Brunel
UNIVERSITY
WEST LONDON

INTELLIGENT IMAGE CROPPING AND SCALING

A thesis submitted for the degree

of Doctor of

Philosophy

by

Joerg Deigmoeller

School of

Engineering and Design

24th January 2011

Abstract

Nowadays, there exist a huge number of end devices with different screen properties for watching television content, which is either broadcasted or transmitted over the internet. To allow best viewing conditions on each of these devices, different image formats have to be provided by the broadcaster. Producing content for every single format is, however, not applicable by the broadcaster as it is much too laborious and costly.

The most obvious solution for providing multiple image formats is to produce one high-resolution format and prepare formats of lower resolution from this. One possibility to do this is to simply scale video images to the resolution of the target image format. Two significant drawbacks are the loss of image details through downscaling and possibly unused image areas due to letter- or pillarboxes. A preferable solution is to find the contextual most important region in the high-resolution format at first and crop this area with an aspect ratio of the target image format afterwards. On the other hand, defining the contextual most important region manually is very time consuming. Trying to apply that to live productions would be nearly impossible.

Therefore, some approaches exist that automatically define cropping areas. To do so, they extract visual features, like moving areas in a video, and define regions of interest (ROIs) based on those. ROIs are finally used to define an enclosing cropping area. The extraction of features is done without any knowledge about the type of content. Hence, these approaches are not able to distinguish between features that might be important in a given context and those that are not.

The work presented within this thesis tackles the problem of extracting visual features based on prior knowledge about the content. Such knowledge is fed into the system in form of metadata that is available from TV production environments. Based on the extracted features, ROIs are then defined and filtered dependent on the analysed content. As proof-of-concept, this application finally adapts SDTV (Standard Definition Television) sports productions automatically to image formats with lower resolution through intelligent cropping and scaling. If no content information is available, the system can still be applied on any type of content through a default mode.

The presented approach is based on the principle of a plug-in system. Each plug-in represents a method for analysing video content information, either on a low level by extracting image features or on a higher level by processing extracted ROIs. The combination of plug-ins is determined by the incoming descriptive production metadata and hence can be adapted to each type of sport individually.

The application has been comprehensively evaluated by comparing the results of the system against alternative cropping methods. This evaluation utilised videos which were manually cropped by a professional video editor, statically cropped videos and simply scaled, non-cropped videos. In addition to and apart from purely subjective evaluations, the gaze positions of subjects watching sports videos have been measured and compared to the regions of interest positions extracted by the system.

Acknowledgment

First and foremost, in memory of a great person and colleague, I would like to thank my advisor at IRT, Dipl.-Ing. Gerhard Stoll. He always believed in me and gave me great support, even in busy times. Special thanks to my supervisor at Brunel University, Dr Takebumi Itagaki, for his valuable guidance and advice.

I wish additionally to thank Dipl.-Inf. Norbert Just, Dipl.-Ing. Oliver Bartholmes, Dipl.-Ing. Tim Weiß and all those people who helped along the way by contributing to this work. Special thanks go to Dipl.-Ing. Ralf Neudel and Dipl.-Inf. Matthias Laabs for the great time in and outside of IRT, as well as for their support in various ways. Last but not least, thanks to my friends and family for always encouraging me.

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1 Motivation

Nowadays, broadcasters distribute their services over various channels. In addition to the traditional broadcast via antenna, satellite or cable, content is provided to the viewer via internet streams, podcasts or adapted broadcast systems for mobile devices, the latter of which is a growing and quite promising market. Especially in Korea (T-DMB) and Japan (One-Seg, based on ISDB-T) the mobile TV market is gaining a substantial market share.

Key requirements for the success of mobile TV services are the adaptation of video content for optimal viewing conditions on mobile devices as well as an appropriate video quality. European mobile TV trials have shown that 24% of users stopped using the service because of quality issues (Arthur, 2007). The study indicates that there is a high demand for made-for-mobile, bite-sized content.

Adaptation of content for mobile devices should be more than just a replication of traditional linear TV content. Mobile TV has to attract an audience with new programming and viewing experiences in order to co-exist with traditional TV on stationary receivers. Watching TV on portable devices should be complementary to the trend towards larger displays at home, such as 42” or even 50” flat screen displays. Unfortunately, all too often, identical TV content is presented on the various distribution channels as the generation of specific content for mobile TV is very costly and time consuming for content providers. The creation of different video formats needs to be implemented at program production level which has direct implications for artistic design. Alternatively, content adaption can be performed manually during postproduction; however, this is not feasible for live productions.

With the introduction of HDTV productions, the effort for content adaptation for small displays increases further. As the scale factor between smallest and highest resolution increases significantly, the required relative cropping region size decreases for optimal viewing conditions. Consequently, greater attention has to be put on the correct focus for the most important areas instead of possibly cropping by static masks (e.g. centred pan & scan).

The proposed work addresses the problem of content adaptation by means of contextual automatic cropping. Compared to other works in the field of intelligent cropping and scaling, metadata information that is available from the broadcaster's production workflow is combined with video analysis methods here. The metadata information feeds the adaptation system with a priori knowledge about the content and is used to guide the feature extraction algorithms. By doing so, the algorithms become aware of the content properties and therefore work more efficiently and deliver more reliable results. If no metadata is available, the system can still be applied on any type of content in a default mode. In this case, salient regions are detected without any background knowledge.

The sample content used in this work consists of different types of SDTV sports productions. The sports genre is generally very popularly viewed on small screens. Compared to movies, sports do not need much contextual information. Hence, they can be watched simultaneously with other activities. Moreover, movies usually have a length of up to several hours, which also makes them less suitable for watching on small displays; a mobile TV user trial from 2006 has shown that the average duration per session is around 17 minutes (Lloyd, Maclean, & Stirling, 2006). As HDTV productions are not yet available for all types of sports production, the work has been carried out based on SDTV content. However, in principle, the presented approaches are also applicable to HDTV after adaption to the higher source format resolution.

1.2 Outline

The thesis is split into nine chapters. The second and third chapters examine the background of computer vision (Chapter 2) and visual attention (Chapter 3) which provide the basis for the technological methods used within this work.

Chapter 4 introduces typical image composition techniques used for TV productions. These techniques applied by cameramen already provide features in video content which point out the most important areas. In later chapters, these techniques are considered to choose appropriate content analysis methods.

In Chapter 5, comparable works from different areas are presented. Finally, their difference to this work is outlined and the advantages of the selected approach in the context of broadcast productions are presented.

Chapter 6 describes the role of metadata in TV productions and which standard has been used within this work to provide previous knowledge for content analysis. Relevant possibilities for content annotation are introduced and a software tool which was implemented to create metadata files for the selected standard is presented.

Chapter 7 elaborates the system implementation. First it discusses the technologies used and the system design. In the following sections, each module of the system and its purpose are explained in detail.

In Chapter 8, the results of the subjective system evaluation are presented based on 15 subjects. Additionally, the screen gaze positions of 10 subjects watching sports videos were measured and compared to the positions of the regions of interest extracted by the system. Deviations between these positions were used to provide a more objective predication of the system's accuracy.

Finally, Chapter 9 concludes the work by summarising the main concepts and discussing the strengths and limitations of the evaluated system. Based on these results an outlook on possible future work is given.

2. Background on Computer Vision

This chapter gives a brief overview of different computer vision methods which are important for this application. The main task for this application is to combine these methods in such a way that regions of interest (ROIs) can be identified in video images.

Within this work, several plug-ins have been implemented, where each plug-in contains a combination of computer vision methods in order to fulfil certain tasks, e.g. detecting moving regions. These modules can be applied in different combinations, dependent on the video content to be analysed.

Each video image that is analysed by a plug-in first passes through the process of *image enhancement* (Sections 2.2 and 2.3). This step prepares the image to support certain patterns in the image which are to be identified. In the next step, *pattern recognition* is applied by extracting certain features from the video image. Several methods serving as the basis for pattern recognition are introduced in Sections 2.4, 2.5 and in Section 3.5 of the following chapter. Then, to label the patterns found in the image as foreground and other signal components as background, *image segmentation* is applied (Section 2.6). Finally, regions which correspond to foreground are classified through *pattern classification* (Section 2.7). The order of these processing steps is common in computer vision (Toennies, 2005).

The use of certain computer vision methods for this implementation is justified in Chapter 7. The discussion of alternative methods in this chapter serves as the basis to weigh up the pros and cons of the application area of analysing broadcast video content.

2.1 Image Formation in Digital Television

In broadcast productions, the standard video systems are SDTV (Standard Definition Television) and HDTV (High Definition Television). A clear distinction between these two systems does not exist. Poynton classifies (Poynton, 2003) SDTV as any video system that has less than ¾ million pixels. Video systems which have more than ¾ million pixels and a native aspect ratio of 16:9, he classifies as a HDTV system.

In the context of this work, exclusively digital component video signals are of interest. Therefore, the author refers to (Poynton, 2003) and (Jack, 2001) for further information on analogue image formation of TV systems.

2.1.1 Studio colour encoding

An image sensor produces three colour channels in a range of red, green and blue (RGB). The values of each channel are proportional to the intensity of incidental light. In most imaging systems, these values are transferred to a nonlinear scale, which is motivated by human perception. This correction (*gamma correction* with an exponent of about 0.4) is denoted by $R'G'B'$. Additionally, human vision has a higher ability to perceive lightness than colour information (Poynton, 2003). Therefore, the ITU (International Telecommunication Union) standardised coefficients to decouple lightness from $R'G'B'$ for SDTV (Rec. ITU-R BT.601-5, 1995) and HDTV (Rec. ITU-R BT.709-3, 1998). Lightness is approximated by luma (Y') which is a weighted sum of $R'G'B'$:

$$Y'_{SDTV} = 0.299 \cdot R' + 0.587 \cdot G' + 0.114 \cdot B' \tag{2.1}$$

$$Y'_{HDTV} = 0.2126 \cdot R' + 0.7152 \cdot G' + 0.0722 \cdot B' \tag{2.2}$$

Luma should not be confused with luminance. Luminance is proportional to intensity and is not an approximation of perceptual response.

Besides luma, colour information is coded by difference components C_B and C_R, also called *chrominance*. They are formed in component digital video, Motion-JPEG, and MPEG by $B' - Y'$ and $R' - Y'$ (Poynton, 2003). To take advantage of the weakness of the human eye, chrominance components are sub-sampled. The notation of sub-sampled images is split into three digits $J\colon a\colon b$. This notation can be interpreted as a region that is J pixels wide and 2 pixels high. J has historically been set to 4, which represents the factor of 3.375 MHz to obtain the horizontal luma sampling rate. The second digit represents the number of chrominance samples (C_B, C_R) in the first line of J luma pixels (Kerr, 2009). Accordingly, the third digit represents the number of chrominance samples in the second line of J luma pixels.

Common sub-sampled images are encoded as 4:2:2 (half of chrominance resolution in the first and second line) or 4:2:0 (half of chrominance resolution in the first line and no chrominance information in the second line).

As mentioned above, the $Y'C_rC_b$ presentation of an image provides facilities to remove redundant information more easily than in the RGB presentation. For this reason, this $Y'C_rC_b$ is commonly used for video compressions or digital video exchange.

2.1.2 Studio image resolutions

A widely used technique for the scanning of images is *interlaced* scanning. By breaking a video frame into two *fields*, the number of images per second can be doubled by halving the vertical resolution of a video image. This approach was originally chosen to overcome the sensibility of the human eye to flicker, while keeping a given data rate. Scanning without interlace is called *progressive*.

The scanning notation of digital images is commonly denoted by the count of vertical pixel resolution (digital active lines), followed by p for *progressive* or i for *interlace* and the frame rate.

Two scanning standards that have been taken from analogue television broadcasting are globally used: $480i29.97$ systems are used especially in North America and Japan, and $576i25$ systems are used especially in Europe, Asia, Australia, Korea, and Central America. Commonly, both systems define a sampling rate of 13.5 MHz (4 x 3.375 MHz) per line, which results in 720 samples per digital active line (digital active line duration x sampling clock = 53.33 µs x 13.5 MHz = 720 samples/line). Alternatively, a sampling rate of 18 MHz can be used for an aspect ratio of 16:9, which results in 960 samples per digital active line (Rec. ITU-R BT.601-5, 1995). To overcome the problem of two different image aspect ratios for SDTV, a common technique is to use anamorphic lenses that squeeze an image from 16:9 to 4:3. By doing this, the vertical and horizontal resolution of an image is the same for both aspect ratios while the display aspect ratio differs. Therefore, the horizontal resolution of 720 samples/line is commonly used to store 4:3 *and* 16:9 images.

For HDTV, two different image resolutions of 1280x720 and 1920x1080 are defined in the broadcast environment. For Europe, the EBU (European Broadcast Union) specifies four production systems of $720p50$, $1080i25$, $1080p25$ and $1080p50$ (High Definition (HD) - Image Formats for Television Production, 2004).

2.1.3 Pixel aspect ratios

In (Rec. ITU-R BT.601-5, 1995), SDTV systems are defined by non square pixels. Non square pixels have a lower sampling rate in the horizontal than in the vertical direction. This sample pitch can be expressed by the *pixel*, i.e. *sample aspect ratio* (PAR/SAR) which is computed by the ratio between horizontal and vertical active image resolution multiplied by the aspect ratio of the intended display (*display aspect ratio*, DAR).

For an intended display of 4:3, this results in the following equation:

$$SAR_{SDTV} = DAR \cdot \frac{\text{horizontal image resolution}}{\text{vertical image resolution}} = \frac{3}{4} \cdot \frac{702}{576} = 0.914 \tag{2.3}$$

As the horizontal image resolution in SDTV is 720 by definition, only 702 samples/line are used in practice. This is due to the fact that the active analogue line length is 52 µs (52 µs x 13.5 MHz = 702 samples/line). The active line period of a digital signal starts 0.71 µs earlier and ends 0.62 µs later to avoid disturbances that might occur by converting analogue to digital signals (Technical Guidelines for the production of Television Programmes for ARD, ZDF and ORF, 2006).

Usually, computer graphics and flat screen TVs employ square pixels. Therefore, most of the non-square issues have been left behind with HDTV. HDTV is exclusively defined by square pixels in broadcast environments. Consequently, the calculation from Equation (2.3) for 1920x1080 results in:

$$SAR_{HDTV} = DAR \cdot \frac{\text{horizontal image resolution}}{\text{vertical image resolution}} = \frac{9}{16} \cdot \frac{1920}{1080} = 1.0 \tag{2.4}$$

2.2 Image Enhancement

Image enhancement is a step in image processing which modifies an image so that the result is more suitable for a specific application. Therefore, the main objective is to suppress non-relevant information and emphasize important image characteristics. This can be done either in the *frequency domain* or the *spatial domain*. This chapter deals with *image enhancement* in the spatial domain as these operations are faster than in the frequency domain as long as the considered patch size for filtering in an image area is comparably small.

Image enhancement methods can be divided into three groups (Toennies, 2005):

1. Point processing: change contrast by manipulating distances between grey values
2. Linear filters: consider neighboured pixels to suppress noise or highlight edges
3. Non linear filters: are constructed in such a way that disadvantages of linear filters are compensated

2.2.1 Point processing

Point operations compute new pixel values which are independent of the local structure in an image. The resulting pixel value a' of a point operation function $f(a)$ is influenced solely by the original pixel value located at the same position.

One of the simplest point operations is a *contrast stretching* that is an increase of the dynamic pixel value range by linear normalisation. Assuming that a_{low} and a_{high} are the lowest and highest pixel values in an image I, the contrast can be stretched to a new maximum level a_{max} and a new minimum level a_{min} by:

$$f_{cs}(a) = (a - a_{low}) \cdot \frac{a_{max} - a_{min}}{a_{high} - a_{low}} + a_{min} \tag{2.5}$$

In most cases, images are quantised by 8 bits. Therefore the maximum and minimum values of contrast stretching are usually set to $a_{min} = 0$ and $a_{max} = 255$. It has to be considered, that according to (Rec. ITU-R BT.601-5, 1995), the signal is only defined on 220 levels; from 16 (black) to 235 (white).

Another point operation is the *gamma correction*, already mentioned in Section 2.1.1. It improves local contrasts by a power function with an exponent referred to as *gamma*:

$$f_g(a) = a^{\gamma} \tag{2.6}$$

As a γ of 1 is simply a copy of each value, a $\gamma > 1$ weights higher values more than lower values. In the case of overexposed images, such a correction helps to get a better distribution over the pixel value range. In turn, if images are underexposed, a $\gamma < 1$ improves the distribution by weighting lower values more than higher values.

2.2.2 Linear filters

Linear filtering is a simple but very effective operation. It is also known as a *linear shift invariant operation*. The term *shift invariant* means that results are dependent on the pattern in an image neighbourhood, rather than on the operation itself. *Linear* means that an operation is homogeneous ($f(ax) = af(x)$) and additive ($f(x + y) = f(x) + f(y)$).

Linear filtering usually makes use of a locally applied pattern of weights. This pattern of weights is called a *kernel* and the process of applying this *kernel* is usually referred to as convolution. Given a *kernel* H, the convolution of image I with H results in the filtered image R.

$$R(x,y) = (H * I) = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} H(i,j) \cdot I(x-i, y-j) \qquad (2.7)$$

The convolution is denoted by the operator $*$. It is assumed that pixels beyond the *kernel H* are zero, because positions outside the matrix are irrelevant for the summation. Common *kernel* sizes are 3x3 or 5x5.

The weights of a kernel depend on the desired effect of the processed image. For example, a *blur filter* sums up all pixels covered by a *kernel* and divides the resulting value by the number of weights in the *kernel*. To highlight centred pixels and decrease the influence of pixels at the boundary of a certain region, a 2D *Gaussian function* (*Gaussian filter*) could be applied instead of a blur filter. Furthermore, *linear filtering* can be applied for *edge detection* by *kernels* weighting different orientations of edges (see Figure 2-1). For further information on *linear filtering* and its usage, the author refers to (Burger & Burge, 2008; Forsyth & Ponce, 2003).

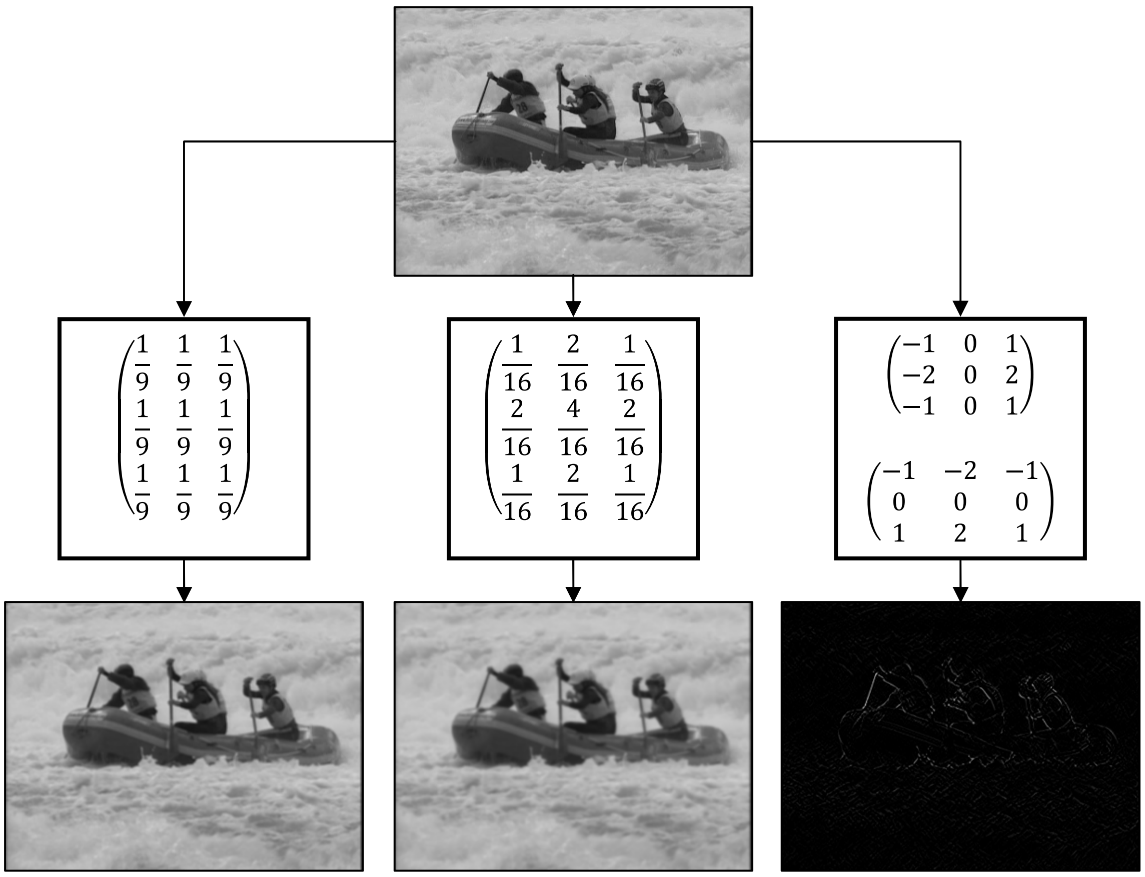

$\begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$
$\begin{pmatrix} \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \frac{2}{16} & \frac{4}{16} & \frac{2}{16} \\ \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \end{pmatrix}$
$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$
$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$

Figure 2-1: Applying linear filters by convolving with a mean filter kernel (left), a Gaussian filter kernel (centre) and Sobel operators for vertical and horizontal edge detection (right).

2.2.3 Non-linear filters

The advantage of *linear filters* is that noise can be reduced by smoothing the image. For example, if the noise that is present is a stationary additive Gaussian noise with a mean value of zero, then a symmetric Gaussian filter kernel might be the right choice. If on the other hand there is noise that is not stationary additive, no general statement about a noise distribution can be made. Such a type of noise is for example *salt and pepper noise*, which induces randomly occurring white and black pixels. Therefore, a kernel function is required that robustly estimates the statistical distribution in a pixel's neighbourhood. This can be achieved by *non-linear filters*.

Such a robust statistical estimator is the *median filter*. The median value is the centre position of neighboured pixel values that are sorted in ascending order. For a set of $2k + 1$ elements, the $k + 1^{\text{st}}$ element is chosen and in the case of $2k$ elements, the average of elements k and $k + 1$ is calculated. The way of applying the median filter on a 2D image is the same procedure as for convolution.

Having a neighbourhood $N_{x,y}$ centred at pixel position x, y, the filter can be expressed by:

$$R(x,y) = med\{\, I(i,j) | i,j \in N_{x,y}\} \tag{2.8}$$

where I is the input image and R is the resulting image. Figure 2-2 depicts the different effects of a median filter and a Gaussian filter applied on an image with salt and pepper noise.

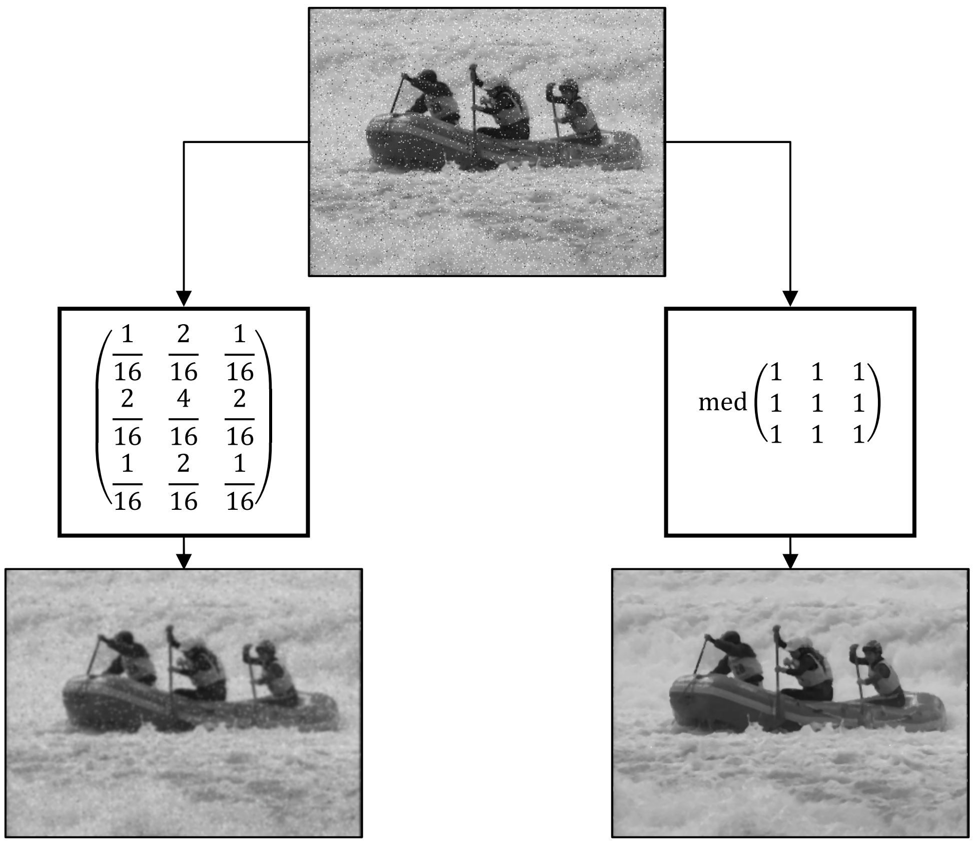

$\begin{pmatrix} \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \frac{2}{16} & \frac{4}{16} & \frac{2}{16} \\ \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \end{pmatrix}$
med $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$

Figure 2-2: Comparison of Gaussian filtering (left) and median filtering (right) applied on salt and pepper noise (top).

