using videos with 10fps. The detection of fire-colored object works well - in a video containing a person dancing with a fire-colored T-Shirt, only 9 frames have been wrongly detected as fire, while comparable methods were leading to 107 and 86 wrong detected frames [16]. However, the used videos had only a frame size of 320×240 $pixel$ captured at 10 $frames$ per second. The average processing time per frame was $16.5ms$. With higher resolutions or frame-rates the processing time would probably increase to a non real-time amount.

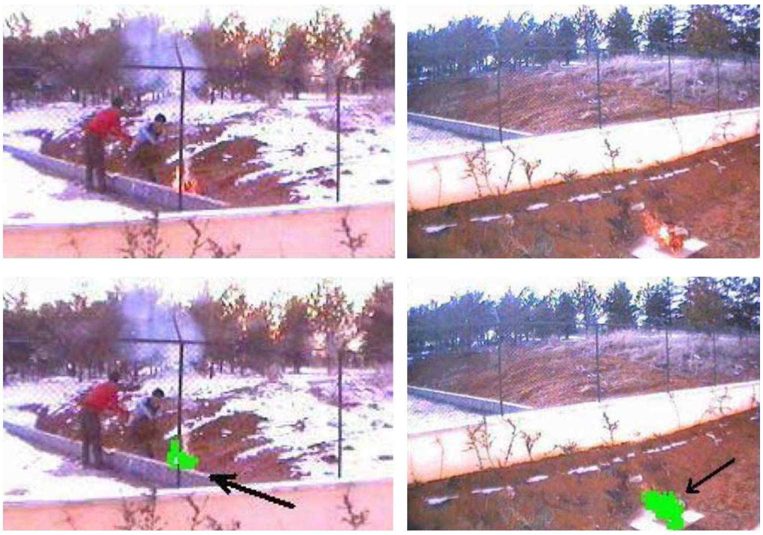


Figure 2.28: On the top: sample frames from two different movies. Bottom: the same frames processed with the proposed method, where all detected fire pixel have been highlighted in green [16].

2.2.2 Fire Detection in Newscast Videos

The method proposed in [5] aims on the automated classification of newscast content. One special characteristic of this technique is that it does not try to identify single fire pixels in a given video frame, just if a fire is represented in the overall frame or not. Another one is that - due to the fact that newscast-videos often rely on handheld cameras [5] - the field of view of the camera can change during a video-shot (when the camera is rotated, for example). The essential features of fire recognition in [5] are color, motion and the geometry of the fire. The color of fire is considered to be the strongest single feature to detect a fire incident in video frames or pictures [5]. The color range for hydrocarbon flames, which are the most common types of flames, reach from red to yellow. Color ranges for other flame-types, like blue petroleum gas flames, are not considered in [5], because the content analysis in newscast-videos aims on detecting fire catastrophes, for example. The first color-restriction is the same as described in Section 'Color Based Detection Metric', where the values of the red, green and blue-color channel are compared to each other. The second color-feature is calculated by sample pictures to obtain useful values. For a fire region, as marked in Figure 2.3, the average values and the variances for the red, green and blue channels are calculated. With these values three Probability Density Functions (PDFs) are created - one for each channel. An observed pixel is then considered to be a potential fire pixel, if the combined values of each PDF are exceeding a given threshold. (Note: The PDF is a Gaussian Model for each color-channel, created by using sample fire pixels from images [5] and has its maximum value if the value of an analyzed pixel has the same value as the average value calculated from these sample pictures.) Those two color characteristics are applied on each frame - every pixel that fulfills these requirements is saved to the Potential Fire Mask (PFM).

The pixels marked in the PFM are then analyzed further with the other fire features. Due to the flickering behavior of fire, its size changes over time. Therefore, the normalized change rate of the fire size is calculated as the second feature, as described in Section 'Randomness of Area Size'. Because the shape of fire changes randomly, another feature of interest is the roughness or randomness of the shape of a fire region. Therefore, in [5] a roughness descriptor B_R is defined as the ratio between the perimeter of the fire region in the PFM and the smallest convex hull surrounding this region, as described in Section 'Boundary Roughness'. Yellow or orange objects, like traffic signs for example, can produce wrong detected fire candidate areas. Because fire has a significant amount of different pixel color values, the variability of the pixel values inside fire-regions can be used as a discriminant. Therefore, the variance is calculated for potential fire regions, as described in Section 'Surface Coarseness'. Fire regions in frames show a high saturation in the red color channel. Therefore, calculating the skewness of the pixels in a fire-area leads to a negative value. As a feature, fire is considered detected if the value for the skewness in the red channel of the examined pixels is below a decision threshold, as described in Section 'Skewness' [5]. In *human recorded* newscast-videos, fire is the most important part of the scene. Therefore, the fire area is centered in the video shot, as described in Section 'Spatial Distribution of Fire'. With this knowledge, a weighting function for the horizontal and vertical position of a pixel is defined. If the result of this function is below a decision threshold for a given pixel, this pixel is set to 0 (in the PFM created by the feature 'Color Based Detection Metric') and therefore no longer considered as potential a fire-pixel in [5]. For each frame a PFM is created by the rules described in Section 'Color Based Detection Metric' and refined by the rules described in Section 'Spatial Distribution of Fire'. In addition, a vector d is provided by the remaining features, namely 'Randomness of Area Size', 'Boundary Roughness', 'Surface Coarseness' and 'Skewness'. With these values a Bayes classifier is trained, using manually selected frames containing fire (positives) or not containing fire (negatives), as described in Section 'Bayes Classifier'. If this trained Bayes classifier is used - to determine if fire is represented in a video frame or not - the naive thresholds from each feature are not applied as a decision rule anymore, only the overall outcome of the Bayes classifier [5]. To evaluate the proposed method, a database containing different newscast-videos has been used in [5]. It contains shots with wild-land and residential fire and burning buildings. Also, records of objects that have a likewise appearance as fire (like sunsets) were used to evaluate this fire detection method. The final database for testing contains $798,000$ frames and each frame has been classified as *contains fire* or *does not contain fire*. The overall method, using all features combined, achieves an average false-positive rate (fire is wrongly detected in a frame) of 0.68% and a false-negative rate (fire is not detected in a frame) of 0.028% [5].

This method provides useful features. The detection rate achieves useful values, even due to the fact that camera motion is allowed in this method.

2.2.3 Fire Detection through Time-of-Flight Imaging

The method proposed in [29] is of special interest, because the basic idea is to merge fire detection from two different branches; from the two-dimensional video signal and the three-dimensional image received from a *Time-of-Flight* (TOF) camera (see Section 'Fire Feature Extraction using Time-of-Flight Cameras' for details). The proposed method uses the amplitude image and the RGB video signal only - the additional available 3D depth map is considered as unreliable in [29] and therefore not used. Figure 2.29 shows the depth map, the amplitude image and the RGB image captured by a TOF-camera. The idea behind using two different sensors is that mis-detections in the video analysis can be corrected through the additional analysis of the amplitude image (and vice-versa). The analysis of the visual information and the amplitude image happens parallel, as shown in Figure 2.30.

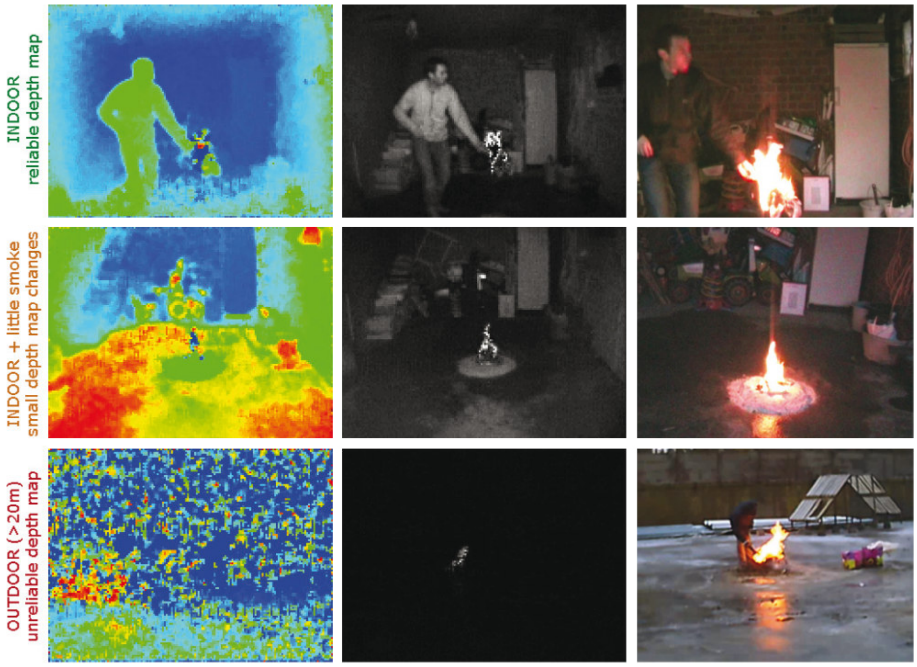


Figure 2.29: From right to left: depth map, amplitude image and RGB video frame of the same scene. The pictures on top are showing an indoor situation, the middle row shows a indoor situation where smoke has already begun to arise. The bottom row shows a outdoor situation. Explanatory notes are made on the left side of each row [29].

The visual part is implemented as a fire detector with low computational costs [29]. First, moving regions are estimated through 'Background Subtraction'. Then, a morphological opening is performed to remove detected noise. After that, the used features are *Spatial Flame Color Disorder* (similar to the approach described in Section 'Color Range'), *Principal Orientation Disorder* (as described in [18]) and *Bounding Box Disorder* (similar to the method from Section 'Randomness of Area Size'). The result is a binary image, where detected fire regions are labeled by $Flames_n^{visual} = 1$, where n represents the frame-number. The amplitude image on the other hand uses other features to detect possible fire candidates. Fire itself leads to fast changing parts in the amplitude image. So, the amount of disorder is measured over three frames consecutively, as described in Section 'Amplitude Disorder Detection'. Through rounding the calculated value, as shown in Equation 2.24, it ensures that areas with fast changing amplitude

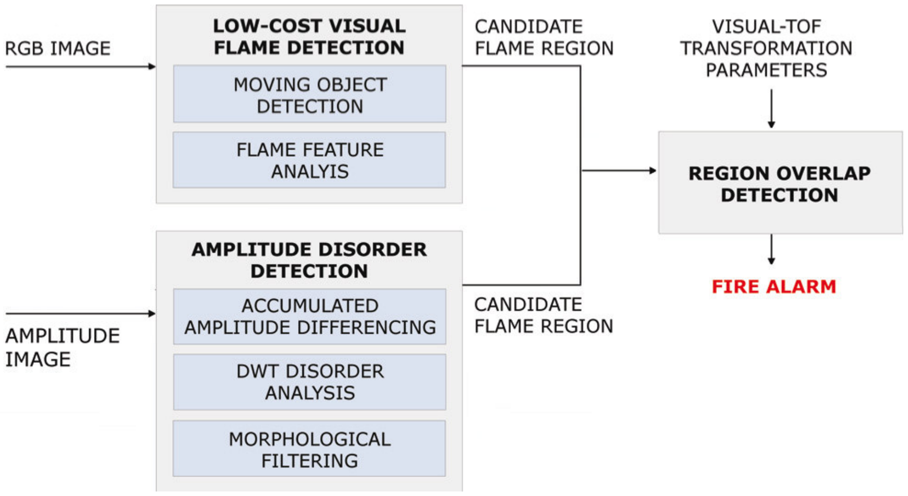


Figure 2.30: General scheme of the parallel fire detection, executed on the visual image and the amplitude image [29].

images get a value < 0 for their corresponding pixel. The feature used next is described in Section 'Discrete Wavelet Transformation of Amplitude Images'. Here, areas that lead to a high contrast in the image (caused through for example borders from an object) lead to a peak inside the horizontal, vertical or diagonal *Discrete Wavelet Transformed* (DWT) sub-image. As described before, fire leads to a high disordered region inside the amplitude image, and this can be seen in the transformed sub-image. A binary image, showing possible fire candidates recognized through wavelet analysis, is created using Equation 2.25 [29]. The final binary image, for the fire detection through amplitude image, is build by the decision rule

$$Flames_n^{amplitude} = \begin{cases} 1, & \text{if } AFD_n^{amp} > 0 \text{ AND } DWT = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{2.37}$$

Here, AFD_n^{amp} is the accumulated amplitude-difference, calculated out of three frames, while $DWT = 1$ if the horizontal, vertical and diagonal discrete wavelet transformed values exceed a decision-threshold. For the last part, regions containing both, a detected fire in the visual and in the amplitude image, have to be calculated. As described in Section 'Image Registration', both binary images have to be registered to do this. The visual detector provides a binary map $Flames_n^{visual}$, where all fire candidate pixel are marked as 1. In addition there is the binary image called $Flames_n^{amplitude}$ for the amplitude-image detection algorithm. To detect the region overlap, a logical *AND* operation is performed. If this merged image contains one or more pixel with the value 1, a fire alarm is given (details for the fusion are described in Section 'Multi-Sensor Fusion'). The experimental results show a fire detection rate between 89% and 95%, using the evaluation metric

$$DetectionRate = \frac{\#DetectedFrames - \#FalseDetectedFrames}{\#GTFireFrames}, \tag{2.38}$$

where the number of ground-truth frames ($\#GTFireFrames$) has been manually counted. One relevant fact of the experimental results is that there is no false-positive detection in any of the four analyzed videos. One explanation for missed frames that are containing fire is that the resolution of the used TOF camera is rather small, leading to miss for example a fire at the beginning because it's simply to small to be recognized [29].

This method uses the amplitude image only, which shows the intensity of a received infrared light. Fire itself emits IR light, so it is possible to detect flames even outside of the operating range of the TOF camera. However, therefore the depth map is not used. Because of the IR emitting behavior of flames, it would also be interesting to turn of the active IR illumination of the TOF camera to see if fire could be detected this way. This would eliminate the problem with artifacts on the boundaries of fast moving (non-flame) objects and decrease the power consumption.

2.2.4 Early Smoke Detection Algorithm

The first step done for smoke detection in [21] is a segmentation that extracts all moving regions through frame-differencing, as described in Section 'Moving Pixel Detection'. After that, smoke-colored pixel are extracted through chromatic feature analysis, as described in Section 'Static Analysis - Chroma'. After checking two dynamic features, the disorder as proposed in Section 'Adapted Disorder Measurement' and the growth rate from Section 'Adapted Smoke Growth Rate', a smoke alarm is given or not. Figure 2.31 shows a the masked frame showing only pixels that fulfill these restrictions. The source frames contain different fuel types, the fire on the upper picture shows the light grayish smoke generated by burning papers, the lower picture shows the dark grayish smoke from burning wood. Figure 2.32 shows the flow chart for this form of smoke detection. The decision if smoke is detected is made by evaluating the two dynamic features: the growth rate and the disorder. The experimental results in [21] are obtained by using a single video. A person with grayish-colored clothes moves through the scene. After using the features *motion detection* and *chromatic features* (see Figure 2.32) this person is detected as a potential smoke region. After the dynamic analysis these false classification disappears. Figure 2.33 shows the single stages of this method; the top row shows three different frames from the original video, the middle row shows the same frame after the *motion detection* and *static color analysis*. The bottom row shows the revealed smoke regions, after executing the whole algorithm. On the bottom the number of recognized smoke pixels per frame is visualized, showing quite well the typical expanding behavior the smoke. This approach shows a useful way to detect smoke. Because non-fire-candidate pixels are reduced early in the stage (only moving pixels with a gray color are considered for further analysis) it is also useful for a real-time analysis.

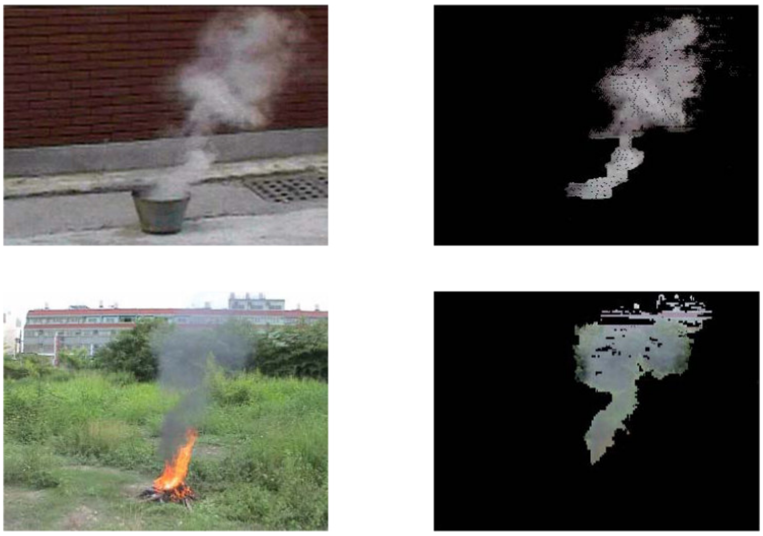


Figure 2.31: Top, left: Smoke generated by burning papers. Right: the same frame masked by the corresponding binary image. Bottom: Smoke generated by burning wood and its corresponding masked frame [21].

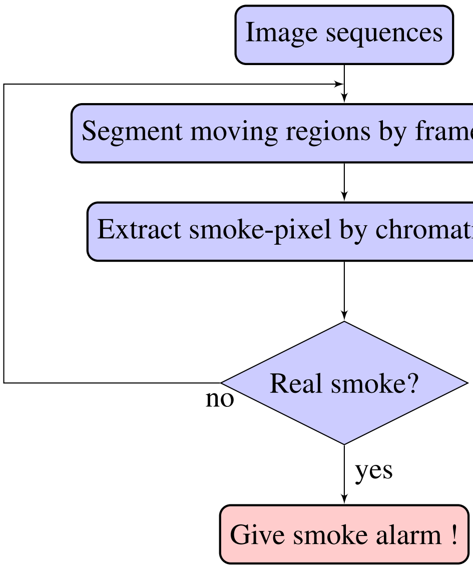


Image sequences
Segment moving regions by frame difference
Extract smoke-pixel by chromatic features
Real smoke?
no
yes
Give smoke alarm !

Figure 2.32: Flow chart of the smoke detection as described in [21]. *Real smoke?* is checked by the two dynamic features.

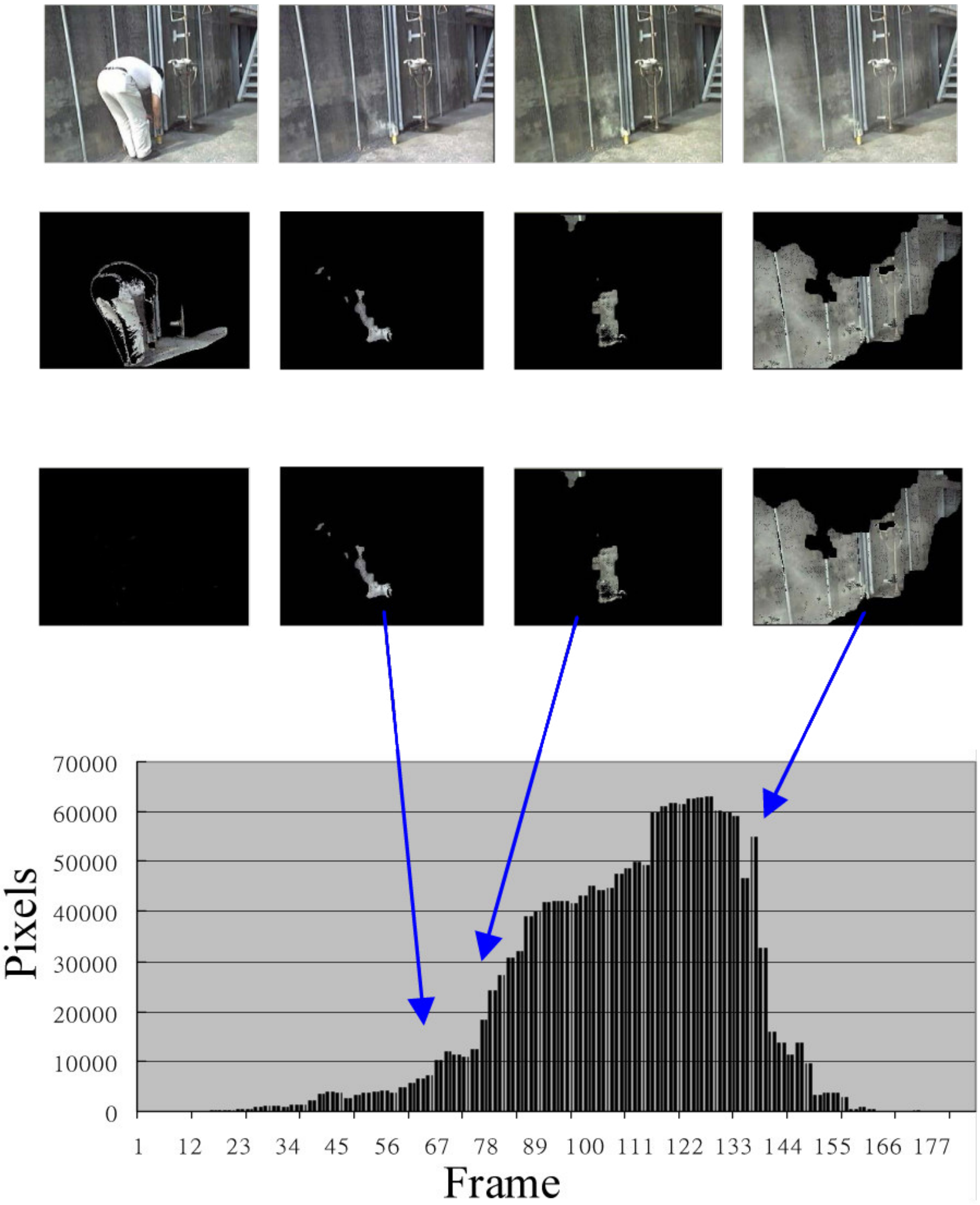


Figure 2.33: Top row: original frames from the video. Middle row: the same images after the *static analysis*. Bottom row: the final result of the smoke detection algorithm. Bottom picture: a histogram showing the amount of detected smoke pixels per frame. The blue arrows indicate the position of the sample images shown. The overall picture shows the typical expanding behavior the smoke [21].

2.2.5 Vision Based Smoke Detection Algorithm

In [23], fire is only indirectly detected through recognizing smoke incidents. The proposed method of detection is only suitable for videos, which have been coded in MPEG-x or H.264/AVC format. The reason for that is that this algorithm needs an calculated *Motion Vector* (MV) for each *Macro Block* (MB), which is intrinsic for both codecs [23]. First of all, all regions with a MV smaller than a threshold are discarded, leaving only areas where motion was detected as described in Section 'Movement Detection using Motion Vectors'. Second, there are chromatic restrictions, as in Section 'Static Analysis - Chroma', to limit these valid areas further. Least of all, a MB is only considered as *containing smoke*, if the last rule, as described in Section 'Upward Characteristics', holds true. Figure 2.34 shows corresponding flow chart for smoke detection using the method from [23]. Each decision step of this pipeline is shown in Figure 2.35, the first picture shows the frame that is evaluated. After applying the constrains - regarding the length of the MV from the corresponding block - the second picture is constructed. It can be clearly seen that also *non smoke blocks*, for example calculated for the trash bin, are marked as *valid*. The third picture includes the chroma-decision, where only blocks within a specific color range are further evaluated. The last picture shows the final output of this method, including the last decision parameter that evaluates the direction of the movement of each valid block.

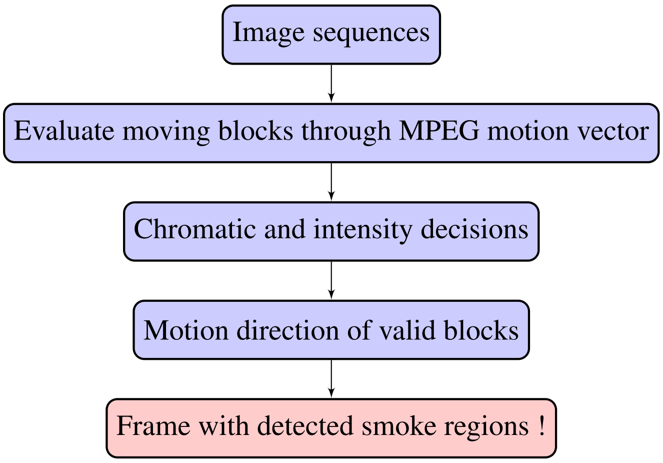


Image sequences
Evaluate moving blocks through MPEG motion vector
Chromatic and intensity decisions
Motion direction of valid blocks
Frame with detected smoke regions !

Figure 2.34: Flow chart of the smoke detection as described in [23].

This paper proposes an innovative and fast method to estimate moving regions and their direction in videos. Problems occur if the smoke direction is affected by airflow, leading to moving directions other than upward.

2.2.6 Forest Fire Smoke Detection in Video

In [39], a method to detect smoke in large areas is proposed. The ambition is to detect forest fires automatically through stationary and static cameras, placed on elevated areas such as artificial watch-posts. Firstly, when the camera is turned on, a background image is saved, representing

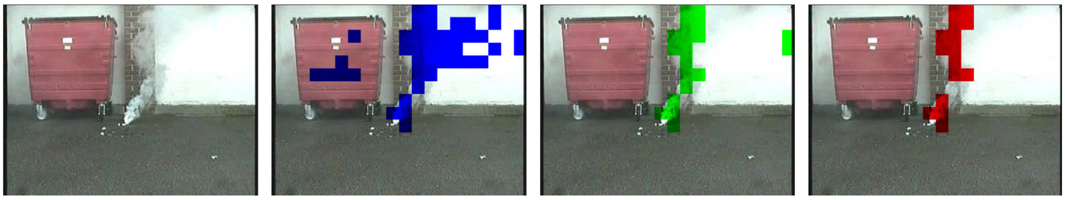


Figure 2.35: From left to right: Original frame, frame after motion detection, frame after chroma decision, final frame after motion direction estimation [21].

the scenery without smoke. After that, moving areas are extracted using the approach described in Section 'Background Subtraction'. To get *Regions of Interest* (ROIs), first erode and dilate are used to get rid of image noise and to build larger clusters of smoke. In [39], for a video resolution of 640x480 a filter size of 8×8 is used. Next, the number of segments are estimated, using a connected component algorithm such as the one proposed in [22]. After that, the changing regions are estimated by their *minimal rectangle area*, using the maximum values of each detected pixel $max(x)$ to estimate the position of the right border and $max(y)$ to estimate the value for the upper border of the rectangle. Respectively, the values for the left and bottom border are calculated using the $min(x)$ and $min(y)$ values of all pixel that have been detected as *moving*. Also, a convex hull algorithm (like the *gift wrapping algorithm* [1]) is used, which usually defines the area of change more exactly than a rectangle could. After that, the frame is divided in N subregions, called ROIs. In [39] a 4×4 grid is used. A ROI is considered as changed, if at least 25% of its area consist of gray pixels, estimated through a color estimation technique similar as the one described in Section 'Static Analysis - Chroma', and each changed ROI is marked as a moving region. If one of the N regions is marked as *changed* for at least 30 frames in a row, a fire alarm is given. If a region was marked as *changed*, but the counter has not reached 30, it is reset to 0 again. Figure 2.36 shows this process as a flow chart. In addition, Figure 2.37 shows an experimental result where smoke was detected. Even smoke that existed from the beginning (=frame 0) was detected. The images are showing four different timestamps with the calculated smoke areas, where the first frame 0 was used as the background-image $imBG$.

One problem is that the background image is only estimated at the beginning of the recording - it is not adapted over time. So, after some time, even clouds lead to a false alarm regions. Another characteristic (that leads to problems when using this approach in for example a $24/7$ application) is that smoke is not detected without sunlight. Also, the detection of the size of the *minimum bounding rectangle* and the *convex hull* of each moving segment (as shown in Section 'Experimental Results' in [39]) lacks meaningful results, because even the static feature, namely if the moving region has a grayish color, is applied afterward. So every tree that is e.g. moving in the wind, is counted as a detection. However, dividing the image in different regions and applying the constraint that a smoke alarm is only given, after smoke has been detected for at least 30 consecutive frames ($= 1$ second) is a good approach to limit the rate of false-positives.

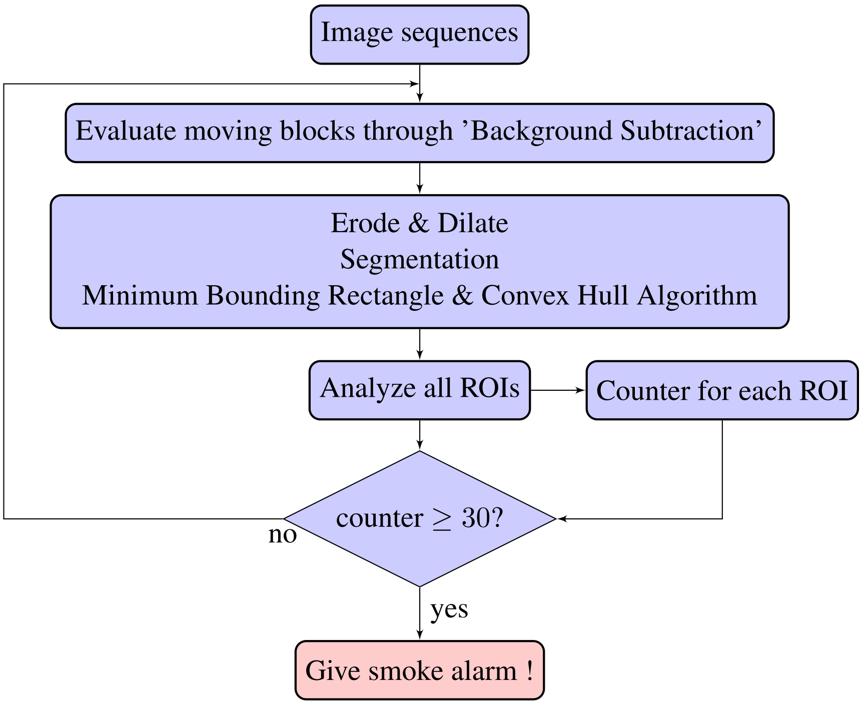


Figure 2.36: Flow chart of the smoke detection as described in [39].

2.2.7 Summary

The basic approach of all fire and smoke detection methods is the same: first, the complexity is reduced by executing restrictions like color or movement. After that, features are calculated. With these results, the final decision is made. It is noteworthy that the same type of feature can be extracted in different ways: e.g. the variety (inherit of fire regions) can be measured by calculating the variance (called *Surface Coarseness* [5]) or by transferring the image to the wavelet domain and executing the *Spatial Wavelet Analysis* [16]. Another case in point are the different types of color restrictions [5] [7] [16] or the different ways to detect the flickering behavior of fire, e.g. through the *Temporal Wavelet Analysis* [16], the *Randomness of Area Size* [5] or the *Bounding Box Disorder* [29].

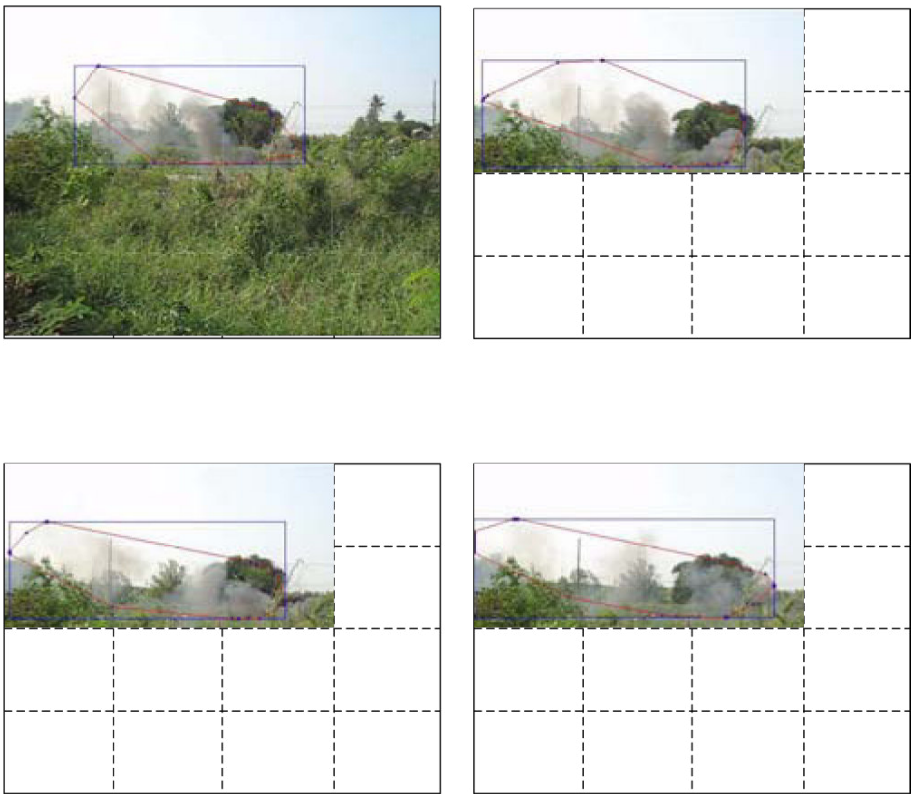


Figure 2.37: Clockwise, starting with the top-left picture: the results of the smoke-detection after one second, after 20 seconds, after 50 seconds and after 90 seconds [39].

CHAPTER 3

Methodology

3.1 Hardware Requirements

In this section, the 3D sensor used - namely the ASUS Xtion Pro - is described. After that, the behavior of this sensor regarding e.g. its image registration, its frame synchronization or the impact of recording fire and smoke is analyzed further. Firstly, the working principle how depth is estimated by the ASUS Xtion Pro is explained.

Both, the ASUS Xtion Pro and the Microsoft Kinect are containing two different sensors: A combination of a RGB and a depth-sensor. The depth estimation method of both sensors is identical [15] and therefore the working principle of the ASUS Xtion is the same as the principle for the Microsoft Kinect and vice versa. The Microsoft Kinect was initially developed as an contactless input-device for the XBOX game console, but was soon used for other areas like computer-vision and motion-detection, for example. Figure 3.1 shows the Microsoft Kinect and its components unfolded when removing the cover. The Infrared (IR)-Sensor sends out IR-dots

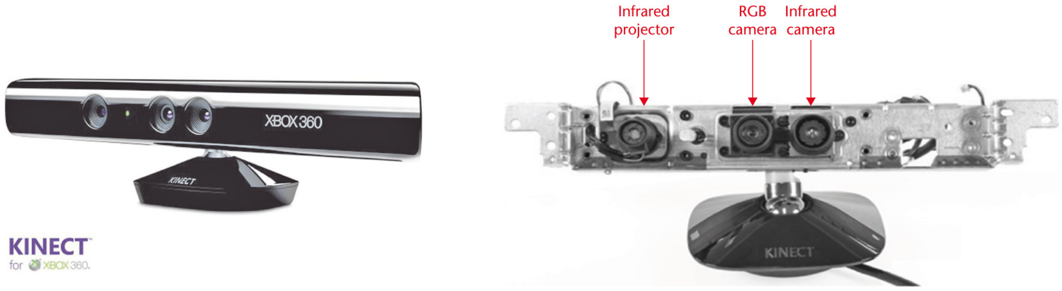


Figure 3.1: Left: the Microsoft Kinect Sensor. Right: the same sensor without its cover, revealing the emitting IR-projector and the two transceiving sensors: one for the RGB color-information and one receiving the infrared-light / the emitted pattern [51].

with a known dot pattern. Due to the fact that the IR-pattern and the geometry between the IR emitter and the IR transceiver is known, the depths of the illuminated scene are calculated by searching and matching local dot patterns with subsequent triangulation [51]. This form of depth-estimation of a scene is called the *Structured-Light Technique* [47]. Figure 3.2 shows the emitted IR-pattern of a Kinect, recorded by a IR-sensitive camera. In Figure 3.3 the principle of the depth-calculation by triangulation and the change of the size of the pattern - depending on the distance to the sensor - is shown. The distance of the detected local pattern influences the position of its projection on the *Imaging plane* and allows therefore the calculation of the distance between the object that reflects the emitted (local) pattern and the sensor. Figure 3.4 shows the corresponding 3D image to Figure 3.2. The brighter a gray value appears the nearer a pixel is located to the camera. For pixels that are *black*, no valid depth could be calculated. This can happen through e.g. objects that are positioned outside the range-limit of the Kinect (the recommended range-limit for the depth sensor is $0.8m - 3.5m$ distance [47]) or they do not reflect an appropriate amount of IR-light [51].

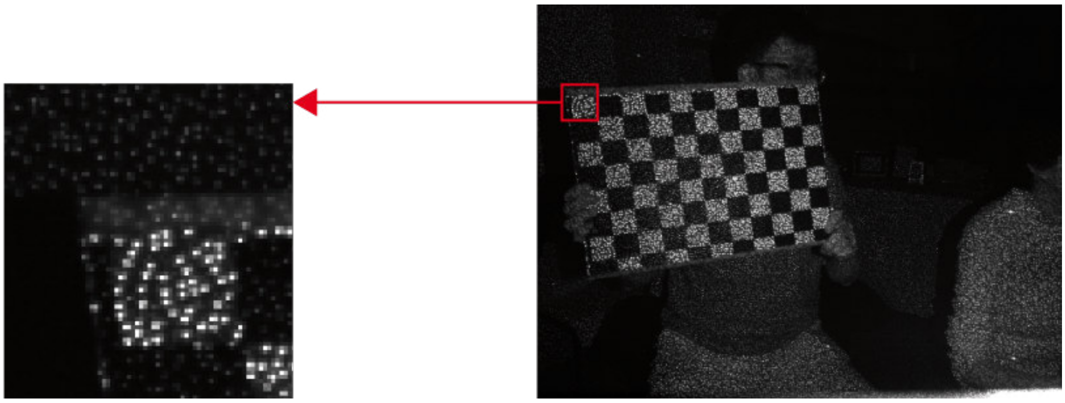


Figure 3.2: The emitted infrared-pattern of a Microsoft Kinect. The red boxed-area is magnified and shown on the left side [51].

3.1.1 Image Registration

Image registration is used to align two or more images of the same scene [2]. Reasons why pictures from the same scene differ can be manifold, they could have been taken to different times, under changing lighting conditions, from different sensors or from different positions. The registration itself is the determination of an image transformation to align one image (reference image) with one or more other images (object images). If points (pixels) from view (called space X) should be mapped to the points of an other view (called space Y), we need a transformation T for each point x from X.

$$x' = T(x) \tag{3.1}$$

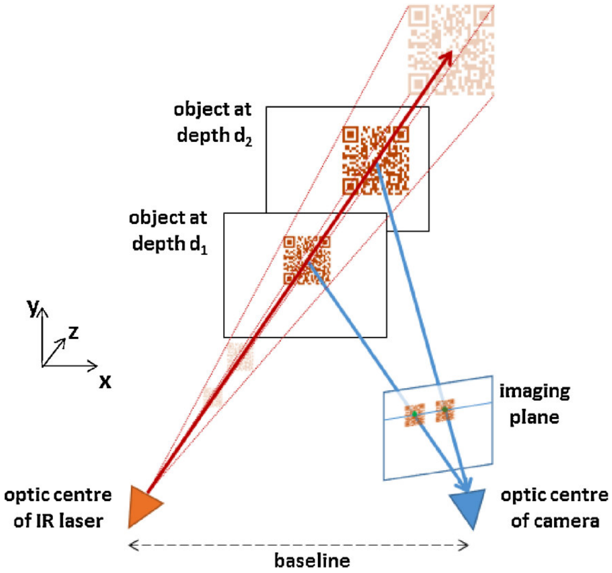


Figure 3.3: The change in the patter-size depends on the distance to the emitter. The distance to the object is calculated by triangulation. The influence of the different depths on the Imaging plane is illustrated by blue rays [47].

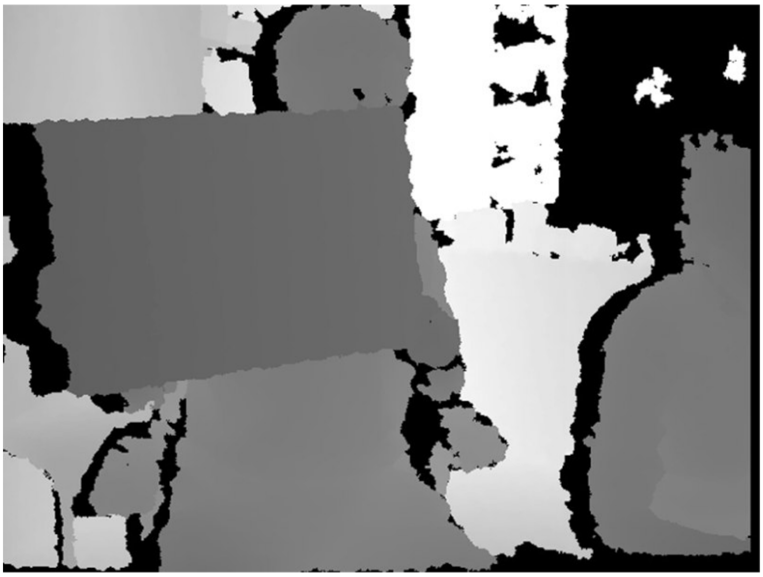


Figure 3.4: The corresponding depth image to Figure 3.2. The depth of the scene is calculated by using the *Structured-Light Technique* [51].

So, the transformation T of a point x generates the transformed point x'. The registration is successful if x' is equal to the corresponding point y from space Y. An unbalanced displacement leads to a nonzero value for $T(x) - y$. This nonzero value is called *registration error* [38]. If the same object is represented in two different images, but on different positions inside each image, a rigid image registration can be used. Here, the matching of those two images is achieved by minimizing or maximizing a mutual function for both images to determine the corresponding rotation and translation parameters. As an example, the *sum of squared differences* can be used as a weighting function. A match is found if this function is at its minimum [2]. This method works well, when images are used that are provided by the same sensor (for example the same scanner) and is called *intra-modality*- or simply *modality*-registration. The alignment of images from different sources is called *multi-modality*- or *inter-modality*-registration. If the matching by the *sum of squared differences* does not work, manual identified landmarks can be used to calculate the needed parameters [2].

So, if two or more sensors are used to detect an event, all sensors must be aligned to the same area or objects. In this case, both sensors are recording frames, so overlapping these frames would lead to the effect that the event from sensor 1 $E_{Sens1_{(x,y)}}$ should also be triggered on the same position at sensor 2, as event $E_{Sens2_{(x,y)}}$. When both sensors are aligned, it is possible to detect events through for example logical operations, as described in Section 'Multi-Sensor Fusion'. Figure 3.5 shows two alligned frames recorded by each sensor of the ASUS Xtion Pro as an superimposed image.

3.1.2 Time Difference between RGB and Depth Frames

When comparing or merging the results from multiple sensors, it is important that the time difference between the measurements of each sensor is as small as possible. In a worst case scenario, one sensor could recognize an specific event at frame n. If there is a time-shift between both sensors, looking at for example the same frame number of another sensor could lead to a non-detection of this event, because this frame was captured after or even before the other frame. So in this case, the same event already happened for example on a frame that was recorded before. Here, it should be evaluated how big the time gap between a recorded color frame and the depth frame is. About 60 seconds have been recorded using an ASUS Xtion Pro. After that, the time differences between the frame numbers - provided by ASUS Xtion Pro - and the time differences between the manually counted frames are analyzed - first thing that was recognized during this analysis is that when comparing a frame that got its frame number assigned by the Xtion (for example a depth-frame with the assigned frame-number 1816) with the manually counted frame-numbers only 1809 frames can be found. So, the ASUS Xtion Pro drops frames itself when the time-difference exceeds a decision threshold (for example a depth frame with the number 1599 can not be found). Table 3.1 shows selected depth-frames from the analyzed video. In this case, frame drops only occurred at the depth sensor, meaning that no frame of the color sensor has been dropped. So for using color frames the equation $frame[assigned] = frame[counted]$ holds true. This leads to the behavior that when comparing a depth frame with the frame-number n and a color frame with the same frame-number there can be a bigger time gap than between the same depth frame and a color frame with a different value (meaning that the frames with the same number do not necessarily need to be the best match for each

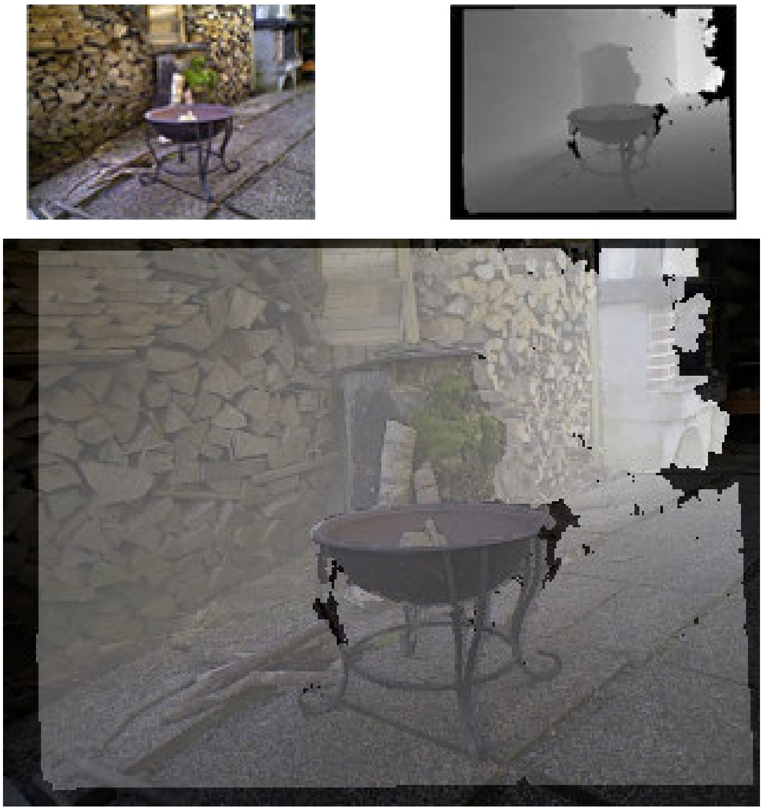


Figure 3.5: Two frames captured with the ASUS Xtion Pro. Image-registration was enabled while recording. Top-row, left: The captured RGB-frame. Top-row, right: The captured Depth-frame. Bottom-row: The superimposed image of RGB- and depth-camera. It can be seen that for example all edges are correctly arranged on top of each other.

other). This behavior can be seen in Table 3.2, where selected frames are represented. If we now compare the timestamps of the (assigned) frame-number 1816 for depth and color frames (leading to a counted frame number of 1809 and 1816, respectively), the resulting time difference would be $|60,565,189\mu s - 60,831,540\mu s| = 0.266351s$, which is a gap of nearly one-third of a second or $8frames$, respectively (when recording with $30fps$). But comparing the two (counted) frame-numbers 1816 - for the depth frame (with the assigned number 1823) - and 1816 for the color frame (with the assigned number 1816), the resulting value of the time difference is $|60,798,773\mu s - 60,831,540\mu s| = 0.032767s$, which is significantly smaller than the time difference between two frames with the same assigned number (and less than $1frame$ using $30fps$). Table 3.3 shows the results of both comparisons. To evaluate why and when such a frame drop occurs, the time differences of all frames used in the record (with a duration of 60 seconds) are analyzed. Here, depth and color frames with the same (manually counted) frame number are compared. The difference between the two frames (depth and color) over time can be

Table 3.1: Analysis of depth-frames provided by the ASUS Xtion PRO (including the drop of frame number 1599). Column Nr[assigned] represents the frame-number given by the ASUS Xtion Pro. Nr[counted] represents the real frame-number, without gaps resulting from frame-drops. The timestamp represents the time when this frame was captured.

Type	**Nr[assigned]**	**Nr[counted]**	**Timestamp[μs])**
DEPTH	1597	1591	53,257,323
DEPTH	1598	1592	53,290,692
DEPTH	1600	1593	53,357,431
DEPTH	1601	1594	53,390,800

Table 3.2: Selected frames to analyze the behavior of the ASUS Xtion PRO. Frame-type describes if this frame was captured by the RGB camera or the depth camera. Nr[assigned] represents the frame-number given by the ASUS Xtion Pro. Nr[counted] represents the real frame-number, without gaps resulting from frame-drops. The timestamp represents the time when this frame was captured. It can be seen that frames from provided by the color-sensor and frames provided by the depth-sensor with the same frame-number, do not necessarily match best when compared to the timestamps when they have been recorded.

Frame Type	**Nr[assigned]**	**Nr[counted]**	**Timestamp[μs])**
COLOR	1809	1809	60,596,928
DEPTH	1816	1809	60,565,189
...	...	...	...
COLOR	1816	1816	60,831,540
DEPTH	1823	1816	60,798,773

Table 3.3: Resulting time gaps using either the same frame numbers as assigned by the ASUS Xtion PRO, or the manually counted frame numbers that compensate dropped frames.

Difference:	**Seconds**	**Frames**
assigned by ASUS Xtion	0.266351s	≈ 1 frame
manually counted	0.032767s	≈ 8 frames

seen in Figure 3.6. As a conclusion, it can be seen that comparing frames with the same frame-number (assigned by the device itself) can lead to inaccurate measurements. In fact, the time difference between a depth frame and a color frame (with the same assigned frame numbers) is linear increasing over time, from $0ms$ at frame number 0 to approximately 300 ms at frame 2000 using a shutter speed of $30fps$. Comparing frames with the same manually counted frame numbers lead to a better result. But also here, the time difference between two consecutive pairs of depth and color frames is increasing over time - until a drop of a depth frame occurs, as shown in Figure 3.6. But here the maximum gap is limited to approximately 1 frame.

It is noteworthy that triggering a frame drop - if the time difference reaches half the frame rate ($\approx 16,5ms$) - would lead to a more precise result, because the maximum value for the time difference between a pair of depth and color frame would be reduced from $+33ms$ to $\pm 16,5ms$.

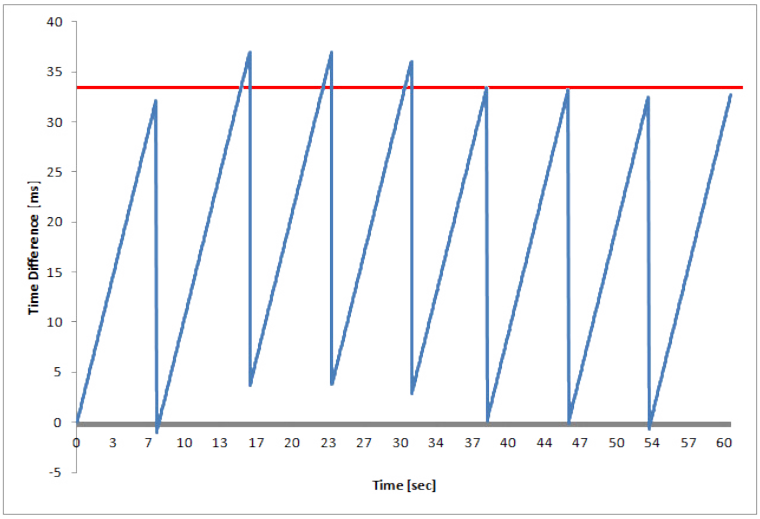


Figure 3.6: Time difference of color and depth frames from a record with a duration of 60 seconds. It can be seen that the time-difference between the frames from the two sensors is growing over time. Here, the depth frames are captured faster than the color frames - leading to this increasing time difference. Whenever the time-difference is near the time difference between two consecutive frames (this would be $\approx 33ms$ using a capture rate of $30fps$, indicated by the red line) one of the *faster* depth frames is dropped to reduce the gap between the frames of the two sensors. These drops can be seen by the vertical lines in the graph, leading to a time-difference value around $0ms$.