A median filter belongs to the group of morphological filters and is commonly applied on grey-scale images. Other morphological operations are *erosion* and *dilation*. For grey-scale image processing, they are quite similar to the median filter. Erosion applied on grey images sets the value of a pixel to a lower level than the median value or even the lowest available in the pixel's neighbourhood. This leads to a contraction of bright areas on a dark background. In turn, dilation chooses a value of higher rank than the median value which results in an expansion of bright areas on a dark background (Erhardt, 2008).

More powerful and less compute-intensive are morphological filters applied on binarised images. Instead of sorting pixels according to ascending values, erosion and dilation can now be computed by simple OR and AND operations with patterns consisting of zeros and ones (see Figure 2-3). Commonly, these patterns have a size of 3x3 or 5x5 pixels. Assuming that S is a structuring element, i.e. a pattern, and I is a binary image, erosion can be expressed as:

$$I \ominus S = \{p \,:\, S_p \wedge I\} \tag{2.9}$$

where S_p is the image obtained by shifting the structuring element S to the pixel p. In the same way, dilation can be described by an OR operation. This means that at least one element of I must match with S, so that the value at position p is kept:

$$I \oplus S = \{p \,:\, S_p \vee I\} \tag{2.10}$$

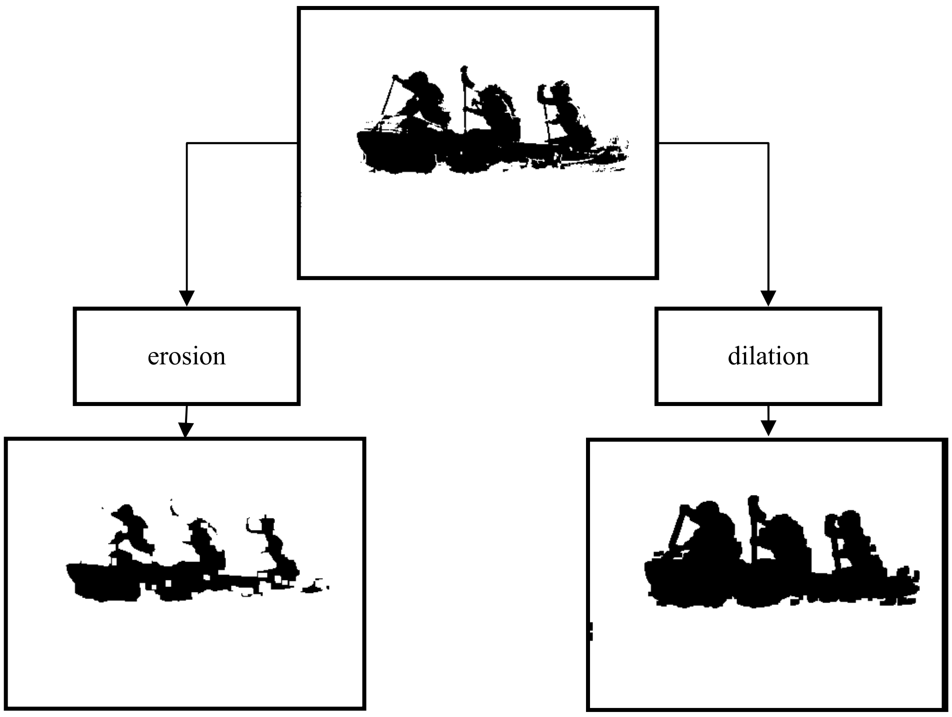


Figure 2-3: Comparison of erosion and dilation applied to a binarised image with a threshold of 100. Both structural elements of the morphological operations have a size of 3x3.

Erosion and dilation operations are not the inverse of each other. For example, applying erosion at first and dilation afterwards rounds object boundaries and removes connections between objects (opening). In turn, closing is a serial execution of dilation and erosion and brings blobs together or closes holes in objects that may remain after a segmentation process.

2.3 Geometric Operations and Interpolation

As will be explained in Section 2.4, scaling of an image requires previous filtering to avoid alias effects. Usually, this filtering process is not applied in the frequency domain, but rather by convolving the image with a kernel. Dependent on up- or downscaling, interpolation is able either to remove or to restore frequency components. Besides scaling, geometric operations such as shifting, rotating and shearing are frequently used as well. Therefore, interpolation plays an important role wherever raster images are deformed and remapped in order that transformed coordinates no longer fall onto a discrete raster.

In this section, a short introduction to 2D mapping functions is first given and then an overview of interpolation functions brings this section to a conclusion. Both topics form an important foundation for the implementations in this work.

2.3.1 Affine transformations

A geometric operation modifies the coordinates of a given image *I* to a new image *I'*:

$$I(x, y) \rightarrow I'(x', y') \tag{2.11}$$

Such a transformation can be applied by simple mapping functions that allow translation, scaling, shearing and rotation:

Translation
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} d_x \\ d_y \end{pmatrix} \tag{2.12}$$

Scaling
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.13}$$

Shearing $$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & b_x \\ b_y & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.14}$$

Rotation $$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.15}$$

These affine transformations can be simplified using homogenous coordinates. In other words, even translations are computed by one vector matrix multiplication. To convert 2D coordinates into homogeneous coordinates, each vector is extended by an additional absolute term:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{2.16}$$

Searching for an affine transformation from image I to I', the matrix of six unknowns has to be solved. This condition is satisfied if three pairs of corresponding points $(x_1, x'_1), (x_2, x'_2), (x_3, x'_3)$ are given. In turn, when applying an affine mapping on an image, all coordinates have to be multiplied by the matrix. An affine transformation is a three point mapping that transforms lines to lines, triangles to triangles and rectangles to parallelograms (see Figure 2-4).

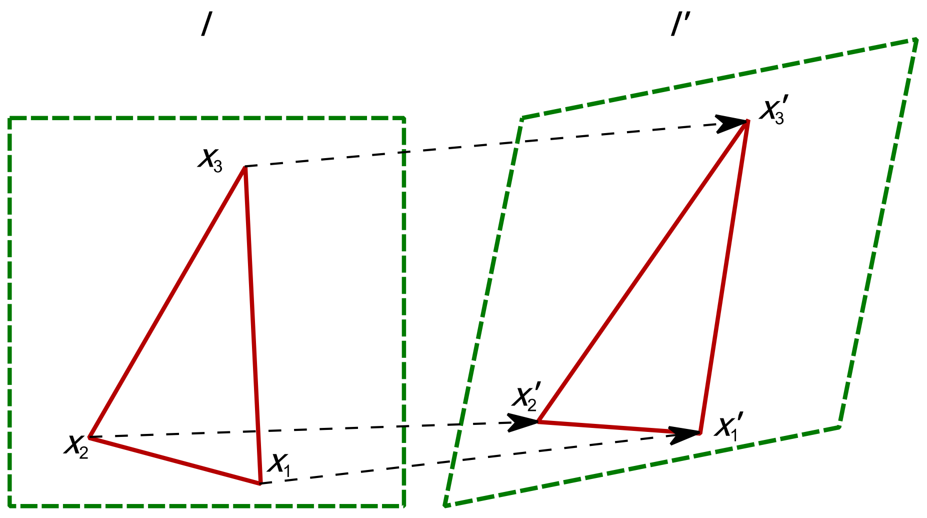


Figure 2-4: Affine 2D transformation using three pairs of corresponding points.

2.3.2 Interpolation operations

Interpolation is a process of regaining lost information in a sampled image or estimating a signal at continuous positions. This task arises in this work when image transformations are applied (two dimensional interpolation) or missing positions of ROIs or cropping areas are determined (one dimensional interpolation). For better illustration, operations are described below for one dimensional interpolation only.

An interpolation of a sampled signal can be presented as a convolution with a continuous function. Dependent on the convolution function, the original signal can be approximated more or less accurately. To address the question of which convolution function has to be chosen to restore specific information content, a look at the spectral domain gives an intuitive answer. Obviously, interpolation of sampled signals goes side by side with Shannon's sampling theorem. This means that a properly sampled signal must have been limited to frequencies not higher than half of the sampling frequency. Therefore, the accuracy of a reconstructed continuous signal is naturally constrained. To retain maximal information from the periodic Fourier Spectrum, an ideal low pass filter cuts off high image frequencies and keeps all lower frequency components. This multiplication in the frequency domain with a square window corresponds to a convolution in the time domain with a *Sinc* function $\left(\frac{\sin(\pi x)}{\pi x}\right)$ (see Figure 2-5).

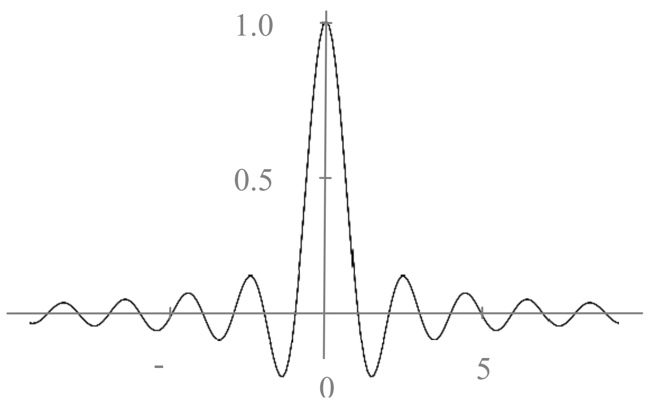


Figure 2-5: One dimensional Sinc function. It is 1 at the origin and has zero values at all integer positions.

Practically, this optimal interpolation is not satisfying for two reasons. Firstly, the convolution mask of a *Sinc* function does not simply include a few neighboured values, but infinitely many values whose influences decrease with higher distances to the origin. Secondly, due to rapid transitions or pulses of the *Sinc* function, disturbing ringing artefacts can occur for image interpolations. For this reason, the Sinc function is approximated by kernel functions dependent on the requirements of the applications. The most common kernels are introduced in the following sections.

2.3.2.1 ***Nearest-neighbour and linear interpolation***

The simplest interpolation function and thus the opposite extreme of Sinc interpolation is the nearest-neighbour interpolation. This approach sets the sample value of a continuous coordinate x to the closest sample value of the function which has to be interpolated. A more closely approximated course of the original function can be achieved by a linear interpolation. It describes the interpolated value by a linear function that is calculated by the two closest samples. The nearest-neighbour and linear interpolation kernel functions are depicted in Figure 2-6.

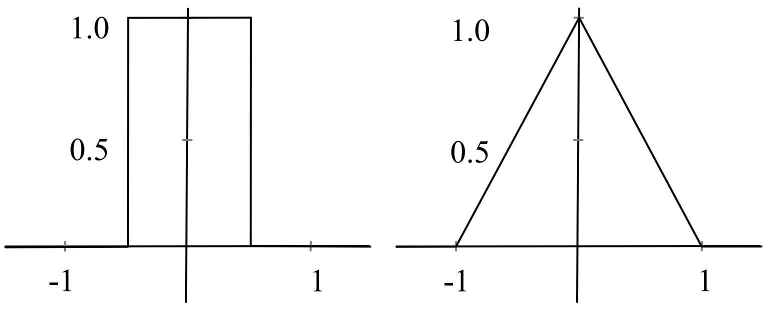


Figure 2-6: Convolution kernels of nearest-neighbour-interpolation (left) and linear interpolation (right).

2.3.2.2 ***Cubic spline interpolations***

Interpolations that come closer to the function course of a *Sinc* function are spline interpolations. An efficient and good approximation of the *Sinc* can be reached by a cubic spline. In general, its *kernel* consists of piecewise cubic polynomials, where each polynomial is controlled by two parameters a and b.

$$w_{cs}(x,a,b) = \frac{1}{6} \cdot \begin{cases} \begin{matrix} (-6a - 9b + 12) \cdot |x|^3 + \\ (6a + 12b - 18) \cdot |x|^2 - 2b + 6 \end{matrix} & for\ 0 \le |x| < 1 \\ \begin{matrix} (-6a - b) \cdot |x|^3 + (30a + 6b) \cdot |x|^2 + \\ (-48a - 12b) \cdot |x| + 24a + 8b \end{matrix} & for\ 1 \le |x| < 2 \\ 0 & for\ |x| \ge 2 \end{cases} \tag{2.17}$$

Specialisations of Equation (2.17) are defined by different values of a and b. Both parameters influence the slope of the function, which affects the amount of overshoot. In the case of image interpolation, overshoot has a strong influence on the sharpness of images. In turn, overshoot can cause ringing artefacts as well. Therefore, it is desirable to find a compromise between these two effects.

For example, parameters of *a = 0.5* and *b = 0* create overshoots that support sharpness in an image (Equation (2.18)). This setting, known as *Catmull-Rom* interpolation, can cause ringing artefacts, but delivers good results in smooth signal areas.

$$w_{cmr}(x) = w_{cs}(x, 0.5, 0) = \frac{1}{2} \cdot \begin{cases} 3 \cdot |x|^3 - 5 \cdot |x|^2 + 2 & for\ 0 \leq |x| < 1 \\ -|x|^3 + 5 \cdot |x|^2 - 8 \cdot |x| + 4 & for\ 1 \leq |x| < 2 \\ 0 & for\ |x| \geq 2 \end{cases} \tag{2.18}$$

Less an interpolation but more an approximation of function courses is the *cubic B-spline* interpolation. What is special about this operation is the fact that reconstructed signals do not necessarily go through all original points. Therefore, it is more a smoothing, comparable to a Gaussian smoothing, than an interpolation. The settings for *a* and *b* are *a = 0* and *b = 1* (Equation (2.19)).

$$w_{cbs}(x) = w_{cs}(x, 0, 1) = \frac{1}{6} \cdot \begin{cases} 3 \cdot |x|^3 - 6 \cdot |x|^2 - 4 & for\ 0 \leq |x| < 1 \\ -|x|^3 + 6 \cdot |x|^2 - 12 \cdot |x| + 8 & for\ 1 \leq |x| < 2 \\ 0 & for\ |x| \geq 2 \end{cases} \tag{2.19}$$

In turn, the *Mitchell-Netravali* interpolation provides the right balance between the two spline functions mentioned above. With a setting of $a = \frac{1}{3}$ and $b = \frac{1}{3}$, it supports sharpness and low ringing artefacts for image interpolations (Equation (2.20)).

$$\begin{aligned} w_{mn}(x) &= w_{cs}(x, \frac{1}{3}, \frac{1}{3}) \\ &= \frac{1}{18} \cdot \begin{cases} 21 \cdot |x|^3 - 36 \cdot |x|^2 + 16 & for\ 0 \leq |x| < 1 \\ -7 \cdot |x|^3 + 36 \cdot |x|^2 - 60 \cdot |x| + 32 & for\ 1 \leq |x| < 2 \\ 0 & for\ |x| \geq 2 \end{cases} \end{aligned} \tag{2.20}$$

2.3.2.3 *Lanczos interpolation*

The Lanczos interpolation kernel is not an approximation of the *Sinc* function by polynomials, but a *Sinc* function that is commonly limited to the second or third overshoot. Additionally, the *Lanczos* kernel is smoothed at its boundaries by a window function to lessen the influences of the overlapping truncated *Sinc* kernels. Due to the good approximations of the *Sinc* kernel by the *Catmull-Rom* or *Mitchell-Netravali* functions, the *Lanczos* interpolation does not offer much advantage.

2.4 2D Fourier Transform

Section 2.2 covered the topic of how to manipulate discrete images by *kernel functions*. Obviously, a discrete image does not provide a proper basis for all operations. For example, shrinking an image should not be done by simply removing each k-th pixel. In doing so, alias effects can become visible; this is caused by disregarding the shift of sampling frequency. This effect could be examined by *changing the basis*.

One way to change the *basis* into another that represents the frequency domain is the *Fourier transform*. The Fourier transform for sampled signals is the DFT (Discrete Fourier Transform). The two-dimensional DFT for a square image $g(x,y)$ of size N×N is defined by:

$$\mathcal{F}(g(x,y))(u,v) = \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} g(x,y)e^{-i2\pi\left(\frac{ux}{N}+\frac{vy}{N}\right)} \tag{2.21}$$

This conversion takes the basis of the complex valued function of x,y (complex valued function with zero imaginary components) and returns a complex valued function of u,v. The exponential is the basis of u,v and can be rewritten as:

$$e^{-i2\pi\left(\frac{ux}{N}+\frac{vy}{N}\right)} = cos\left(2\pi\left(\frac{ux}{N}+\frac{vy}{N}\right)\right) - i\,sin\left(2\pi\left(\frac{ux}{N}+\frac{vy}{N}\right)\right) \tag{2.22}$$

Equation (2.21) can be interpreted as follows: the value of each pixel $g(x,y)$ is calculated by multiplying the spatial image with the corresponding base function of u,v and summing the result. As can be seen in Equation (2.22), the complex number is composed of sine and cosine waves. The values of the trigonometric functions can be interpreted as measuring the amount of given frequency and orientation in the signal.

Consequently, each complex number of the sum represents a *spatial frequency component* whose precision depends on the number of pixels x,y used for the transformation, where $\mathcal{F}\big(g(x,y)(0,0)\big)$ describes the DC-component, i.e. the average brightness of the image, and $\mathcal{F}\big(g(x,y)(N-1,N-1)\big)$ describes the highest frequency.

To visualise each component, the complex function is usually inconvenient to draw. One possibility is to plot the real term and the imaginary term separately. Another option is to consider magnitude and phase of the complex numbers. Commonly, just the magnitude spectrum is chosen for visualisation, because it is in direct relation to the number of frequencies present in the image. Looking at the magnitudes of a transformed image, fast-diminishing values can be observed with increasing frequency. To counteract this effect, a logarithmic transformation is usually applied before plotting. Furthermore, for a better illustration, the magnitude spectrum is rearranged by shifting the origin of the four quadrants to the centre (see Figure 2-7).

To save computing power, the DFT can be simplified by a few mathematical tricks. First, the DFT is separable and can be calculated by two successive one-dimensional sums, instead of a double sum:

$$\mathcal{F}\big(g(x,y)\big)(u,v) = \frac{1}{N}\sum_{y=0}^{N-1} g'(u,y)e^{-i2\pi\frac{vy}{N}} \tag{2.23}$$

where $$g'(u,y) = \frac{1}{N}\sum_{x=0}^{N-1} g(x,y)e^{-i2\pi\frac{ux}{N}}$$

This reduces the computational complexity from $O(N^4)$ to $O(N^3)$ for a two-dimensional image with a size of NxN pixels (Toennies, 2005).

Further significant computational savings can be achieved by employing the *Fast Fourier Transform* (FFT). The total expense is finally $O(N^2 logN)$. For further information on FFT, the author refers to (Brigham, 1988).

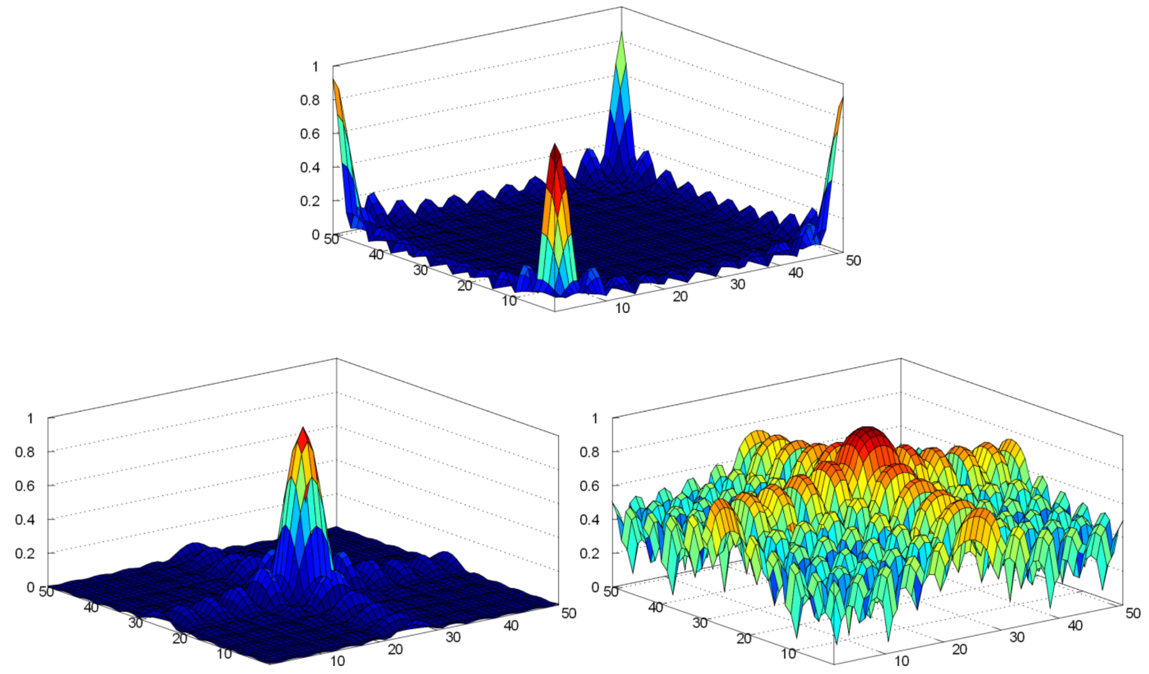


Figure 2-7: Mirrored magnitude spectrum after a DFT (top) and rearranged magnitude spectrum with origins shifted to the centre without taking the logarithm (bottom left) and with taking the logarithm (bottom right). All magnitude ranges are normalised to 0,…,1.

2.5 Motion Estimation

Motion contains important information for the fields of video compression and video processing. While video compression follows the approach of reducing the number of bits needed, image processing aims at quality improvement or motion-based video segmentation. In the context of this work, the focus is on motion-based segmentation, i.e. the local allocation of foreground and background in a video sequence. In the first part of this section, *local motion estimation* is discussed in order to obtain a motion vector field. Afterwards, an approach for global motion estimation from related vectors in a motion vector field is introduced.

2.5.1 Local motion estimation

Dependent on the area of the application, one local motion estimation method might be more appropriate than the other. In the case of video compression, the computed motion need not necessarily be the real motion as long as the encoded video maintains an acceptable quality and some minimum bit rate can be achieved. On the other hand, video processing relies on precise motion information. For example, if a missing video image is computed by motion-compensated temporal interpolation (e.g. for frame rate conversions), the estimated motion needs to resemble the true motion between video frames. Obviously, the true motion information is of more interest in this application.

This section gives an overview of the most common local motion estimation methods: block matching, phase correlation and optical flow.

2.5.1.1 ***Block matching***

Block matching is a widely used method in video compression to detect similar rectangular regions of intensity values in consecutive video frames. It is the simplest estimation of local motion, while at the same time producing quite accurate results dependent on the block size. The idea of block matching is to minimise the sum of an estimation criterion Φ at pixel position $\mathrm{x} = (x, y)^T$, applied on all blocks $\mathcal{B}_m$ in a pixel's neighbourhood $\mathcal{P}$:

$$\min_{\vec{\mathrm{u}} \epsilon \mathcal{P}} \mathcal{E}(\vec{\mathrm{u}}), \qquad \mathcal{E}(\vec{\mathrm{u}}) = \sum_{\mathrm{x} \epsilon \mathcal{B}_\mathrm{m}} \Phi(\varepsilon_t(\mathrm{x})) = \sum_{\mathrm{x} \epsilon \mathcal{B}_m} \Phi\big(I_t(\mathrm{x}) - I_{t+1}(\mathrm{x} - \vec{\mathrm{u}})\big) \tag{2.24}$$

where $\varepsilon_t(\mathrm{x})$ is the prediction error for pixel position x at time t and $\vec{\mathrm{u}}$ is the displacement vector for each compared block $\mathcal{B}_m$ at time $t + 1$.