3.1.3 Fire Color Range

The color restrictions used for most videos, as described in Section 'Colorhistogram' and Section 'Color Range', do not apply for the ASUS Xtion Pro. In fact, fire is recorded as *nearly white* pixel. Figure 3.7 shows a frame captured by the ASUS Xtion Pro and a picture from the same scene taken with a Samsung Galaxy S3. It can be seen that the ASUS Xtion Pro captures fire with less spatial variability than the Samsung Galaxy S3, for example. Even when disabling the *Auto-White-Balance* from the ASUS Xtion Pro, the spatial variation of the fire pixel colors still remains very low. Figure 3.8 shows the image of a fire taken with disabled Auto-White-Balance. Due to this reason, the color-values as described in Section 'Color Range' have to be adapted to detect fire using the RGB camera of the ASUS Xtion Pro. The color restrictions presented in Table 3.4 have been identified to represent fire recorded by an ASUS Xtion Pro. (To get more comparable results it is recommended to set the *AutoWhiteBalance()*-function to *false*)

3.1.4 Smoke Color Range

As shown in Figure 3.9, gray pixel have the same values in every color-channel - namely Red (R), Green (G) and Blue (B) - leading to the condition $R = G = B$ for gray pixel. When observing smoke recorded by the RGB-Sensor of the ASUS Xtion Pro, a small difference ϵ is allowed

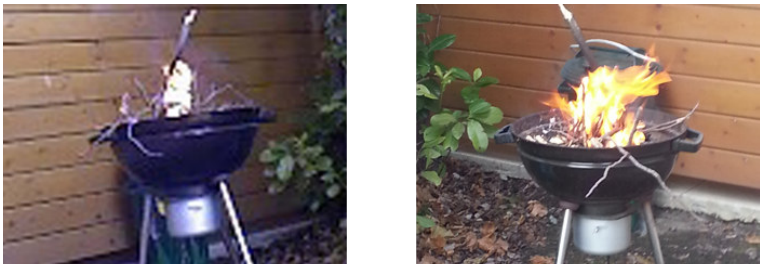


Figure 3.7: Two images from the same scene, left taken by the RGB camera of the ASUS Xtion Pro, right taken by the camera of a Samsung Galaxy S3. It can be observed that the ASUS Xtion Pro records fire as nearly white pixel

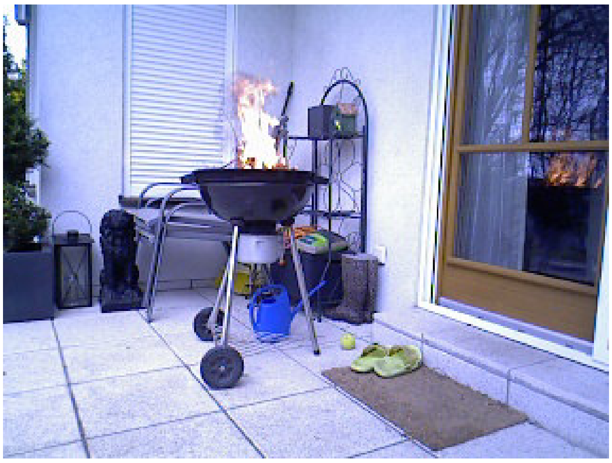


Figure 3.8: The image of a fire captured by an ASUS Xtion Pro with disabled Auto-White-Balance. It can be seen that the color-variation in the fire region is still very low.

between the color-channels. As further described in Section 'Static Analysis - Chroma' and paper [21], smoke can be divided into two different color-ranges; Light smoke and dark smoke. Therefore, smoke recorded by the ASUS Xtion Pro leads to pixel-intensity-values between 230 and 255 when observing light-smoke and between 165 and 192 when observing dark-smoke, considering the intensity-values (V) from a frame that is converted to the HSV-color space. The final restrictions for smoke colored pixel recorded by an ASUS Xtion Pro are shown in Table 3.5. Figure 3.10 shows a frame containing smoke recorded by the ASUS Xtion Pro.

Table 3.4: All five restrictions to extract fire-colored pixel from a frame recorded by an ASUS Xtion Pro.

Channel	Min. Value	Max. Value
RED	245	255
GREEN	240	255
BLUE	240	255
HUE	0	18
VALUE	250	255

Figure 3.9: Verification that the color gray consist of an equal amount of red, green and blue. RGB values (f.l.t.r.): [70/70/70], [120/120/120], [175/175/175], [220/220/220]

Table 3.5: All three restrictions to extract smoke-colored pixel from a frame recorded by an ASUS Xtion Pro.

Channel	Min. Value	Max. Value
COLOR	$(R \pm \epsilon) = (G \pm \epsilon) = (B \pm \epsilon)$	
HUE Light Smoke	230	255
HUE Dark Smoke	165	192

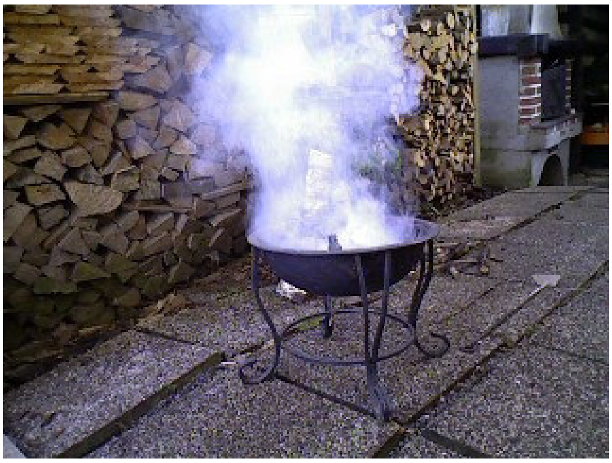


Figure 3.10: Smoke recorded by the RGB-sensor of the ASUS Xtion Pro.

3.1.5 Depth Camera and Fire

In this section, the change that the presence of a fire causes in the stream of the depth-sensor is evaluated. The ASUS Xtion Pro uses infrared-light (IR) to calculate the depth of objects in a scene. Due to the fact that fire emits IR light, this leads to the effect that the depth can not be estimated in areas containing fire. Figure 3.11 shows the change in the depth-video when fire is represented. (If the depth of a pixel/object can not be calculated it is represented as a black pixel in the depth-video) Figure 3.12 shows the superimposed image of a RGB- and a depth frame.

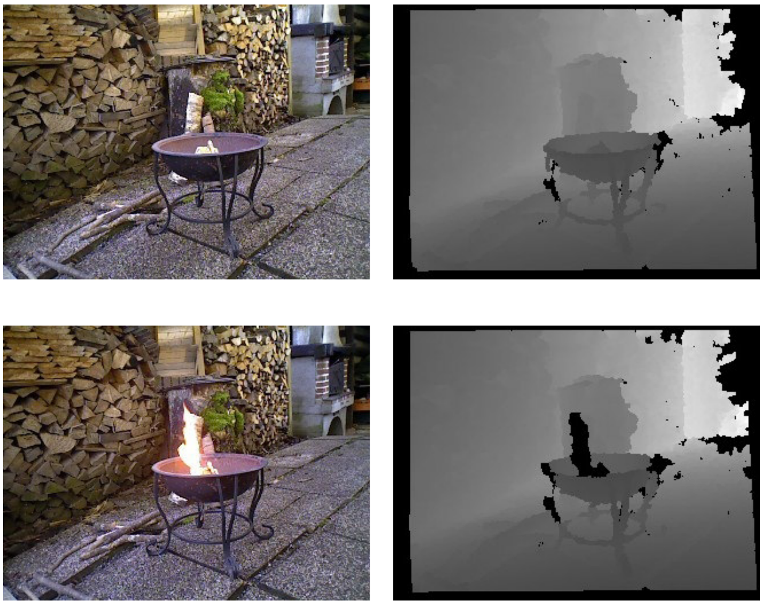


Figure 3.11: Two frames recorded by the ASUS Xtion Pro depth camera. Top row: $frame_{824}$ recorded by the RGB and Depth camera, showing a scene without fire. Bottom row: $frame_{11779}$ from the same scene. It can be seen that inside the fire-area the depth of the scene can not be calculated (illustrated by black pixel occurring in the depth-image).

The black areas of the depth image are removed, visualizing that the area - where no depth could be calculated - is in the same spot as the fire itself (Depth-image and RGB-image are rectified).

3.1.6 Flickering Depth Image

Fire itself leads to areas where the depth of these regions can not be estimated, but also when using a non-moving camera monitoring a static scene, regions where the depth could not be estimated occur. This happens so often that - when watching the video-stream from the depth-sensor - these black areas occur in a flickering behavior. Figure 3.13 shows two consecutive depth-frames, recorded by a non-moving camera monitoring a static scene. Although there is no movement in the scene there are changes in the areas where the depth of an object can not be

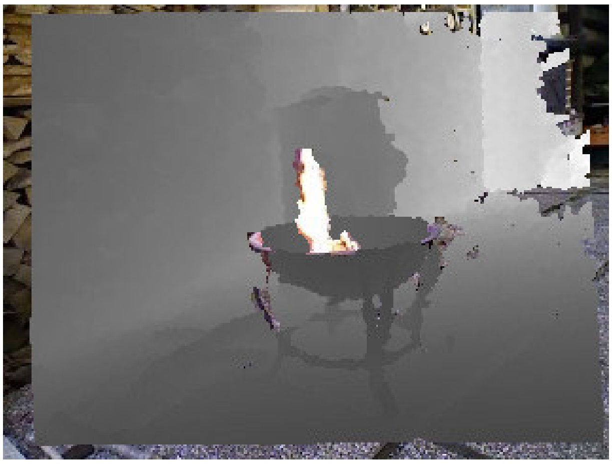


Figure 3.12: The superimposed image of RGB- and depth-camera from $frame_{11779}$, where black-pixels inside the frame of the depth-camera have been removed. It can be seen that the fire is located at the same area where the depth-camera is not able to calculate the depth of the scene.

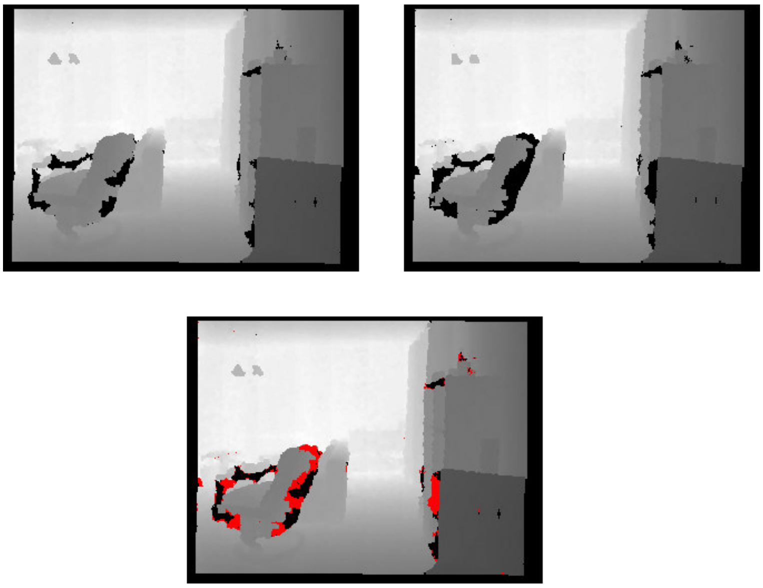


Figure 3.13: Two consecutive frames recorded by the ASUS Xtion Pro depth camera. Although there is no movement in the scene there are changes in the areas where the depth of an object can not be estimated (represented in black). Top row: Two consecutive frames $frame_1$ and $frame_2$ from a non-changing scene. Bottom row: Areas where the depth could be estimated in $frame_1$ - but not in $frame_2$ - are marked red. This leads to a flickering behavior of the depth stream.

estimated (represented in black). Due to this fact, the appearance of a new region - where the depth of the scene can not be estimated - is not sufficient enough to detect the presence of fire.

3.1.7 Depth Camera and Smoke

Here, it is analyzed how smoke changes the depth-image calculated by the depth-sensor. Even heavy smoke does not lead to an significant change in the depth-images of the ASUS Xtion Pro. Figure 3.14 shows two different frames, recorded by the RGB sensor and the depth sensor. It can be seen that in $frame_{13}$ (top row) the structure of the elements behind the fire-source is clearly visible in the RGB-image as well as the depth-image. In the RGB image of $frame_{329}$ (bottom-row, left) heavy smoke covers these elements. However, these elements are still visible in the depth-image. Therefore, the video-stream provided by the depth-camera of the ASUS Xtion Pro is not suitable for smoke detection.

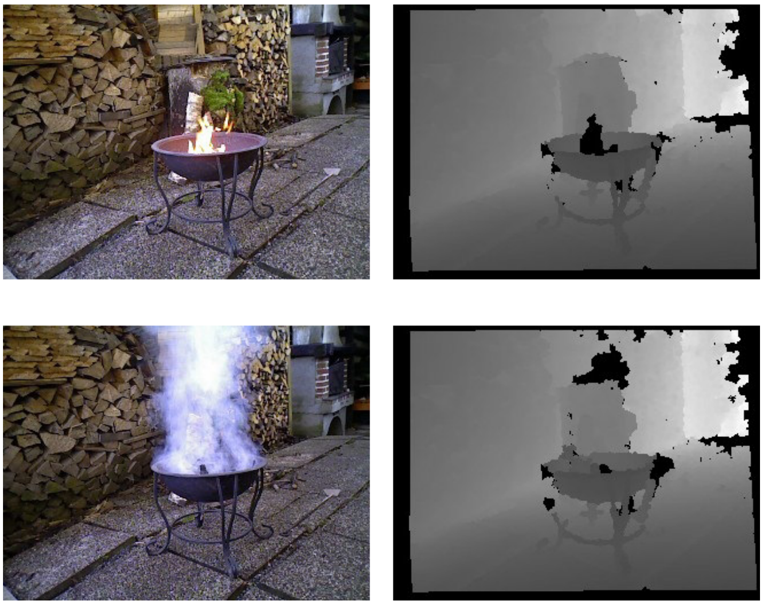


Figure 3.14: Two frames recorded by the ASUS Xtion Pro depth camera. Top row: $frame_{13}$ recorded by the RGB and Depth camera. Bottom row: $frame_{329}$ from the same scene. It can be seen that smoke does not lead to a significant change in the depth-image. For example: The wooden log is covered with smoke and for this reason it can not be seen in the RGB frame, while it is still visible in the depth-frame.

3.1.8 Depth Camera and Sunlight

Sunlight interferes with the light-pattern emitted by sensors like the Microsoft Kinect or the ASUS Xtion Pro [30]. Therefore, the depth-information provided by theses sensors is only usable indoors [20]. Here, the influence of sunlight on the depth-estimation is analyzed further, to show why the method proposed in this Thesis is limited for indoor-use only. Figure 3.15 shows a frame captured by the ASUS Xtion Pro and its corresponding depth-image, recorded in direct sunlight. It can be seen that direct sunlight in a scene prevents the sensor from calculating valid depths for these regions.

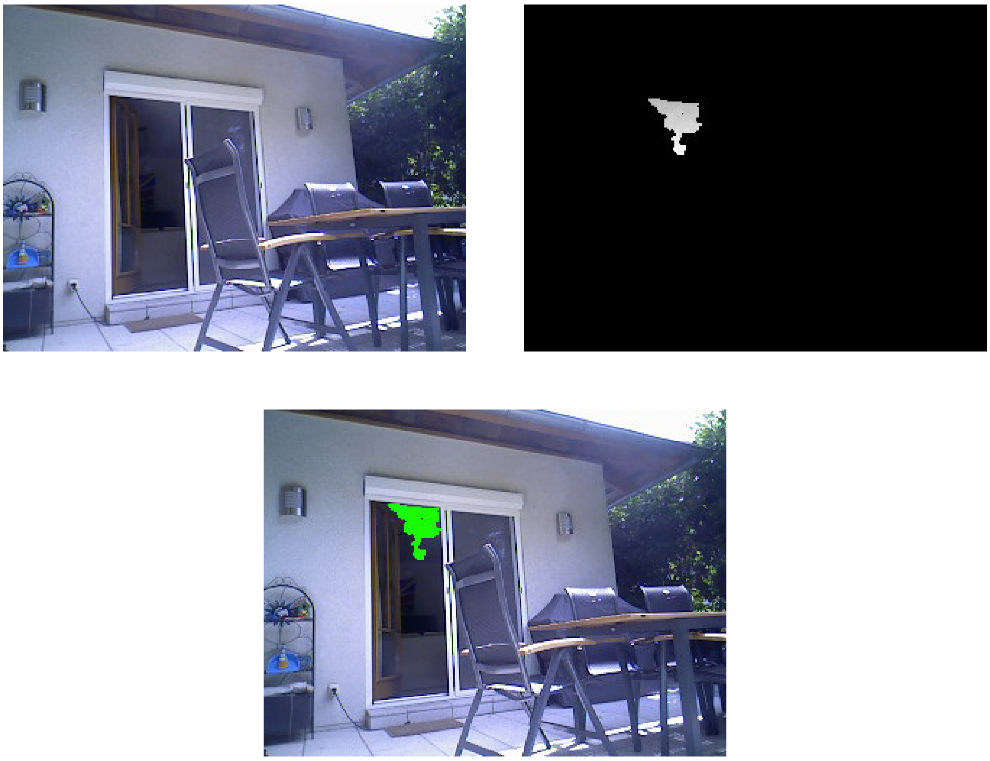


Figure 3.15: Top row, left: a frame recorded by the ASUS Xtion Pro in direct sunlight. Right: The estimated depth-map of the same scene. Bottom: the superimposed image, where areas - were a valid depth could be calculated - are highlighted in green. It can be seen that the depth-map does not provide useful results when the ASUS Xtion is used for example outdoor in direct sunlight.

3.1.9 Framework

Here, the software-framework OpenNI2 and the computer-vision library OpenCV are presented. Both are used in the proposed method for this thesis. *OpenNI* stands for *Open Natural Interaction*. The OpenNI framework is an open-source *Software Development Kit* (SDK), useful to develop 3D sensing applications and middleware[1]that supports Windows, Mac and Linux as a platform [37]. The idea is to standardize the interoperability of *Natural Interaction* (NI) devices, as the ASUS Xtion Pro or the Microsoft Kinect and it was partially developed by *Primesense*,

---

[1] OpenNI. The standard framework for 3D sensing, `http://openni.org/`, Accessed: 06.01.2014

one of the creators of the Microsoft Kinect. OpenNI provides *Application Programming Interfaces* (APIs) that allow programmers to access the NI-devices[1]. Figure 3.16 shows the architecture of OpenNI SDK and how NI-hardware interacts with developed applications. *OpenCV*

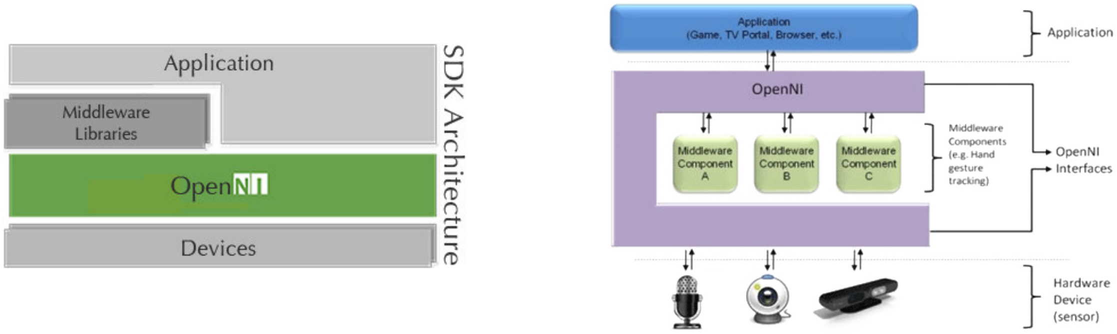


Figure 3.16: Left: The architecture of the *Open Natural Interaction* (OpenNI) Software Development Kit[1]. Right: The interaction of (different) hardware via OpenNI with developed applications [37].

stands for *Open Source Computer Vision Library*. The big advantage of OpenCV is that it contains more than 2500 optimized computer-vision algorithms and is available for free. OpenCV provides methods to store, save, manipulate, access, a.s.o. images. Widely methods used, like morphological operations, color transformations, face and object recognition, histogram calculation, a.s.o. are also provided by the OpenCV framework [11]. The current version OpenCV 2.0 includes main changes to the C++ interface and can be found on the official website[2].

3.2 Methods

In this Section methods used, like *Integral-Images* or *Morphological Operations*, are introduced and briefly explained.

3.2.1 Otsus Method

Otsus idea was that two regions - representing for example two classes - that should be separated are more homogeneous considering pixel belonging to each region than using pixel from different regions [6]. This is useful when e.g. a grayscale image is converted in a binary image. One measurement of this region homogeneity is its variance σ. Regions containing *quite similar* pixel have a lower variance than regions with lots of different pixel (e.g. noise). So, this method iterates all possible intensity values $1...I_{max}$ and divides the difference image in two classes, one for pixels with a lower value than the threshold and one for pixels with a similar or higher value. After that, the variance for both individual classes, namely σ_{class1} and σ_{class2}, is

---

[2] Open Source Computer Vision Library, `http://opencv.org/`, Accessed: 21.05.2014

estimated. The intensity value where the combined variance of both classes is minimal is then used for thresholding [6].

3.2.2 Morphological Operations

Morphological operations are used for image processing [35]. Here, the two basic approaches *erode*, *dilate* and their combinations *opening* and *closing* are explained further. Due to the fact that the approach proposed in this thesis uses morphological operations only on binary images, other approaches (for example suitable for color-images) are not explained further. Morphological-operations can be used to reduce noise or to fill small gaps between extracted structures or objects [35]. *Dilation* adds pixels to the boundaries of regions. It is used to close small gaps, but is also enlarges the structures [35]. *Erosion* is the opposite of dilation. Here, boundaries are made smaller. Erosion is usually used to eliminate noise, because structures smaller than the filter-size disappear after an erode is performed [35]. *Opening* is the operation called, when an erosion is performed followed by dilation. Opening smooths the boundaries of regions and it opens narrow connections [35]. *Closing* is when a dilation is followed by an erosion. Here, small gaps and holes inside the region are filled without increasing the size of the remaining structure [35].

3.2.3 Integral Images

Areas in images can be calculated very fast using integral-images as introduced by Viola and Jones in [43]. Here, the sum of values in a region can be calculated in $O_{(n)}$, by using only the values defined by the four corners of the rectangle [19]. This is done by first calculating the sum of all pixel-values that lie above and left of each pixel $p_{(x,y)}$. This Integral-Image can be calculated with only one pass over the original-image. After that, the sum of the pixel-values that lie within a rectangle R can be calculated by using the four corners, as shown in Figure 3.17. In this figure, Corner $C1$ covers the summed-up values for every pixel that lies above and left of it, visualized by rectangle A. Corner $C2$ also represents the values above and left of it, visualized by combining the rectangles $A + B$. Corner $C3$ consists of all values combining rectangle $A + C$ while corner $C4$ includes all values of the rectangles $A + B + C + R$. The final value for a Region R can be calculated by $C4 - C2 - C3 + C1$ [43].

3.3 Analysis of Movement Detection

When using static cameras, motion is widely used as a feature of fire and smoke, as in [16] and [39], for example. In this section, the quality as well as advantages and disadvantages of selected movement detections methods are examined in detail using MATLAB[3]. This leads to an deeper understanding on the behavior of movement detection when examining recorded fire.

---

[3] MATLAB - The Language of Technical Computing, `http://www.mathworks.com/products/matlab/`, Accessed: 27.06.2014

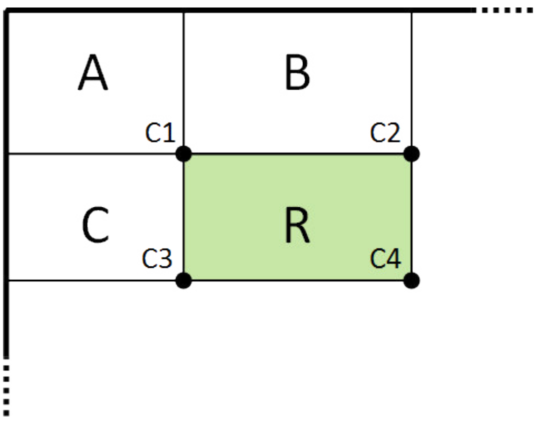


Figure 3.17: The sum of all pixel within the Rectangle R can be calculated by using its corner-values ($C1$, $C2$, $C3$, $C4$) provided by the initially calculated integral image. The value for R is then calculated by $C4 - C2 - C3 + C1$ [43].

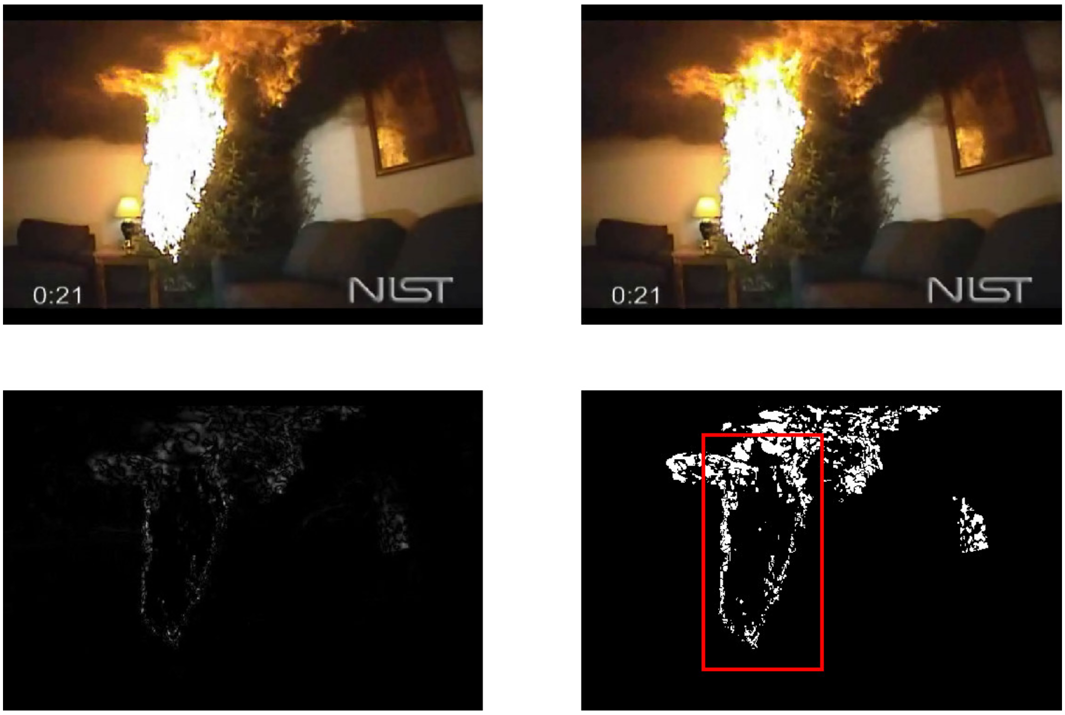


Figure 3.18: Frame from video[4], where the effect that the region inside the flame itself is considered as *non changing* can be seen. Clockwise starting with the top left image: $frame_{(800)}$, $frame_{(800+1)}$, the difference image and the difference image converted to a binary image by using an intensity threshold of 17, where the region inside the fire (which is wrongly detected as *non moving*) is marked by a red rectangle

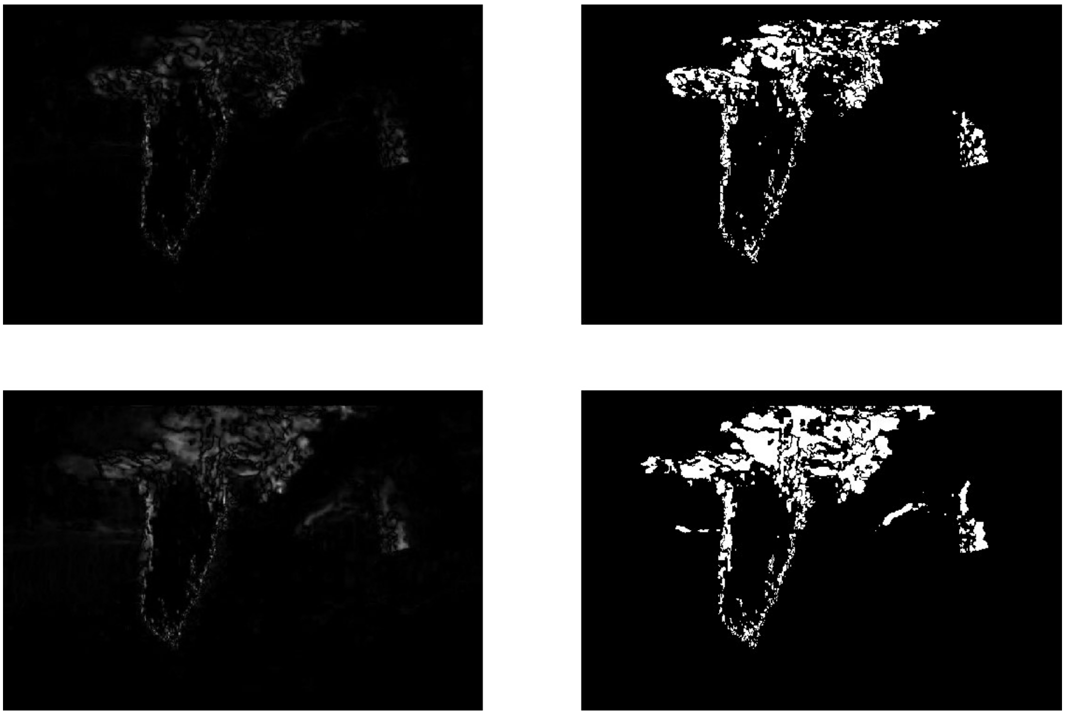


Figure 3.19: Difference image and binary image from video[4]. The left side shows the difference images, the right side the corresponding binary images. The frames used at the top row are $frame_{(800)}$ and $frame_{(800+1)}$ while for the bottom row $frame_{(800)}$ and $frame_{(800+5)}$ are used to estimate the difference- and the binary-image. It can be seen, that using a step-size of 5 frames provides better results.

3.3.1 Frame Differencing

Six different videos showing fire were evaluated to find out, if frame differencing could be a useful method to detect movement in fire videos. Firstly, two frames are selected and converted to grayscale frames. After that, the absolute difference between the two frames is calculated. Also a threshold is estimated to convert the resulting difference image into a binary image (for further details see Section 'Frame Differencing' in Chapter 'State of the Art').

First a threshold is estimated. Therefore, *MATLAB R2010a*[3] is used. MATLAB provides the *graythresh* method, which calculates a value that is able to distinguish between two assumed classes represented in a image. This automated thresholding is done by *Otsu's Method* [34]. Six videos are analyzed with this method. The fire intensity in these videos spreads from *just started burning* to a *fully developed fire*. The detected thresholds varied from intensity difference 1 to an intensity difference of 45 (out of 255). However, there were only those two values as an extrema (far from the median value), all other thresholds are near the intensity value 22. Figure 3.18 shows the main problem of frame differencing: when a fire is already very large, the region inside the flame appear as a single bright area, leading to no big differences between consecutive frames. For these frames, the - through Ostu's method - estimated threshold of 17 was used for these frames. The red rectangle shows that the area inside the flame was not detected as *moving*. An attempt to improve the binary image was made by increasing the step-size between the frames from 1 to 5, which leads to a time-difference $\Delta t = \frac{5}{30} sec = 0.167 sec$. The new resulting binary image, showing the detected moving pixel, seems better usable than that resulting from the smaller step-size 1. Figure 3.19 shows the direct comparison of the difference images to the left and the binary images to the right. The step size of 1 can be seen in the top row, the step size of 5 in the bottom row.

3.3.2 Background Subtraction

The first approach was taking the first frame as the background image. However, due to the noise existing in videos, noisy pixel could be seen as a single peak in the intensity image. To overcome this problem, generating the background image is adapted. Therefore, the median value of three consecutive frames is calculated and used as the background image. The video[5]used shows the comparison between a burning moist tree and a dry one. Around $frame_{(180)}$ the initial flame is started. Although the fire region on the left side is only about 5×5 pixel in total, it is recognized quite well. Figure 3.20 shows magnified parts of $frame_{(180)}$ on top, and an overlay of this frame in its original size with pixel detected as moving highlighted in green on the bottom. The used threshold is the same as estimated in Section 'Frame Differencing', resulting in intensity values bigger than 22 considered as moving. Another problem that occurs is that - due to for example changing lighting conditions or slight camera movement - a background image estimated just at the beginning of the recording can get very unreliable. Figure 3.21 shows the comparison of the same background image as before, but this time compared to a frame captured about 20

---

[4] Dry tree fire in non-sprinklered room (camera view-couch), `http://fire.nist.gov/tree_fire.htm`, Accessed: 26.11.2013

[5] Comparison of dry tree and properly maintained (high moisture) tree fires, `http://fire.nist.gov/tree_fire.htm`, Accessed: 26.11.2013

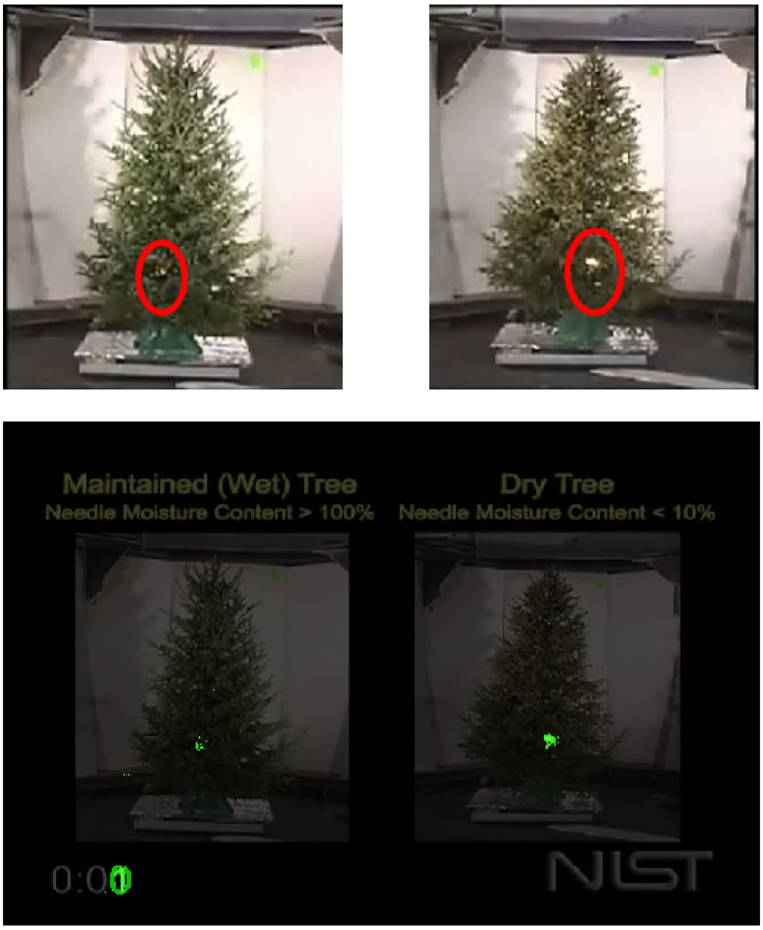


Figure 3.20: Frames from video[5]. The initial flame can be seen at $frame_{(180)}$ at the top row (magnified), the bottom row shows an overlay of the same frame in original size, with pixel detected as moving highlighted.

seconds later, at $frame_{(690)}$. Therefore, it is tried to dynamical adapt the background image over time. So, the *MATLAB* code is adapted to estimate the median value over time, in this case, the median of three pictures within a time interval of 5 frames is used. Figure 3.22 shows the adaptive generated background image, the binary image where movement was detected and an overlay of the binary map and the analyzed $frame_{(690)}$. Comparing Figure 3.22 to Figure 3.21 it can be seen that this new adaptive method is more reliable than using the background image only. In this case, noise is reduced so well that the intensity threshold can be reduced to 11 and still provides useful results.

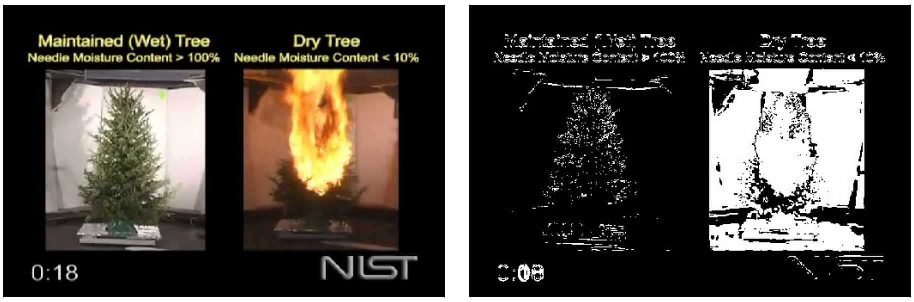


Figure 3.21: $frame_{(690)}$ from video[5] shown on the left side. Due to the time difference of about 22 seconds to the time when the background image was generated, the calculated difference image (shown on the right side) is not very reliable anymore.

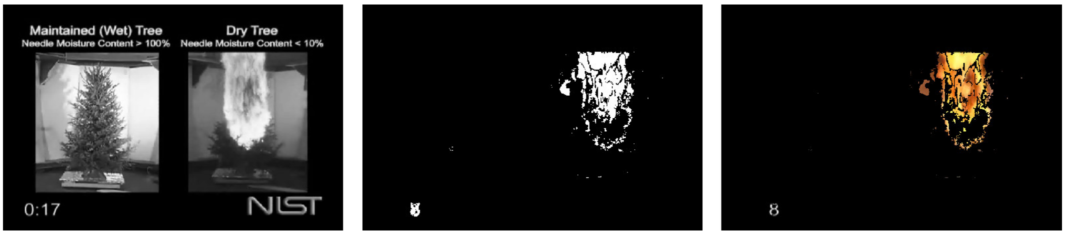


Figure 3.22: $frame_{(690)}$ from video[5] is analyzed. The picture on the left shows the new adaptive background image, calculated out of the median of three frames with time-gaps of 5 frames each. In the middle, the pixel detected as moving can be seen. The right picture shows an overlap of the analyzed frame and the binary image. Compared to Figure 3.21, this method seems less fragile to light changing conditions or slight camera movement.

3.4 Analysis of Fire-Features

In this section, RGB features for fire are analyzed further, to clarify the advantages and disadvantages the selected features have.

3.4.1 Color Range

Frames from video[4] and video[5] have been analyzed, using the color restrictions proposed in Section 'Color Range'. The condition for the value of the blue color channel (B) with $B < 80$ is to restrictive when using video[5] as an example video. Raising the limit to 100 delivers better results. Figure 3.23 shows one of the analyzed frames from video[5] on the left side, and a comparison of the thresholded frame with the values proposed in 'Color Range' in the middle, and the same image with the adapted threshold $B < 100$ for the blue color channel on the right side.

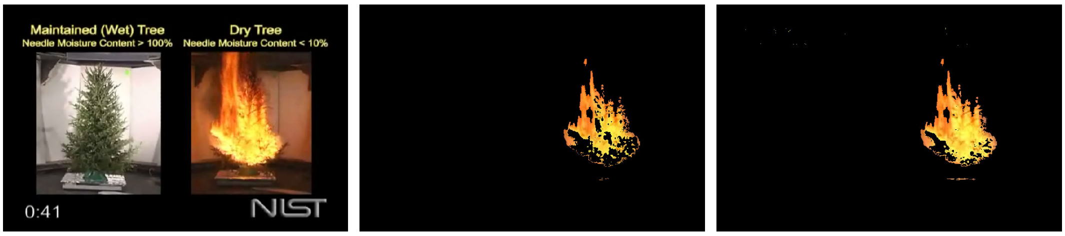


Figure 3.23: The analyzed $frame_{(1380)}$ from video^5 on the left side. In the middle the image after restricting the color values as described in the section 'Color Range'. On the right side, the restriction of the blue color channel was relaxed from $B < 80$ to $B < 100$.

3.4.2 Color Histogram

The adapted version of the color histogram restriction, as described in Section 'Colorhistogram', is evaluated here. Therefore, three rules need to be applied for each pixel:

- The values for the color channels have to satisfy the condition $R > G > B$
- The value for the R channel needs to be above a threshold λ_R. In this case, the threshold value described in Section 'Color Range' was used for the red channel.
- The saturation has to be above a threshold λ_{Sat}.

The value chosen for λ_R is 190, the value for λ_{Sat} is 50%. Figure 3.24 shows a frame containing fire and the resulting frames after applying each rule.

3.4.3 Comparison of Color Range and Color Histogram

Both methods are applied on the same frame. For the color range, the adapted version was used, where the value of the blue channel was limited to $B < 100$ instead of $B < 80$ as described in Section 'Color Range'. For the color histogram, all three rules with the estimated thresholds have been applied, as described in Section 'Color Histogram'. The result of both methods, applied on $frame_{(1380)}$ can be seen in Figure 3.25. It can be seen that 'Colorhistogram' estimates fire colored pixel better than the proposed method from 'Color Range'.

3.4.4 Surface Coarseness

The surface coarseness is a measurement for the degree of disorder inside a Region R. In Section 'Surface Coarseness', the standard deviation σ was used to measure if an area has a high amount of disarray. In this case, the variance var is used, because it spreads the values better than σ. The calculation of var is fairly simple: $var = \sigma^2$.

Different images have been analyzed regarding their surface coarseness / variance. First, all images have been thresholded using the values described in Section 'Comparison of Color Range and Color Histogram'. After that, the mean value and the variance of all remaining regions have

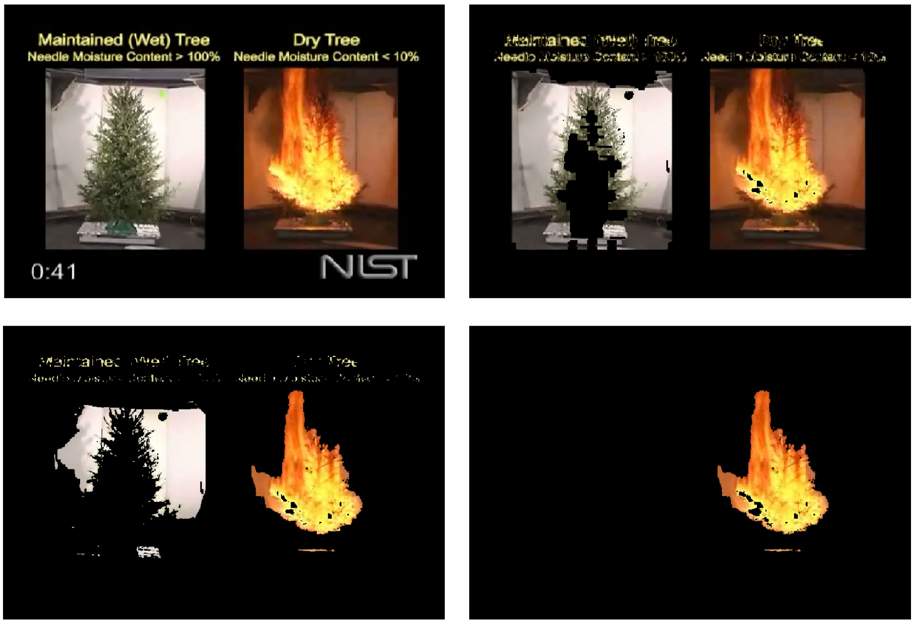


Figure 3.24: Top-row, left: the analyzed $frame_{(1380)}$ from video[5]. Right: The same frame with restriction 1 applied, namely $R > G > B$. Bottom row, left: the same image using restriction 1 & 2, leading to $(R > G > B) \wedge (R > \lambda_R)$. Right: $frame_{(1380)}$ again, with all three restrictions $(R > G > B) \wedge (R > \lambda_R) \wedge (Saturation > \lambda_{Sat})$ applied.

been calculated. Example images that have been used are shown in Figure 3.26. They show a person wearing an orange shirt, an image of an office burning, an artificial image where lots of pixel have the value $R = max$ for their red channel and an image of an burning tree. The results for the variance and the mean values are shown in Table 3.6 It can be seen that the variance of fire regions is much higher than the variance of for example the regions considered in the *orange shirt* image. However, the artificial generated image seems to have a high amount of disorder, but the value for the variance is very small.

Table 3.6: The mean and variance values from the images shown in Figure 3.26.

	mean	**variance**
Orange Shirt	217.37	250.8
Burning Office	237.39	479.82
Artificial Image with R_{max}	254.08	3.29
Burning Tree	239.49	390.54

---

[6] Your office fire, `http://www.youtube.com/watch?v=G6lLbDQcJyA`, Accessed: 01.12.2013

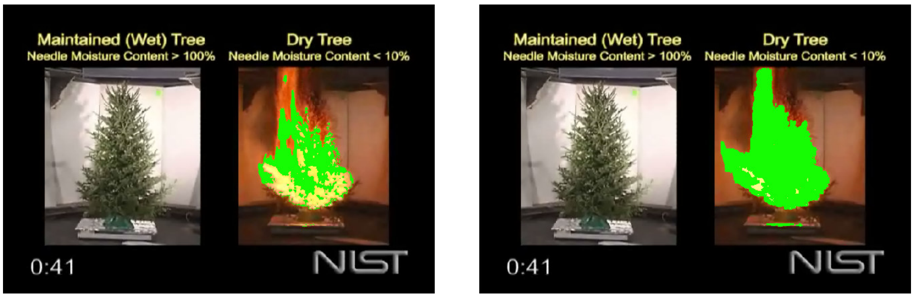

Maintained (Wet) Tree
Needle Moisture Content > 100%
Dry Tree
Needle Moisture Content < 10%
0:41
NIST
Maintained (Wet) Tree
Needle Moisture Content > 100%
Dry Tree
Needle Moisture Content < 10%
0:41
NIST

Figure 3.25: The analyzed $frame_{(1380)}$ from video[5]. Pixel evaluated as *fire* are marked in green. On the left side, the resulting image after using the color restriction from Section 'Color Range' can be seen, the right side shows the resulting frame using all three restrictions from Section 'Color Histogram'.

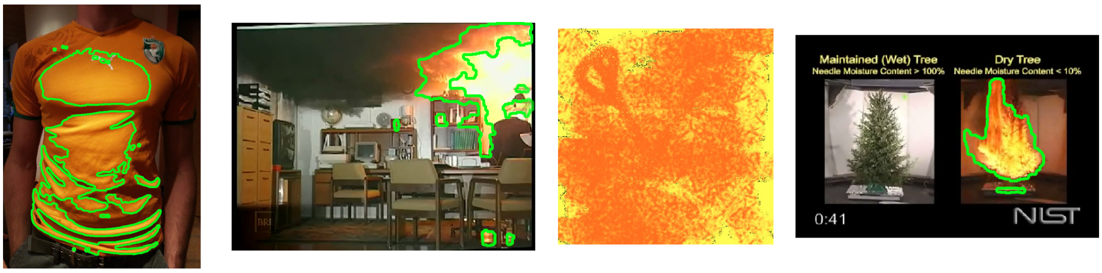


Figure 3.26: An image of a person wearing an orange shirt, an office burning[6], an artificial image where most pixel have the maximum value for their red color channel, and a burning tree[5]. Regions that have been analyzed are marked by a green border.

3.4.5 Skewness

The skewness is a value that describes the distribution of measured values. Here, only the red channel of selected areas is considered. Flame regions tend to have lots of pixel with high values for the R channel, as described in Section 'Skewness'. Figure 3.27 shows the histograms of the four images shown in Figure 3.26. Note: all histograms are limited to the values $180 \leq R \leq 255$. It can be seen that real fire regions have a high amount of pixel with a red channel value $r \approx Max_{(=255)}$, while for example an orange shirt has a significant lesser amount of pixel with the highest possible value for R. This leads to the following values for the skewness (calculated from possible fire regions values for the red color channel $R \geq 180$), as shown in Table 3.7. It can be seen that fire regions have a negative skewness, and therefore are more distributed to the right side, as for example regions extracted from a fire colored shirt.

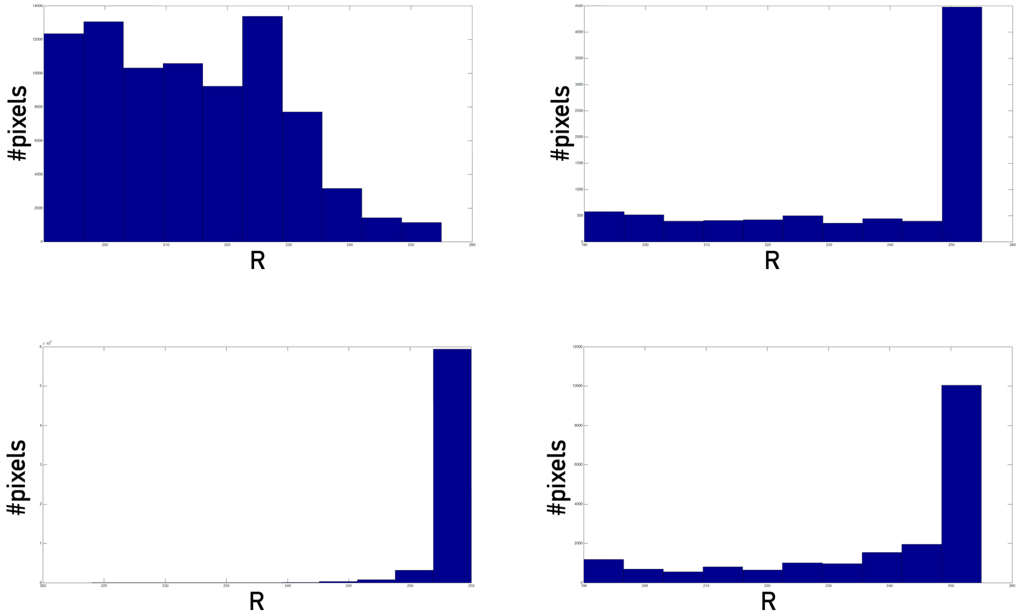


Figure 3.27: Histograms from possible fire candidate regions as shown before in Figure 3.26. Top left: histogram from the red color channel, calculated from *orange shirt*. Top right: histogram *office fire*[6]. Bottom left: histogram is from the artificial image. Bottom right: histogram belongs of the *burning tree* video[5]. Note: all histograms are limited to the range $180 \leq R \leq 255$.

Table 3.7: The mean and variance values from the images shown in Figure 3.26.

	mean	**variance**	**skewness**
Orange Shirt	217.37	250.8	0.072
Burning Office	237.39	479.82	-0.85
Artificial Image with R_{max}	254.08	3.29	-3.76
Burning Tree	239.49	390.54	-1.18

3.4.6 Comparison between Surface Coarseness and Skewness

It can be seen that the variance alone could be insufficient to determine fire regions from ordinary fire-colored regions. In fact, during testing some images showing e.g. an orange shirt, lead to variances up to $var = 374.46$. However, these images had a positive skewness, in this case a $skewness = 0.47$. So, both features should be used to discriminate real fire from fire colored objects. It is noteworthy that, to calculate the skewness of a distribution, the standard deviation has to be calculated anyway - this can be seen in Equation 2.19. So, if the skewness should be used as a feature, the surface coarseness can be used as an additional feature with low additional computational costs.

3.4.7 Randomness of Area Size

Here, the feature describing the spatial and temporal variation of fire, as described in Section 'Randomness of Area Size', is analyzed. Therefore, the video[6] was analyzed from $frame_{(3075)}$ to $frame_{(3299)}$ (from the beginning of a fire to the frame where this fire-scene ends), leading to a duration of 9 seconds. Figure 3.28 shows the amount of pixel declared as fire pixel per frame on the left side, and the corresponding value for ΔA_i on the right side. Fire pixel have been identified using the method described in Section 'Comparison of Color Range and Color Histogram'. It is noteworthy that, to receive a better overview, in this case only every second frame was used to calculate size and difference between frames in Figure 3.28. It can be seen that the size of the fire-area is fluctuating. In fact, the minimal area size recognized in this time slot is 8204 pixel, while the maximum size is 14630 pixel. The mean value for $\Delta \bar{A}_i = 0.094$. In Figure 3.28, it can be seen that the area size itself is varying significantly. The *rate of change between frames* ΔA_i is also fluctuating too much and seems therefore not stable enough to be a useful feature for e.g. thresholding with a value.

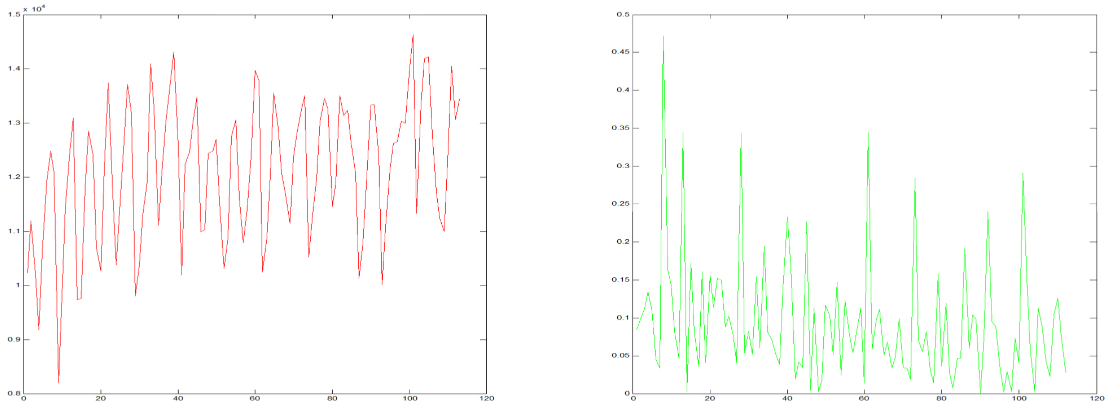


Figure 3.28: Left side, red: The amount of pixel recognized as fire pixel per frame in video[6], starting with $frame_{(3075)}$. Right side, green: The corresponding values for ΔA_i, describing how much the amount of pixel is changing from frame to frame.