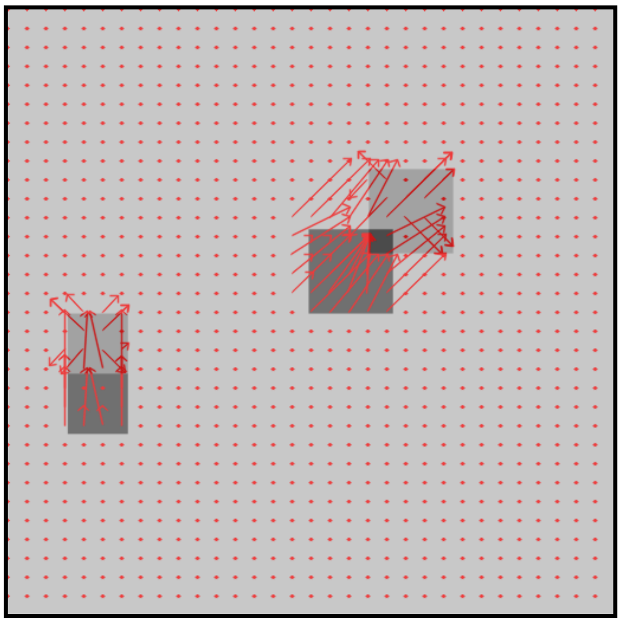


Figure 2-8: Motion vector estimation by applying block matching on two moving squares. The transparent squares describe the position in the second video frame. The size of the video frames is 512 x 512 pixels and the block size is 16 x 16 pixels.

To complete Equation (2.24), the estimation criterion Φ still has to be defined. Widely used criteria are the sum of squared difference (SSD) $\Phi\big(\varepsilon_t(\mathrm{x})\big) = \big(\varepsilon_t(\mathrm{x})\big)^2$ and the sum of absolute difference (SAD) $\Phi\big(\varepsilon_t(\mathrm{x})\big) = |\varepsilon_t(\mathrm{x})|$. More complex approaches, for example computing the median of squared errors, are rarely used in order to save computing power. Some speed optimisations can be addressed by, for example, sub-sampling (cf. colour subsampling, 2.1.1) or coarser quantisation stages (Bovik, 2009). Figure 2-8 shows the motion vector estimation of two moving squares by applying block matching.

2.5.1.2 ***Phase correlation***

The advantage of block matching is a precise estimation of shifted pixels. On the other hand, the disadvantage of block matching is the extensive search by checking many possibilities of local displacements. Phase correlation motion estimation cannot deliver precise information in the space-time domain as block matching does. In spite of that, phase correlation has the ability to identify present motion in the frequency domain robustly. Combining the two provides a promising approach to computing likely candidates of motion by phase correlation at first, and then using this information to limit the search area for block matching in the next step. The underlying idea of phase correlation is a cross correlation measurement between two video frames at time t and $t+1$ in Fourier-Domain:

$$C_{t,t+1}(u,v) = \hat{I}_t(u,v)\hat{I}^*{}_{t+1}(u,v) \tag{2.25}$$

where u, v are frequency components, $\hat{I}$ is the Fourier transformed image of I, and $\hat{I}^*$ is the conjugate complex of $\hat{I}$. In the Fourier domain, it is of great benefit that not only is intensity information available, but phase information as well. This fact is of great importance for a robust search for similar structures in consecutive video frames. For example, if the illumination of an object changes in video images, e.g. if an object moves into shade, intensity values might suddenly change whereas the structure remains. Motivated by this, phase correlation applies a normalisation by the magnitude on the cross correlation function. After an inverse Fourier transform, an image Ψ is obtained with numerous peaks which describe dominating replacements between image I_t and I_{t+1} (see Figure 2-9):

$$\Psi_{t,t+1}(x,y) = F^{-1}\left\{\frac{\hat{I}_t(u,v)\hat{I}^*{}_{t+1}(u,v)}{\left|\hat{I}_t(u,v)\hat{I}^*{}_{t+1}(u,v)\right|}\right\} = F^{-1}\left\{e^{-i\,arg\{\hat{I}_t(u,v)\}}\cdot e^{-i\,arg\{\hat{I}^*{}_{t+1}(u,v)\}}\right\} \tag{2.26}$$

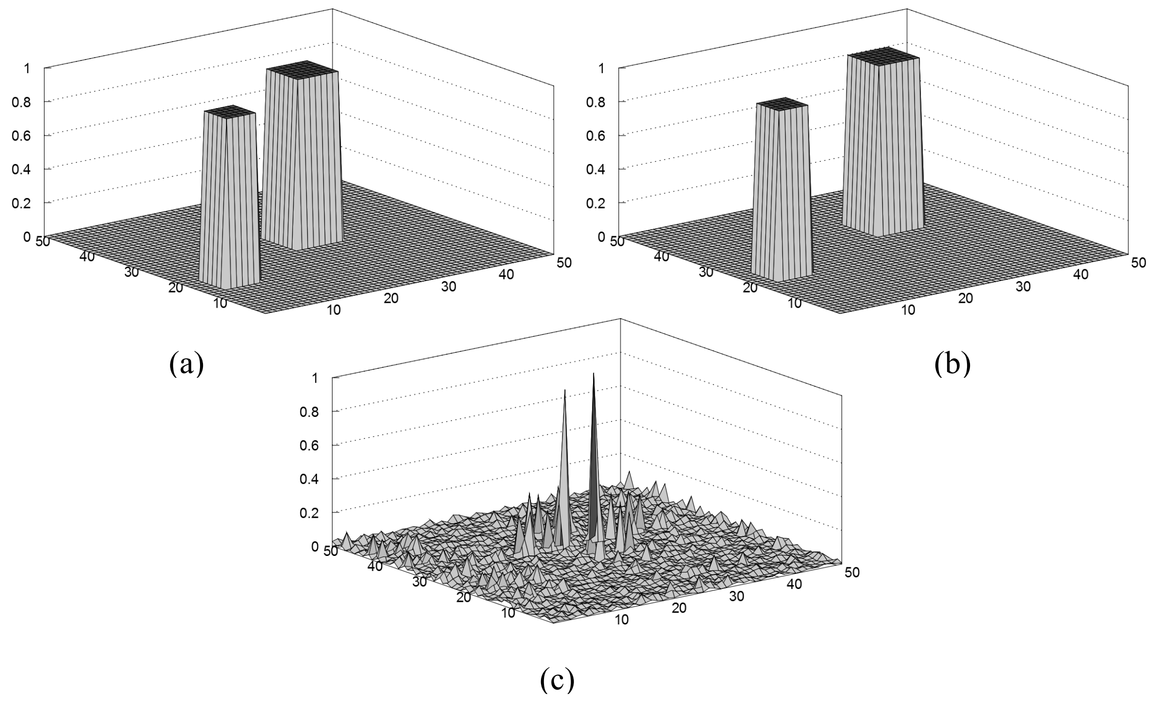


Figure 2-9: Illustration of phase correlation motion estimation for moving objects in two consecutive video frames ((a) and (b)) resulting in two peaks (c) that describe the amount of motion.

2.5.1.3 *Optical flow*

The *optical flow* approach expects that a pixel at position $\mathrm{x} = (x, y)^T$ at time t reappears in a subsequent frame at time $t + 1$ at a new position $\mathrm{x} + \vec{\mathrm{u}} = (x + u, y + v)^T$. It is assumed that this displacement can be approximated by a first-order Taylor series (Paragios, Chen, & Faugeras, 2005):

$$I(\mathrm{x} + \vec{\mathrm{u}}, t + 1) \approx I(\mathrm{x}, t) + \vec{\mathrm{u}} \cdot \nabla_{\mathrm{x}} I(\mathrm{x}, t) + \frac{\partial I(\mathrm{x}, t)}{\partial t} \tag{2.27}$$

Subtracting $I(\mathrm{x}, t)$ to estimate the intensity difference leads to the optical flow constraint equation:

$$\vec{\mathrm{u}} \cdot \nabla_{\mathrm{x}} I(\mathrm{x}, t) + \frac{\partial I(\mathrm{x}, t)}{\partial t} = 0 \tag{2.28}$$

Obviously, the translation $\vec{\mathrm{u}}$ cannot be calculated from the optical flow constraint equation only. This under-constrained condition is reflected in the aperture problem that appears as ambiguities perpendicular to the image gradients (see Figure 2-10). Therefore, another constraint has to be defined to solve Equation (2.28).

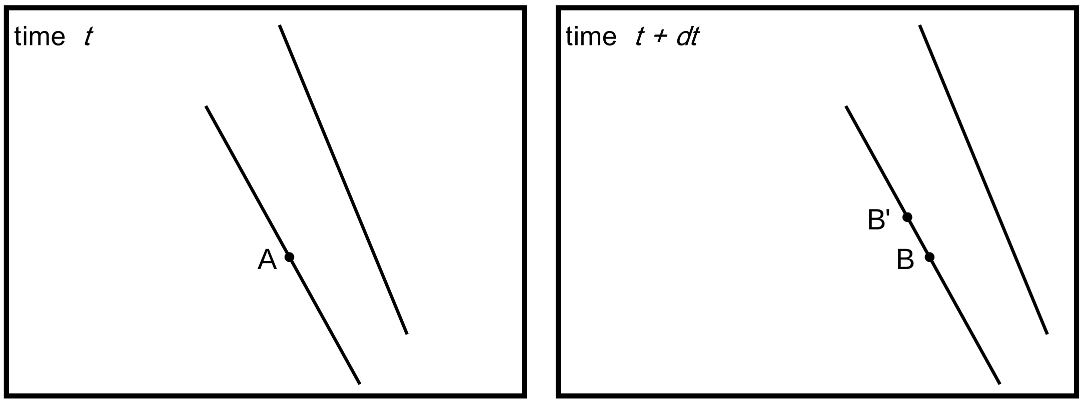


Figure 2-10: Aperture problem caused by ambiguities in optical flow computation. No clear assignment of point A at time t can be made to either point B or B' at time t+dt.

A key assumption which leads to a solution is that neighboured pixels undergo similar motion and hence those pixels are included in the gradient constraints. The squared error is finally minimised by the least-squares estimator:

$$E(\vec{\mathrm{u}}) = \sum_{x} g(\mathrm{x}) \left(\vec{\mathrm{u}} \cdot \nabla_{\mathrm{x}} I(\mathrm{x}, t) + \frac{\partial I(\mathrm{x}, t)}{\partial t} \right)^2 \tag{2.29}$$

where $g(\mathrm{x})$ is a kernel weighting the region of considered pixels. This approach has been proposed by Lucas and Kanade (Lucas & Kanade, 1981). The translation $\vec{\mathrm{u}}$ can finally be calculated by minimising $E(\vec{\mathrm{u}})$ with:

$$\frac{\partial E(\vec{\mathrm{u}})}{\partial u} = 0 \quad \text{and} \quad \frac{\partial E(\vec{\mathrm{u}})}{\partial v} = 0 \tag{2.30}$$

Dependent on the image structure, and specifically in case of the aperture problem, it happens that Equation (2.30) cannot be solved. This leads to a sparse vector field, where only clearly identified pixels can be tracked by this method.

An alternative motion estimation method has been proposed by Horn and Schunck (Horn & Schunck, 1981). Compared to the local approach of Lucas and Kanade, an energy function is minimised here which supports smoothness of the flow over the image:

$$E(\vec{\mathrm{u}}) = \sum_{x} \left[\left(\vec{\mathrm{u}} \cdot \nabla_{\mathrm{x}} I(\mathrm{x}, t) + \frac{\partial I(\mathrm{x}, t)}{\partial t} \right)^2 + \lambda(||\nabla_{\mathrm{x}} u||^2 + ||\nabla_{\mathrm{x}} v||^2) \right] \tag{2.31}$$

where λ is a regularisation constant to control the smoothness of the vector field, i.e. larger values of λ result in a smoother optical flow field.

The advantage of Horn and Schunck over Lucas and Kanade is a dense flow field, which allows motion information even for unstructured areas through the global smoothness constraint. On the other hand, the Lucas and Kanade approach allows a fast computation, whereas Horn and Schunck is an iterative method to minimise the energy function. Additionally, Lucas and Kanade is less sensitive to noise as it considers only clearly identified pixels.

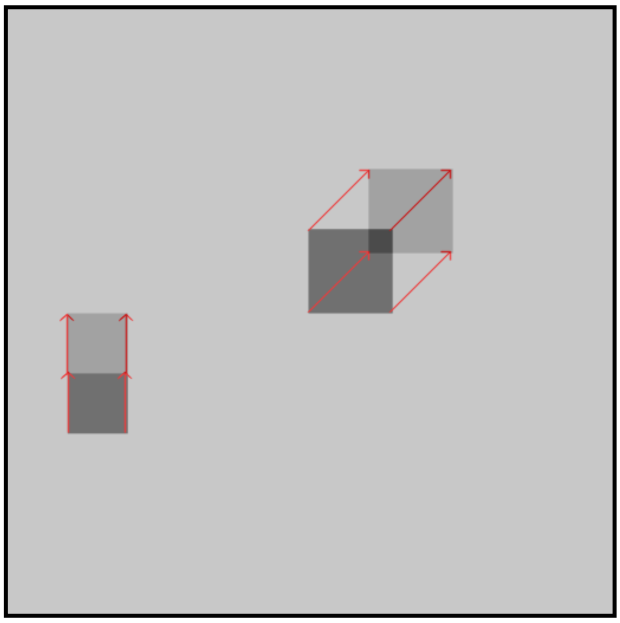


Figure 2-11: Motion vector estimation by applying the optical flow approach by Lucas and Kanade on two moving squares. The transparent squares describe the position in the second video frame. The size of the video frames is 512 x 512 pixels.

2.5.2 Global motion estimation

Motion of pixels in 2D images can be seen as a result of projections from moving objects as well as camera motion in a 3D world coordinate system. For determining ROIs, it is of interest only to detect areas in images which correspond to moving objects. For this, an obvious approach is to detect and compensate camera motion in order to separate the two motion types in 2D images. This can be done either by physically measuring the movement of the camera directly or by estimating it from local motion information, i.e. a measured motion vector field. Even if the former method is the preferred and more accurate solution, it cannot be expected that measured camera parameters will always be available in a broadcast production workflow. Therefore, the computation of camera motion is considered in this section. Due to the fact that the main region in an image often corresponds to background, global motion is considered as camera motion here.

To estimate global motion model parameters, a geometric transformation is searched that, as much as possible, maps image coordinates from a first image to a consecutive one and vice versa. Such a mapping can be estimated by assuming that all corresponding coordinates for which a solution is searched lie approximately on a plane (see Figure 2-12). This constraint is known as planar homography. To achieve this, it is assumed that a world coordinate point $\boldsymbol{X_w}$ is projected on the first and second image planes (see Figure 2-12). This results in the image coordinates $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$, where each corresponds to its own camera coordinate system with origins o_1 and o_2. Accordingly, a world coordinate point $\boldsymbol{X_w}$ is then $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$ expressed in the two camera systems. Further assuming that the world coordinate system is the same as the first camera coordinate system, then $\boldsymbol{X_w} = \boldsymbol{X_1}$ and hence $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$ can be mapped by:

$$\boldsymbol{X_1} = \boldsymbol{X_2} \cdot R + T \tag{2.32}$$

where R is a 3x3 rotation matrix and T is a 3D translation vector. The condition that all points lie on a plane can be described with the aid of the plane equation $N^T\boldsymbol{X_1} = d$, where N is the normal vector perpendicular to the plane and d is the distance from the plane to the origin o_1. Inserting this condition into Equation (2.32) results in:

$$\boldsymbol{X_2} = \boldsymbol{X_1} \cdot R + T\frac{1}{d}N^T\boldsymbol{X_1} = \left(R + T\frac{1}{d}N^T\right)\boldsymbol{X_1} = H \cdot \boldsymbol{X_1} \tag{2.33}$$

The matrix H is known as planar homography matrix.

Ignoring any camera calibration transformation (which can be done here, as the real position of world coordinates is not of interest) the coordinates $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$ can be mapped to homogeneous image coordinates by:

$$\mathrm{Z}_1\boldsymbol{x_1} = \mathrm{Z}_1\begin{pmatrix}\mathrm{x}_1\\ \mathrm{y}_1\\ 1\end{pmatrix} = \begin{pmatrix}\mathrm{X}_1\\ \mathrm{Y}_1\\ \mathrm{Z}_1\end{pmatrix} \text{ and } \mathrm{Z}_2\boldsymbol{x_2} = \mathrm{Z}_2\begin{pmatrix}\mathrm{x}_2\\ \mathrm{y}_2\\ 1\end{pmatrix} = \begin{pmatrix}\mathrm{X}_2\\ \mathrm{Y}_2\\ \mathrm{Z}_2\end{pmatrix},$$

$$\text{with } \boldsymbol{X_1} = \begin{pmatrix} \mathrm{X}_1 \\ \mathrm{Y}_1 \\ \mathrm{Z}_1 \end{pmatrix} \text{ and } \boldsymbol{X_2} = \begin{pmatrix} \mathrm{X}_2 \\ \mathrm{Y}_2 \\ \mathrm{Z}_2 \end{pmatrix} \tag{2.34}$$

Substituting the homogeneous image coordinates above for the world coordinates in Equation (2.33) and eliminating Z_1 and Z_2 (see (Ma, Soatto, Kosecka, & Sastry, 2004) for further details), the planar epipolar constraint is estimated by:

$$\boldsymbol{x_2} \times H\boldsymbol{x_1} = 0 \tag{2.35}$$

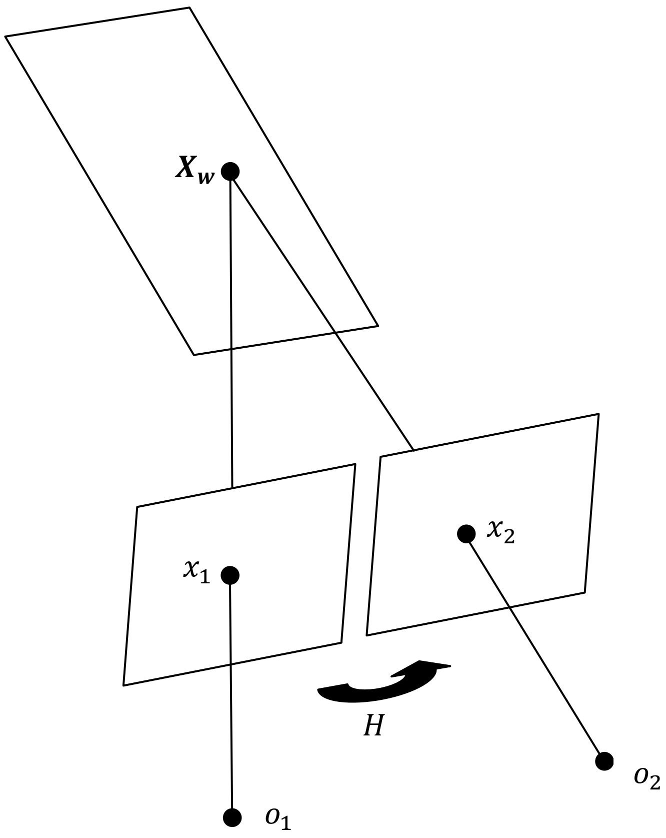


Figure 2-12: Projection of a world coordinate lying on a plane onto image planes. Each image plane has its own camera coordinate system with origins o_1 and o_2.

Geometrically, Equation (2.35) can be interpreted as follows: the matrix H maps two position vectors $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ in such a way that they point in the same direction but not necessarily with the same magnitude and hence their cross product is zero (see Figure 2-12).

A special class of homographies represents affine transformations, already discussed in Section 2.3.1. By applying this, Equation (2.35) significantly simplifies to:

$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} \times \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \vec{0} \tag{2.36}$$

where parameters h_{11}, h_{12}, h_{21}, h_{22} allow scaling, shearing and rotating, and h_{13}, h_{23} describe translation along the x- and y-direction. As the last row of the affine transformation is $(0\ 0\ 1)$, it is assumed that the coordinate system o_2 does not shift into the z-direction in comparison to o_1. In other words, homogeneous coordinates $\boldsymbol{x_2}$ with $z_2 = 1$ and $\boldsymbol{x_1}$ with $z_1 = 1$ must always satisfy $z_2 = (0\ 0\ 1) \cdot \boldsymbol{x_1} = \mathbf{1}$. To solve Equation (2.36), three motion vectors are required to describe such a mapping.

With regard to camera motion estimation, an affine transformation limits the camera motion to zooming (which corresponds to scaling in the x- and y-direction), rotation around the z-axis, as well as moving up and to the side (which corresponds to translation in the x- and y-direction). For pan and tilt (which correspond to rotations around the x- and y-axis), interesting special cases exist, where the homography matrix H is $H = R$. Thus, depth information gets lost as no translation exists and the scene becomes planar. This means that for the motion vector field all global motion vectors have the same magnitude independent of a pixel's depth (cf. Figure 2-13).

2.5.2.1 ***Robust estimation***

Due to the fact that many more than three matched points are usually available from a motion vector field, the system in Equation (2.36) is over-determined. Additionally, measured point positions are inexact and hence the equation will not result in a zero solution due to noisy data. This deviation from zero, i.e. the difference between estimated and predicted values is called *residual*. Therefore, instead of demanding an exact solution, an approximated solution has to be found by considering point correspondences as much as necessary. To keep the notation of the previous section, $\boldsymbol{x_{1,i}}$ and $\boldsymbol{x_{2,i}}$ are now the i-th point correspondence of a set of matched image points in a motion vector field.

In the following, methods are proposed to receive a solution for the given problem of Equation (2.36). H is assumed to be still affine, as solving a full perspective transformation would be beyond the scope of this work. Due to the over-determined system, Equation (2.36) serves as a residual measure for each point correspondence:

$$\boldsymbol{x}_{\boldsymbol{1},\boldsymbol{i}} \times H\boldsymbol{x}_{\boldsymbol{2},\boldsymbol{i}} = r_i \tag{2.37}$$

One possibility to solve for H is to minimise the sum of squared residuals by the *linear least-squares* approach:

$$\min_H \sum_{i=0}^{n} \|r_i\|^2 \tag{2.38}$$

Alternatively, (2.38) can be minimised by means of the Singular Value Decomposition (SVD) (see (Ma, Soatto, Kosecka, & Sastry, 2004; Kalman, 1996; Hartley & Zisserman, 2008) for further details).

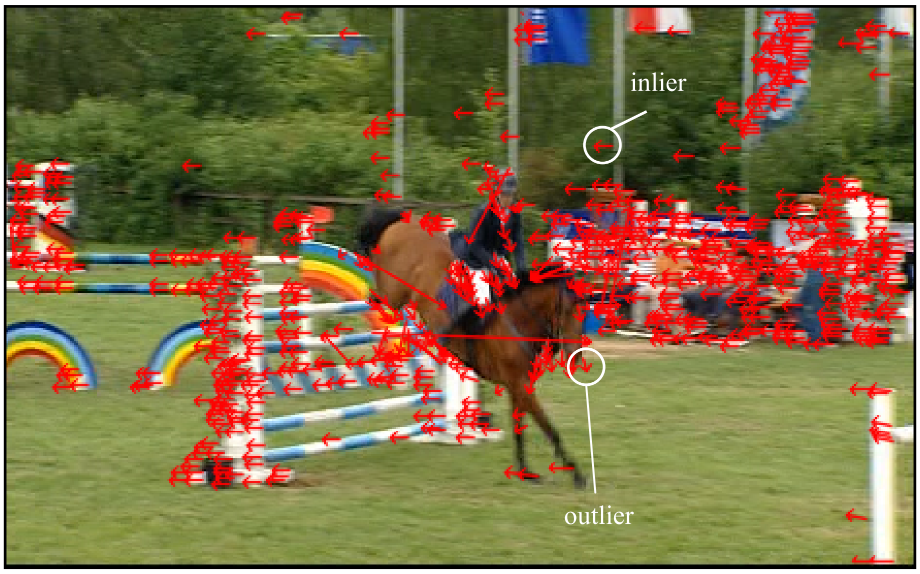


Figure 2-13: Motion vector field estimated by optical flow. Motion vectors from the object represent outliers to motion vectors that correspond to the global motion.

Least-squares fitting is an appropriate method to remove noise that is caused by inexact measured data, i.e. the residuals are expected to be normally distributed. Besides such a type of noise, the global motion estimation is further negatively influenced by motion vectors that correspond to object motion, i.e. those that differ from global motion. Such interferences represent *outliers* to the noise distribution of imprecise coordinate measurement and follow a different and unmodeled distribution (see Figure 2-13).

Under this condition, a minimisation by least-squares might not result in a satisfying solution. To overcome this, an approach that first separates *inliers* from *outliers* and finally applies least-squares fitting obviously leads to a more stable estimation (see Figure 2-14). Such methods belong to the group of robust estimators. In the following, three robust estimators are briefly introduced which reduce the sensitivity to outliers.