3.4.8 Fire Growth Rate

However, considering the red graph in Figure 3.28, it can be seen that the fire area itself is growing over time. So the mean value of the last 25 frames (= 1 second) is generated for every frame (this method is called *moving average*). Figure 3.29 shows the amount of detected fire pixel on the left side, and the mean value consecutively calculated for the last 25 frames on the right side. This behavior was further estimated, using a 32 seconds timeslot, namely from $frame_{(3425)}$ to $frame_{(4225)}$ extracted from video[6]. The different times are used because of the longer timeslot for a developing fire. It can be seen that the moving average of fire region is growing from approximately 5600 pixel to approximately 10200 pixel. Figure 3.30 shows the moving average for the last 25 analyzed frames on the left side, and the moving average of the last 50 frames on the right side. Figure 3.31 shows the 50-frames moving average with two example frames and their position inside the graph. Other than when using the growth rate of smoke as mentioned in Sections ’Smoke Growth Rate’ and ’Adapted Smoke Growth Rate’, the graph of the size of a fire has to be smoothed by for example the moving average of the last 50 frames to deliver reliable results. This procedure is needed, because of the fluctuating behavior of fire, as described in Section ’Randomness of Area Size’ and visible in Figure 3.29 (red graph).

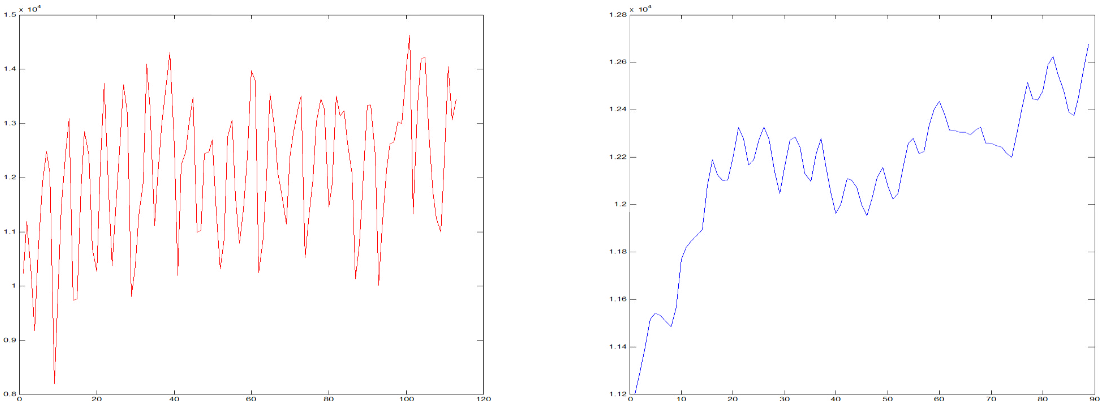


Figure 3.29: Left side, red: The amount of pixel recognized as fire pixel per frame from Figure 3.28. Right side, blue: The consecutive mean value for the last 25 analyzed frames (= 1 second) from video[6].

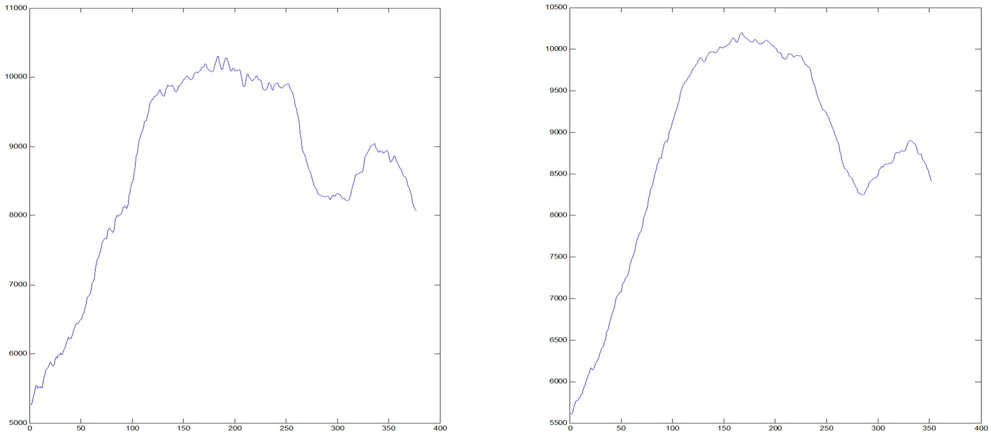


Figure 3.30: Analyzed timeslot from the video[6], starting at 2:17 and ending at 2:49. On the left side, a plot of the moving average of the last 25 frames can be seen, on the right the smoother moving average of the last 50 frames. For the first 200 frames, the growing behavior of the fire area can be seen.

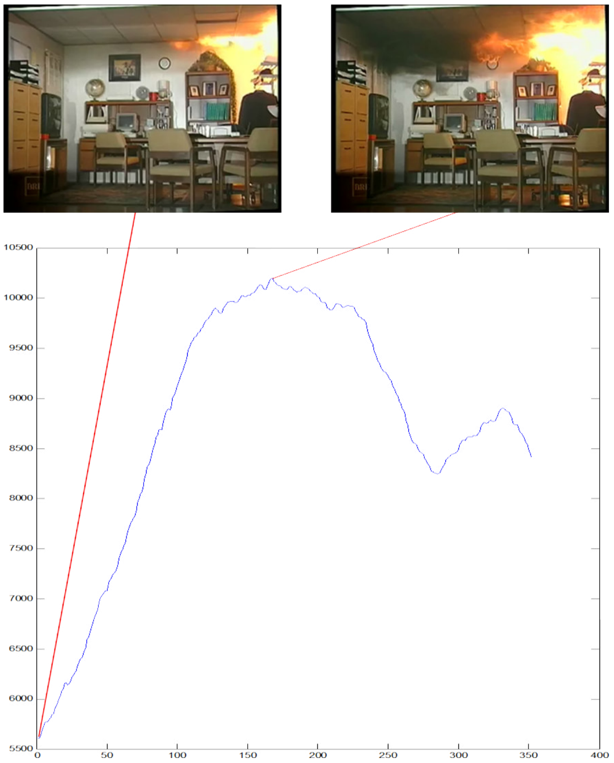


Figure 3.31: $frame_{(3425)}$ and $frame_{(3855)}$ from the video[6]. The moving average of the last 50 frames is rising from approximately 5600 pixel to approximately 10200 pixel.

3.5 Implemented Method for Fire Detection

The basic approach for this proposed method for fire detection (using the first sensor - namely an RGB camera - only) is similar to other methods, as for example in [5] [16] and [29]. Firstly, motion is detected. Secondly, values from different features are calculated. After that, a decision if fire is represented in the current frame is made by considering all results provided by the features used. One additional step is the final estimation if fire is detected in the overall video stream - the idea behind this additional decision is that a single *false positive* frame can not lead to a (wrongly) raised fire alarm. The additional depth sensor is used to verify if the potential fire region - detected by the RGB sensor - is a valid fire region. Therefore, this additional sensor is used to decrease the false-positive rate. Figure 3.32 shows the corresponding flow chart for fire detection using the proposed method, including the multi-sensor fusion.

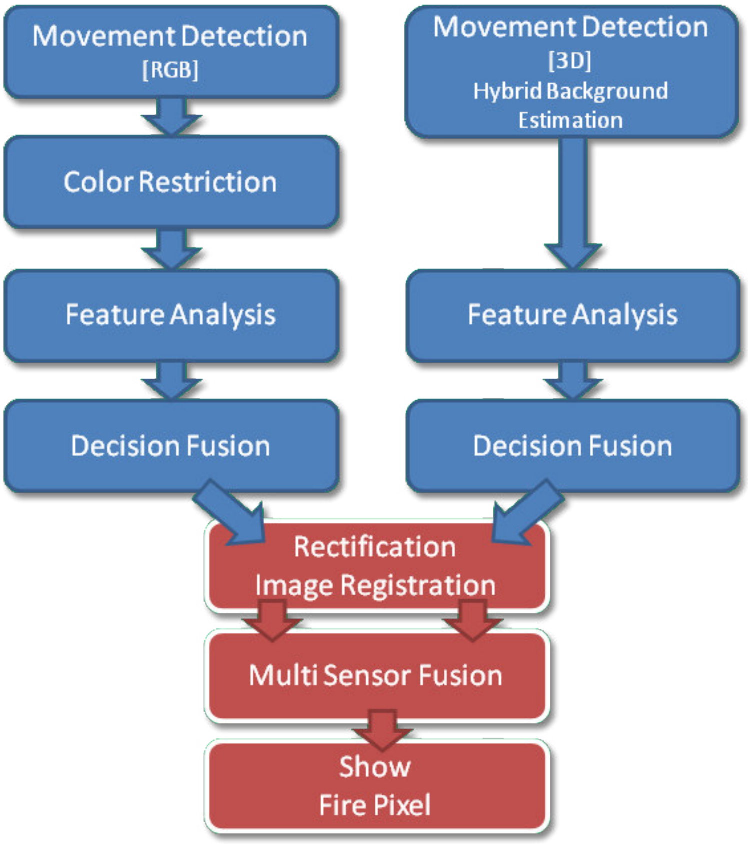


Figure 3.32: Flow-chart for the proposed fire detection algorithm, using the 2D and 3D multi-sensor fusion. Left side (blue) shows the detection algorithm using the RGB color sensor, right side (blue) shows the detection method for the 3D depth sensor. Blocks in red visualize the stages needed for a useful multi-sensor fusion.

3.5.1 RGB Camera

In this section the fire detection via RGB camera is described. Firstly, motion is estimated using frame-differencing. Secondly, every moving pixel is analyzed if it fulfills the color-restrictions for fire pixel. Thirdly, other features like the *mean-value* and the *skewness* are extracted. After that, the decision if fire is represented in the overall RGB frame is made by an adapted version of the *T-out-of-v voting* [16]. It is noteworthy that the final decision if a fire-alarm is raised is not made until the information of both sensors is merged.

Motion Estimation by the RGB Sensor

First, the current $frame_{(n)}$ and its preceding $frame_{(n-1)}$ are converted to grayscale images. After that, a matrix is calculated containing the absolute-differences between every corresponding pixel $p_{n_{(x,y)}}$ from $frame_{(n)}$ and $p_{n-1_{(x,y)}}$ from $frame_{(n-1)}$. A decision threshold is estimated using 'Otsus Method', dividing the matrix in two different classes: pixel with an absolute-difference above this threshold are considered as *moving* and therefore labeled as 1 in a binary-matrix, and pixel with a difference below the same threshold considered as *not-moving* - labeled by 0 in the same matrix. This binary-matrix is called a *Potential Fire Mask* (PFM) [5]. This motion estimation is a feature with the power of veto - pixel that are considered as *not-moving* - and therefore labeled as 0 in the PFM - can not be fire pixel. To eliminate artifacts resulting from noise, this PFM is smoothed by an *erode-filter* with a 2×2-kernel. To close small gaps between the remaining regions of moving pixel, an operation for image-closing is performed by a dilation followed by an erosion (in this case both methods are executed with a 6×6-kernel). The remaining PFM is called $PFM_{Movement}$. Figure 3.33 shows an RGB-frame containing fire with its corresponding binary-mask $PFM_{Movement}$.

Color Restriction

The Section 'Fire Color Range' describes the values for the Red (R), Green (G), Blue (B), Hue (H) and Saturation (S)-channel when examining fire-pixel recorded with an ASUS Xtion Pro. To get the final PFM for the color-restriction, namely PFM_{Color}, the current $frame_{(n)}$ that is analyzed is split into three different matrices containing the values of the R, G and B-channel, respectively. Every matrix (containing one color channel of the whole frame) is analyzed using the upper- and lower-threshold of each color channel, leading to three different binary-arrays B_R, B_G and B_B, where a pixel $p_{(x,y)}$ is set to 1 if the pixel fulfills the restrictions made in this channel and 0 if it does not. $Frame_{(n)}$ is further converted to the *HSV* color-space. The resulting image is divided again, leading to three sub-images, containing the values for the H, S and V-channel. In this case, only the arrays with the H- and S-values are analyzed further. By applying the restrictions from Section 'Fire Color Range', two more binary-arrays are generated: B_H and B_V. The final binary-mask PFM_{Color} is then calculated by *AND*-combining all five binary sub-arrays, leading to

$$PFM_{Color} = B_R \cap B_G \cap B_B \cap B_H \cap B_V. \tag{3.2}$$

After that, an image-closing operation is performed on the PFM_{Color}, using a 2×2-kernel for the dilation and a 3×3-kernel for the following erosion. Figure 3.34 shows the resulting

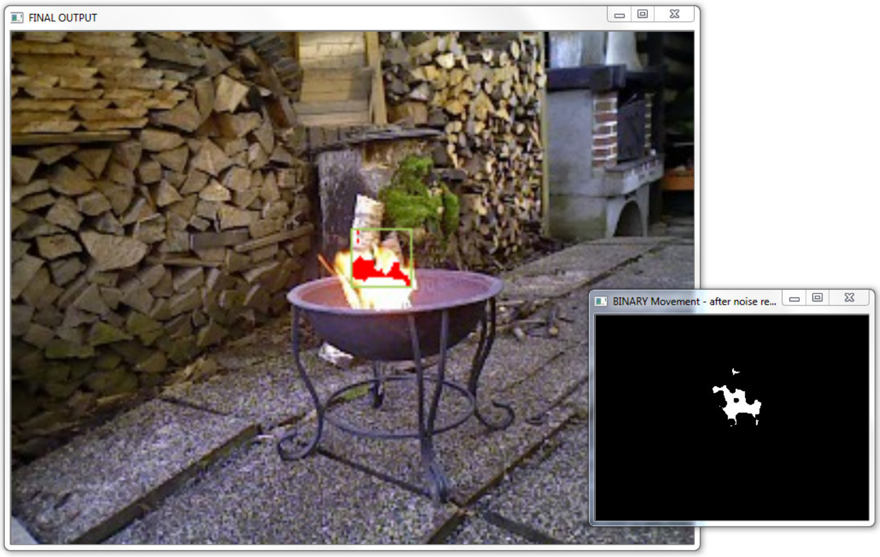


Figure 3.33: A frame recorded by the ASUS Xtion Pro RGB-camera and its corresponding binary-mask $PFM_{Movement}$ (after noise-reduction with the morphological operations *erode* followed by an *image-closing*).

binary-array PFM_{Color} after noise-reduction. The color-restriction is the second feature with the power of veto - every pixel that is marked as 0 in the PFM_{Color} is not further considered as a potential fire-pixel. The final PFM_{RGB} that is further analyzed is then calculated by *AND*-combining the $PFMs$ from the movement- and color-restrictions, leading to the binary-array

$$PFM_{RGB} = PFM_{Movement} \wedge PFM_{Color}. \tag{3.3}$$

Feature: Mean

Due to the fact that recorded fire leads to very bright pixel in the video-stream of the ASUS Xtion Pro, those pixel have high-values for their R, G and B channel, respectively. To extract this information, the proposed method uses the grayscaled $frame_{(n)}$ (already converted by the feature 'Motion Estimation by the RGB Sensor'). Here, the values of the R, G and B channels are merged to one value, representing the brightness of the corresponding pixel. After that, this grayscaled $frame_{(n)}$ is masked by $PFM_{Movement}$, leaving only those pixel that are considered as *moving*. Within the resulting pixel array, the mean-value $\bar{x}$ is calculated by

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \cdots + x_3}{n}, \tag{3.4}$$

where x_i represents the grayscale-value of the corresponding pixel and n is the number of moving pixel detected. In this case, the result of this equation is a real number and not a boolean value.

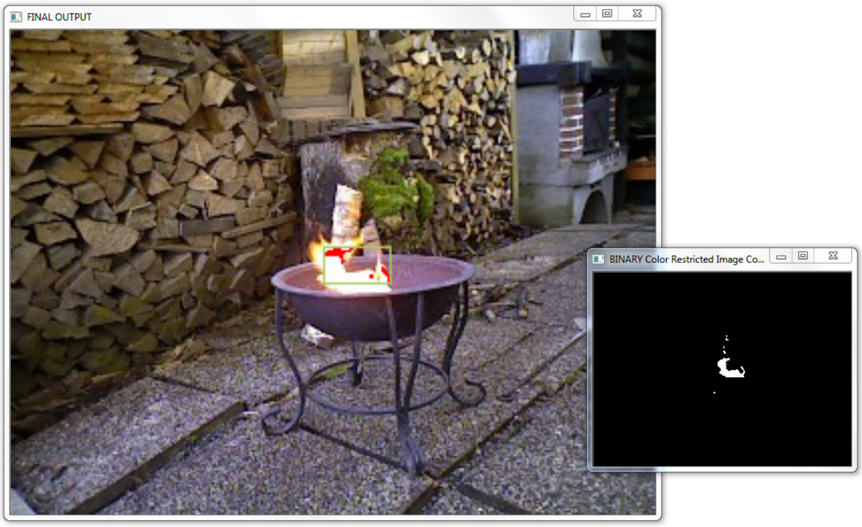


Figure 3.34: A frame recorded by the ASUS Xtion Pro RGB-camera and its corresponding binary-mask PFM_{Color} (after noise-reduction with the morphological operations *erode* followed by an *image-closing*).

Feature: Skewness

Here, the skewness of the red-color channel of pixel - considered as potential fire-pixel - is analyzed further. Therefore, the current $frame_{(n)}$ is first reduced to an image containing its red color-channel only. This sub-image is then masked with the binary-mask PFM_{RGB} calculated by the feature described in Section 'Color Restriction'. For the remaining red pixel-values the skewness is calculated by Equation 2.19. Unlike to the values from the features calculated in the Sections 'Motion Estimation by the RGB Sensor' and 'Color Restriction', the resulting value is a real-number and not a boolean-value. This value has no power-of-veto and is further used for the *decision fusion*. Due to the fact that fire-pixel have a high saturation in the red color-channel, fire is leading to a negative value for the skewness [5].

Feature: Surface Coarseness

As shown in the analysis of the feature 'Surface Coarseness' on page 67, there is a high spatial-variation inside fire-regions recorded in a video-frame, compared to for example a fire-colored object moving through a scene. This variation leads to an high value for the standard-deviation σ of the region. Sigma is calculated by Equation 2.21. As described in Section 'Fire Color Range', fire appears as very bright (actually nearly white) pixels, especially inside the center of the fire, when recorded by an ASUS Xtion Pro. Therefore, the standard-deviation in fire-regions recorded by the Xtion is as low as the standard-deviation for fire-colored objects, for example. Examining Figure 3.8 it can be recognized that fire-pixel on the edge of the fire region are still more orange-colored than white and therefore there is more variation within the pixel at the outline than within the pixel located inside the fire-area itself. The analysis of 'Frame Differencing' shows that pixel located in the core of a fire region are considered as *non-moving* due to the fact that the center of the fire is leading to recorded pixel with high brightness-values and therefore the difference between the pixel of the current $frame_{(n)}$ and its preceding $frame_{(n-1)}$ is not high enough to be marked as moving. Due to these facts, the calculation of the standard-deviation is done by eliminating the disadvantage of the low spatial-variation inside the fire-region by using only the pixel on the edge of the fire. This is done by masking the grayscale-converted $frame_{(n)}$ by the Potential Fire Mask $PFM_{Movement}$. After that, the standard-deviation is calculated by Equation 2.21. Here, the result of this equation is a real number again (and not a boolean value).

Decision Fusion (Color-Sensor)

Here, the decision if fire is detected in the current frame is made by considering all extracted features from the RGB stream of the ASUS Xtion Pro. Two out of five features - namely 'Motion Estimation by the RGB Sensor' (D_{Mov}) and 'Color Restriction' (D_{Col}) - have the power of veto, meaning that if the equation

$$[(\forall pixel_{D_{Mov}} = 0) \vee (\forall pixel_{D_{Col}} = 0)] = true, \tag{3.5}$$

is fulfilled there is no fire detected in this frame. More strict, there has to be at least one pixel that complies both restrictions at the same location (x, y), leading to Equation

$$\exists pixel_{(x,y)}, \text{ where } (pixel_{D_{Mov(x,y)}} = 1) \wedge (pixel_{D_{Col(x,y)}} = 1). \tag{3.6}$$

This means, that if there is at least one pixel that is considered as fire-colored and moving, the frame is evaluated further. The remaining three out of five features - namely 'Feature: Mean' (D_{Mean}), 'Feature: Skewness' (D_{Skew}) and 'Feature: Surface Coarseness' (D_{Dev}) have real-numbers as an outcome. With these values, an adapted version of the *T-out-of-v voting* [16] - as described in Section 'Voting Based' - is used to decide if fire is detected in the overall frame or not. Firstly, every feature is adapted by an adjustment-value (gained by experiments) representing the *zero-point* for the corresponding feature. This allows a faster recognition, how strong each feature has detected fire (leading to a positive value) for the evaluation. For this reason, the value 154 is subtracted from the calculated value by D_{Mean}. This zero-point adjustment leads to the behavior that areas with a higher mean-value (which are more likely to be fire regions) have an higher impact on the overall decision than regions with a lower value. The feature D_{Dev} is adapted by the value 50, the feature D_{Skew} does not need to be adapted, because it is already symmetric to 0. So the resulting behavior for all calculated values of the features are that the more negative a resulting value for a feature is, the more it is unlikely that this feature was calculated inside a real fire region. On the other hand, the larger the number the more likely it is a fire region. Please note that it is actually the other way round when examining the feature D_{Skew}, because in this case a negative Skewness is a characteristic for fire-regions. Secondly, every decision parameter is weighted by a factor, for D_{Mean} this factor is 1.0, for D_{Skew} it is 20.0 and for D_{Dev} this factor is 1.0. The larger factor for the skewness is reasonable, because of the small outcome values of this feature which lies in between -3 and $+3$ while the values for the standard deviation and the mean are much higher - even after subtracting adjustment values - lying within ranges between -154 and $+101$. The final-decision if there is a fire detected by the RGB sensor is made by adding up all values, leading to

$$D_{Final} = D_{Mean} \cdot 1.0 - D_{Skew} \cdot 20.0 + D_{Dev} \cdot 1.0 \tag{3.7}$$

if the resulting value lies above a decision threshold $\lambda_{RGB} = 0$, the current frame is marked as a frame *containing fire*.

3.5.2 Depth Camera

In this section the novel approach for fire detection including the depth-sensor is described. Firstly, a background-image is created merging the first three frames from the depth-sensor video-stream. Secondly, changes in the depth-frames are estimated by the background-subtraction method, therefore the background-image is updated every 30 frames. Regions with a valid depth-value in the background image but no valid value for the current frame are considered as potential fire-regions. There is no fire-detection made by the depth-sensor itself, it is used to decrease the false-positive rate compared to the fire-recognition by the RGB sensor only.

Hybrid Background Estimation

One precondition for this implemented plausibility-check is that the ASUS Xtion Pro is used as a stationary camera - meaning that the camera itself must not be moved. Due to this fact, a background image can be calculated and used to determine significant changes between the background-image and the current depth-frame $Dframe_{(n)}$. The initial background-image is calculated by a more time consuming method (than calculating the updated background images), using the first three depth-frames and merging them using the median-values of every pixel by using

$$DinitBG_{(x,y)} = Med(Dframe_{1(x,y)}, Dframe_{2(x,y)}, Dframe_{3(x,y)}), \tag{3.8}$$

where Med is calculating the median of the three values, $Dframe_1$, $Dframe_2$ and $Dframe_3$ are the first three frames recorded by the depth-sensor and $DinitBG$ is the resulting initial background-image. Due to the flickering behavior of the depth-sensor images - as described in Section 'Flickering Depth Image' - using the first image only could lead to false depth-values for those flickering pixel. Therefore a background-image pixel is set to for example 0 (= depth could not be determined), only if at least two out of the first three frames have a value of 0 on the same position. Due to the fact that this method is very slow (approx. 50 frames of the stream are not analyzed while calculating the initial background image) it is only used to calculate the first background image during the set-up stage of the proposed algorithm. Figure 3.35 shows a frame recorded by the RGB sensor and its corresponding initial background-image, calculated by the median-values of the first three frames received from the depth-sensor.

Fast Background Update

The initial background estimation technique using the median value provides very stable results, however this calculation is also very slow. Due to the real-time constraints for this method a

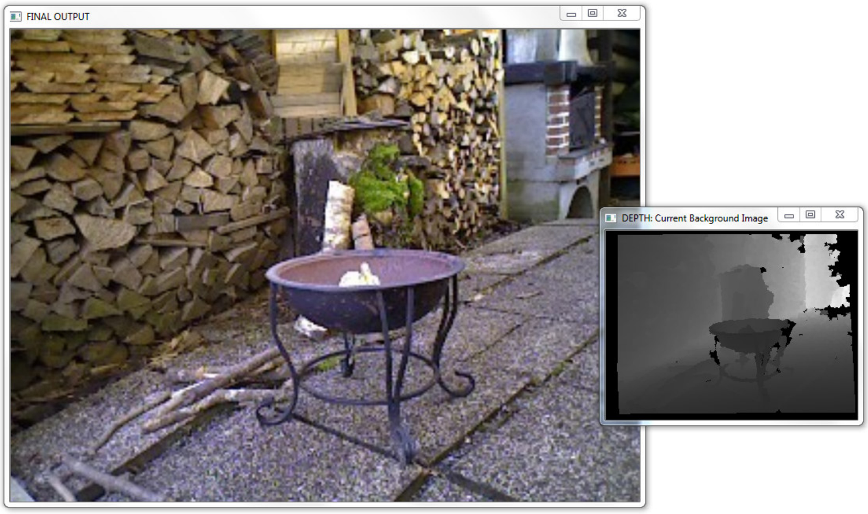


Figure 3.35: A frame recorded by the ASUS Xtion Pro RGB-camera and its corresponding initial background-image $DinitBG$, calculated by using the median values of the first three depth-frames $Dframe_1$, $Dframe_2$ and $Dframe_3$. For regions visualized as *black*, no valid depth could be estimated.

different approach is needed for continuously updating this background image. If a pixel of the current depth-frame $Dframe_{n(x,y)}$ has a valid value for its depth, it is linear combined with the corresponding pixel of the current background-image $DinitBGadapt_{n(x,y)}$ by

$$DinitBGadapt_{n(x,y)} = \frac{Dframe_{n(x,y)} + DinitBGadapt_{n-30(x,y)}}{2}, \tag{3.9}$$

where $DinitBGadapt_{n(x,y)}$ is the new calculated background pixel value on position (x, y) and $Dframe_{n(x,y)}$ is the corresponding pixel from the current depth image. $DinitBGadapt_{n-30(x,y)}$ is the preceding background image. The index $(n-30)$ is caused by the fact that the background-image is adapted every 30 frames by this method. If the value for a pixel from the current depth-frame $Dframe_{n(x,y)}$ could not be calculated - leading to the value 0 - this linear combination is executed including a weighting function. This reduces the impact that a flickering depth-pixel - as described in Section 'Flickering Depth Image' - has on the adapted background image $DinitBGadapt_n$, leading to

$$DinitBGadapt_{n(x,y)} = \frac{Dframe_{n(x,y)} \cdot 1 + DinitBGadapt_{n-30(x,y)} \cdot 3}{4}, \tag{3.10}$$

where the current depth-pixel $Dframe_{n(x,y)}$ is weighted by the factor 1, and the value of the corresponding pixel of the preceding background-image $DinitBGadapt_{n-30(x,y)}$ is weighted by the factor 3. This leads to a fast and stable method to continuously adapt the background image. Figure 3.36 shows $frame_n$ of the RGB video-stream and the corresponding background-image $DinitBGadapt_n$, which has been updated for 40 times since the detection began.

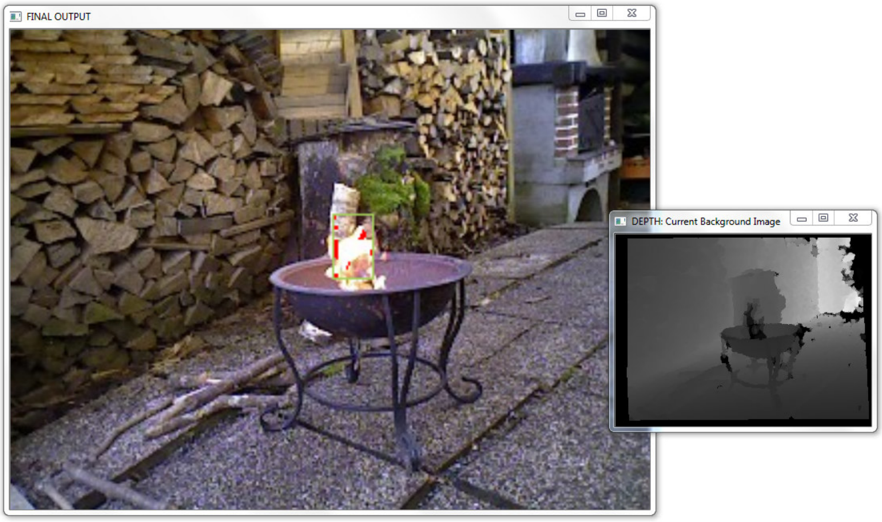


Figure 3.36: A frame recorded by the ASUS Xtion Pro RGB-camera and its corresponding background-image $DinitBGadapt_n$, calculated by using the linear combination of the preceding background-image $DinitBGadapt_{n-30}$ and current depth-frame $Dframe_n$. As the index $(n - 30)$ suggests this linear combination continuously takes place every 30 frames.

Motion Estimation by the Depth-Sensor

Motion can be detected in depth-frames similar as in RGB-frames, as described in Section 'Background Subtraction'. For this method, the difference between the calculated and adapted background-image $DinitBGadapt_n$ and the current depth-frame $Dframe_n$ is made by subtracting every pixel from the adapted background image from the current depth image. If the value is above a decision threshold, this pixel is labeled as *moving*. To improve the result, noise is reduced by using an *erode-filter* with a 2×2-kernel. After that, an image-closing operation is performed using a 5×5 filter-kernel, leading to the *Potential Fire Mask* (PFM) of the movement detected by the depth sensor, called $PFM_{Depth(Mov)}$. Figure 3.37 shows a video-frame recorded by the RGB sensor and the corresponding $PFM_{Depth(Mov)}$, where detected motion is visualized by white pixel.

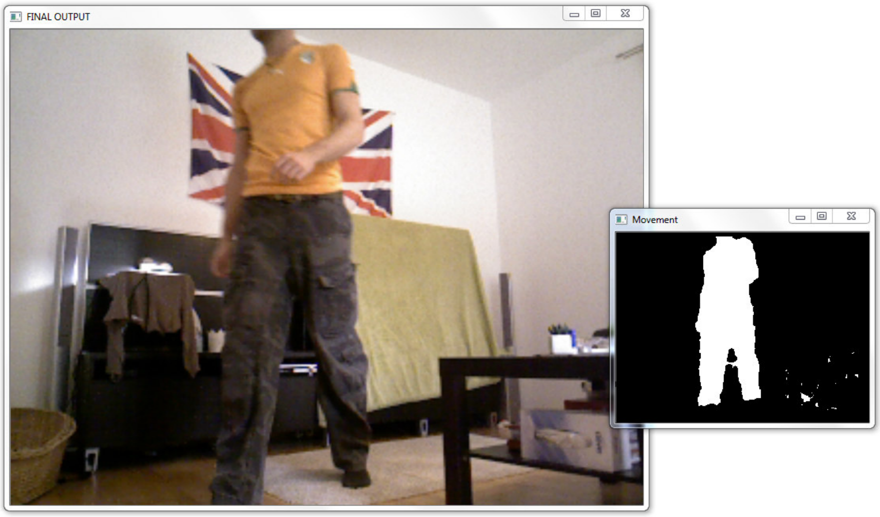


Figure 3.37: A frame recorded by the ASUS Xtion Pro RGB-camera and the corresponding binary-image $PFM_{Depth(Mov)}$ for this feature, calculated by subtracting the adapted background image $DinitBGadapt_n$ (calculated by the informations provided by the 3D sensor only) from the current depth-frame $Dframe_n$. By thresholding and using morphological-operations the final array $PFM_{Depth(Mov)}$ is calculated.

New Areas with invalid Depth-Values

As described in Section 'Depth Camera and Fire', fire leads to areas where the depth of the region can not be determined by the depth-sensor of the ASUS Xtion Pro. Therefore, the current depth-frame $Dframe_n$ is compared to the determined (adapted) background-image $DinitBGadapt_n$. If a pixel has the value 0 (=no depth calculated) in $Dframe_n$, but a value $\neq 0$ in $DinitBGadapt_n$ it is marked as a potential fire pixel in the Potential Fire Mask (PFM)

$PFM_{Depth(Black)}$. Figure 3.38 shows the current RGB-frame $frame_n$, the determined, adapted background-image $DinitBGadapt_n$, the current depth-image $Dframe_n$ and the binary-image representing the potential fire-regions $PFM_{Depth(Black)}$ for this feature. It can be clearly seen that the flickering behavior of the depth image (as described in Section 'Flickering Depth Image') results in false regions of potential fire in the $PFM_{Depth(Black)}$.

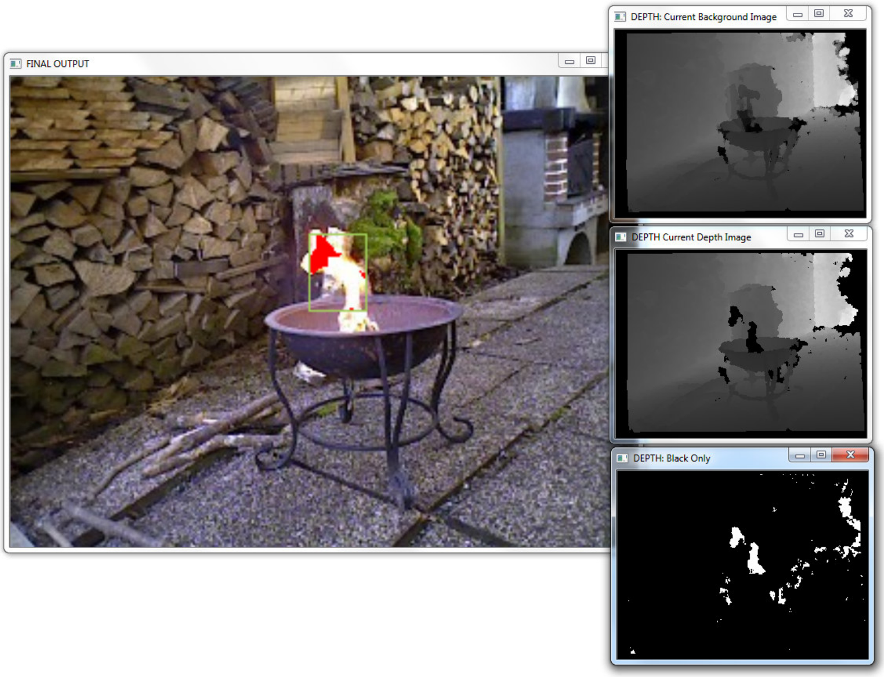


Figure 3.38: A frame recorded by the ASUS Xtion Pro RGB-camera and its corresponding background-image $DinitBGadapt_n$, the current depth-frame $Dframe_n$ and the resulting binary-image $PFM_{Depth(Black)}$ for this feature. The flickering behavior of the depth-image (as described in Section 'Flickering Depth Image') leads to false areas in $PFM_{Depth(Black)}$.

Decision Fusion (Depth-Sensor)

For the final detection by the depth-camera, the features explained in the Sections 'New Areas with invalid Depth-Values' and 'Motion Estimation by the Depth-Sensor' are combined. The final Potential Fire Mask (PFM) is created by

$$PFM_{Depth} = PFM_{Depth(Mov)} \wedge PFM_{Depth(Black)}, \tag{3.11}$$

where $PFM_{Depth(Mov)}$ is the resulting binary-image from the movement detection of the depth-camera, $PFM_{Depth(Black)}$ is the resulting binary-image from the detection of new areas with invalid depth-determination and PFM_{Depth} is the resulting binary image. When used alone, the proposed method for the depth-sensor is not able to detect fire-events in a meaningful way. The main reason for this is the flickering behavior of the depth-sensor, generating regions where no valid depth-values can be determined - which results in the same behavior as when recording fire incidents (see Sections 'Depth Camera and Fire' and 'Flickering Depth Image' for details). Therefore the occurrence of a *new* region - where the depth of a scene can not be determined - is not alone sufficient to conclude that there is fire in the scene, so the results provided by the depth-sensor analysis are combined with the results from the RGB-Sensor. The idea is that the information provided by the depth-analysis should decrease the *false-positive*-rate of the overall method. Figure 3.39 shows a RGB-frame provided by the ASUS Xtion Pro and the resulting binary-image, calculated by combining the results provided by the 3D features described in the Sections 'New Areas with invalid Depth-Values' and 'Motion Estimation by the Depth-Sensor'. Falsely detected regions can be seen on the right border of the binary image (highlighted by the red ellipse).

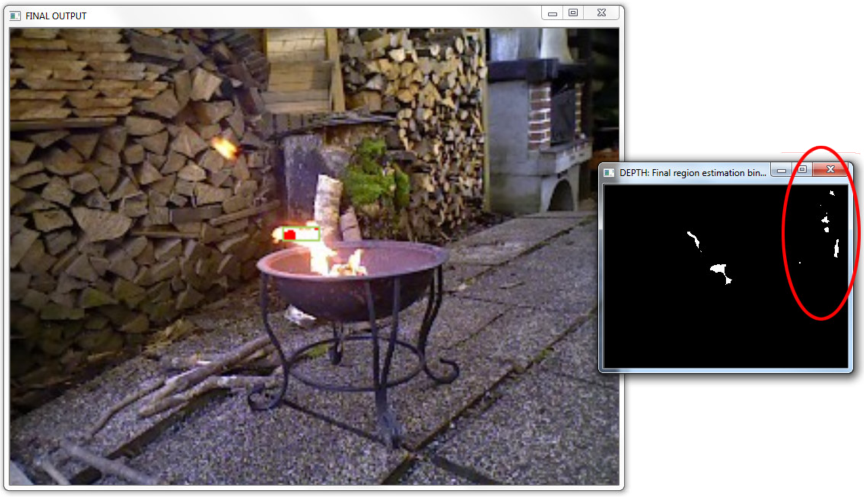


Figure 3.39: A frame recorded by the ASUS Xtion Pro RGB-camera and the final PFM_{Depth}, calculated by combining the binary-images $PFM_{Depth(Mov)}$ and $PFM_{Depth(Black)}$. Falsely detected fire-regions are marked in red.

3.5.3 Decision made by Sensor Fusion

The final decision - if fire is represented in the overall frame - is made by combining the different results of both sensors, the RGB color-sensor and the depth-sensor. To achieve this, two preconditions have to be made; First, the video-streams provided by both sensors have to be synchronized. In fact, the time difference between the frames should not be bigger than $1/30$ seconds. Second, both video-streams have to be rectified, so that a pixel $frame_{n_{(x,y)}}$ from the RGB-stream and the pixel $Dframe_{n_{(x,y)}}$ from the stream of the depth-sensor are referring to the same point - for example an object - in the scene (for further details for synchronization see Section 'Time Difference between RGB and Depth Frames' and for additional information for image-registration see Section 'Image Registration'). For the final decision, if fire is recognized in this frame, firstly the Potential Fire Masks PFM_{Color} (provided by analyzing the RGB-sensor) and PFM_{Depth} (provided by analyzing the depth-sensor) are combined, leading to the final binary-image PFM_{Final}, calculated by

$$PFM_{Final} = PFM_{Color} \wedge PFM_{Depth}. \tag{3.12}$$

So, if both sensors have detected potential fire-regions in the same area, pixel from the binary-image PFM_{Final} are set to *true*. Figure 3.40 shows an frame recorded by the RGB-camera and the corresponding final Potential Fire Mask PFM_{Final}, which is made by combining the informations of both sensors. Secondly, for the overall decision, the information calculated in Equation 3.7 is analyzed: If the result for the value D_{Final} is above the decision value $\lambda_{RGB} = 0$, and the binary-mask PFM_{Final} contains at least one *true*-value, the existence of fire in the overall frame is assumed, leading to the final decision described by the equation

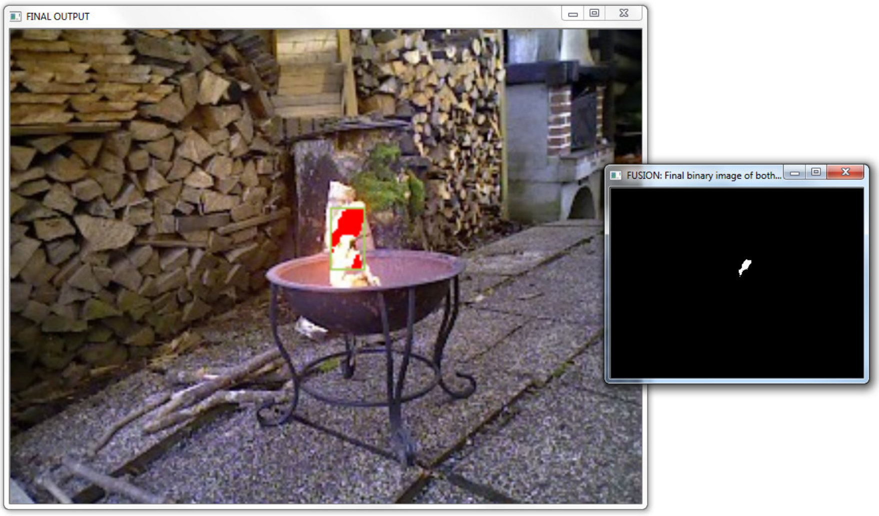


Figure 3.40: A frame recorded by the RGB-camera and the final PFM_{Final}, calculated by combining the information provided by the RGB color-sensor and the depth-sensor of the ASUS Xtion Pro.

$$(D_{Final} > \lambda_{RGB})\ AND\ (PFM_{Color} \cap PFM_{Depth}) \neq \emptyset. \tag{3.13}$$

3.5.4 Final Decision

To reduce the number of false alarms, the final fire-alarm is not raised until at least 75% out of the last 30 analyzed (consecutive) frames are recognized as *containing fire*. Figure 3.41 shows the implemented method for fire-detection using the sensor fusion of the 2D RGB-sensor and the 3D depth-sensor of the ASUS Xtion Pro.

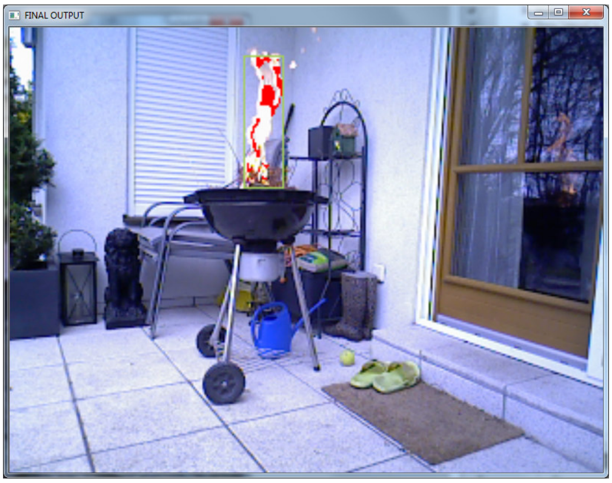


Figure 3.41: The implemented method for fire-detection, using the 2D RGB- and the 3D depth-sensor of the ASUS Xtion Pro. The analyzed regions are labeled as red (surrounded by a green rectangle).

3.6 Implemented Method for Smoke Detection

Here, the implemented method for smoke detection is described. The basic approach stays the same as for other smoke detection methods as described in [21] [23] and [39], with preceding movement-detection and color-restrictions. After that, this method divides the color-frame in 8×6 subregions (resulting in 48 different regions $Reg_{(x,y)}$), where different features are calculated, as shown in Figure 3.42. The final decision - if smoke is detected inside a region - is then made by a decision tree. Due to the fact that smoke does not lead to significant changes in the depth image - as described in Section 'Depth Camera and Smoke' - the information provided by the depth-sensor is not included in the smoke-detection part of this implemented method (in contrast to the fire-detection method, where the information of the depth-sensor is used to decrease the false-positive rate). Figure 3.43 shows the flowchart for the smoke detection, with steps that are not needed for this approach faded to gray.

3.6.1 Motion Estimation

Motion is estimated by using frame-differencing. Due to the fact that the motion estimation for the RGB-stream of the fire-detection is also using frame-differencing there is no additional computation time needed (see Section 'Motion Estimation by the RGB Sensor' for details). For the smoke-detection the size of the filter-kernel used afterward is different. To reduce noise the

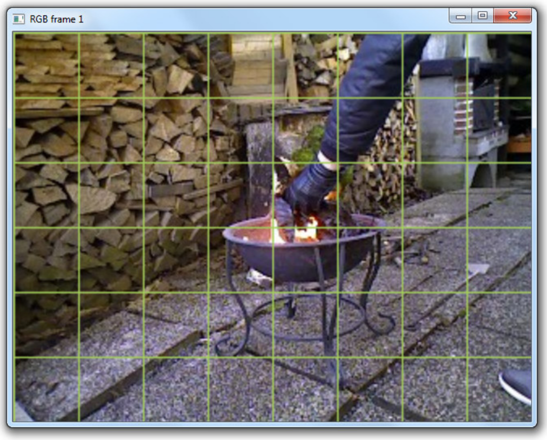


Figure 3.42: A video-frame recorded with the ASUS Xtion Pro. The frame is subdivided into regions where different features are calculated. This regions are visualized by a green grid.

same 2×2 filter is used, but the image-closing operation uses a 15×15 filter. The resulting binary-image of this operation is called *Potential Smoke Mask* PSM_{Mov}.

3.6.2 Color Restriction

The Section 'Smoke Color Range' describes the typical color-values that are recorded, when observing smoke with the RGB-sensor of the ASUS Xtion Pro. Those three restrictions, regarding the distribution of the color-values on the RGB-channels and the Intensity-value from the HSV-model, are combined to calculate the resulting PSM_{Color}. The first restriction to detect smoke-colored pixel is that the values for the Red (R), Green (G) and Blue (B) channels have to be nearly the same, only a small difference ϵ is allowed. This method restricts the maximal difference between the color-channels to $\epsilon = 8$, resulting in the binary-image $PSM_{R=G=B}$. The second restriction is that smoke is either dark or light. Therefore the intensity-value of a pixel (converted to the HSV color-model) is restricted to the values described in Section 'Smoke Color Range', leading to two Potential Smoke Masks PSM_{Light} and PSM_{Dark}. Due to the fact that smoke is either *light* or *dark*, the final binary-image PSM_{Color} is created by

$$PSM_{Color} = PSM_{R=G=B} \wedge (PSM_{Light} \vee PSM_{Dark}). \tag{3.14}$$

The resulting binary-mask is first eroded by a 2×2-filter to reduce false-detections caused by noise, and second smoothed by an imclosing-operation realized by a 6×6-filter.

3.6.3 Movement and Color

By using the information provided by the *Potential Smoke Mask* PSM_{Mov}, which is the result of the movement-estimation described in Section 'Motion Estimation', and PSM_{Color}, which is

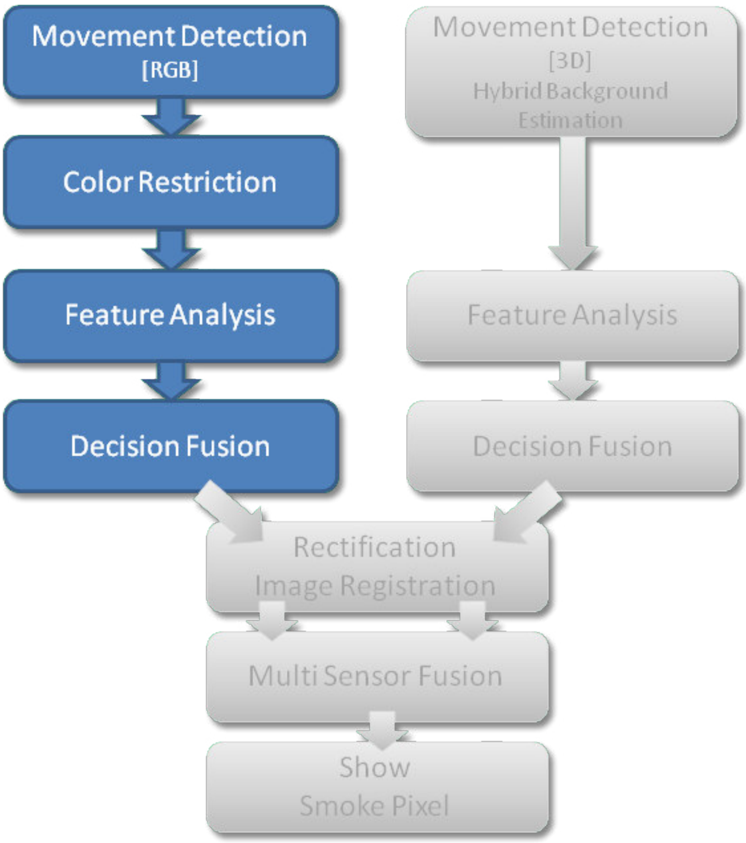


Figure 3.43: Flow-chart for the proposed smoke-detection algorithm, using the 2D color-sensor only. Blocks in gray are not needed, because in this case no multi-sensor fusion takes place.

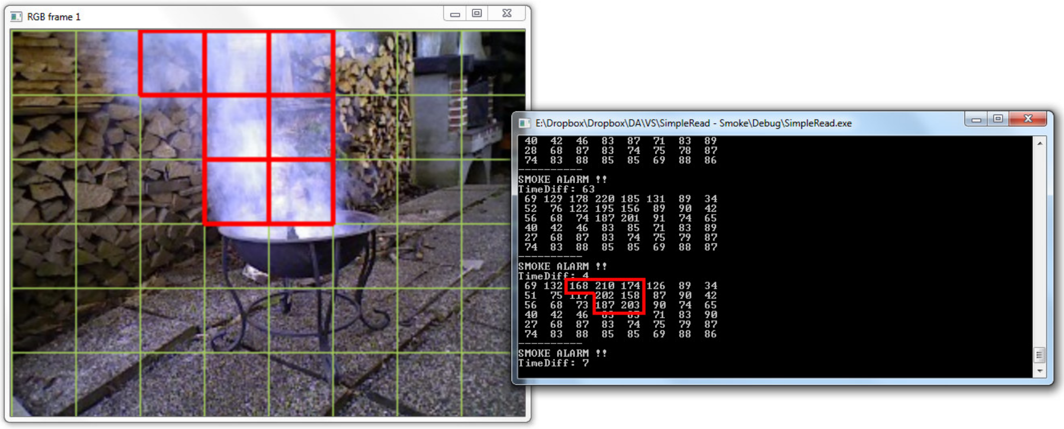
the result of the color-restriction described in Section 'Color Restriction', the final binary-mask PSM_{RGB} is constructed by

$$PSM_{RGB} = PSM_{Mov} \wedge PSM_{Color}. \tag{3.15}$$

This binary-image is then converted to an *Integral-Image*. For each Region $Reg_{(x,y)}$ the amount of pixel that fulfill the requirements *movement* and *color* is calculated. If the amount of those pixel inside a region is above 20% the Decision (D) for this region is labeled as 1, resulting in $D_{Col(Reg_{(x,y)})} = 1$ for a potential smoke-detection in this region (regarding only the restrictions made by the features *movement* and *color*).

3.6.4 Average Grayscale Value

As described in Section 'Smoke Color Range', smoke has a typical color range. Du to this fact, the average value of the pixels inside a smoke region lies above a threshold. To calculate this feature, the current $frame_n$, recorded by the RGB-camera, is converted to to a grayscale-image. Using this image, an *Integral Image* is calculated. By dividing the value calculated within a Region $Reg_{(x,y)}$ by the amount of pixel inside this region the average value is calculated. If this value is above a decision threshold $\lambda_{Grey} = 140$ the decision for this region is labeled as $D_{Grey(Reg_{(x,y)})} = 1$ for a potential-smoke detection by this feature. Figure 3.44 shows a color-frame recorded by the ASUS Xtion Pro divided in its subregions. In the command-window the mean-values of the grayscale pixel inside each region are shown. It can be seen that regions covered by smoke have a higher value than regions not covered by smoke.


RGB frame 1
E:\Dropbox\Dropbox\DA\VS\SimpleRead - Smoke\Debug\SimpleRead.exe

```
 40  42  46  83  87  71  83  89
 28  68  87  83  74  75  78  87
 74  83  88  85  85  69  88  86
----------
SMOKE ALARM !!
TimeDiff: 63
 69 129 178 220 185 131  89  34
 52  76 122 195 156  89  90  42
 56  68  74 187 201  91  74  65
 40  42  46  83  85  71  83  89
 27  68  87  83  74  75  79  87
 74  83  88  85  85  69  88  87
----------
SMOKE ALARM !!
TimeDiff: 4
 69 132 168 210 174 126  89  34
 51  75 117 202 158  87  90  42
 56  68  73 187 203  90  74  65
 40  42  46  83  85  71  83  90
 27  68  87  83  74  75  79  87
 74  83  88  85  85  69  88  86
----------
SMOKE ALARM !!
TimeDiff: 7
```

Figure 3.44: A video-frame recorded with the ASUS Xtion Pro subdivided into regions, where the average grayscale value of the pixel inside every region is calculated. The command-window shows every mean value calculated in every Region $Reg_{(x,y)}$. It can be seen that regions covered in smoke have a higher mean value than regions without smoke (values highlighted in red).

3.6.5 Regional Grayscale Difference

This feature is used, when a the average grayscale value of a Region $Reg_{(x,y)}$ - calculated at the beginning of the detection - is below a decision threshold. This means that this region contains a lot of dark colors. Due to the fact that smoke has a specific color, as described in Section 'Smoke Color Range', an increasing value (when comparing the average grayscale value of the background image with the average grayscale value of the current image) is an indicator for smoke covering the background. Therefore, if the difference of the average grayscale values is above a decision threshold, this region is marked as $D_{GreyDiff(Reg_{(x,y)})} = 1$.