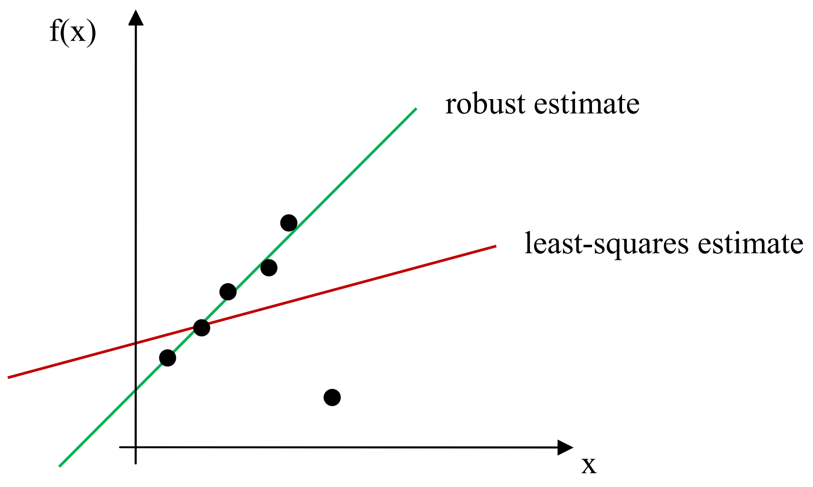


Figure 2-14: Effect of a single outlier in a bunch of arbitrary values x on least-squares fitting (red line). The distance between points and the fitted line is the error to be minimised. In turn, a robust estimator (green line) is not influenced by the outlier.

M-estimators

M-estimators ("M" for "maximum likelihood-type") try to reduce the influence of outliers by using an alternative function to the squared residuals $\|r_i\|^2$ from the least squares method:

$$\min_{H} \sum_{i=0}^{n} \rho(r_i) \tag{2.39}$$

where ρ is called the influence function. Dependent on the given data set and its distribution, different functions ρ can be used to influence the results of the estimator. For further information on different M-estimator functions, the author refers to (Zhang Z. , 1997).

RANSAC

The idea of RANSAC (RANdom SAmple Consensus) is to randomly choose a minimal number of samples to fit a model – three for affine homography. Residuals that lie within a threshold define the support for this model. This process of randomly selecting samples is repeated several times and motion vectors that correspond to the model with most support are considered as inliers. Finally, the model parameters are re-estimated by applying least-squares on inliers only.

According to (Hartley & Zisserman, 2008), the RANSAC algorithm applied on a data set S that contains outliers can be summarised as follows:

1. Select s random data points from S and instantiate the model from this subset.
2. Find a set of data points S_i that are within an error threshold t of the model. Hence, S_i defines the inliers of S.
3. If the number of inliers in S_i is greater than a threshold T, re-estimate the model (e.g. by applying least-squares) using all points in S_i and terminate.
4. If the number of inliers in S_i is less than T, select a new subset and repeat steps 1 to 3.
5. After N trials, choose the largest consensus set which has been estimated.

For RANSAC, three parameters have to be specified: t, T and N. As N defines the number of samples, it should be set to a sufficiently large value to ensure that with a probability p, at least one subset contains exclusively inliers. The probability to choose s random inliers within one of N trials is:

$$p = 1 - (1 - (1 - \varepsilon)^s)^N \tag{2.40}$$

where ε is the fraction of outliers. With the assumption that p should be nearly 1, N can be computed from s and ε.

For example, $\varepsilon = 50\%$, $p = 99\%$ and $s = 3$ (three vectors to solve a matrix H), N can be estimated by:

$$N = \frac{\log(1 - p)}{\log(1 - (1 - \varepsilon)^s)} = \frac{\log(1 - 0.99)}{\log(1 - (1 - 0.4)^3)} \approx 35 \tag{2.41}$$

In turn, T should correspond to the expected number of inliers and t has to be estimated empirically.

Least Median of Squares

An optional robust estimator is the Least Median of Squares (LMedS) method (Rousseeuw & Leroy, 2003). It is defined by the nonlinear minimisation problem:

$$\min \underset{i=1,\ldots,N}{\text{med}} \|r_i\|^2 \tag{2.42}$$

LMedS randomly chooses a set of samples for N models, as RANSAC does. Finally, it chooses the inliers of the model with minimal error to re-estimate the model parameters. The error that corresponds to a model is defined as the median value of squared residuals for the whole data set. Therefore, LMedS can only tolerate up to 50% outliers in a data set. Alternatively, one can use the Least Quantile of Squares (LQS) if it is expected that fewer inliers exist. This might lead to a statistically worse condition and has to be used carefully. As RANSAC can deal with a higher number of outliers, LMedS requires no threshold setting.

2.6 Segmentation

Segmentation is a key process in detecting objects. It groups corresponding pixels to segments by detecting discontinuities and similarities in, for example, intensity values. Instead of handling each pixel individually, this process gives a much more meaningful image representation for further analysing.

2.6.1 Thresholding

Histogram based methods rely on the fact that related pixels can be found by peaks and valleys in their intensity distribution. Due to separating modes in an intensity distribution, this method is also referred to as *thresholding*.

The simplest technique of *thresholding* is defining a single global threshold to split a bimodal distribution. Each pixel is labelled as background or foreground when it lies below or above a previously defined intensity threshold T. The output is a binary image that represents each segment as black or white. This approach is computationally efficient when illumination can be controlled, for example in industrial inspection.

In the case of uneven illumination, a histogram cannot be separated by a single threshold. In such cases, an *adaptive thresholding* can be applied that divides an image into several sub-images. For each sub-image, a threshold is statistically examined. The simplest way to examine the *local threshold* in a sub-image S, is by using the *mean* value of intensities:

$$T_{mean} = \frac{1}{m \cdot n} \sum_{j=0}^{n-1} \sum_{i=0}^{m-1} S(i,j) \tag{2.43}$$

Alternatively, thresholds of sub-images can be computed by, e.g. the *median* value or a *weighted sum*. For further information on local thresholding, the author refers to (Gonzales & Woods, 2002) and (Fisher, 2007).

2.6.2 Region labelling with flood fill

When a local or global thresholding has been applied on an image, pixels are labelled as foreground or background. To get a more precise split into possibly several foreground objects, a local connection between foreground pixels is made. This process is called *region labelling*. A simple algorithm for region labelling is *flood fill* (Gonzales & Woods, 2002; Burger & Burge, 2008). This operation is initialised at an unmarked foreground pixel and flows out across a flat region. Neighboured pixels are added to a segment piece by piece as long as the absolute intensity level difference to the starting point is below a threshold. Due to the fact that the growing region starts from one reference pixel, this method is also called *seeded region growing*. After a segment has been finalised, this region is marked and the next “seed” point is chosen.

2.6.3 Edge linking

As described in Chapter 2.2.2, linear filters for edge detection yield pixels lying on edges. Due to noise and non-uniform illumination in images, highlighted edges can be interrupted. Therefore, a linking of edges is required afterwards to combine edge pixels into meaningful segments.

A simple approach for edge linking is to consider points in a small local neighbourhood that are labelled as edge points. Pixels are linked together if their magnitude and angle of the gradient fulfil criteria of resemblance. The gradient of an image $I(x, y)$ is defined by its first derivative:

$$\nabla I = \begin{pmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{pmatrix} \tag{2.44}$$

Thus, the magnitude and the angle of the gradient are:

$$mag(\nabla I) = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \qquad ang(\nabla I) = tan^{-1}\left(\frac{\frac{\partial I}{\partial y}}{\frac{\partial I}{\partial x}}\right) \tag{2.45}$$

The magnitude and angle of the gradient are computed for each edge pixel (x_0, y_0) as well as for the pixels (x, y) in its neighbourhood (usually 3x3 or 5x5 neighbourhood). The similarity of the magnitude is defined by a threshold E:

$$\left|mag\big(\nabla I(x, y)\big) - mag\big(\nabla I(x_0, y_0)\big)\right| \leq E \tag{2.46}$$

In the same way, the similarity of angles is expressed by a threshold A as follows:

$$\left|ang\big(\nabla I(x, y)\big) - ang\big(\nabla I(x_0, y_0)\big)\right| < A \tag{2.47}$$

Each neighbourhood (x, y) is linked to a pixel (x_0, y_0) if both criteria are fulfilled. This process is repeated for the whole image.

Alternatively, more complex methods can be applied that process edge images globally, for example *Hough Transform* or *Graph Theoretic Techniques*. For further information on edge linking by global processing methods, the author refers to (Gonzales & Woods, 2002).

2.6.4 Clustering

Clustering can be described as a process that organises observed elements into groups whose members are related in some sense (Fasulo, 1999). It is a widely used technique applied in different areas such as data mining, pattern recognition and computer vision. In computer vision it is commonly a post-processing of segmentation and a pre-processing of classification. In general, cluster methods can be divided into three main steps. Firstly, a pattern representation has to be defined, e.g. by extracted features. Secondly, a measure of similarity has to be defined that is adapted to the pattern representation. And thirdly, found associations are organised into groups, i.e. clusters (Jain, Murty, & Flynn, 1999).

There exist two types of clustering methods, *hierarchical clustering* and *partitioning* (also referred to as *k-clustering*) (Fasulo, 1999). In this section, the basic ideas of both methods are described.

2.6.4.1 ***Hierarchical clustering***

The approach of *hierarchical clustering* is to subdivide a set S of elements into subsets. This creates a tree of data with S representing the root and the subsets defined by internal nodes. There exist two approaches to building up such a branched structure: *divisive* and *agglomerative algorithms*. The former recursively splits a set S until a stopping criterion is satisfied. The latter starts with each element as a singleton set and successively merges them until an abort criterion is met. As *agglomerative methods* are much more common, *divisive methods* are not treated here. For further information on divisive methods, the author refers to (Jain, Murty, & Flynn, 1999).

Most *agglomerative methods* proceed in a similar way. In a first step, S is split into singleton sets $S_1, S_{2,} \dots, S_n$, where each singleton set is assumed to be an initial cluster. Next, pairs of sets $\{S_i, S_j\}$ are searched by a defined *cost function* $c(S_i)$. Every pair found is replaced by the intersection $S_i \cup S_j$ of singletons. This process is repeated until an abort criterion is met. If no abort criterion is specified, the algorithm terminates when only one cluster is left (see Figure 2-15). The core of *hierarchical clustering* is the *cost function*. Dependent on criteria such as properties of the data set and processing time, the *cost function* has to meet different requirements. Equations (2.48)-(2.50) list the most common cost functions, where $d(x_i, x_j)$ denotes the distance between element x_i of set S_i and element x_j of set and S_j.

Single-linkage
$$\min_{x_i \epsilon S_i, x_j \epsilon S_j} d(x_i, x_j) \tag{2.48}$$

Average-linkage
$$\frac{1}{|S_i||S_j|} \tag{2.49}$$

Complete-linkage
$$\max_{x_i \epsilon S_i, x_j \epsilon S_j} d(x_i, x_j) \tag{2.50}$$

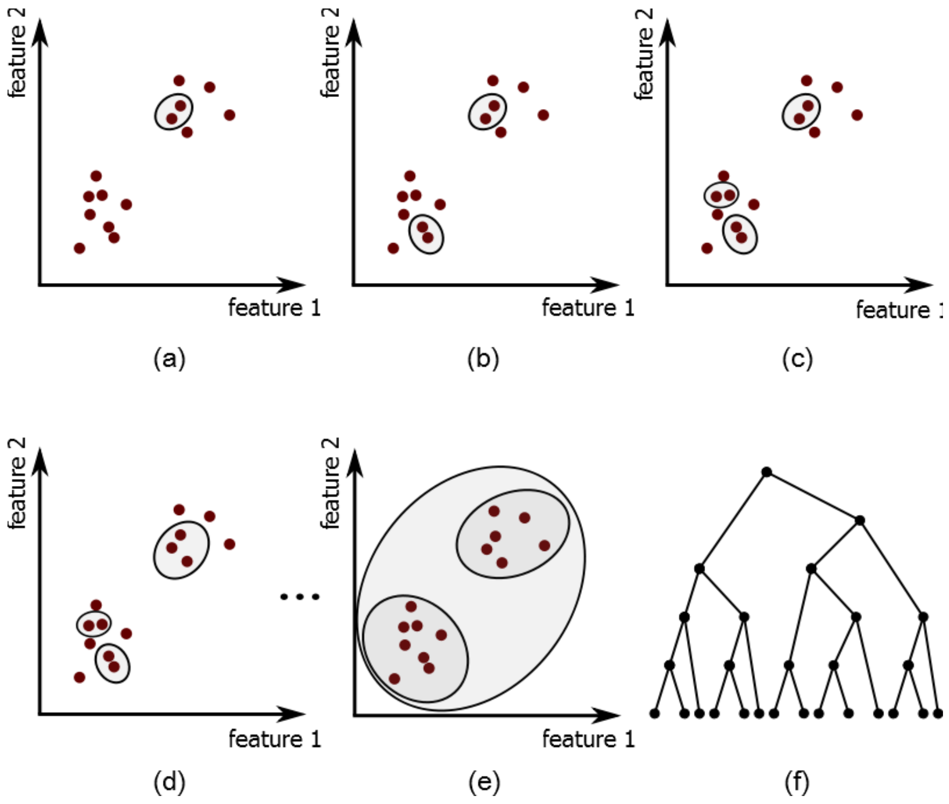


Figure 2-15: Agglomerative clustering for a two-dimensional feature space without an abort criterion ((a)-(e)) and the resulting tree (f).

2.6.4.2 *Partitioning*

Partitioning algorithms usually divide a set S by a predefined number k of desired subsets. Perhaps the most common *partitioning algorithm* is the *optimisation algorithm*. For this, a cost function $c(S_i)$ is defined that is associated with each subset. It is expected that the optimal allocation of elements to clusters has been achieved if the sum of costs per cluster $\sum_{i=0}^{k-1} c(S_i)$ is minimised.

A popular *partitioning method* is the *k-means* algorithm whose cost function relies on the *sum-of-squares criterion*. The main idea is – as for *hierarchical clustering* – to use the distance between the r'th elements of S_i (denoted by x_r^i) and the centroid of each cluster S_i (denoted by $\bar{x}^i$) as measure of similarity. If $|S_i|$ the number of elements in S_i, the *sum-of-squares criterion* is defined as follows:

$$c(S_i) = \sum_{r=1}^{|S_i|} \left(d\left(\bar{x}^i, x_r^i\right)\right)^2 \tag{2.51}$$

Some elements may not belong to a cluster and hence present outliers for a cluster. A more robust measure for the cluster centre is the *median value*, already mentioned in 2.2.3. The algorithm is referred to as *k-medioids*. This leads to Equation (2.52), where $\hat{x}^i$ denotes the median of each S_i.

$$c(S_i) = \sum_{r=0}^{|S_i|} d\left(\hat{x}^i, x_s^i\right) \tag{2.52}$$

2.7 Classification

Classification is a method that takes a set of feature examples and creates a class label. This process is called training and the feature examples are called training data. After training a classifier, any new example is assigned to the class with the lowest risk of mismatch.

2.7.1 Bayes classifier

Classification not only links an element s to a class c_i, but also specifies a quality of the decision. This qualitative statement is the *posterior* $P\left(s = c_i | \vec{f}(s)\right)$, where $\vec{f}(s)$ is the feature vector of s.

The *Bayes rule* makes a connection between the *posterior*, the probability distribution of $\vec{f}(s)$ for each class and the *prior probability* (Toennies, 2005). The *prior probability* $P(s = c_i)$ (often just called the *prior*), is the anticipated likelihood or belief for each class before any training data has been observed. For uncorrelated features, the probability distribution of $\vec{f}(s)$ for a class c_i is estimated by the product of each feature $\vec{f_j}(s)$:

$$P(\vec{f}(s)|s = c_i) = P(f_1(s)|s = c_i) \cdot P(f_2(s)|s = c_i) \cdot \ldots \cdot P(f_m(s)|s = c_i) \tag{2.53}$$

According to the *Bayes rule*, the *posterior* can be calculated as follows:

$$P\left(s = c_i|\vec{f}(s)\right) = \frac{P(\vec{f}(s)|s = c_i) \cdot P(s = c_i)}{\sum_{k=0}^{k-1} P(\vec{f}(s)|s = c_k)\, P(s = c_k)} \tag{2.54}$$

where the sum in the denominator is used for normalisation.

For example, having normally distributed training data for two classes a and b with two features f_1 and f_2, the probability distributions of $\vec{f}$ for each class are:

$$\begin{aligned} P(\vec{f}(s)|s = a) &= P(f_1(s)|s = a) \cdot P(f_2(s)|s = a) \\ &= \frac{1}{2\pi} \cdot \frac{1}{\sigma_a(f_1)} e^{\left(-\frac{(f_1(s)-E_a(f_1))^2}{2\sigma_a(f_1)}\right)} \cdot \frac{1}{\sigma_a(f_2)} e^{\left(-\frac{(f_2(s)-E_a(f_2))^2}{2\sigma_a(f_2)}\right)} \end{aligned} \tag{2.55}$$

$$\begin{aligned} P(\vec{f}(s)|s = b) &= P(\vec{f_1}(s)|s = b) \cdot P(\vec{f_2}(s)|s = b) \\ &= \frac{1}{2\pi} \cdot \frac{1}{\sigma_b(f_1)} e^{\left(-\frac{(f_1(s)-E_b(f_1))^2}{2\sigma_b(f_1)}\right)} \cdot \frac{1}{\sigma_b(f_2)} e^{\left(-\frac{(f_2(s)-E_b(f_2))^2}{2\sigma_b(f_2)}\right)} \end{aligned} \tag{2.56}$$

where $\sigma_a(f_j), \sigma_b(f_j)$ and $E_a(f_j), E_b(f_j)$ are the variance and mean of the Gaussian distributions. Furthermore, it is known that class a is chosen n-times more than class b. Hence, the *priors* are $P(a) = \frac{n}{n+1}$ and $P(b) = \frac{1}{n+1}$.

For each new segment s with features $f_1(s), f_2(s)$ the *posterior* can now be computed by:

$$P\left(s = a|\vec{f}(s)\right) = \frac{P(f_1(s)|s = a) \cdot P(f_2(s)|s = a) \cdot P(a)}{P_{total}} \tag{2.57}$$

$$P\left(s = b|\vec{f}(s)\right) = \frac{P(\vec{f}_1(s)|s = b) \cdot P(\vec{f}_2(s)|s = b) \cdot P(b)}{P_{total}} \tag{2.58}$$

where P_{total} is:

$$\begin{aligned} P_{total} = P(f_1(s)|s = a) \cdot P(a) + P(f_2(s)|s = a) \cdot P(a) + P(f_1(s)|s = b) \cdot P(b) \\ + (f_2(s)|s = b) \cdot\ P(b) \end{aligned} \tag{2.59}$$

Having the two class classifier of the previous example, there exists a *decision boundary* in the *feature space* where $P\left(s = a|\vec{f}(s)\right) = P\left(s = b|\vec{f}(s)\right)$. A *feature vector* that lies on this *decision boundary* cannot be assigned to any class. Therefore, a decision for one class has to be made. Figure 2-16 shows the feature space of the example above with variances of $\sigma_a(f) = \sigma_b(f) = 0.1$, means of $E_a(f_1) = E_a(f_2) = 0.3,\ E_b(f_1) = E_b(f_2) = 0.6$ and a *prior* of $P(a) = \frac{2}{2+1} = 0.67$ and $P(b) = \frac{1}{2+1} = 0.33$.

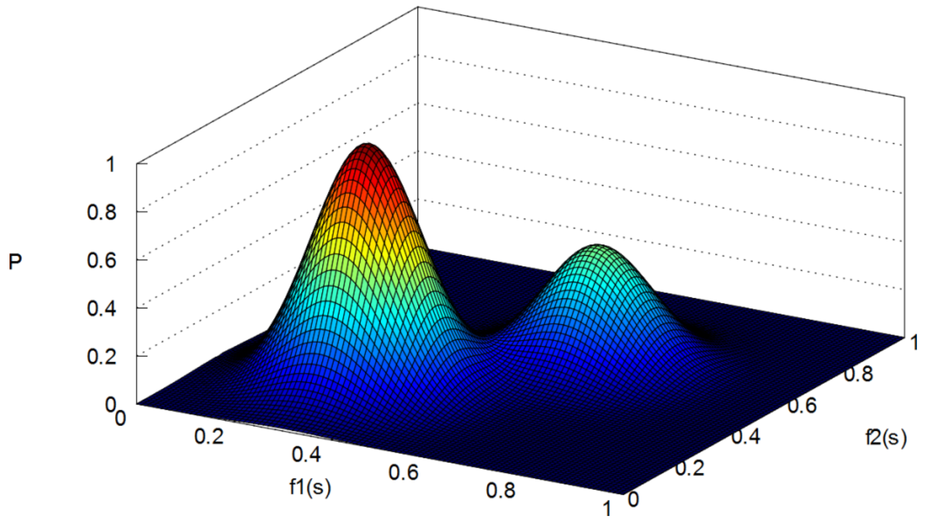


Figure 2-16: Feature space of two normally distributed classes *a* and *b* with $E_a(f_1) = E_a(f_2) = 0.3,\ E_b(f_1) = E_b(f_2) = 0.6$ and $P(a) = 0.67, P(b) = 0.33$, normalised to a value range of 0,...,1.

2.7.2 Nearest neighbour classifier and linear classifiers

A *nearest neighbour* method assumes that the closeness of a new segment to an already classified point indicates the class for the new segment. This means that $P\left(s = c_i | \vec{f}(s)\right) = 1$ if an example exists in c_i that has a minimum distance to $\vec{f}(s)$. This is a significant simplification of the *Bayes rule*, because it is quite difficult to obtain an appropriate probability distribution from training data. In the case of a large number of training examples, *nearest neighbour* is a good approximation of the *Bayes classifier*. The difference from the Bayesian decision can be further decreased by considering a higher number of neighbours for classifying. This leads to the *k-nearest neighbour* approach which takes k neighbours into account instead of a single point. Given a feature vector $\vec{f}(s)$ and a *nearest neighbour* classifier $A(k, l)$, where l is a predefined minimum number of nearest training examples in class c_i, the *k-nearest neighbour* algorithm can be summarised by the following steps (Forsyth & Ponce, 2003):

1. Find k training examples that are closest to $\vec{f}(s)$.
2. Determine the class c_i that contains the majority of feature vectors n in the set of k found *nearest neighbours*.
3. If $n > l$, classify $\vec{f}(s)$ as c_i, else refuse $\vec{f}(s)$.

To simplify the computation of the *nearest neighbours*, some points in the feature space that do not influence the decision can be removed.

The idea of nearest neighbour is to focus on the decision boundaries only, instead of considering a whole data set. This approach can be simplified even more when a function can be found that defines the decision boundary. Having training data that is linearly separable, a linear classification function provides the easiest solution to describe the decision boundary. A classifier which allows learning a linearly discriminative function is called a *linear classifier*. Examples of *linear classifiers* are *perceptron* (Freund & Schapire, 1999) and *support vector machines* (Schölkopf, Burges, & Smola, 1999) using linear kernels.

2.8 Summary

This chapter gave a summary of computer vision and related methods which are relevant for this application. The sections describe methods from *image enhancement*, *pattern recognition*, *image segmentation* and *pattern classification*. The introduction and discussion of different algorithms serves as a basis to create modules which identify ROIs in broadcast video images. These modules will be introduced in Chapter 7.

The following chapter is an introduction to systems which implement visual attention models for image analysis. Such systems make use of computer vision methods to detect salient regions in images. In other words, they interpret images on a higher level by processing the previously mentioned steps from image enhancement to image segmentation. They deliver possible ROIs which are then classified on the next higher level in this application.

3. Visual Attention

Attention is a common and well known term that everybody uses in everyday language. From a psychophysical point of view, attention is a complex process that allows us to find the way in our crowded environment. The visual attention mechanism is part of this process and is described in (Tsotsos J. K., 1995) by the following basic components:

- The selection of a region of interest in the visual field
- The selection of feature dimensions and values of interest
- The control of information flow through the network of neurons that constitute the visual system
- The shifting from one selected region to the next in time

In this chapter, psychophysical approaches to understanding the visual process of humans are briefly explained. Then two visual attention models are introduced that form the basis for many computer models. Finally, the implementations of attention models by applying computer vision methods are explained. Such systems allow a first step into identifying regions of interest by applying computer vision methods.

3.1 Bottom-Up and Top-Down Attention

Visual attention allows people to efficiently process visual information by focussing on regions of interest. Consequently, a complete scene needs only to be scanned at the most relevant positions. Finding and shifting the focus of attention is influenced by two main factors: *bottom-up* and *top-down*.

Bottom-up attention depends on the saliency of features in a visual scene, e.g. by high contrasts. Therefore, the attention is controlled without conscious intention. This mechanism is also called exogenous.

Top-down is an endogenous searching process. In this case, attention is driven by the previous knowledge, expectations and current goals of a subject. If someone is searching for an item that e.g. has a specific colour, the gaze is more attracted by regions of this known colour.

3.2 Visual Search

Visual search is a common task and a fundamental tool in psychophysical experiments on visual attention. For this, subjects have to scan a scene for targets among other objects or features (distractors). Such a scene is usually an artificial set-up of targets with different features such as colour, size, orientation or shape. The quantity of targets and distractors is called set size or display size. One of the main questions behind these tests is: what are the *basic features* in visual perception?

Two different tests are of interest in visual search. The first one is whether a subject finds a target in a given period. The subject has to specify the item afterwards. The second test is how much time a subject requires to find the target. The resulting *reaction time* or *response time* (RT) is a measure of visual search efficiency. The RT is presented as a function of set size.

The flatter the slope of the RT functions, the more efficient the visual search. Results depend extremely on the type of search process. Hereby, a distinction is made between *serial search* and *parallel search.* As may be imagined, the *parallel search* is the more efficient one in processing and its slope is nearly zero, i.e. the *response time* is independent of the set size. This effect occurs when *targets* and *distractors* differ in one single *feature*. Therefore, the search is called a *feature search* and due to the fast RT, the effect is called the *pop-out effect*. On the other hand, a *serial search* occurs when *targets* and *distractors* differ in more than one feature; therefore, it is also called a *conjunction search*. In this case, the RT increases with the number of distractors (see Figure 3-1).

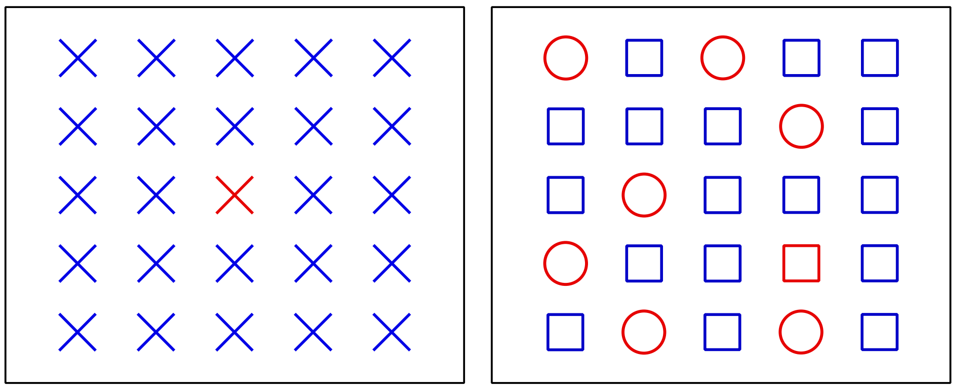


Figure 3-1: Example of a feature search (left) and a conjunction search (right). It takes much longer to find the red square among red circles and blue squares than to find the red X among blue X (only one feature differs).

So far, one might conclude that a *feature search* causes a *pop-out effect* and a conjunction of several features increases the search time by each added target. Anne Treisman showed in (Treisman, 1986) that a feature search does not necessarily result in a *pop-out effect*. She stated that it is easier to find the presence of a basic feature than its absence. This effect is called *search asymmetries*. Figure 3-2 depicts this phenomenon by a vertical line among squares (presence of basic feature) and a square without vertical line (absence of basic feature) among squares with vertical lines.

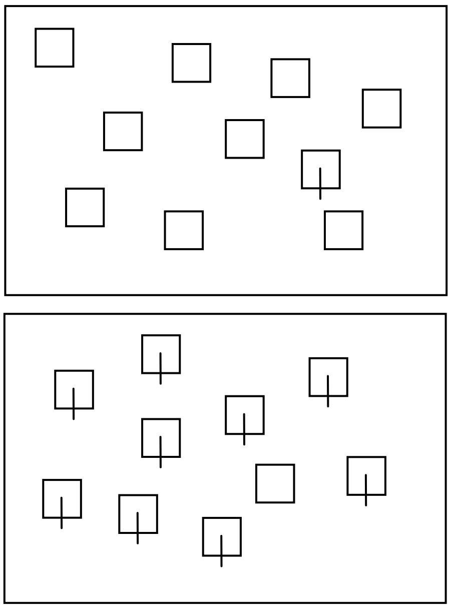


Figure 3-2: Presences of feature (top) and absence of feature (bottom).

3.3 Feature-Integration Theory of Attention (FIT)

The most famous model of visual attention is the *Feature-Integration Theory* (FIT) of Anne Treisman (Treisman A., 1980). This model describes the visual perception in several levels. The first level is the *pre-attentive level* and is responsible for the detection of features. On this level, “features are registered early, automatically, and in parallel across the visual field.” (Treisman A., 1980). In subsequent stages (*attentive level*), each object is identified by focussed attention. Figure 3-3 depicts all levels of the FIT.

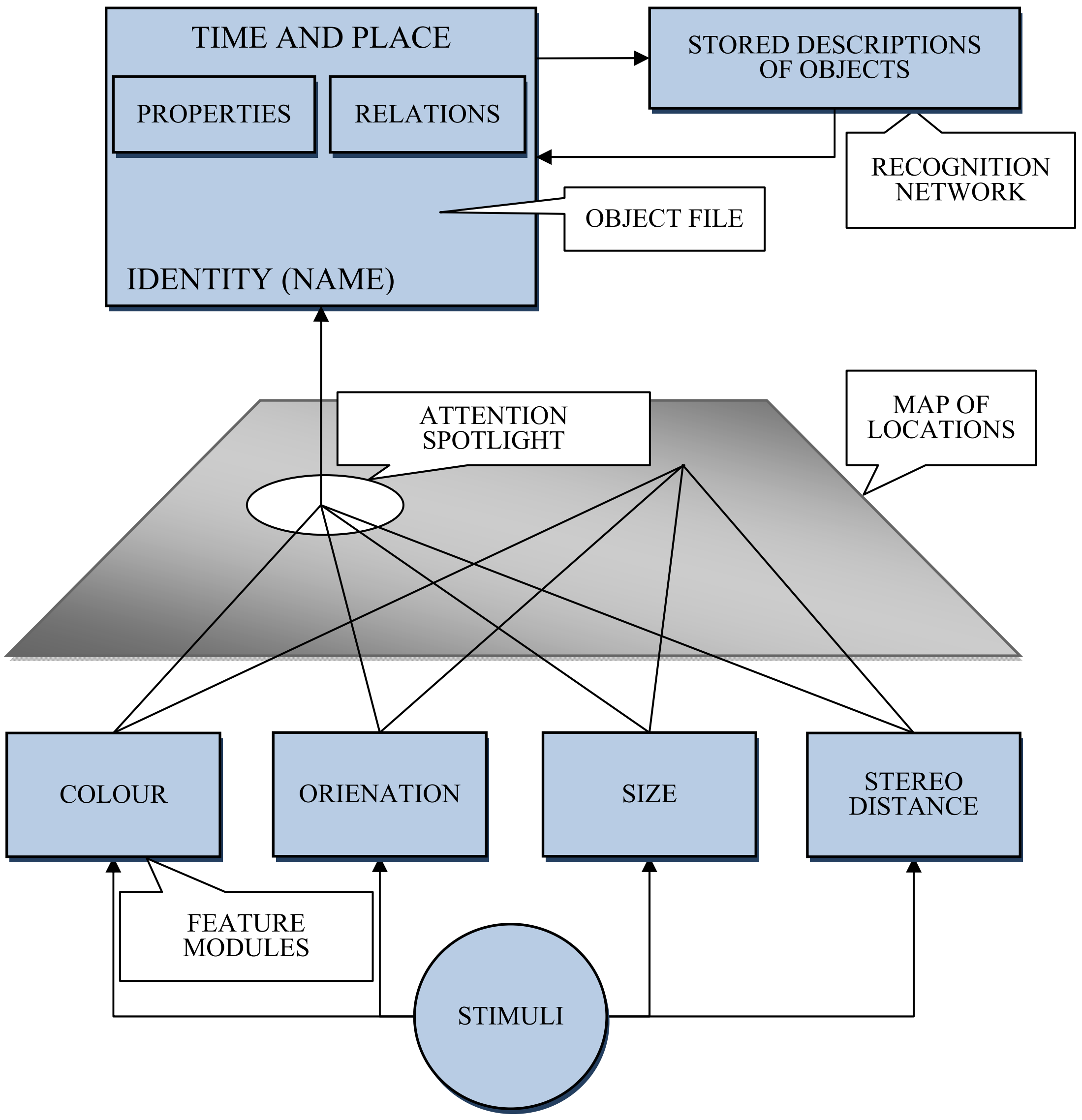


Figure 3-3: Model of Feature Integration Theory according to Treisman (Treisman, 1986).

3.3.1 Feature maps

Treisman describes feature detection on the *pre-attentive level* by *feature modules*. Each module detects the presence of the respective feature. The *feature modules* can be seen as a stack of maps, each representing a special property. In the case where a target differs from its distractors by exactly one feature and the distractors are homogenous, the target is detected quickly and in parallel (*pop-out effect*).

On the next level, all maps yield a *master map* of locations. This map contains spatial information about features, but no information about what kind of features they are. Comparable to a roaming spotlight, focal attention makes use of this *master map* and successively processes all regions. At each location, all present feature maps are combined, which results in a conjunctive search.

3.3.2 Object detection

After locating and combining all features, this information is used to create an object description, a so-called *object file*. Finally, the contents of this file are compared to descriptions from a recognition network. This *top-down* process finally allows the recognition of an object. The recognition network combines attributes, behaviour, names and significance of familiar objects (Treisman, 1986). The *object file* is continually updated from new feature information and from the recognition network.

3.4 Guided Search

The Guided Search Model of Wolfe (Wolfe, 1994) is another very important work in visual attention along with the FIT. The two are quite similar. The main difference is an alternative approach for the *conjunction search*. Wolfe's test results have shown that some slopes of RTs for targets that differ in more than one feature are too flat for a *serial search*. Wolfe disputes a strict separation between *pre-attentive level* and *attentive level*. He claims that the *attentive level* receives information from the *pre-attentive level*, which allows a more efficient search. Additionally, instead of a *stack of maps* for each feature module, the *Guided Search* distinguishes between feature dimensions, such as colour, orientation, etc.

In addition to *bottom-up* feature maps, there are *top-down* feature maps in the *Guided Search* approach. This allows a better separation of target features from distractor features. All *feature maps* are fused in an *activation map* that is comparable to Treisman's *master map of location*.

3.5 Visual Attention Systems

3.5.1 Visual-attention system by Itti, Koch and Niebur

One of the most popular computational attention systems is the one by Itti *et al*. (Itti L., 1998). The FIT serves as a basis for this system. The *pre-attentive level*, i.e. the *bottom-up* part of FIT is implemented by applying known computer vision methods on a still colour image.

Motivated by receptive fields, a set of linear centre-surround operations (denoted by "$\ominus$") compute local variations of different features. For this, difference images across scales of image pyramids are estimated. Each scale of an image pyramid represents a low-pass filtered copy of the previous scale with quarter resolution. The lowest level of an image pyramid is the unfiltered and non-scaled input image. For subtraction, the centre pixel is at scale c ∈ {2, 3, 4}. Surround pixels are at scales s = c + δ, with δ ∈ {3,4}. As coarser scales have a lower resolution, images have to be up-scaled again for point-to-point subtraction. These operations are done for three early visual features: *intensity*, *colours* and *orientations*.

3.5.1.1 ***Feature maps***

A *feature map* is a computed feature of the corresponding *feature dimension*. According to Treisman's theory, *feature maps* of each *feature dimension* form a feature stack.

The first feature is motivated by the intensity contrast of dark centres on a bright surround and vice versa. Such contrasts are sensitively detected by the neurons of mammals. The feature *intensity* (*I*) is calculated by the average of the colour channels r, g and b:

$$I = \frac{r+g+b}{3} \tag{3.1}$$

The resulting intensity image is used to create an image pyramid from scales σ, with $\sigma \in [0 \ldots 8]$, to compute centre-surround relations:

$$\mathcal{I}(c,s) = |I(c) \ominus I(s)| \tag{3.2}$$

For the *feature dimension colour*, four colour channels of red, green, blue and yellow are generated. This is justified by the fact that parsed colour from a trichromatic primate retina is encoded in colour-opponent signals of 'red/green' and 'blue/yellow' (Conway, 2002). First, the input channels r, g and b are normalised by I to decouple hue from *intensity*. Afterwards, they are broadly tuned in the range of red, green, blue and yellow:

$$R = r - \frac{g+b}{2} \tag{3.3}$$

$$G = g - \frac{r+b}{2} \tag{3.4}$$

$$B = b - \frac{r+g}{2} \tag{3.5}$$

$$Y = \frac{r+g}{2} - \frac{|r-g|}{2} - b \tag{3.6}$$

To simulate spatial and chromatic opponency of the primary visual cortex, all channels are further processed in two image pyramids:

$$\mathcal{RG}(\mathrm{c},\mathrm{s}) = |(R(c) - G(c)) \ominus (G(s) - R(s))| \quad (3.7)$$

$$\mathcal{BY}(\mathrm{c},\mathrm{s}) = |(B(c) - Y(c)) \ominus (Y(s) - B(s))| \quad (3.8)$$

Orientation represents the third implemented feature. It is computed by Gabor pyramids $O(\sigma,\theta)$, with θ ϵ [0°, 45°, 90°, 135°]. Centre surround information is determined across scales again:

$$\mathcal{O}(\mathrm{c},\mathrm{s},\theta) = |O(\mathrm{c},\theta) \ominus O(\mathrm{s},\theta)| \quad (3.9)$$

3.5.1.2 *Saliency map*

A saliency map combines all feature maps into one and indicates salient locations. Before all maps are combined, they need to be normalised to similar ranges. Therefore, each map is normalised to a fixed range of [0...M], usually [0...1]. To separate salient objects from noise or less salient objects, Itti *et al.* introduce two different weighting methods. Together, normalisation and one of the two weighting operations are expressed by the normalisation operator $\mathcal{N}(.)$.

The first and originally introduced weighting method makes use of the local and global maximum values of each map. For this, the global maximum value M is determined, as well as the average of all local maximum values $\overline{m}$. Finally, the whole map is multiplied by $(M - \overline{m})^2$. The advantage of this method is a gain of a single salient area. The disadvantage is an extreme suppression of salient regions if more than one region of similar values appears. In that case, the average value of local maxima is close to the global maximum and hence the multiplier drifts to zero.

The second operation is biologically more plausible, but computationally more complex. Corresponding to (Itti, Models of Bottom-Up and Top-Down Visual Attention, 2000), three conditions shall be fulfilled by this method. Firstly, by uncommon relations between a centre and its surroundings, an inhibition of the surround should happen. Secondly, inhibition shall be strongest if surroundings share similar stimulus properties to the centre. Finally, related to centre surround distances, inhibition has to be strongest at a certain distance and weaken radially symmetrically from and to the centre. An approach that comes close to the previously mentioned requirements is a two-dimensional difference-of-Gaussian pattern. For the two Gaussian functions, one small and one high σ is chosen. The small one serves as excitation, the higher one serves as inhibition (see Figure 3-4).

Each feature map $\mathcal{M}$ is iteratively subjected by the two-dimensional difference of Gaussian filter and added to the original map:

$$\mathcal{M} \leftarrow |\mathcal{M} + \mathcal{M} * \mathcal{DoG} - C_{inh}|_{\geq 0} \tag{3.10}$$

All negative values are set to zero. C_{inh} is a constant that adversely affects the excitation in cases where excitation and inhibition balance each other (in (Laurent, 2000), C_{inh} is set to 0.02).

After normalisation, all feature maps are combined in three *conspicuity maps*, $\bar{\mathcal{I}}$ for *intensity*, $\bar{\mathcal{C}}$ for *colour* and $\bar{\mathcal{O}}$ for *orientation*. These maps are fused on scale 4 of the image pyramids:

$$\bar{I} = \bigoplus_{c=2}^{4} \bigoplus_{s=c+3}^{c+4} \mathcal{N}(\mathcal{I}(c,s)) \tag{3.11}$$

$$\bar{C} = \bigoplus_{c=2}^{4} \bigoplus_{s=c+3}^{c+4} [\mathcal{N}(RG\ (c,s)) + \mathcal{N}(\mathcal{BY}(c,s))] \tag{3.12}$$

$$\bar{\mathcal{O}} = \sum_{\theta \epsilon \{0°,45°,90°,135°\}} \mathcal{N}\left(\bigoplus_{c=2}^{4} \bigoplus_{s=c+3}^{c+3} \mathcal{N}(\mathcal{O}(c,s,\theta)) \right) \tag{3.13}$$

The *conspicuity maps* are normalised again. The sum of all maps finally gives the *saliency map* $\mathcal{S}$:

$$\mathcal{S} = \frac{1}{3}\left(\mathcal{N}(\bar{\mathcal{I}}) + \mathcal{N}(\bar{C}) + \mathcal{N}(\bar{\mathcal{O}})\right) \tag{3.14}$$

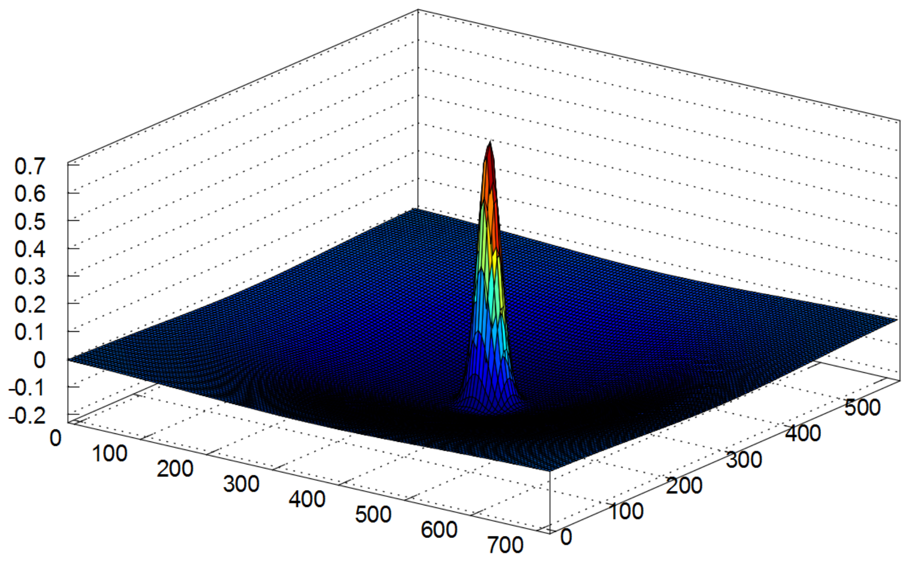


Figure 3-4: Difference of two two-dimensional Gaussian functions with $\sigma_{ex} = 2$ % and $\sigma_{inh} = 25$ % as proposed in (Itti, Models of Bottom-Up and Top-Down Visual Attention, 2000) for an arbitrary scale.

3.5.1.3 *Winner take all*

After fusing all maps to a single *saliency map*, the most salient regions are defined by a neuronally inspired approach called *winner-take-all* (WTA). The model consists of neurons, charged by the synaptic input. The neuron that first reaches a threshold fires and the *focus of attention* (FOA) is shifted to this neuron. After that, a global inhibition is applied to reset all neurons. Then the charge and fire process restarts, whereby the FOA and its close surroundings are suppressed. This shifts the FOA and avoids a return to the previous one (*inhibition of return*).

3.5.2 Spectral Residual

An alternative visual attention system to the one by Itti *et al.* is Spectral Residual by Xiaodi Hou (Hou & Zhang, Saliency Detection: A Spectral Residual Approach, 2007). This approach relies on the fact that man-made images share similar structures that can be considered as redundancy. Non-redundant features are therefore deviations from the norm. A visual system only signals such unexpected features to the next level of processing (Koch & Poggio, 1999). From the point of view of information theory, this decomposition of information H in an image I can be described as follows:

$$H(I) = H(\text{Innovation}) + H(\text{Prior Knowledge}) \tag{3.15}$$

where $H(\text{Innovation})$ is the novelty part and $H(\text{Prior Knowlegde})$ denotes redundancy. Hou proposes a method which detects the “innovation” part of an image by defining and removing statistically redundant components. He proposes the log-spectrum of a DFT as a good representation of redundant components in images due to its well-distributed frequency components in the magnitude spectrum (cf. Section 2.4). By examining the averaged log-spectra of a wide range of typical man-made images, a clear similarity can be noticed in the magnitude course. This sharing of similar trends is based on statistical singularities in images.

The Fourier space domain is highly sensible to structural changes in an image, e.g. an object on a plain background. Hence, abrupt and uncommon local structural changes can be easily detected. To compute saliency by Spectral Residual, an input image I is converted to a grey image and then transformed into the frequency domain by a DFT:

$$A(f) = |\mathcal{F}(I(x))| \tag{3.16}$$

$$P(f) = arg\left(\mathcal{F}(I(x))\right) \tag{3.17}$$

where $A(f)$ is the magnitude and $P(f)$ is the phase of each frequency component. Next, the natural logarithm of the magnitude is calculated:

$$L(f) = ln(A(f)) \tag{3.18}$$

The Spectral Residual is defined by subtracting an approximated copy of $L(f)$ from $L(f)$:

$$R(f) = L(f) - L(f) * h_n(f) \tag{3.19}$$

where $h_n(f)$ is a local average filter with a kernel size of $n \times n$. Hou proposes $n = 3$ due to negligible changes when using bigger kernel sizes such as 5x5 and 7x7. Corresponding to Equation (3.15) $R(f)$ represents the “innovation” part and $L(f) * h_n(f)$ the prior knowledge.

Finally, the saliency map is determined by an inverse Fourier transform of $R(f)$ from frequency domain to spatial domain:

$$S(x) = g(x) * \mathcal{F}^{-1}\left[exp\big(R(f) + i \cdot P(f)\big)\right]^2 \tag{3.20}$$

where $g(x)$ is a Gaussian filter with a standard deviation of $\sigma = 8$ and kernel size of 3x3. The Gaussian filter is applied for a better segmentation of corresponding areas. The squaring of the inverse transformed image is for image enhancement (gamma correction) to emphasise high intensities and suppress small intensities.

Different results can be achieved dependent on the size of the input image. The higher the image resolution, the less salient are large features. In turn, small images support large features and omit details. Hou proposes a size of 64 pixels (either width or height) which is a good approximation of normal visual conditions.

3.6 Summary

This chapter introduced two system implementations motivated by visual attention. They define regions of interest by applying common computer vision methods.

The system by Itti is the more biologically motivated one and is close to the visual attention models introduced at the beginning of this chapter. The approach by Hou, in contrast, relies on statistical singularities in images which can be best identified in the log-spectrum of DFT transformed images. As can be imagined from the complexity of both systems, the one by Hou is much less computationally expensive than the one by Itti.

The implementation of visual attention for video images provides an important contribution to this work. Therefore, the proposed systems are revisited in Chapter 7 to evaluate their suitability for determining ROIs in the area of broadcast applications.

4. Image Composition

This chapter steps a bit out of line in that it discusses some artistic aspects. Nonetheless, these are important and very interesting aspects that must accompany the subject of visual attention, if one is to consider the extraction of features by computer vision and the composition of a new image by means of cropping. Images shot by a cameraman, whether they are still images or moving pictures, are not random in their composition. A cameraman's intention is to convey information by attracting the viewer's visual focus to a specific point. Designing information is perhaps the key element in image composition: "Information design is defined as the art and science of preparing information so that it can be used by human beings with efficiency and effectiveness." (Jacobson, 2000)

The interpretation of information can be divided into three levels (Weber, 1990). The first level is *perception*. *Perception* is a physiological habitude of human beings and is a result of interplays of past experience and interpretation of the perceived. The second level is *recognition*. It is a semantic aspect and involves identifying objects or events. On the last level are *emotions*. *Emotions* are reactions on the recognition.

Compared to the previous chapter, image composition and visual attention systems might close a circle at this point; a cameraman tries to attract a viewer's attention while a visual attention system works like the early perception of a human being. Unfortunately, the early perception is just the first level of information interpretation. Therefore, much information is not "understood" by a bottom-up visual attention system.

As the proposed implementation focuses on sports applications, the examples of image compositions in this chapter are taken from sports productions.

4.1 Composition Makes Order Out of Confusion

It might sound weird to talk about rules of compositing in photography. Even so, there are clear concepts about how to attract a viewer's attention. The aim of a cameraman is to lead the attention of the viewer to those elements that the cameraman found most relevant.

4.2 Positioning Objects in an Image

In sports photography, the most important question of composition is how to position *objects* in the space enclosed by the frame (*field*). The shape that is formed around an object, thus the difference between *field* and *object*, is called *negative space*. Changing the *negative space* by positioning *objects* at different positions provides a different visual sensation (Clements B., 1980).

Objects located more at the centre of an image convey the impression of being balanced and *static*, while objects tending to the image boundary or image corner are more *dynamic*, i.e. the image composition is unstable. Shapes and lines seem to be in motion or to fall over. In their turn, objects at the top of an image seem to be light while objects at the bottom seem to be rooted. In the same way, objects at the top or bottom can convey the impression of depth (Feininger, 1965).