3.6.6 Decreased Amount of Edges

Smoke covers objects and structures located behind it. Due to this fact, for every Region $Reg_{(x,y)}$ the decreasing amount of edges is calculated [13]. Therefore, an initial background image is reduced to the binary representation of its edges by using the *Canny edge detector*. So for every Region the initial amount of pixel representing edges is stored. For every frame, the current amount of edges inside a region is calculated using the same Canny-operator as used for the background image. If the amount of edges inside a region is decreased by more than 70% the examined region is considered to be a smoke-region and therefore marked with $D_{EdgeDiff(Reg_{(x,y)})} = 1$. If the calculated amount of edges for a Region $Reg_{(x,y)}$ is below a decision threshold this feature is not used for the specific region. Figure 3.45 shows a RGB frame containing smoke and its corresponding images containing the edges calculated initially and the edges of the current frame. It can be seen that, due to the covering characteristics of smoke, the amount of detected edges decreases inside the smoke region.

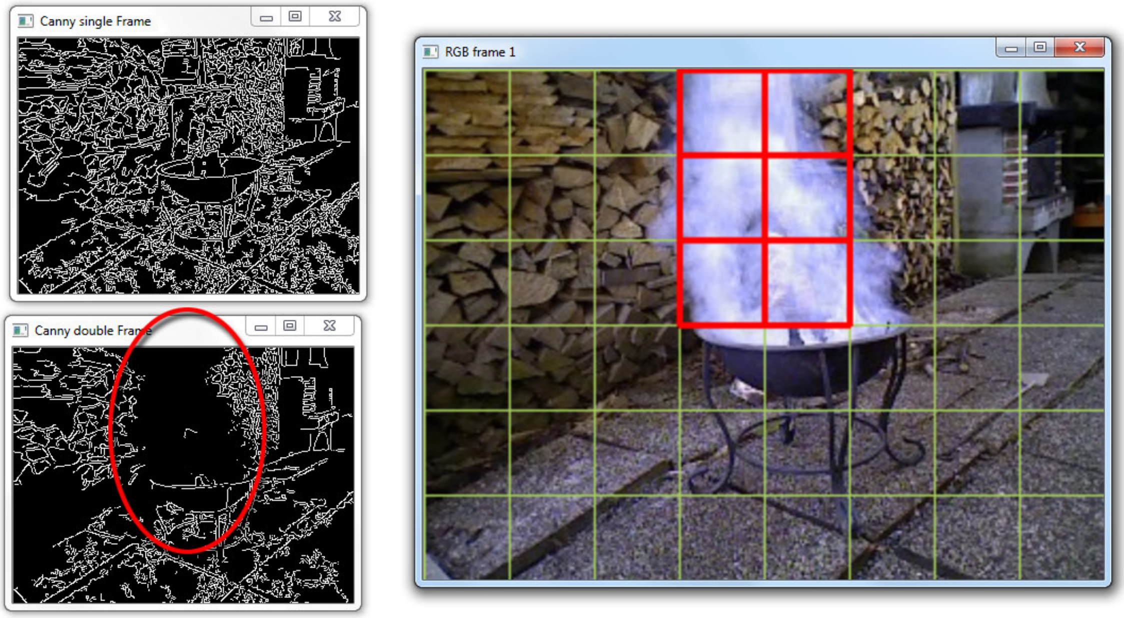


Figure 3.45: On the right side a RGB frame recorded by the ASUS Xtion Pro is shown. On the left side on top, the initially generated background image containing edges only is shown. The image below shows the edge-detection done by the same Canny-algorithm for the current frame. The region where smoke covers a significant amount of edges is marked in red.

3.6.7 Decision Fusion per Frame

The naive thresholds (as described in the Sections 'Movement and Color', 'Average Grayscale Value', 'Regional Grayscale Difference' and 'Decreased Amount of Edges') are used for the decision, if smoke is considered detected in the current frame or not. Therefore, the final decision is made by a voting-based approach where the decision - if smoke is detected or not - is made by an unanimous vote, considering all three features, leading to the decision

$$D_{Final(Reg)} = D_{Col(Reg)} \wedge D_{Grey(Reg)} \wedge D_{GreyDiff(Reg)} \wedge D_{EdgeDiff(Reg)}, \tag{3.16}$$

for every region. If the decision for at least one Region $D_{Final(Reg_{(x,y)})}$ is set to 1 the current frame is considered as *containing smoke*.

3.6.8 Final Decision

The history of the last $45 frames$ for every Region $Reg_{(x,y)}$ is stored in an array. If the amount of smoke detections per region exceeds the decision threshold of 50%, meaning that the analysis of at least $23 frames$ out of the last $45 frames$ must have led to the detection of smoke in the corresponding region, this region is labeled as a final smoke-region. If at least three different regions fulfill this requirement at the same time the overall fire-alarm is raised.

CHAPTER 4

Evaluation

4.1 Experimental Results

In this section, the performances of the proposed methods are analyzed. Firstly, the detection rates (*false-positives*, *false-negatives*, *true-positives* and *true-negatives*) are explained. Secondly, these detection rates are evaluated. After that, the relevance of the results is measured by calculating *Precision*, *Recall* and the *F1-Score*. In addition, impacts on the detection-rates by using the depth-sensor also for fire recognition are examined in detail.

4.1.1 Precision, Recall and F1-Score

The values *Precision* and *Recall* measure the relevance of the results achieved by an algorithm [14]. The results provided have to be binary - in this case for example *frame contains fire* or *frame does not contain fire*. Therefore, every single result can be (manually) classified as one out of four classes:

- *True Positives* (TP) - A fire inside a frame was correctly detected [41].
- *True Negatives* (TN) - The absence of fire inside a frame was correctly detected [41].
- *False Positives* (FP) - A fire inside a frame was wrongly detected [41].
- *False Negatives* (FN) - The absence of fire inside a frame was wrongly detected [41].

With these classes, the values Precision and Recall can be calculated by [14]

$$Precision = \frac{\#TruePositives}{\#TruePositives + \#FalsePositives} \tag{4.1}$$

and [14]

$$Recall = \frac{\#TruePositives}{\#TruePositives + \#FalseNegatives}. \tag{4.2}$$

The resulting value of Equations 4.1 and 4.2 lies within the range of 0.0 to 1.0, where a (perfect) precision-score of 1.0 means that each fire-incident detected was a real fire event while a (perfect) recall-score of 1.0 means that all fire-incidents have been detected [14]. Both values can be combined to a resulting (single) value that represents the quality of the retrieval, called *F1-Score* [14]. Here, the values for precision and recall are merged to their weighted average by

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}, \tag{4.3}$$

resulting again in a value between 0.0 and 1.0, where the optimal F1-Score is 1.0 [14].

4.1.2 Detecting Fire

In this section, the performance of the proposed method for fire-detection, using an ASUS Xtion Pro, is evaluated and the experimental results are given. 24 videos with 10.536 frames have been analyzed, with 3 videos containing real fire events. Due to the fact that this methods aims on detecting fire in the home-environment, the non-fire videos are containing for example people moving, sitting or dancing. Also, events with a fire-like appearance are evaluated - like someone dancing in a fire-colored shirt or a television showing the recorded video of a fireplace. Figure 4.1 shows sample frames from four selected videos. All videos have been manually analyzed and all frames have been labeled if they are containing fire or not. After that, the results of the fire-detection of the color-sensor is used to calculate the *True-Positive* rate (TP, fire is correctly assumed in a frame), the *True-Negative* rate (TN, the absence of fire is correctly assumed in a frame), the *False-Positive* rate (FP, fire is wrongly assumed in a frame) and the *False-Negative* rate (FN, the absence of fire is wrongly assumed in a frame).

RGB Detection

In this section, the detection rate using the information provided by the color-sensor only is analyzed. Therefore, only the features described in Section 'RGB Camera' are used to analyze the video-frames and to decide if fire is represented in the overall frame or not. Table 4.1 shows the results of the fire detection using the RGB sensor only. Here, especially video 1 and video 6 lead to a high amount of false-positives, because video 1 is the record of a TV-screen showing a fireplace while video 6 shows someone dancing in a fire-colored T-Shirt. The retrieval-scores *Precision*, *Recall* and the *F1-Score* that the approach achieves are listed in Table 4.2. These scores result in using the RGB-sensor only, without additional validation by the depth-sensor analysis.

Combined Detection

It this section, the information provided by the depth-sensor is merged with the information provided by the RGB sensor leading to a *multi-sensor fusion*. The primary detection of fire is made by the RGB-Sensor, the final validation if the detected fire is in a plausible area, is made by the depth-sensor. The idea is that this approach decreases the overall false-positive rate. Table 4.3 shows the detection-rates when using both implemented methods combined. An significant

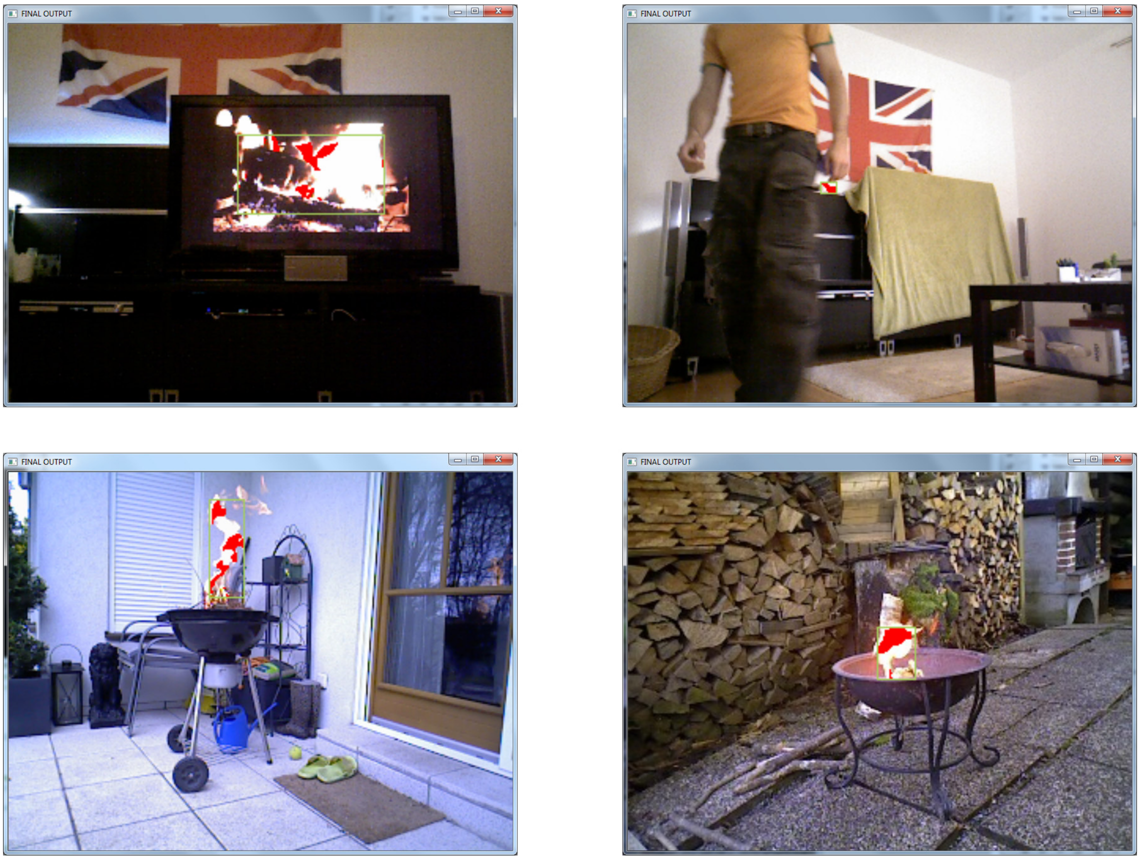


Figure 4.1: The top row shows frames from two videos that include motion and objects with a fire-like appearance. The image on the top-left shows the video of a fireplace replayed on a television. The image on the top-right shows someone dancing while wearing a fire-colored shirt. Both sample images on the bottom show real fire events. All videos have been recorded by the ASUS Xtion Pro.

decrease of the FP-rate is achieved. The retrieval-scores *Precision*, *Recall* and the *F1-Score* in Table 4.4 are the result of using the combined multi-sensor fusion by using the RGB-sensor to detect fire-events with an additional plausibility-check by the depth-sensor.

Table 4.1: The results of the fire-detection algorithm using the information extracted by the RGB sensor only. The left part of the table shows the number of detected frames, the right side the detection-rates in percent.

RGB Sensor								
	TN	TP	FN	FP	TN	TP	FN	FP
Fire-Television	177	0	0	574	23.57%		0.00 %	76.43%
Ext.d-Fire Closeup	757	0	0	0	100.00%		0.00%	0.00%
Fire-BBQ	202	665	17	1	99.51%	97.51%	1.92%	0.11%
Fire-Bowl	752	1772	125	34	95.67%	93.41%	4.66%	1.27%
Fire-Bowl II	0	348	9	0		97.48%	2.52%	0.00%
Fire-Colored Dance	585	0	0	269	68.50%		0.00%	31.50%
Ambient-Env. I	114	0	0	0	100.00%		0.00%	0.00%
Ambient-Env. II	60	0	0	5	92.31%		0.00%	7.69%
Ambient-Env. III	104	0	0	1	99.05%		0.00%	0.95%
Ambient-Env. IV	114	0	0	0	100.00%		0.00%	0.00%
Ambient-Env. V	123	0	0	0	100.00%		0.00%	0.00%
Ambient-Env. VI	127	0	0	2	98.45%		0.00%	1.55%
Ambient-Env. VII	139	0	0	1	99.29%		0.00%	0.71%
Ambient-Env. VIII	93	0	0	7	93.00%		0.00%	7.00%
Ambient-Env. IX	39	0	0	0	100.00%		0.00%	0.00%
Breakfast Preparing	362	0	0	37	90.73%		0.00%	9.27%
Working with refl. Mat.	404	0	0	32	92.66%		0.00%	7.34%
Physiotherapy I	323	0	0	36	89.79%		0.00%	10.03%
Physiotherapy II	402	0	0	21	95.04%		0.00%	4.96%
Living Room	1175	0	0	5	99.58%		0.00%	0.42%
Skeleton Test	39	0	0	0	100.00%		0.00%	0.00%
Home Env. I	89	0	0	0	100.00%		0.00%	0.00%
Home Env. II	10	0	0	0	100.00%		0.00%	0.00%
Dark-Colored Dance	385	0	0	0	100.00%		0.00%	0.00%
Average					91.29%	96.13%	0.57%	7.95%

Table 4.2: The achieved values for precision, recall and the F1-score for all videos, using only the information provided by the RGB-Sensor of the ASUS Xtion Pro, without additional validation by the depth-sensor.

Name	**Score**
Precision	0.73097
Recall	0.94857
F1-Score	0.82567

Table 4.3: The results of the fire-detection algorithm using the information extracted by the RGB sensor with additional validation by the depth-sensor. Left part of the table shows the number of detected frames, the right side the detection-rates in percent.

	Combined Sensors (RGB+Depth)							
	TN	TP	FN	FP	TN	TP	FN	FP
Fire-Television	749	0	0	2	99.73%		0.00 %	0.27%
Ext.-Fire Closeup	757	0	0	0	100.00%		0.00%	0.00%
Fire-BBQ	202	641	41	1	99.51%	93.99%	4.63%	0.11%
Fire-Bowl	781	1695	205	2	99.74%	89.21%	7.64%	0.07%
Fire-Bowl II	0	293	64	0		82.07%	17.93%	0.00%
Fire-Colored Dance	854	0	0	0	100.00%		0.00%	0.00%
Ambient-Env. I	114	0	0	0	100.00%		0.00%	0.00%
Ambient-Env. II	65	0	0	0	100.00%		0.00%	0.00%
Ambient-Env. III	105	0	0	0	100.00%		0.00%	0.00%
Ambient-Env. IV	114	0	0	0	100.00%		0.00%	0.00%
Ambient-Env. V	123	0	0	0	100.00%		0.00%	0.00%
Ambient-Env. VI	129	0	0	0	100.00%		0.00%	0.00%
Ambient-Env. VII	140	0	0	0	100.00%		0.00%	0.00%
Ambient-Env. VIII	100	0	0	0	100.00%		0.00%	0.00%
Ambient-Env. IX	39	0	0	0	100.00%		0.00%	0.00%
Breakfast Preparing	393	0	0	6	98.50%		0.00%	1.50%
Working with refl. Mat.	436	0	0	0	100.00%		0.00%	0%
Physiotherapy I	356	0	0	3	99.16%		0.00%	0.84%
Physiotherapy II	416	0	0	7	98.35%		0.00%	1.65%
Living Room	1175	0	0	5	99.58%		0.00%	0.42%
Skeleton Test	39	0	0	0	100.00%		0.00%	0.00%
Home Env. I	89	0	0	0	100.00%		0.00%	0.00%
Home Env. II	10	0	0	0	100.00%		0.00%	0.00%
Dark-Colored Dance	385	0	0	0	100.00%		0.00%	0.00%
Average					99.76%	88.42%	1.26 %	0.20%

Table 4.4: The achieved values for precision, recall and the F1-score - using both sensors of the ASUS Xtion PRO. The combination of the fire-detection using the depth-sensor in addition to the color-sensor leads to the *Multi-Sensor Fusion*.

Name	Score
Precision	0.99021
Recall	0.89452
F1-Score	0.93994

Comparison of Rates: RGB and Multi-Sensor Detection

In this section, the detection-rates of both methods are compared. After that, the impact of using test-videos providing special events, like an already burning-fire before the fire-detection was started or someone walking within the minimum distance-limit from the ASUS Xtion Pro, are described further. Table 4.5 shows the detection rates, as described in Section ’RGB Detection’ and Section ’Combined Detection’, side-by-side for a better comparison, it shows the correct dismissed FPs as well as the decreasing TP-rate.

Table 4.5: The results of the fire-detection algorithm using the information extracted by the RGB sensor only is compared to the detection-rates when using the *Multi-Sensor Approach* (Using the results calculated by the RGB sensor with additional validation by the depth-sensor).

	RGB only				**RGB and Depth**			
	TN	TP	FN	FP	TN	TP	FN	FP
Fire-TV	23.57%		0.00 %	76.43%	99.73%		0.00 %	0.27%
Ext-Fire Close.	100.00%		0.00%	0.00%	100.00%		0.00%	0.00%
Fire-BBQ	99.51%	97.51%	1.92%	0.11%	99.51%	93.99%	4.63%	0.11%
Fire-Bowl	95.67%	93.41%	4.66%	1.27%	99.74%	89.21%	7.64%	0.07%
Fire-Bowl II		97.48%	2.52%	0.00%		82.07%	17.93%	0.00%
Fire-Col. Dance	68.50%		0.00%	31.50%	100.00%		0.00%	0.00%
Amb.-Env. I	100.00%		0.00%	0.00%	100.00%		0.00%	0.00%
Amb.-Env. II	92.31%		0.00%	7.69%	100.00%		0.00%	0.00%
Amb.-Env. III	99.05%		0.00%	0.95%	100.00%		0.00%	0.00%
Amb.-Env. IV	100.00%		0.00%	0.00%	100.00%		0.00%	0.00%
Amb.-Env. V	100.00%		0.00%	0.00%	100.00%		0.00%	0.00%
Amb.-Env. VI	98.45%		0.00%	1.55%	100.00%		0.00%	0.00%
Amb.-Env. VII	99.29%		0.00%	0.71%	100.00%		0.00%	0.00%
Amb.-Env. VIII	93.00%		0.00%	7.00%	100.00%		0.00%	0.00%
Amb.-Env. IX	100.00%		0.00%	0.00%	100.00%		0.00%	0.00%
Breakf. Prep.	90.73%		0.00%	9.27%	98.50%		0.00%	1.50%
W. w. refl. Mat.	92.66%		0.00%	7.34%	100.00%		0.00%	0%
Physioth. I	89.79%		0.00%	10.03%	99.16%		0.00%	0.84%
Physioth. II	95.04%		0.00%	4.96%	98.35%		0.00%	1.65%
Living Room	99.58%		0.00%	0.42%	99.58%		0.00%	0.42%
Skeleton Test	100.00%		0.00%	0.00%	100.00%		0.00%	0.00%
Home Env. I	100.00%		0.00%	0.00%	100.00%		0.00%	0.00%
Home Env. II	100.00%		0.00%	0.00%	100.00%		0.00%	0.00%
Dark-Col. Dance	100.00%		0.00%	0.00%	100.00%		0.00%	0.00%
Average	91.29%	96.13%	0.57%	7.95%	99.76%	88.42%	1.26 %	0.20%

Special Videos and Events Here, interesting events and significant values of for example false-positives are described further. The following enumeration explains selected videos:

- *Fire-Television* - Here, a recorded fire was played back on a television-screen, to test the behavior of both sensors when examining recorded (and therefore artificial) fire. This leads to a very high false-positive rate of 76.43% when using the RGB-sensor only.

- *Fire-Bowl II* - Here, the fire was already burning before the ASUS Xtion PRO was activated. This leads to an area in the depth-image where no depth could be calculated for the initial background-image. If the initial depth can not be calculated for a region (like in this case), fire inside this region can not be validated by the depth-sensor.

- *Fire-Colored Dance* - A person dancing in a fire-colored T-Shirt in front of a bright surface reflecting artificial light. This leads to a high-rate of false-positives detected by the RGB sensor.

- *Working with reflective Material* - Here, the light of an artificial light-source is reflected into the camera, generating false positives.

- *Physiotherapy I+II, Living Room* - Here, the person moving in the scene passes by the ASUS Xtion PRO inside its minimum-detection limit, generating large areas where no valid depth can be estimated.

Figure 4.2 shows four screenshots out of the described videos, where the impact of the special events in the RGB or Depth-Image can be seen side-by-side.

Decrease of the False-Positive Rate

A *false-positive* event is a detected fire-incident by the method, when there is no real-fire presented in the scene. In this case, a robust approach should be developed meaning that the false-positive rate should be near 0.00%. The basic approach proposed in this thesis is the fire-detection via analyzing the information provided by a RGB-Sensor. In addition, the depth-sensor is used to validate the fire-detection for the corresponding region. The idea is that this should lead to a decrease of the false-positive rate, while there should not be a significant decrease of the true-positive rate. Table 4.6 shows detected false-positives from the videos by using the color-sensor only and the detection-rate when using the combined approach. The FP-rate is significantly decreasing by a total of 97.46%, when using the multi-sensor fusion. In fact, in 14 videos false-positives occurred when considering only the information provided by the color-sensor. Only in two videos the FP-rate stayed the same (in video *Living-Room* the person moving through the scene passes the depth-sensor inside its minimal detection-distance, creating large areas where no depth could be calculated. Exactly in this moment the false-positives were triggered - see Figure 4.2 for an example image of the behavior of the depth-sensor when an object passes within its minimum distance). An increasing FP-rate could not be observed in any of the test-videos.

Table 4.6: The number of detected false-positives (FP) - using the color-sensor only - compared to the FP rate when using the multi-sensor fusion combining color- and depth-sensor. The corresponding decrease-rate is shown in the right column. It can be seen that the additional validation by the depth-sensor leads to a significant decrease of the false-positive rate, with an total of 97.46%.

	RGB only	**Combined**	
	FP (#frames)	FP (#frames)	**Decrease-Rate**(%)
Fire-TV	574	2	99.65%
Fire-BBQ	1	1	0.00%
Fire-Bowl	34	2	94.12%
Fire-Col. Dance	269	0	100.00%
Amb.-Env. II	5	0	100.00%
Amb.-Env. III	1	0	100.00%
Amb.-Env. VI	2	0	100.00%
Amb.-Env. VII	1	0	100.00%
Amb.-Env. VIII	7	0	100.00%
Breakf. Prep.	37	6	83.78%
W. w. refl. Mat.	32	0	100.00%
Physioth. I	36	3	91.67%
Physioth. II	21	7	66,67%
Living Room	5	5	0.00%
Sum	1025	26	97.46%

Decrease of the True-Positive Rate

When a method tries to decrease the false-positive rate, in the best case the true-positive rate stays the same. Here, the impact of the decreasing false-positive rate, as analyzed in Section 'Decrease of the False-Positive Rate', on the true-positive rate is analyzed. Table 4.7 shows how the detection-rate of real fire events (TP) is decreased also, when using the multi-sensor fusion. The overall decrease-rate of 5.60% is justifiable when comparing it to the achieved decrease-rate of 97.46% of false-positives using the multi-sensor approach.

Table 4.7: The number of correct identified fire-events when using the RGB-sensor only is compared to the multi-sensor approach. The overall decrease-rate is 5.60%.

	RGB only	**Combined**	
	TP (#frames)	TP (#frames)	**Decrease-Rate**(%)
Fire-BBQ	665	641	3.61%
Fire-Bowl	1772	1695	4.35%
Fire-Bowl II	348	293	15.80%
Sum	2785	2629	5.60%

Rates of Final Fire-Alarms Triggered

As mentioned in Section 'Final Decision', the overall fire-alarm is not triggered when a single fire event in one frame is recognized. This approach should reduce the possibility of false triggered fire-alarms. Therefore, only when 75% out of the last $30 frames$ are marked as frames containing fire the overall alarm is triggered. By using this final decision-threshold the detection rate of fire is 100%, evaluated by using the 24 videos from Table 4.1, while the rate of falsely triggered fire-alarms is 0.0%.

4.1.3 Detecting Smoke

In this section, the performance of the proposed method for smoke-detection, using an ASUS Xtion Pro, is evaluated and the experimental results are given. 18 videos have been analyzed, with 3 videos containing smoke events.

RGB Detection

In this section, the smoke detection rates - using the RGB sensor only - are evaluated further. The method, to detect smoke in the video-stream provided by the ASUS Xtion Pro, is described in Section 'Implemented Method for Smoke Detection' in detail. Table 4.8 shows the results of the smoke-detection using the RGB sensor only, leading to an overall score of 99.52% TNs. Only video *Fire-Bowl II* leads to a high amount of FP, because in this video a fire starts burning. Due to the typical growth rate of fire, as described in Section 'Fire Growth Rate', in the end of this video the fire is big enough to cover details from objects in the background. Due to the fact that fire is recorded as *nearly white pixels* by an ASUS Xtion Pro, as described in Section 'Fire Color Range', this leads to some false smoke detections at the end of the video. The retrieval-scores *Precision*, *Recall* and the *F1-Score* are calculated using the RGB-sensor only, because when detecting smoke the (additional) usage of the 3D depth-sensor does not provide useful results. Table 4.9 shows the retrieval-scores achieved by the proposed algorithm.