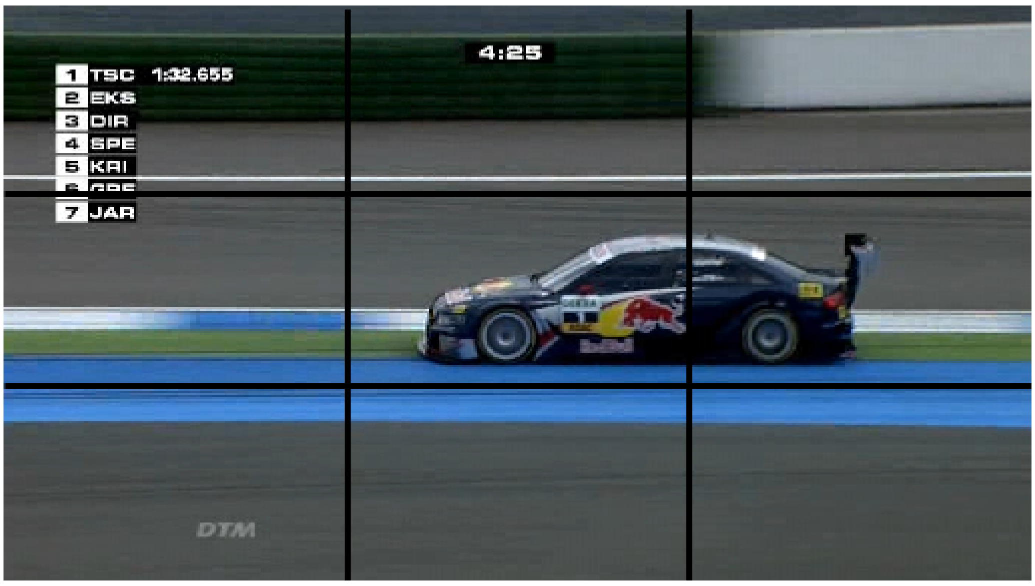


Figure 4-1: Example of the rule of thirds applied to an object moving from right to left. Due to fast camera motion, the effect of motion blur additionally appears in this example.

A common separation of an image and maybe one of the major concepts of composition is the *rule of thirds* (see Figure 4-1). It is an approximation of the *golden ratio* that has a relation of 1,618:1. The *golden ratio* has a very harmonious and pleasing proportion. This fact has been proven by many scientific tests (Doczi, 1981). For example, the *rule of thirds* is applied to objects moving from left to right or vice versa. Usually, more space is allowed on the side into which the object is moving, while the object is kept relatively static and the camera is moving. As mentioned above, this promotes an “in-motion” impression. Besides that, it gives the viewer an outlook on the area into which the object is moving.

4.3 Depth of Field and Motion Blur

Depth of field is a three dimensional acuity and can be roughly distinguished in two types – shallow and great. The former allows the separation of foreground and background from a subject by blurring the background. The latter moves nearly everything of a 3D scene into focus.

Motion blur is a function of the exposure time – the greater the motion offset during exposure time, the more motion blur appears in the image. In the case of a fast subject, tracked by a camera, subject and background/foreground are clearly separated by this effect (see Figure 4-1).

4.4 Composition Guidelines for European Public Broadcasters

In broadcast productions, there are a few limitations in image composition which aim to ensure a proper workflow. All requirements (technical and artistic) for such a workflow are specified in guidelines. For European public broadcasters, those guidelines are defined by the EBU (European Broadcasting Union) and generally in a similar way on national level by each public broadcaster.

The EBU defines two image format standards for broadcasting: 4:3 (12:9) and 16:9. Additionally, the guidelines state how graphics and all essential action shall be positioned in an image (EBU Technical Recommendation R95-2000 , 2000). This assures a proper image composition, starting from the shot image to the graphic designer to the possibly prepared image on the end device (e.g. missing edge of the screen on a CRT or adaptations of flat screens to the TV raster). In addition, format conversions from 16:9 to 4:3 and vice versa are defined.

4.4.1 Safe areas

Safe areas define an enclosed region in an image that contains necessary information. The *action safe area* defines the region for essential action (see Figure 4-2) and the *graphics safe area* defines the regions of overlaid graphics (see Figure 4-3).

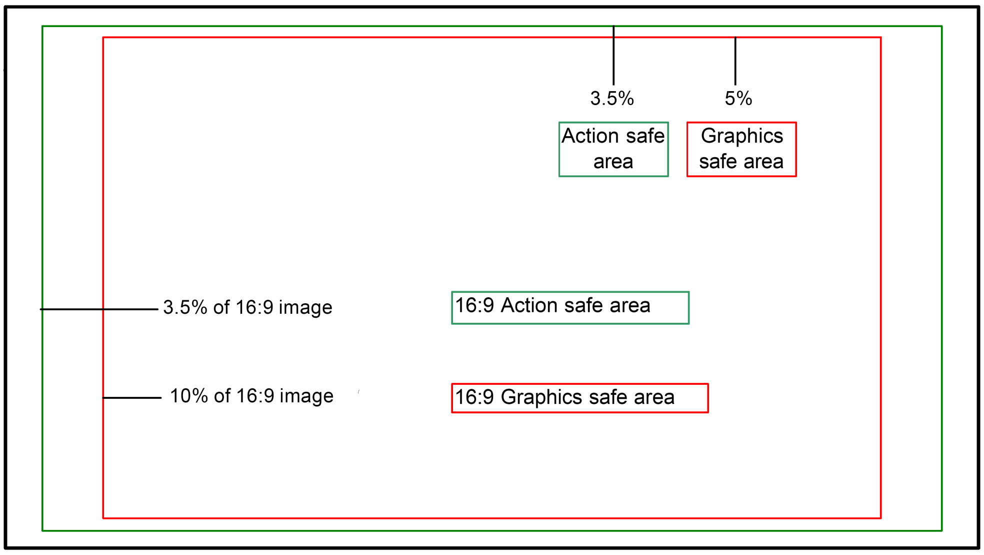


Figure 4-2: ***Action safe area*** **and** ***graphics safe area*** **for 16:9 image formats (Technical Guidelines for the production of Television Programmes for ARD, ZDF and ORF, 2006).**

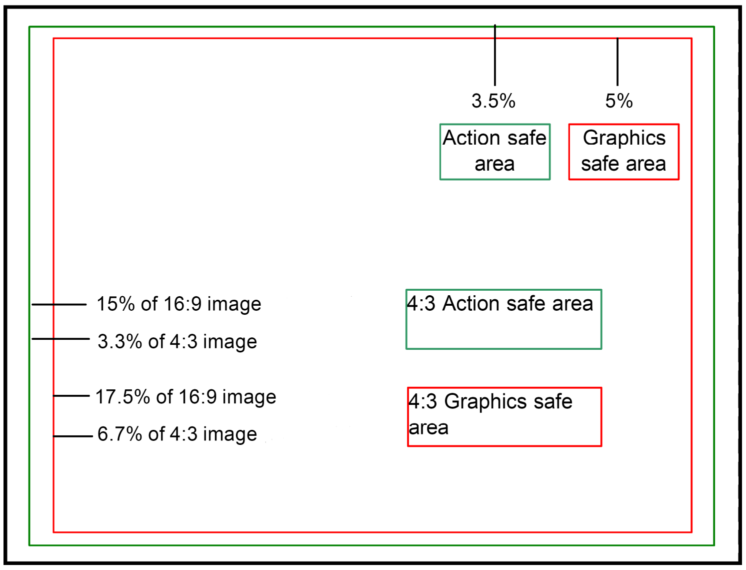


Figure 4-3: ***Action safe area*** **and** ***graphics safe area*** **for 4:3 image formats (Technical Guidelines for the production of Television Programmes for ARD, ZDF and ORF, 2006).**

4.4.2 Scanned image areas

For conversions between 16:9 and 4:3, either the images keep their original aspect ratio (consequently, empty areas of the target format are blackened) or the original image is scanned by a mask that has the aspect ratio of the target format.

4.4.2.1 *Conversion from 16:9 to 4:3*

Corresponding to previously mentioned guidelines, 16:9 images have to be scanned by 4:3 masks for conversion. The scanning mask has to have the largest possible resolution, i.e. one which fits into a 16:9 image. The scanning process can be done in two ways. In the first way, the 4:3 extract is statically positioned in the middle of the

16:9 image. This approach requires that the extracted 4:3 picture frames the main subject (*shoot-to-protect*) and complies with the normal artistic practice of framing in 4:3 (EBU Technical Recommendation R95-2000 , 2000). In the second, the mask is statically positioned out of centre or is tracked by an operator (*pan & scan*).

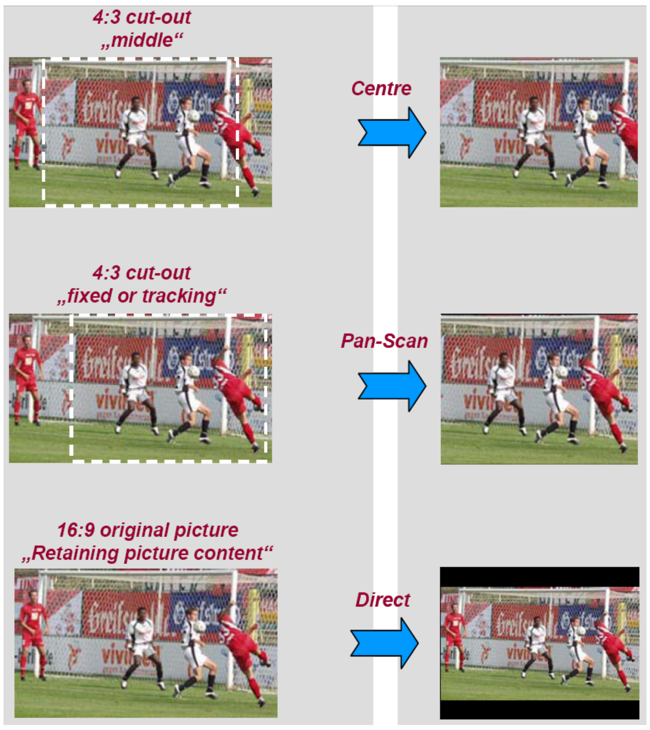


Figure 4-4: Conversion from 16:9 to 4:3 image formats by centre cut-out, pan & scan or letterbox (Technical Guidelines for the production of Television Programmes for ARD, ZDF and ORF, 2006). In this example, a centre cut-out should be avoided.

4.4.2.2 *Conversion from 4:3 to 16:9*

In the same way as 16:9 is adapted to 4:3, a 16:9 mask scans the 4:3 picture to obtain the widescreen standard. Here, a *shoot-to-protect* approach is less common. Therefore, a static extract can hardly be applied.

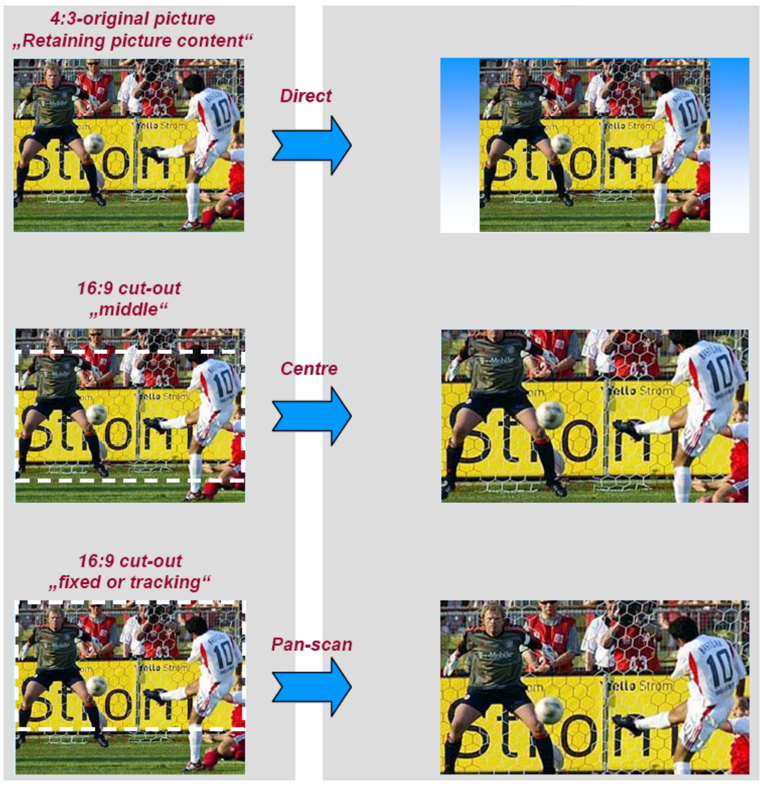


Figure 4-5: Conversion from 4:3 to 16:9 image formats by adding graphics, centre cut-out or pan & scan (Technical Guidelines for the production of Television Programmes for ARD, ZDF and ORF, 2006). In this example, a centre cut-out should be avoided as well.

4.5 Summary

This chapter gave a brief overview of image compositions and guidelines for broadcast productions. Image compositions have a strong influence on information encoded in an image, i.e. encoded in its signal components. Such information originates from the intentions of the cameramen while shooting a scene. Hence, that information can be interpreted as top-down information.

Image compositions will be referred to again at various points within this work when making decisions about certain methods.

5. State of the Art ROI Extraction in Video Applications

In the previous chapters, an overview has been given of methods and systems which build a basis for detecting ROIs in images. Many works currently exist which are heading in the same direction by applying such methods. Most of them have the same objective of cropping out regions that contain the most relevant content. As a survey, related works are presented in this chapter. Finally, the summary at the end of this chapter discusses the motivation of the proposed implementation compared to existing approaches.

The first applications for the detection of ROIs were developed for still images. Generally, such algorithms adopt the approach of the previously introduced attention model by Itti and apply such additional extraction methods as text and face detection (Chen, Xie, Fan, Ma, Zhang, & Zhou, A visual attention model for adapting images on small displays, 2002; Chen, Xie, Fan, Ma, Zhang, & Zhou, Visual attention based image browsing on mobile devices, 2003). A more general approach computes the energy of an image by derivations of the intensity in the x- and y-direction (Avidan & Shamir, 2007). In this way, homogenous image parts with low energy are interpreted as less interesting. Instead of cropping, parts with low energy are spatially compressed and interesting parts are kept in their original shape. This is called seam carving. Within this work, this approach is not of interest, because image distortion can come into the picture which is highly undesirable for TV productions.

For video content, motion becomes available as an important additional feature to automatic ROI detection and cropping algorithms. (Zhang H.-J. , 2002) creates a motion saliency map by exclusively considering information extracted from MPEG motion vectors. The map is computed by allocating each vector to a motion histogram and analysing the distribution of the histogram for several video frames. Finally, the video is skimmed by defining temporal segments of intensive motion. An application developed to summarise videos for mobile devices is proposed in (Kopf, Haenselmann, Farin, & Effelsberg, 2004). It does not use a visual attention model, but defines ROIs by shot boundary, texts, faces and motion detection. Moving objects are extracted by defining a background template for each shot and the difference between current frame and background model. Additionally, objects are tracked and only those that remain tracked for several frames are considered as reliable. Besides cropping, videos are temporally compressed by removing shots of no interest. In (Cheng, Chu, & Wu, 2005), an attention model relying on intensity, colour and motion is applied. Intensity and colour interpretation rely on the model of Itti. Motion is detected and categorised (e.g. camera motion / no camera motion, zoom, pan, etc.) by tensor histograms. Dependent on the motion type, salient extracted areas are weighted for each frame. Finally a median filter is applied to smooth ROIs temporally and spatially. In (Deselaers, Dreuw, & Ney, 2008), a motion map is determined by optical flow and an appearance map is obtained from colour information. A saliency map is additionally computed by the Spectral Residual approach, already mentioned in Section 3.5.2. The combination of all maps results in a weighted map. From this map, a sequence of cropping areas is chosen that encloses as many relevant image parts as possible. The cropping areas are defined by applying zoom, pan and scan.

In (Numata, Senoo, & Shishikui, 2008), NHK presented the broadcast service AdapTV. AdapTV is a video trimming system which combines broadcasted metadata and user profile information to adapt video content to mobile display viewing conditions. The system requires content and object related metadata which has to be annotated by the broadcaster. In addition, the system combines a simple long and short shot detection to choose an optimised cropping area on the receiving side. It should be mentioned that AdapTV does not extract ROIs automatically.

A prototype solution ("Helios") was presented by Snell & Wilcox (Zaller, 2007). This system is able to zoom in if clearly defined ROIs are available. Furthermore, Snell & Wilcox's publications explain techniques planned for future professional conversion tools using two approaches for foreground and background separation (Knee, 2008). The first approach estimates global motion by phase correlation motion estimation. Additionally, intra-frame saliency is determined by colour elements that are more likely to attract the viewer's attention. The second approach uses a clustering method which assigns pixels of similar properties to a segment. These segments are matched in consecutive frames. Available ROI information is used to dynamically adapt the source material to the desired target resolution. Snell & Wilcox solutions also make use of seam carving.

In 2008, Thomson (now Grass Valley) developed the ViBE Mobile TV encoder which also relies on ROI detection for repurposing video content (ViBE Mobile TV Encoder

Data Sheet, 2009). The applied visual attention model which is proposed in (Le Meur, Le Callet, & Barba, Predicting visual fixations on video based on low-level visual features, 2007) analyses contrast in the Krauskopf colour space (achromatic component, red / green antagonist components, blue / yellow antagonist components) by applying a contrast component function in the frequency domain. Additionally, visual masking effects and orientation are considered. Temporal saliency is detected by hierarchical block matching and M-estimator regression for fitting 2D global affine motion. Thomson provides results of user tests, indicating that 90% of the users were watching the image areas that were proposed by the system as ROIs. The test set-up consisted of 16 observers and four video clips (Le Meur, Cloarec, & Guillotel, Automatic content repurposing for mobile applications, 2008). 3-Italia, Europe's largest DVB-H service provider, has made use of the Thomson head-end since September 2008.

Most of the approaches presented do not make use of prior knowledge about the content in order to analyse and automatically choose important areas. Therefore, the rating of the importance of ROIs can only be done independently of the context and is based on general assumptions.

However, there exist content specific methods as well. In (Dearden, Demiris, & Grau, 2006), a football player tracking system is presented. Players are extracted by histogram backprojection of the field colour. In the next step in the process, particle filters are used to track players. A simpler approach for football player tracking is proposed in (Figueroa, Leite, Barros, Cohen, & Medoini, 2004). This approach extracts players by subtracting the current image from a background model. Found blobs are classified by shape and intensity distribution and linked between frames.

Examples for combining bottom-up and top-down information are also known from the field of robotics. For example in (Frintrop, 2006), the Neuromorphic Vision Toolkit by Itti *et al.* (Itti, iLab Neuromorphic Vision C++ Toolkit, 2010) is optimised for real time robotic applications in the attention system VOCUS (Visual Object detection with a CompUtational attention System). In addition to run-time optimizations for bottom-up saliency map computation, feature vectors can be trained manually or automatically. Those feature vectors can be used to weight extracted bottom-up features.

5.1 Summary

In summary, it can be said that top-down information for ROI extraction in broadcast applications is rarely used. Specific applications based on such information work exclusively for a single type of content and sometimes even need a special set-up on site. On the other hand, a classifying or weighting of ROIs can hardly be applied based on bottom-up information only.

In this work, prior knowledge about the type of broadcast content is obtained from metadata information, which has recently become available in the production workflow as a result of the transition to tapeless production. In file-based production environments metadata is available in electronic form, which facilitates feeding any post-production system with additional information. The next chapter gives an introduction to the metadata format used within this work. The following chapters explain the combination of this data with computer vision methods in the context of the developed system.

6. Metadata in the Production Workflow

As the purpose within this work is to use previous knowledge to guide feature extraction, this information must be fed into the system. The most intuitive way for broadcast productions is to make use of available content metadata. Such data is more and more available thanks to the transition from tape to tapeless productions. Unfortunately, there exists no uniform metadata standard for media content in the broadcast production environment. For this reason, there is a great necessity and demand for unification. Such a standardisation would simplify the exchange of metadata within broadcasting corporations as well as between broadcasters.

One possible metadata specification meeting the requirements of uniformity for broadcast productions is introduced in this chapter. Firstly, however, parts of this specification necessary for this work are explained. Then a tool is presented which has been programmed to create metadata files that feed the system with the required content information.

6.1 The BMF Specification

In collaboration with public broadcasters, an open metadata specification has been worked out at IRT (Institut fuer Rundfunktechnik). This specification, called BMF (BMF – Broadcast Metadata exchange Format, 2007), is therefore tailor-made for content description in the scope of broadcast productions. It is designed to serve as an electronic record report starting from the production site up to designing complete programmes and annotating content. Besides that, BMF is foreseen to be used as a pure XML format as well as to be wrapped in a container format, e.g. the widely used MXF standard (Material eXchange Format).

In the context of this work, the metadata format is not limited to BMF as long as it delivers the necessary information in a defined structure. Nevertheless, BMF has been chosen as metadata type for this work due to its special design for broadcast applications and its open specification.

Because of the complexity of BMF, this chapter introduces only those parts of the specification that are necessary for this work. These are the annotation of genre type and information about scene changes in a video.

6.1.1 Programme annotation

The BMF specification differentiates between two presentation form types, namely “fictional presentation form” and “non-fictional presentation form”. While the former includes content such as movies, the latter describes genres such as documentaries, news and reportages. A more detailed breakdown is purposely not provided by the specification itself in order to guarantee flexibility and extensibility. It is up to each broadcaster to decide how those annotation types are to be specified in more detail. To do so, broadcasters have to create thesauri as enumerated data types as well as batched enumerated data types for sub-divisions. Those enumerated types have already been established by German public broadcasters (ARD, ZDF) and are specified in the BMF documentation (BMF – Broadcast Metadata exchange Format, 2007).

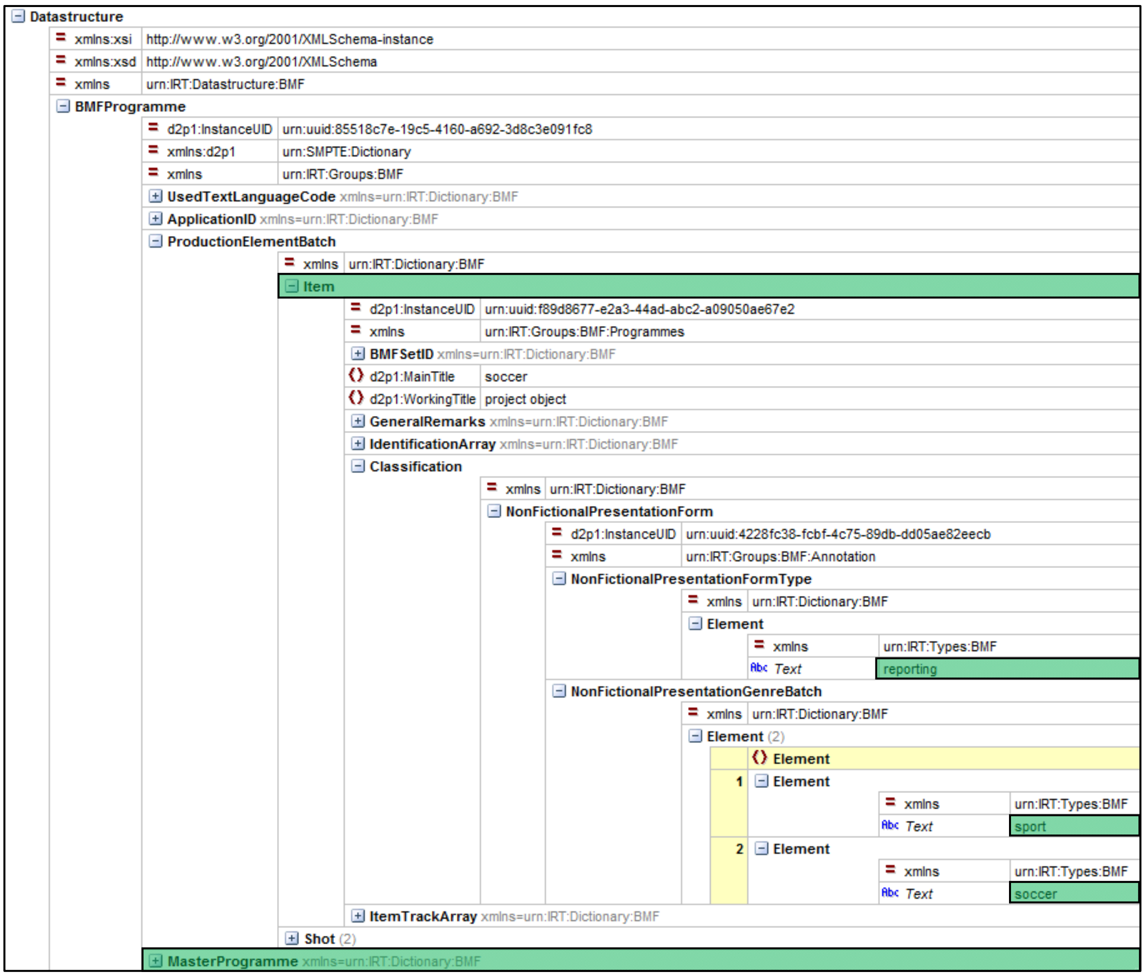
Fictional Presentation Form Type List
televisiondrama
televisionseries
filmlet
featurefilm
animatedfilm
musicclip
documentaryfeature
experimentalfilm

Non-Fictional Presentation Form Type
documentaryItem
docMix
magazine
reporting
address
call_in
discussion
documentaryfeature
interview
commentary
collection
course
news_collection
news_programme
spot
studio_action
reading
speech
semiDocumentation

Genre Type List				
architecture	education	leisuretime	society	communication
culture	fineArts	visualArts	literature	medicine
music	politics	law	religion	sport
technology	entertainment	environment	economy	traffic
cross_section	science	dailynews		

Figure 6-1: Excerpt from the BMF documentation (BMF – Broadcast Metadata exchange Format, 2007) of the thesauri specified by German public broadcasters. A Fictional Presentation Form Type (see upper boxes) can be described in more detail by a Genre Type (see lower box) for annotating a programme.

An excerpt of the thesauri for different presentation form types is listed in Figure 6-1. The thesauri do not specify a genre in more detail. For the purpose of this work, it is desired to know the type of sport as well. Therefore, beyond the BMF documentation, the “Genre Type List” of the BMF documentation is extended for the purposes of this application by any type of sport. To keep the hierarchy as proposed by German public broadcasters, the type of sport is added as a further element in the “Genre Type List”. An example of a BMF file annotated for the genre type soccer is given in Figure 6-2. Annotating such genre information could be easily done by an editor during a production workflow by an appropriate GUI.

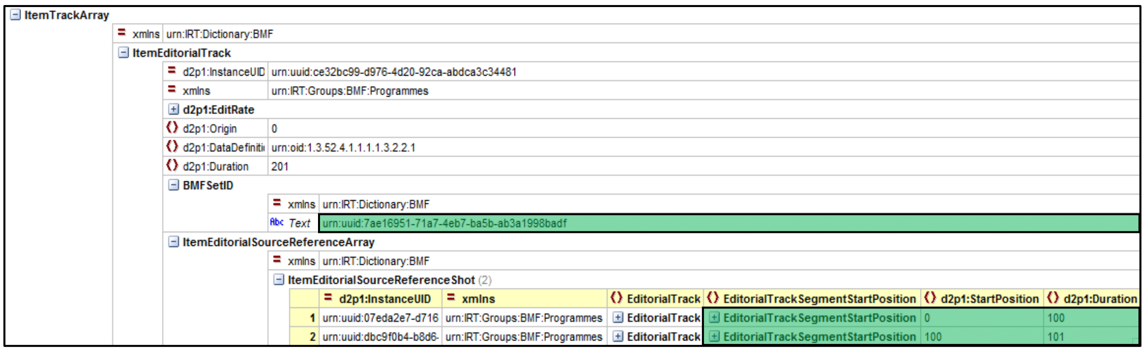


```
Datastructure
    xmlns:xsi   http://www.w3.org/2001/XMLSchema-instance
    xmlns:xsd   http://www.w3.org/2001/XMLSchema
    xmlns       urn:IRT:Datastructure:BMF
    BMFProgramme
        d2p1:InstanceUID  urn:uuid:85518c7e-19c5-4160-a692-3d8c3e091fc8
        xmlns:d2p1        urn:SMPTE:Dictionary
        xmlns             urn:IRT:Groups:BMF
        UsedTextLanguageCode xmlns=urn:IRT:Dictionary:BMF
        ApplicationID xmlns=urn:IRT:Dictionary:BMF
        ProductionElementBatch
            xmlns  urn:IRT:Dictionary:BMF
            Item
                d2p1:InstanceUID  urn:uuid:f89d8677-e2a3-44ad-abc2-a09050ae67e2
                xmlns             urn:IRT:Groups:BMF:Programmes
                BMFSetID xmlns=urn:IRT:Dictionary:BMF
                d2p1:MainTitle    soccer
                d2p1:WorkingTitle project object
                GeneralRemarks xmlns=urn:IRT:Dictionary:BMF
                IdentificationArray xmlns=urn:IRT:Dictionary:BMF
                Classification
                    xmlns  urn:IRT:Dictionary:BMF
                    NonFictionalPresentationForm
                        d2p1:InstanceUID  urn:uuid:4228fc38-fcbf-4c75-89db-dd05ae82eecb
                        xmlns             urn:IRT:Groups:BMF:Annotation
                        NonFictionalPresentationFormType
                            xmlns  urn:IRT:Dictionary:BMF
                            Element
                                xmlns  urn:IRT:Types:BMF
                                Text   reporting
                        NonFictionalPresentationGenreBatch
                            xmlns  urn:IRT:Dictionary:BMF
                            Element (2)
                                    Element
                                1   Element
                                        xmlns  urn:IRT:Types:BMF
                                        Text   sport
                                2   Element
                                        xmlns  urn:IRT:Types:BMF
                                        Text   soccer
                ItemTrackArray xmlns=urn:IRT:Dictionary:BMF
            Shot (2)
        MasterProgramme xmlns=urn:IRT:Dictionary:BMF
```

Figure 6-2: Table view of an example BMF file for the type of sport soccer. The whole structure of the programme is described in the node MasterProgramme (see lowest marked node). All items that are part of this programme are arranged in a batch (see upper marked node). Each item is referenced in the programme by a UUID, whereas an item in turn is further specified by its presentation form (see marked nodes in the middle).

BMF allows only referencing of media. Therefore, the essence is not part of the metadata. This has the advantage that one media content could be used for multiple programme descriptions by simply specifying a universally unique identifier (UUID). This UUID is a unique assignment between metadata and corresponding media.

Besides simply referencing the essence, BMF allows one to describe the media in more detail. This possibility is used in this work to further divide a video into shots by indicating their boundaries. To do so, an item is further described by a batch of shots. In turn, several items can be part of one programme. To allow such a nesting, UUIDs are used again to make a connection between programme and items. Such a list of shots for an item is depicted in Figure 6-3 based on the example of Figure 6-2. So far, only manually annotated shot boundary information is processed by the system (see next section). Such information could be easily received by recording the editor action as metadata in future.


```
ItemTrackArray
    xmlns  urn:IRT:Dictionary:BMF
    ItemEditorialTrack
        d2p1:InstanceUID  urn:uuid:ce32bc99-d976-4d20-92ca-abdca3c34481
        xmlns             urn:IRT:Groups:BMF:Programmes
        d2p1:EditRate
        d2p1:Origin       0
        d2p1:DataDefiniti urn:oid:1.3.52.4.1.1.1.1.3.2.2.1
        d2p1:Duration     201
        BMFSetID
            xmlns  urn:IRT:Dictionary:BMF
            Text   urn:uuid:7ae16951-71a7-4eb7-ba5b-ab3a1998badf
        ItemEditorialSourceReferenceArray
            xmlns  urn:IRT:Dictionary:BMF
            ItemEditorialSourceReferenceShot (2)
```

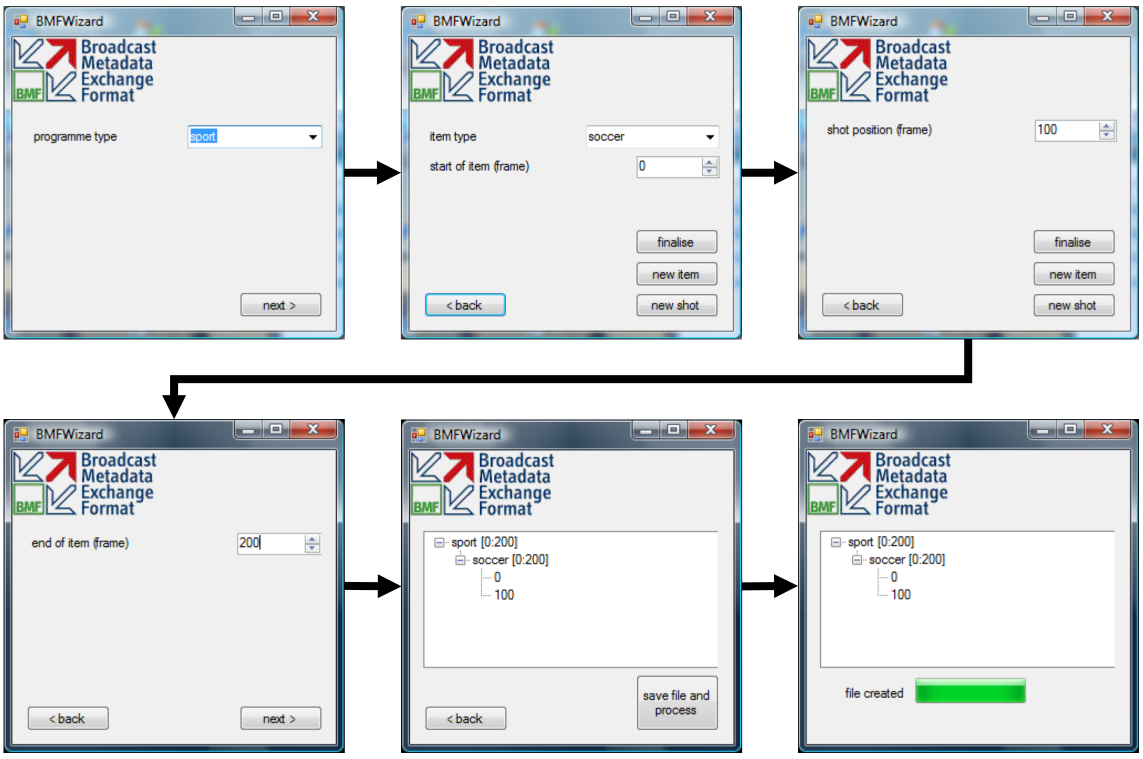
	d2p1:InstanceUID	xmlns	EditorialTrack	EditorialTrackSegmentStartPosition	d2p1:StartPosition	d2p1:Duration
1	urn:uuid:07eda2e7-d716	urn:IRT:Groups:BMF:Programmes	EditorialTrack	EditorialTrackSegmentStartPosition	0	100
2	urn:uuid:dbc9f0b4-b8d6-	urn:IRT:Groups:BMF:Programmes	EditorialTrack	EditorialTrackSegmentStartPosition	100	101

Figure 6-3: More detailed table view of the example of Figure 6-1. The item track in this item array is uniquely identifiable by a UUID (see upper marked element). Thus this item can be referenced by a programme as depicted in Figure 6-1. This item further contains a list of shots where each specifies a start position and duration (see lower marked elements).

The BMF specification does not specify a fixed hierarchy of items and programmes, but rather enables them to be loosely related by inheritance. This allows a dynamic program design. For the purposes of this work, a simple structure of items as described above is entirely sufficient. For further information on possible BMF structures, the author refers to (BMF – Broadcast Metadata exchange Format, 2007).

6.1.2 BMF annotation tool

Up to now, BMF exists only as a specification and hence no tools for file creation or file conversion exist. Within this work, a simple BMF-XML writer based on the BMF-XML-schema has been programmed. It allows the description of one programme that contains several items by a graphical step-by-step-wizard (see Figure 6-4). Each item is assigned to a type of subgenre. Additionally, shot boundaries can be defined for each item. Due to this limitation this tool makes use of just a small part of the whole BMF specification.


programme type: sport

item type: soccer

start of item (frame): 0

shot position (frame): 100

end of item (frame): 200

```
sport [0:200]
  soccer [0:200]
    0
    100
```

file created

Figure 6-4: Example for creating a BMF file for soccer content by the BMF wizard.

6.2 Summary

This chapter gave a very short overview of the relevant parts of the Broadcast Metadata Exchange Format (BMF) for this work. The most important information includes the type of genre and shot boundary positions. This descriptive data could be easily annotated in a metadata file by editors of broadcast productions. For example, the shot boundary information could be records of an editor's action to identify the cameras that are on air.

Such an electronic description is used to feed the system with previous knowledge about the video content. This allows guiding of the feature extraction in order to identify ROIs more reliably. How this information influences the ROI extraction is described in the next chapter in Section 7.2.2.

7. Implementation

Based on the previous chapters, each component of the developed application is explained in this chapter. As already mentioned, the implementation makes use of computer vision methods and visual attention systems (cf. Chapters 2 and 3) and combines these with previous knowledge encoded in video images by image composition (cf. Chapter 4) and context related knowledge from descriptive metadata (cf. Chapter 6). Such a fusion allows a more specific search for ROIs than relying on bottom-up information only. Nevertheless, the system is not dependent on such information and can run in a default mode as well. As the final goal of the application is to automatically define cropping areas, extracted ROIs are used to find the most attracting and contextually important regions in a sequence of video images.

The following sections give an insight into the developed application by presenting the design and functionalities of the system. First of all, an overview of the technologies and libraries used for this implementation is given. In Section 7.2 the idea of a modular system is brought closer and its realisation within this implementation is discussed. Furthermore, the linkage between metadata and computer vision algorithms is presented. Section 7.3 deals with the implementation of each plug-in that can be loaded by the system. A distinction is made between two groups of plug-ins: extraction plug-ins and plug-ins working on higher processing levels. The former are responsible for extracting ROIs on low level computer vision methods, whereas the latter interpret ROIs to categorise their contextual relevance and finally define cropping areas.

7.1 Technology

7.1.1 Microsoft Visual C++

The application has been developed with the IDE (Integrated Development Environment) *Microsoft Visual Studio 2008*. It allows the execution of two different types of C++ applications. One uses *native C++ code* whereas the other option is to run C++ code under the control of the *.NET Framework*. The *.NET Framework* is the central concept of all *.NET* development products provided by Microsoft including high level programming languages such as C#, F#, Visual Basic and C++. Basically it consists of a set of libraries (*.NET Framework class libraries*) and the *Common Language Runtime* (CLR) which executes an application. Programs that are based on

C++ but run under this extension are called C++/CLI (*Common Language Infrastructure*) programs or CLR programs.

The CLR provides garbage collection (GC) for dynamically allocated memory. Therefore, CLR code is often referred to as managed code. This memory management provides a simplification of allocation and release processes compared to native code.

Fundamental parts of .NET form assemblies that are building blocks containing intermediate language code. They are generated after successful compilation of any .NET application. There exist *process assemblies* (with the file extension .exe) and *library assemblies* (with the file extension .dll). The former contain all required information in a single package. The latter can be used for larger applications by numerous assemblies. All assemblies contain a manifest[1] file which contains the information necessary for using or executing it. This technology of independent blocks which can be loaded at runtime provides a central concept used within this work. It supports the basic idea of loading different modules, dependent on the type of incoming video content.

7.1.2 OpenCV

OpenCV (Open Source Computer Vision) is a C/C++ library which was originally developed by Intel. Nowadays, the library is maintained by Willow Garage (Willow Garage, 2010). This library mainly provides real time image processing methods such as object identification, segmentation, recognition, motion estimation, multiple-camera depth computation and much more. It is a very comprehensive collection of functions which also supports reading of uncompressed or decoded video input from file or cameras as well as writing uncompressed video, i.e. image data. The most common image format in OpenCV is the *IplImage* structure which is inherited from the Intel Image Processing Library. This structure provides raw image data from one to four channels and quantisation of 8-bit signed/unsigned, 16-bit signed/unsigned and single/double precision floating points. Most OpenCV operations support this image format.

[1] A manifest contains metadata that is necessary for executing the application including all externally referenced assemblies

7.1.3 FFmpeg

FFmpeg is a collection of open source libraries for decoding, encoding, recording, converting and streaming a wide range of video and audio formats (FFmpeg, 2010). It includes the leading audio/video codec library *libavcodec* as well as *libavformat,* which is an audio/video container multiplexing and demultiplexing library. It is written in C and hence can be used cross-platform. Well known applications that make use of FFmpeg are MPlayer and VLC-player (VideoLAN).

The combination of OpenCV and FFMpeg provides a very powerful framework for image processing. For this application, both libraries are used and included in the Microsoft Visual Studio .NET IDE.

7.2 System Design

The idea underlying the implementation is a dynamic loading of computer vision methods adapted to the content to be analysed. In this way, the extraction of ROIs can be optimised to the present video content. In the context of this work, the BMF metadata specification has been chosen (see Chapter 6) which provides “type of genre” information and shot information. What extraction methods are chosen can be influenced by defining profiles that are assigned to each type of content. After ROIs have been identified by the system, a cropping area is chosen in a final step for every video frame that encloses as many high-weighted ROIs as possible.

To meet the demands of a system which offers the previously mentioned functionality, some requirements have to be fulfilled. Due to the fact that, dependent on the video content, different extraction methods and settings have to be loaded, *flexibility*, *reusability* and *expandability* are of central importance. The implementation is mainly intended as a test environment. Therefore, *maintainability* should also not be disregarded. *Performance* is of more secondary importance. These requirements lead to following conceptual features used in the system:

1. *Modular approach*. Components with related functions have the same structure and interfaces. Furthermore, modules define the smallest possible combination of associated operations with the smallest possible external interface. This allows high *reusability* of components as well as high *expandability*.

2. *Function-Specialisation*. Components that are based on the defined structure receive a method signature according to their functionality. The signature defines the required in- and output data. Besides this, components of identical functionality have the same method signature.

3. *Plug-In Concept*. The plug-in concept allows a high *flexibility* of the system. As a plug-in wraps one or several modules, the exchange of modules does not influence the interface between plug-in and system core. This provides a high *maintainability*.

4. *System Core*. The system core manages all configurations and modifications based on the modular structure.

5. *Independent Input/Output Operations*. All file operations, such as reading a video file, can be processed independently from the system core.

6. *Logging*. States and processes can be logged. This allows tracking of each component and its operations.

Each plug-in functions independently. All modules which are part of a plug-in must however incorporate some basic functionality. This involves managing parameters as well as information about the module itself and its environment. A distinction is drawn between extraction plug-ins which are a collection of computer vision methods and plug-ins that work on higher level. The latter ones include the Cropping Plug-in and the Classification Plug-In.

The *plug-in-system* which is controlled by the system core is located at the next higher level. It loads all required plug-ins and builds up the structure of the application. What plug-ins are loaded is determined by the incoming metadata. If no metadata is available, a default setting is loaded. Figure 7-1 depicts the overall structure of the application.

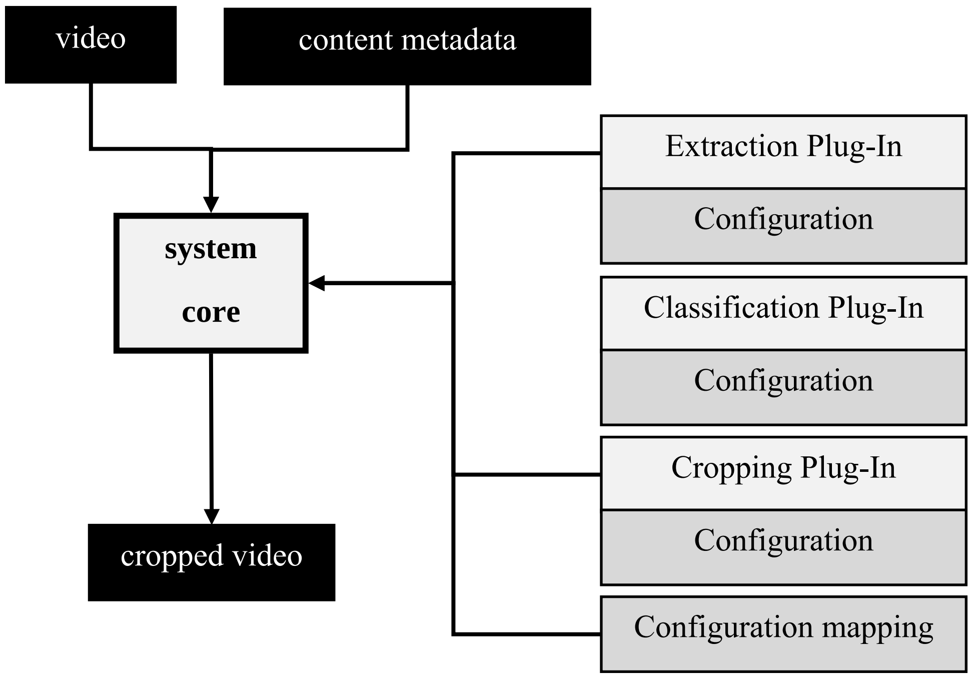


Figure 7-1: Overall structure of the system. Black boxes represent input and output data whereas dark grey boxes represent data used for system set-up and configuration. Light grey boxes represent the software components. The system core manages incoming data as well as the configuration of the application by loading plug-ins.

7.2.1 Modular structure and module settings

The modular structure forms the basis of the application. It follows the *composite pattern*. Hence, a module can in turn consist of several or no further modules (sub-modules). In the case of several sub-modules, the main module serves as a container and administrator for all underlying components. In this way, complex algorithms can be split into blocks of operations. This allows a flexible changing of logic units. To allow parallel processing, the module class is extended by an abstract class for multithreading.

Each module defines an associated parameter set. The parameter set contains all values that are necessary to run the module, where a set can be defined either by the module itself or by feeding data in the form of an XML file. The type and ranges of the values in a data set can be limited by minimal and maximal values to avoid incorrect initialisation.

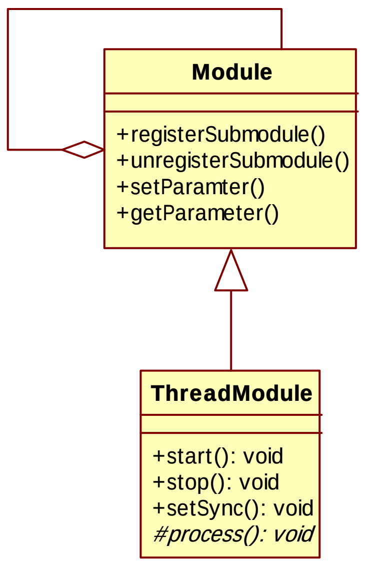


Figure 7-2: The composite pattern applied on modules (top) and module extensions for multithreading (bottom).

7.2.2 The plug-in system

Here, a plug-in is a block of full functionality which can be included in the system at runtime. Within this application, a plug-in is realised by a library assembly (*.dll*). All plug-ins have to conform to the *IPlugIn*-interface to offer a consistent method signature (see Figure 7-3). Internally, plug-ins define their own parameters and methods. As mentioned in 7.2.1, plug-ins can consist of additional sub-modules, where each of these can run as separate thread.

The plug-in system loads all necessary plug-ins for running the application and sets all parameters that are defined, either by the module itself or by a XML file. Beside this, the plug-in system ensures that all data is correct, a valid module plus method signature has been specified and the module structure is correct. Modelling and cycling of each thread is managed by the plug-in system as well. This approach to computer programming is based on the *strategy pattern* (see Figure 7-3).

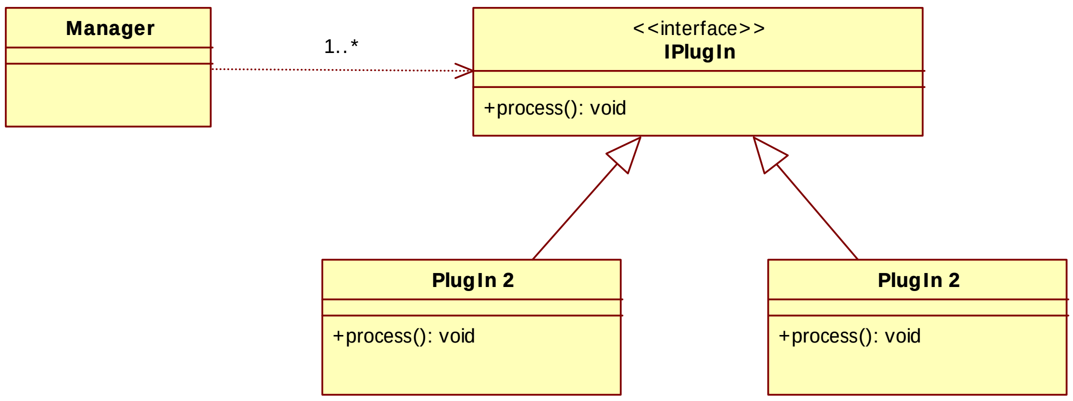


Figure 7-3: Management of plug-ins according to the strategy pattern.

7.2.2.1 *Plug-in configuration*

Plug-in parameters can be set by loading their corresponding XML file which contains different parameter settings. Each parameter set is intended for certain use cases, i.e. different type of genres. Therefore, each module can be adapted individually to the video content. This allows the system to be fed with content related background knowledge which is the main advantage of this work compared to other approaches in the field of broadcast applications.

The internal processing of such information relies on a description of each video by its properties. These properties are not arbitrary but have to be predefined by the plug-in developer. They represent possible properties that can be interpreted by the implemented plug-in. Linking the properties that can be interpreted to the received content enables the plug-in system to define which modules have to be loaded with which parameter settings. In summary, this linking of video content with plug-ins implies the following prior knowledge:

1. The genre type of the video has to be known by receiving descriptive metadata from the production work flow

2. All video content properties that can be extracted by the modules have to be defined by the plug-in developer

3. The video content properties of each genre type have to be defined by either the developer or the application user

Having this information, optimised combinations of modules and their settings can be loaded. The type of genre information is fed in as BMF metadata. If this knowledge is unavailable, the plug-in system falls back to default settings.