Combined Detection

Due to the fact that smoke does not lead to significant changes considering the 3D depth-sensor of the ASUS Xtion Pro, smoke is detected using the RGB camera only - without the multi-sensor approach. (see Section 'Depth Camera and Smoke' for details).

Rates of Final Smoke-Alarms Triggered

As described in Section 'Final Decision', the overall-smoke alarm is not triggered when a smoke event in a single frame is detected. Therefore, smoke has to be detected in three different regions (or more) for at least 23 out of the last $45 frames$. With this additional condition, the overall smoke-detection rate is 100%, using the all videos included in Table 4.8.

Table 4.8: The results of analyzing the smoke-detection algorithm proposed in this Thesis, using the RGB sensor only. The left part of the table shows the number of detected frames, the right side the detection-rates in percent.

RGB Sensor								
	TN	TP	FN	FP	TN	TP	FN	FP
Fire-Television	549	0	0	0	100.00%		0.00 %	00.00%
Fire-Bowl	1863	0	0	97	95.05%		0.00%	4.95%
Fire-Bowl II	265	0	0	3	98.88%		0.00%	1.12%
Fire-Colored Dance	606	0	0	3	99.51%		0.00%	0.49%
Ambient-Env. I	114	0	0	0	100.00%		0.00%	0.00%
Ambient-Env. II	69	0	0	0	100.00%		0.00%	0.00%
Ambient-Env. III	98	0	0	0	100.00%		0.00%	0.00%
Breakfast Preparing	364	0	0	1	99.73%		0.00%	0.27%
Working with refl. Mat.	392	0	0	1	99.75%		0.00%	0.25%
Living Room	886	0	0	14	98.44%		0.00%	1.56%
Skeleton Test	57	0	0	0	100.00%		0.00%	0.00%
Home Env. I	98	0	0	0	100.00%		0.00%	0.00%
Home Env. II	28	0	0	0	100.00%		0.00%	0.00%
Smoke I	64	484	6	0	100.00%	98.78%	1.08%	0.00%
Smoke II	24	0	0	0	100.00%	58.51%	30.95%	0.00%
Smoke III	7	345	52	0	100.00%	86.90%	12.87%	0.00%
Sitting	24	0	0	0	100.00%		0.00%	0.00%
Dark-Colored Dance	276	0	0	0	100.00%		0.00%	0.00%
Average					99.52%	81.40%	2.49%	0.48%

Table 4.9: The achieved values for precision, recall and the F1-score for smoke-detection, using the color-sensor of the ASUS Xtion PRO only.

Name	**Score**
Precision	0.88136
Recall	0.90112
F1-Score	0.89113

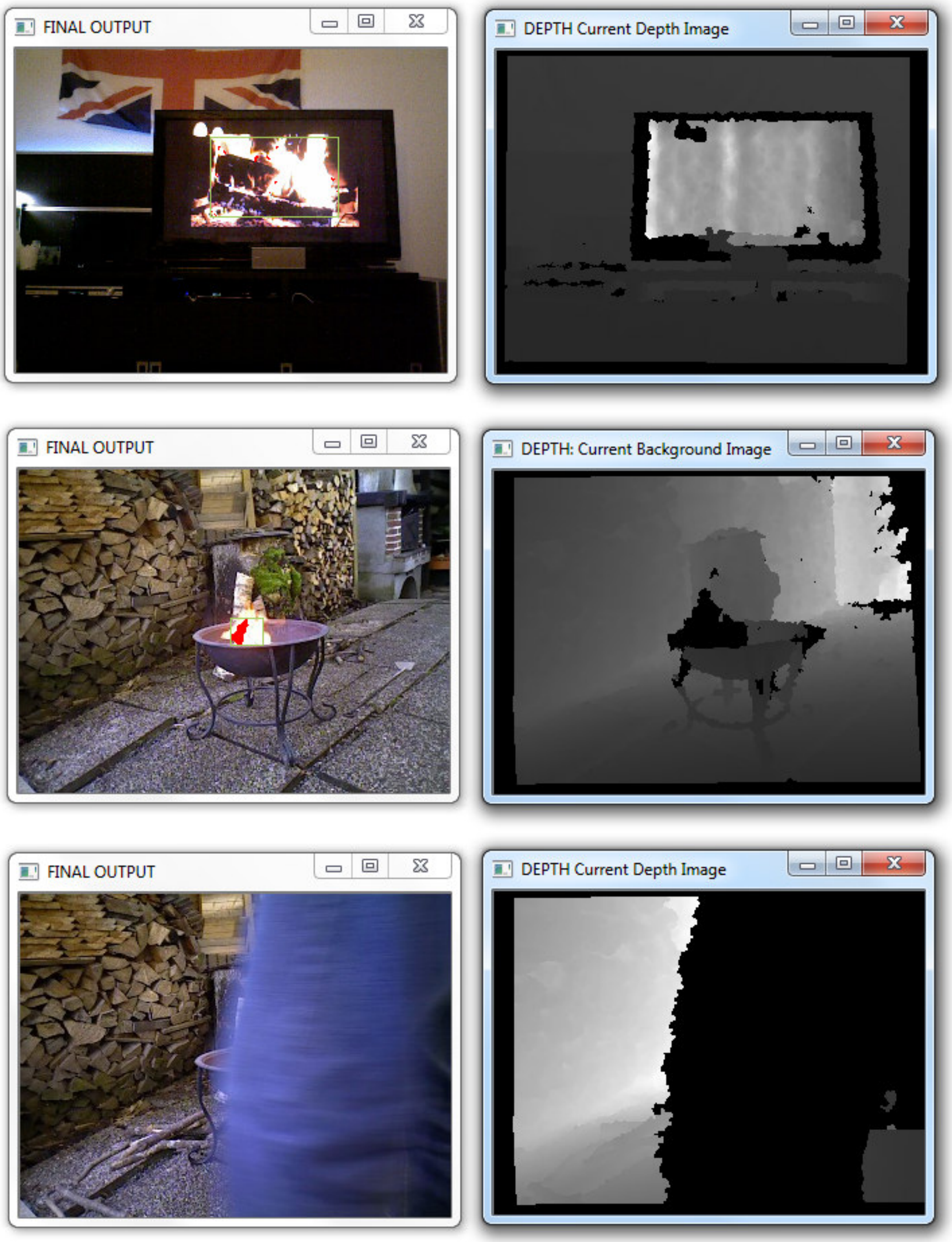

FINAL OUTPUT
DEPTH Current Depth Image
FINAL OUTPUT
DEPTH: Current Background Image
FINAL OUTPUT
DEPTH Current Depth Image

Figure 4.2: Left side: RGB-image. Right-side: Depth-image. The top row shows the impact of recorded fire to the color-sensor, while the depth-sensor is able to calculate a depth for the scene and therefore dismisses the fire-decision of the RGB-sensor. The second row shows the depth-image of the calculated *background-image* (not the current depth-frame as in row 1 and 3). Here, the missing region estimation can be clearly seen, because the recording of the fire was started after the fire was already burning. The last row shows the impact of a close-passing leg to the RGB-camera and the resulting problem estimating the depth in the depth frame.

4.2 Comparison with Related Work

In this section, the proposed methods for fire- and smoke detection are compared to already existing approaches. Due to the fact that using the depth map of the ASUS Xtion Pro for fire detection is a novel approach - and that the smoke detection algorithm does not include the 3D depth sensor - the comparison focus mainly on the different detection methods using the RGB sensor.

4.2.1 Fire Detection

For the proposed fire detection method, firstly motion is estimated by using frame differencing (similar as in for example [16], [21] and [27]). After that, fire colored pixel are extracted using a color restriction method (as in [5], [16] or [40]) with the difference that - due to the fact that the color camera of the ASUS Xtion Pro provides low quality RGB-images only - the color restrictions of those existing approaches can not be used for providing meaningful results. Therefore, a tailor-made color restrictions was created using sample-images extracted from videos containing fire incidents. The features calculated are the *mean-value*, the *standard-deviation* and the *skewness*, as for example in [5], but with the difference that only pixels located on the edge of the fire area are taken in consideration. This is necessary due to the fact that the areas inside fire regions (recorded by an ASUS Xtion Pro) contain nearly none spatial variation. The mean value and the standard deviation are calculated, after merging all three color channels to one (by reducing the evaluated pixels to grayscale), while the skewness is calculated using the values provided by the red color channel only. The decision - if fire is detected using the RGB sensor only - is calculated using all features. While the features *movement* and *color* have the power of veto the remaining features *mean*, *deviation* and *skewness* are merged to a final decision by using an adapted version of the *T-out-of-v method* [16]. In this case, the values taken in consideration are not binary values (weighted by a user defined value), here every feature provides a numeric result finally weighted and merged to the decision value. If this value lies above a decision threshold a frame is marked as a fire frame. Due to the fact that there was no method found using a depth image calculated by the *structured-light approach*, the detection of fire candidate areas by the 3D sensor is a novel approach. The multi sensor fusion itself is done similar as in [29], but in this case the RGB and the depth-image are merged (and not the RGB and the intensity-image). The rectification and alignment constraints are the same as in [29]. The technique that, when the color sensor detects fire in a specific area that the depth sensor checks this region regarding its plausibility, has also no equivalent in existing methods.

4.2.2 Smoke Detection

Here, motion is also the first feature extracted. Due to the fact that this is already calculated for the fire detection method, the computational effort is not increased. The next restriction is also *color*. As mentioned before the RGB camera of the ASUS Xtion Pro provides low quality images only, leading to the fact that existing smoke color ranges, as proposed in for example [21] or [39], can not be used for this method. Due to this reason, an own restriction is modeled by analyzing frames extracted from videos containing smoke recorded by the ASUS Xtion Pro.

Each frame is divided into sub-regions, as in [39], and for each region different features are calculated using integral images. Instead of converting the frames to the wavelet domain, as in [13], the feature for the typical smoothing behavior of smoke is calculated by spliting it into two separate features: firstly, the average pixel value of each region is calculated and compared to the initial generated value. Secondly, an edge detection algorithm is used to store the amount of initially detected edges of each region, which is compared to the current amount of edges detected for each frame later on. If the average pixel value is the same as for smoke covered regions and the amount of edges is additionally decreased, these features consider the presence of smoke inside the corresponding region. The final decision, if smoke is detected in a frame or not, is made by using a decision tree as proposed in [9].

CHAPTER 5

Summary and Future Work

5.1 Summary

In this section, the method for fire-detection and the method for smoke detection implemented for this thesis are summarized. After that, a conclusion, regarding how suitable it is using two sensors to detect fire, is given.

5.1.1 Fire-Detection

In this section, the approach to detect fire, using the *multi-sensor approach* by combining a 2D RGB-sensor and a 3D depth-sensor, is proposed. Both sensors are already combined in a Microsoft Kinect or the ASUS Xtion Pro. For this method, the ASUS Xtion Pro is used as the (stationary) hardware monitoring a scene. Due to the fact that this sensor estimates the depth of e.g. objects by using the *structured-light approach*, the proposed method is suitable for indoor fire-detection only, because direct sunlight interferes with the pattern emitted by the infrared sensor. For this method, fire is recognized using the RGB color sensor. When a potential fire area is located, the second sensor (namely the 3D depth sensor) is used to evaluate if a fire-detection at this location is plausible or not. To achieve this, firstly the information provided by the RGB sensor is restricted to areas that are considered as *moving* and fulfill a certain *color-restriction*. Secondly, different features are calculated and their outcome is merged to the overall decision, if and where fire is located in a video frame. The depth sensor also extracts regions with possible fire candidate areas, by detecting *motion* and areas with potential *new fire incidents*. By comparing the potential fire area, detected by the RGB sensor, with plausible fire areas calculated by the depth sensor algorithm, it is evaluated, if fire is represented in this frame. To achieve that, the frames of both sensors have to be synchronized, rectified and aligned accordingly to allow a useful *multi-sensor fusion*. The idea is that using this multi-sensor approach decreases the number of false-positives (FPs). In fact, the number of FPs have been reduced from 6.64% to 0.20% while the fire-detection itself (shown by the true-positive rate - TP) still provides reliable results. So, 97.46% of the FPs could be dismissed, while only 5.60% of the initial valid TPs -

detected by the RGB sensor - were (falsely) detected as *false-negatives* afterward by the multi-sensor approach. By adding the final restriction, that fire has to be detected for multiple frames to raise the overall fire alarm, every fire incident inside the test videos was recognized.

5.1.2 Smoke-Detection

Due to the fact that smoke does not lead to significant changes in the 3D depth-map, the smoke detection is implemented by using a single sensor only, namely the RGB-sensor. The RGB-camera used by the ASUS Xtion Pro provides only low-quality videos, therefore - when implementing a method for smoke detection only - using for example a high quality webcam is more recommended instead. To detect smoke, a frame is divided in an 8×6 grid, resulting in 48 different sub-regions. For each region restrictions like *motion*, *color* or the *decrease of edges* are calculated. If each feature calculated for a region lies above a decision threshold, the region is considered as a smoke region. Because of the restriction using one sensor only, the flow chart for smoke detection differs from the chart for fire detection, because steps like the image rectification or a synchronization of both sensors are not necessary here. By adding the final restriction that at least three different regions have to detect smoke over a given timeslot, every smoke incident was detected using the test videos.

5.1.3 Conclusion

Using sensors like the ASUS Xtion Pro to detect fire events provides useful results. Due to the fact that a false-triggered alarm can lead to high fines and penalties (when a fire-brigade is called, for example) it is preferable that the number of false-positives (FPs) stays low, while the TP-detection still provides useful results. This approach shows that the number of false-detected fire events can be significantly decreased by using the multi-sensor fusion. Evaluation showed that the ASUS Xtion Pro is very suitable for fire detection inside buildings. However, it suffers from restrictions like that sunlight interferes with the infrared light needed to estimate the depth map or the restrictions for the maximum and minimum distance of the depth-sensor, which prevents a usage for large covered areas (like tunnels). Smoke detection on the other hand can be achieved better by using a high quality webcam (mainly because the depth-sensor does not provide useful results and is therefore not used).

5.2 Discussion of Open Issues

Although the fire-detection using this multi-sensor approach works quite well, there are still areas for potential improvement. A better quality for the RGB camera would lead to more stable results, because it will be more simple to distinguish between real fire and for example fire colored objects. This happens due to the fact that e.g. the variety inside fire regions will be recorded better compared to using the currently available ASUS Xtion Pro. The recently announced Microsoft Kinect 2[1] - that will be available as a public beta in July 2014[2] - has the potential to overcome this problem. However, due to the fact that the Kinect 2 will be a TOF-sensor, a different approach to detect fire via a multi-sensor fusion is needed. Another field of improvement is that the proposed method does not detect objects that are passing the sensor within its minimum working distance. So an additional consideration if wide areas of the depth image became suddenly undetectable (regarding their depth) would prevent false raised alarms, because such a close pass generates areas where fire incidents are considered as *plausible* by the depth sensor analysis. On the other hand, the ASUS Xtion PRO provides only poor results for smoke detection, due to the fact that the depth sensor can not be used for meaningful smoke detection and the RGB camera itself does not provide high quality images. However, the proposed smoke detection method still contains areas for improvement. Firstly, the edge detection feature proposed provides only useful results in well structured scenes. Therefore, converting a frame to the wavelet domain (measuring the *high-frequency* areas of images, as generated by edges) and comparing the sub-regions regarding their wavelet energy could provide more useful results. This is caused by the fact that in this case not only *hard edges* would be considered in this comparison. In addition, the horizontal, vertical and diagonal wavelet transformed frames could still be calculated using the proposed *integral image* approach. Secondly, the regional grayscale difference does not provide useful information, when the RGB camera is facing for example a single gray-colored area. Therefore, a weakness of the provided smoke detection algorithm (using the RGB color sensor only) could occur, when a lot of the sub-regions (where features are calculated) are located on for example gray surfaces with less variation inside.

---

[1] Microsoft Developer Network - KINECT for Windows, `http://blogs.msdn.com/b/kinectforwindows/archive/2014/03/27/revealing-kinect-for-windows-v2-hardware.aspx`, Accessed: 29.06.2014

[2] Microsoft.com - KINECT for Windows, `http://www.microsoft.com/en-us/kinectforwindows/`, Accessed: 29.06.2014

Bibliography

[1] F. Yaacoub, Y. Hamam, A. Abche and C. Fares. Convex hull in medical simulations: A new hybrid approach. In *32nd Annual Conference on IEEE Industrial Electronics*, pages 3308–3313, 2006.

[2] K. J. Friston, C. D. Frith, R. J. Dolan, C. J. Price, S. Zeki, J. T. Ashburner and W. D. Penny. *Human Brain Function, 2nd Edition, Chapter 2*. Academic Press, 2004.

[3] M. Atefian and H. Mahdavi-Nasab. A robust mean-shift tracking using gmm background subtraction. *Journal of Basic and Applied Scientific Research*, pages 596–607, 2013.

[4] A. Somov, D. Spirjakin, M. Ivanov, I. Khromushin, R. Passerone, A. Baranov and A. Savkin. Combustible gases and early fire detection: an autonomous system for wireless sensor networks. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, pages 85–93, New York, 2010. ACM.

[5] P.V.K. Borges and E. Izquierdo. A probabilistic approach for vision-based fire detection in videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(5):721–731, 2010.

[6] Y. Xu C. Yu, C. Dian-Ren and Chen Lei. Otsu's thresholding method based on gray level-gradient two-dimensional histogram. In *2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR)*, volume 3, pages 282–285, March 2010.

[7] I. Chakraborty and T.K. Paul. A hybrid clustering algorithm for fire detection in video and analysis with color based thresholding method. In *International Conference on Advances in Computer Engineering (ACE)*, pages 277–280, 2010.

[8] S.C.S. Cheung and C. Kamath. Robust techniques for background subtraction in urban traffic video. In *Visual Communications and Image Processing*, volume 5308, pages 881–892. SPIE, 2004.

[9] G. Chung and E.W. Coiera. Are decision trees a feasible knowledge representation to guide extraction of critical information from randomized controlled trial reports? *BMC Medical Informatics and Decision Making*, 8:48, 2008.

[10] L. YunChang, Y. Chunyu and Z. Yongming. Nighttime video smoke detection based on active infrared video image. In *International Conference on Electrical and Control Engineering (ICECE)*, pages 1359–1362, June 2010.

[11] I. Culjak, D. Abram, T. Pribanic, H. Dzapo and M. Cifrek. A brief introduction to opencv. In *MIPRO 2012, Proceedings of the 35th International Convention*, pages 1725–1730, May 2012.

[12] H. Wang, A. Finn, O. Erdinc and A. Vincitore. Spatial-temporal structural and dynamics features for video fire detection. In *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, pages 513–519, Jan 2013.

[13] B.U. Toreyin et al. Wavelet based real-time smoke detection in video. *Signal Processing: Image Communication*, 20:255–26, 2005.

[14] H. Shen, J. Fang and J. Zhao. Efindbugs: Effective error ranking for findbugs. In *IEEE Fourth International Conference on Software Testing, Verification and Validation (ICST)*, pages 299–308, March 2011.

[15] J. Kober, M. Glisson and M. Mistry. Playing catch and juggling with a humanoid robot. In *12th International Conference on Humanoid Robots (Humanoids)*, pages 875–881, Nov 2012.

[16] B.U. Töreyin, Y. Dedeoğlu, U. Güdükbay and A.E. Çetin. Computer vision based method for real-time fire and flame detection. *Pattern Recognition Letters*, 27(1):49–58, January 2006.

[17] S. Verstockt, S. Van Hoecke, N. Tilley, B. Merci, B. Sette, P. Lambert, C. Hollemeersch and R. Van de Walle. *Hot topics in video fire surveillance*, pages 443–458. Video Surveillance. Intech, 2011.

[18] S. Verstockt, S. Van Hoecke, N. Tilley, B. Merci, B. Sette, P. Lambert, C.-F.J. Hollemeersch and R. Van de Walle. Firecube: A multi-view localization framework for 3d fire analysis. *Fire Safety Journal*, 46(5):262 – 275, 2011.

[19] B. Bilgic, B.K.P. Horn and I. Masaki. Efficient integral image computation on the gpu. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 528–533, June 2010.

[20] R.A. El-Laithy, J. Huang and M. Yeh. Study on the use of microsoft kinect for robotics applications. In *Position Location and Navigation Symposium (PLANS)*, pages 1280–1288, April 2012.

[21] T.-H. Chen, Y.-H. Yin, S.-F. Huang and Y.T. Ye. The smoke detection for early fire-alarming system base on video processing. In *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 427–430, 2006.

[22] A. AbuBaker, R. Qahwaji, S. Ipson and M. Saleh. One scan connected component labeling technique. In *IEEE International Conference on Signal Processing and Communications*, pages 1283–1286, 2007.

[23] C. Ha, G. Jeon and J. Jeong. Vision-based smoke detection algorithm for early fire recognition in digital video recording system. In *Seventh International Conference on Signal-Image Technology and Internet-Based Systems (SITIS)*, pages 209–212, 2011.

[24] L. Jianhua and M.-L. Liou. A simple and efficient search algorithm for block-matching motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(2):429–433, 1997.

[25] Y.-S. Lim, S. Lim, J. Choi, S. Cho, C.-K. Kim and Y.-W. Lee. A fire detection and rescue support framework with wireless sensor networks. In *International Conference on Convergence Information Technology*, pages 135–138, 2007.

[26] J.Y. Kwak, B.C. Ko and J.-Y. Nam. Forest smoke detection using ccd camera and spatial-temporal variation of smoke visual patterns. In *Eighth International Conference on Computer Graphics, Imaging and Visualization (CGIV)*, pages 141–144, Aug 2011.

[27] K.H. Cheong, B.C. Ko and J.Y. Nam. Vision sensor-based fire monitoring system for smart home. *International Conference on Ubiquitous Information Technology and Applications*, pages 1453–1463, 2007.

[28] D. Manolakis, S. Kogon and V. Ingle. *Statistical and adaptive signal processing : spectral estimation, signal modeling, adaptive filtering, and array processing*. Artech house signal processing library. Artech house, Boston, 2005.

[29] S. Verstockt, P. De Potter, S. Van Hoecke, P. Lambert and R. Van De Walle. Multi-sensor fire detection using visual and time-of-flight imaging. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2011.

[30] G. Flores, S. Zhou, R. Lozano and P. Castillo. A vision and gps-based real-time trajectory planning for a mav in unknown and low-sunlight environments. *Journal of Intelligent and Robotic Systems*, 74(1-2):59–67, 2014.

[31] R.C. Luo and K.L. Su. Autonomous fire-detection system using adaptive sensory fusion for intelligent security robot. *IEEE/ASME Transactions on Mechatronics*, 12(3):274–281, June 2007.

[32] H. Rapp U. Köthe B. Jähne M. Frank, M. Plaue and F.A. Hamprecht. Theoretical and experimental error analysis of continuous-wave time-of-flight range cameras. *Optical Engineering*, 48(1), 2009.

[33] M.-H. Hung, J.-S. Pan and C.-H. Hsieh. Speed up temporal median filter for background subtraction. In *First International Conference on Pervasive Computing Signal Processing and Applications (PCSPA)*, pages 297–300, 2010.

[34] L. Jinghong, L. Riqing and Z. Xiaohui. Design and research of video fire detection system based on fpga. In *Chinese Control and Decision Conference (CCDC)*, pages 1677–1679, 2011.

[35] N. Jamil, T. Mohd, T. Sembok and Z.A. Bakar. Noise removal and enhancement of binary images using morphological operations. In *International Symposium on Information Technology*, volume 4, pages 1–6, Aug 2008.

[36] P. Siciliano. Enabling technologies for ambient assisted living. Slides, 2010. Institute for Microelectronics and Microsystems CNR-IMM, Lecce (Italy).

[37] G.-F. He, S.-K. Kang, W.-C. Song and S.-T. Jung. Real-time gesture recognition using 3d depth camera. In *IEEE Second International Conference on Software Engineering and Service Science (ICSESS)*, pages 187–190, July 2011.

[38] M. Sonka and M.J. Fitzpatrick, editors. *Handbook of Medical Imaging Volume 2. Medical Image Processing and Analysis*. SPIE Press, Bellingham, Washington, 2004.

[39] S. Surit and W. Chatwiriya. Forest fire smoke detection in video based on digital image processing approach with static and dynamic characteristic analysis. In *First International Conference on Computers, Networks, Systems and Industrial Engineering (CNSI)*, pages 35–39, 2011.

[40] B.U. Toreyin and A.E. Cetin. Online detection of fire in video. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–5, 2007.

[41] M. Dolejsi, J. Kybic, S. Tuma and M. Polovincak. Reducing false positive responses in lung nodule detector system by asymmetric adaboost. In *5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 656–659, May 2008.

[42] V. Vasyukov and E. Kalennikova. An adaptive procedure of smoke and background discrimination in the early fire detection video system. In *6th International Forum on Strategic Technology (IFOST)*, volume 2, pages 844–847, 2011.

[43] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–511–I–518 vol.1, 2001.

[44] H. Haibing, W. Jinjun, F. Jun, L. Weiliang and Z. Yongming. Design a low power wireless fire detector based on cc430. In *International Conference on Intelligent Computation Technology and Automation (ICICTA)*, volume 1, pages 1107–1110, 2010.

[45] T.-H. Chen, P.-H. Wu and Y.-C. Chiou. An early fire-detection method based on image processing. In *International Conference on Image Processing*, volume 3, pages 1707–1710, 2004.

[46] Y. Xiao-Juan and F. Hong-Cai. An abrupt shot change detection algorithm based on the yuv space. In *International Conference on Electrical and Control Engineering (ICECE)*, pages 4630–4633, 2010.

[47] J. Han, L. Shao, D. Xu and J. Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE Transactions on Cybernetics*, 43(5):1318–1334, Oct 2013.

[48] X. Yu, P. Xue and Q. Tian. A statistical approach for object motion estimation with mpeg motion vectors. In *IEEE International Conference on Multimedia and Expo*, volume 1, pages 519–522, 2004.

[49] J. Cheon, J. Lee, I. Lee, Y. Chae, Y. Yoo and G. Han. A single-chip cmos smoke and temperature sensor for an intelligent fire detector. *IEEE Sensors Journal*, 9(8):914–921, 2009.

[50] F. Yuan. A fast accumulative motion orientation model based on integral image for video smoke detection. *Pattern Recognition Letters*, 29(7):925–932, May 2008.

[51] Z. Zhang. Microsoft kinect sensor and its effect. *IEEE MultiMedia*, 19(2):4–10, Feb 2012.

[52] Z. Xi, X. Fang, S. Zhen and M. Zhibin. Video flame detection algorithm based on multi-feature fusion technique. In *24th Chinese Control and Decision Conference (CCDC)*, pages 4291–4294, May 2012.