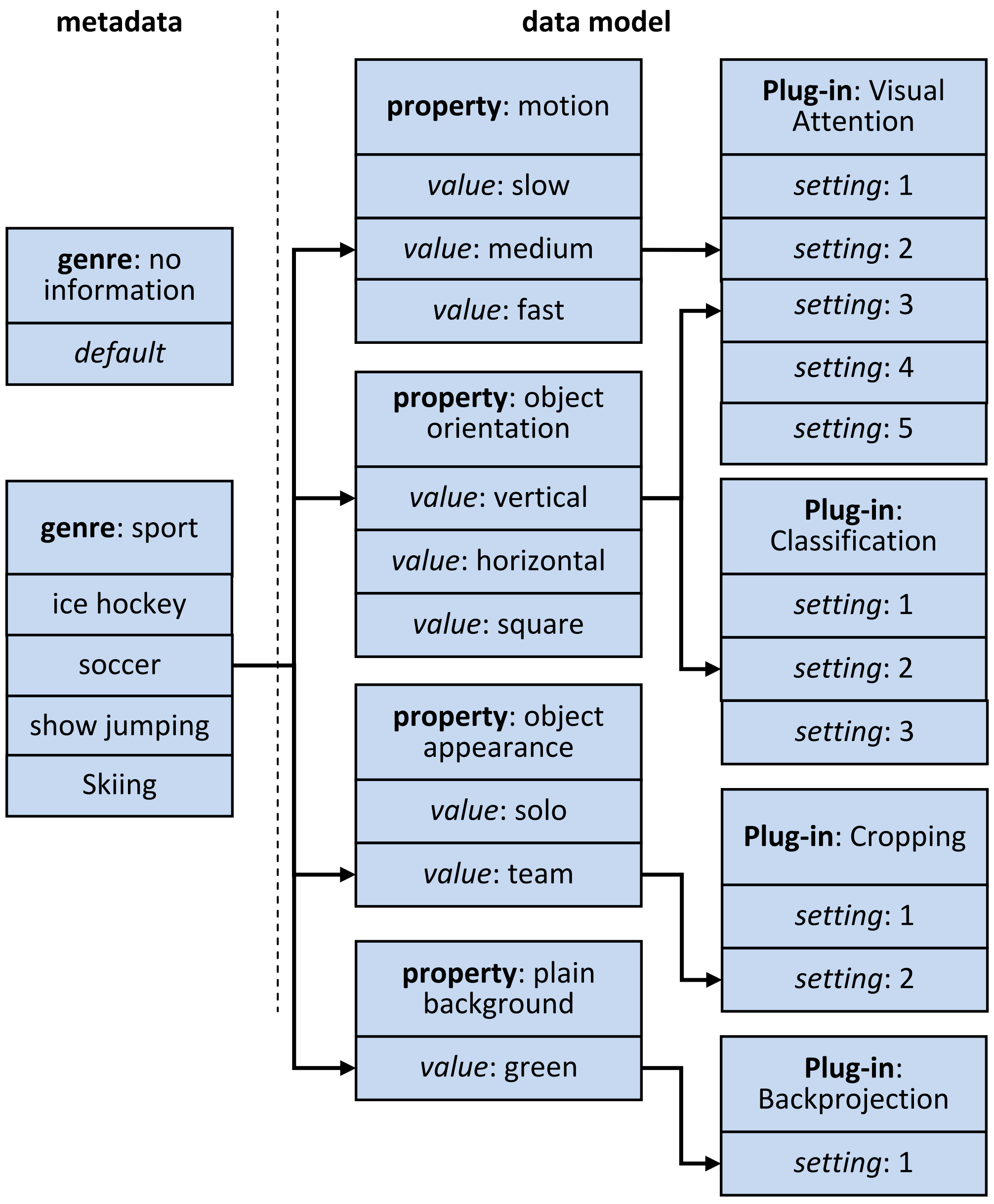


Figure 7-4: Assignment of properties of a specific sport production (left) and extractible properties (middle) for the example soccer. The listed plug-ins (right) are currently implemented. The system is, however, not limited to these plug-ins and can be extended at a later time.

The linkage of extractable properties and the properties present in a video signal is a clear and simple assignment. A property (e.g. motion) has to be further specified by property values (e.g. fast motion). A property value can be assigned to multiple plug-in settings. In turn, a plug-in setting can be assigned multiple property values. Within this work, several properties and property values have been defined.

Figure 7-4 shows an example of assignments between parameter settings and plug-ins. The system is, however, not limited to these properties and can be extended at a later time, e.g. if new plug-ins are available.

7.2.2.2 ***Loading plug-ins***

The *ParameterLoader* class administrates the whole process of parsing metadata information and loading plug-ins. It consists of three further classes (see Figure 7-5). The constructor of the *ParameterLoader* class expects a BMF file. The class functions then return information about shot boundaries, type of genre and plug-in parameters at a certain position in the corresponding video file. This allows modifying settings of the system at any position in a video, for example when the type of genre changes.

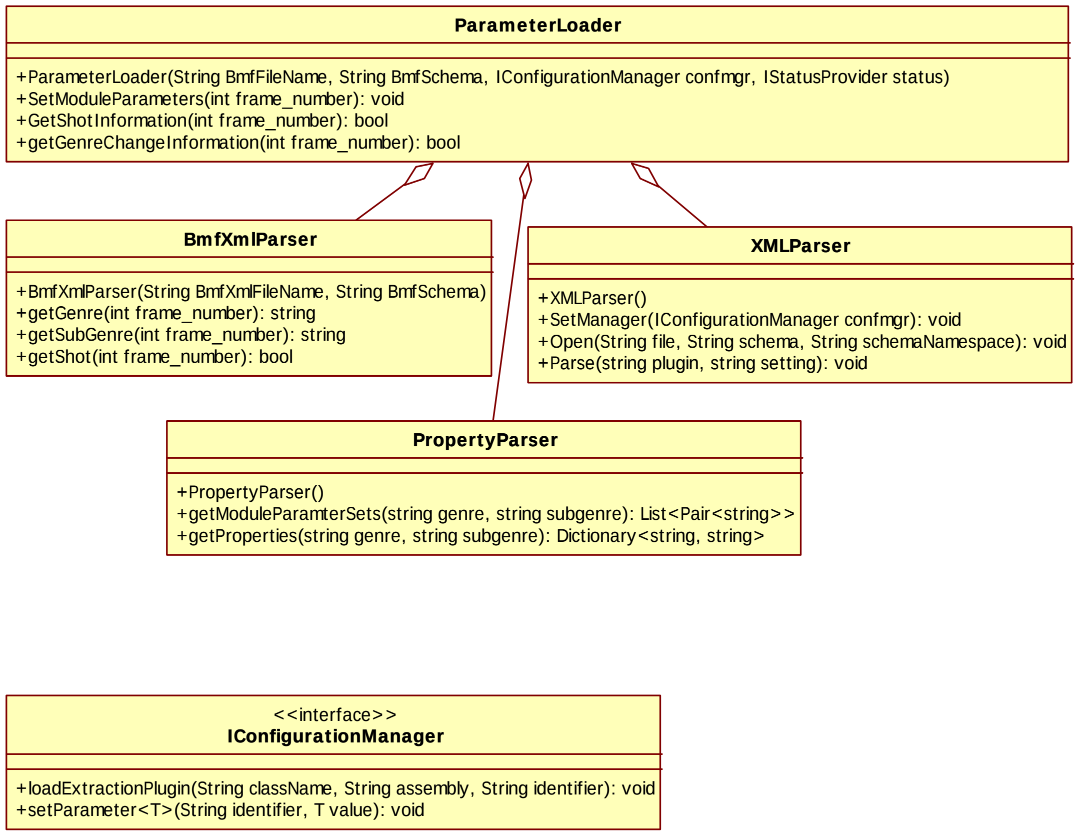

ParameterLoader
+ParameterLoader(String BmfFileName, String BmfSchema, IConfigurationManager confmgr, IStatusProvider status)
+SetModuleParameters(int frame_number): void
+GetShotInformation(int frame_number): bool
+getGenreChangeInformation(int frame_number): bool
BmfXmlParser
+BmfXmlParser(String BmfXmlFileName, String BmfSchema)
+getGenre(int frame_number): string
+getSubGenre(int frame_number): string
+getShot(int frame_number): bool
XMLParser
+XMLParser()
+SetManager(IConfigurationManager confmgr): void
+Open(String file, String schema, String schemaNamespace): void
+Parse(string plugin, string setting): void
PropertyParser
+PropertyParser()
+getModuleParamterSets(string genre, string subgenre): List<Pair<string>>
+getProperties(string genre, string subgenre): Dictionary<string, string>
<<interface>>
IConfigurationManager
+loadExtractionPlugin(String className, String assembly, String identifier): void
+setParameter<T>(String identifier, T value): void

Figure 7-5: Class diagram of *ParameterLoader* and its associated classes *BmfXmlParser, PropertyParser* and *ParameterLoader*.

The first class that is used in *ParameterLoader* is the *BmfXmlParser* class. It provides three functions which return genre, subgenre and shot information at a specific frame number of a parsed BMF file. Shot information is not of importance for loading plug-ins, but will be accessed at a later stage for the definition of cropping areas (see Section 7.3.4).

The second class is the *PropertyParser* which receives data from the *BmfXmlParser* describing the present genre and subgenre. Based on this information, it returns the corresponding plug-in configurations.

The *XMLParser* is the third class in *ParameterLoader*. It receives the required module names and corresponding settings from the *PropertyParser*. The loading of each plug-in and setting of parameters is finally done with aid of a class implementing the interface *IConfigurationManager*.

7.2.3 Core element

The central control element for all configurations and activities within the application is the *CV* class. It functions as the "supervisor" of all program parts and provides a management interface for the whole application. Consequently, the *CV* class represents the logic layer of the software. Here, video and plug-in controlling come together.

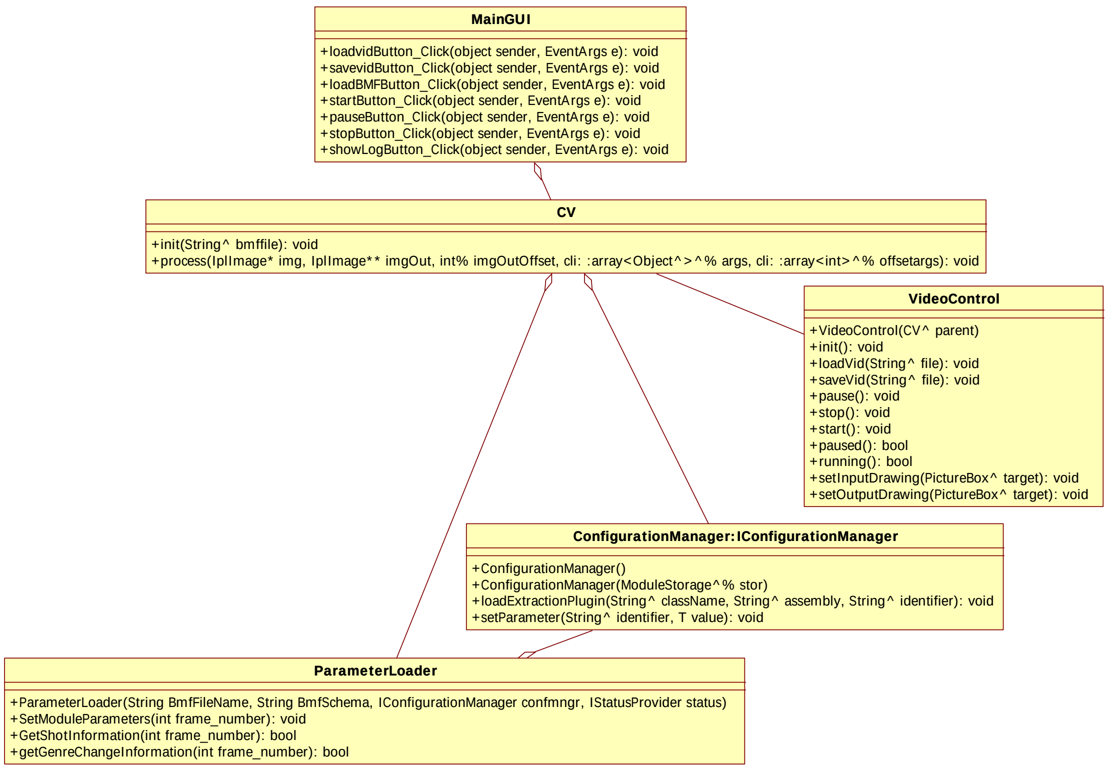
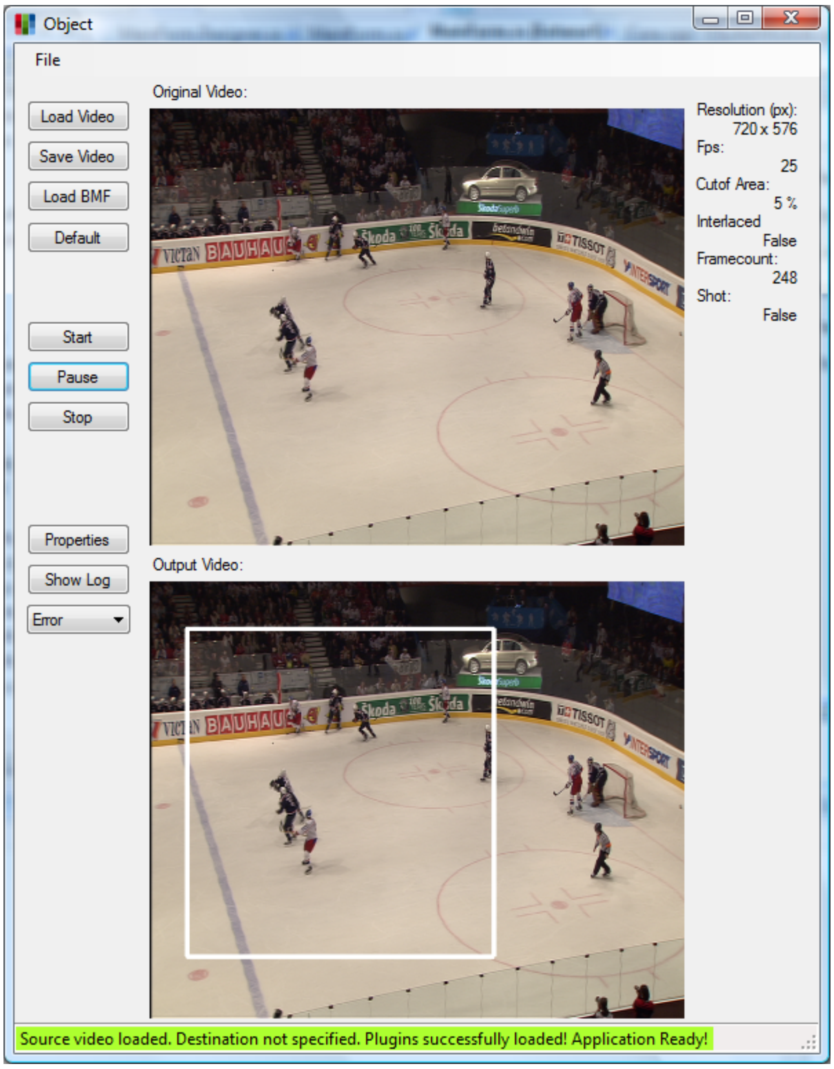

MainGUI
+loadvidButton_Click(object sender, EventArgs e): void
+savevidButton_Click(object sender, EventArgs e): void
+loadBMFButton_Click(object sender, EventArgs e): void
+startButton_Click(object sender, EventArgs e): void
+pauseButton_Click(object sender, EventArgs e): void
+stopButton_Click(object sender, EventArgs e): void
+showLogButton_Click(object sender, EventArgs e): void
CV
+init(String^ bmffile): void
+process(IplImage* img, IplImage** imgOut, int% imgOutOffset, cli: :array<Object^>^% args, cli: :array<int>^% offsetargs): void
VideoControl
+VideoControl(CV^ parent)
+init(): void
+loadVid(String^ file): void
+saveVid(String^ file): void
+pause(): void
+stop(): void
+start(): void
+paused(): bool
+running(): bool
+setInputDrawing(PictureBox^ target): void
+setOutputDrawing(PictureBox^ target): void
ConfigurationManager:IConfigurationManager
+ConfigurationManager()
+ConfigurationManager(ModuleStorage^% stor)
+loadExtractionPlugin(String^ className, String^ assembly, String^ identifier): void
+setParameter(String^ identifier, T value): void
ParameterLoader
+ParameterLoader(String BmfFileName, String BmfSchema, IConfigurationManager confmngr, IStatusProvider status)
+SetModuleParameters(int frame_number): void
+GetShotInformation(int frame_number): bool
+getGenreChangeInformation(int frame_number): bool

Figure 7-6: Core element and its aggregations for loading plug-ins (*ConfigurationManager*), managing of metadata parsing, property parsing and module loading (*ParameterLoader*)

The class *MainGUI* is the graphical user interface (see Figure 7-7) and hence represents the entry point of the application. It instantiates the *CV* class.

The class *VideoControl* is the interface for controlling the video input. In turn, the *VideoControl*-constructor receives a handle on the instantiated *CV*-object. Therefore, both classes form a *bidirectional association*, while the *VideoControl* triggers processes in the *CV* class by new incoming video images.

The interactions between the classes *MainGUI*, *CV*, *VideoControl* and the *ParameterLoader* class mentioned in the previous section are depicted in Figure 7-6. The *VideoControl* class will be explained in more detail in the following section.



Resolution (px):
720 x 576
Fps:
25
Cutof Area:
5 %
Interlaced
False
Framecount:
248
Shot:
False

Source video loaded. Destination not specified. Plugins successfully loaded! Application Ready!

Figure 7-7: The GUI of the application. The left-hand part represents the control panel, the centre is the display area of input- and output-video, and the right hand part represents video information. The lower image displays the computed cropping area.

7.2.4 Video input/output

As already mentioned, the video input and video output are controlled through the *VideoControl* class. This is an aggregation of three additional classes: *VideoOperations*, *VideoIn* and *VideoOut* (see Figure 7-8).

The *VideoOperations* constructor receives the *VideoIn* and *VideoOut* objects that were instantiated in *VideoControl*. *VideoOperations* allows several overloaded functions for drawing on the input video and output video panel of the GUI (see Figure 7-7). Furthermore, *VideoOperations* allows the user to draw rectangles on each panel, i.e. in the video frames, to highlight for example extracted regions of interest or cropping areas. A very important function in *VideoOperations* is the insertion of a delay between video input and video output by indicating a buffer-size in the constructor. This allows the offline processing of video frames. As will be seen later, this functionality is used for buffering and filtering regions of interest for a certain period of time.

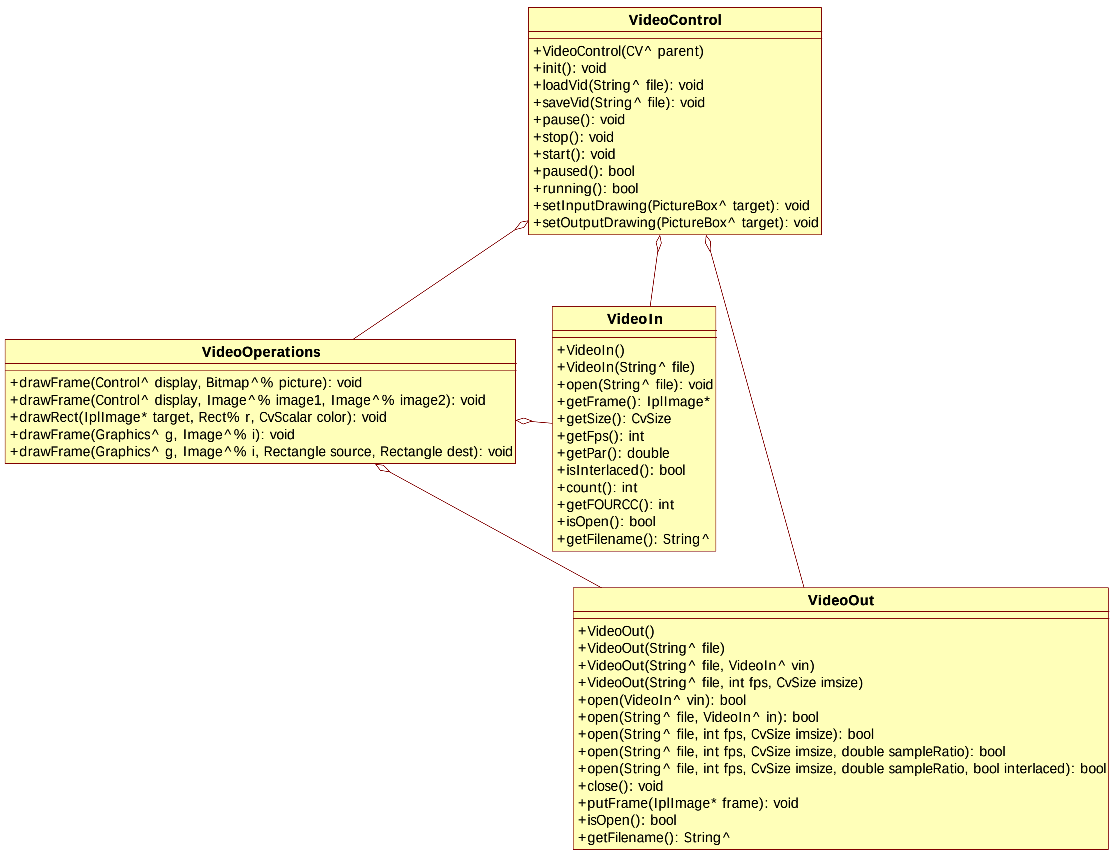

VideoControl
+VideoControl(CV^ parent)
+init(): void
+loadVid(String^ file): void
+saveVid(String^ file): void
+pause(): void
+stop(): void
+start(): void
+paused(): bool
+running(): bool
+setInputDrawing(PictureBox^ target): void
+setOutputDrawing(PictureBox^ target): void
VideoOperations
+drawFrame(Control^ display, Bitmap^% picture): void
+drawFrame(Control^ display, Image^% image1, Image^% image2): void
+drawRect(IplImage* target, Rect% r, CvScalar color): void
+drawFrame(Graphics^ g, Image^% i): void
+drawFrame(Graphics^ g, Image^% i, Rectangle source, Rectangle dest): void
VideoIn
+VideoIn()
+VideoIn(String^ file)
+open(String^ file): void
+getFrame(): IplImage*
+getSize(): CvSize
+getFps(): int
+getPar(): double
+isInterlaced(): bool
+count(): int
+getFOURCC(): int
+isOpen(): bool
+getFilename(): String^
VideoOut
+VideoOut()
+VideoOut(String^ file)
+VideoOut(String^ file, VideoIn^ vin)
+VideoOut(String^ file, int fps, CvSize imsize)
+open(VideoIn^ vin): bool
+open(String^ file, VideoIn^ in): bool
+open(String^ file, int fps, CvSize imsize): bool
+open(String^ file, int fps, CvSize imsize, double sampleRatio): bool
+open(String^ file, int fps, CvSize imsize, double sampleRatio, bool interlaced): bool
+close(): void
+putFrame(IplImage* frame): void
+isOpen(): bool
+getFilename(): String^

Figure 7-8: VideoControl class and its aggregations. This class provides a control interface of incoming and outgoing video data.

The classes *VideoIn* and *VideoOut* form the direct connection to the ffmpeg library (see Section 7.1.3). *VideoIn* allows the decoding of a wide range of video files and provides the most important information about the loaded video file, such as frames per second, pixel aspect ratio and image size. *VideoOut* supports uncompressed video formats in an AVI container as well as encoding according to the MPEG-2 video standard. All internal image processing is done with the *IplImage* format of OpenCV (see Section 7.1.2).

7.2.5 Logging

To ensure the tracking of all processes, a logging system has been implemented which facilitates the collection of information that can be viewed at a later time. Additionally, it allows the user to specify a level of importance for logging. The logging should be possible at any position in the application. For this, an approach based on the *singleton pattern* is chosen, which allows the usage of one single object across the whole system.

7.3 Plug-ins

In the following sections, each plug-in which has been implemented is introduced. Each section describing a plug-in is divided into a first part which treats the implemented algorithms and a second part which describes the class structure and parameter settings.

As already mentioned in 7.2, plug-ins are not an inherent part of the application. They are loaded at run-time. For each execution of the application, the Cropping Plug-In (see Section 7.3.4) and the Classification Plug-In (see Section 7.3.3) are required, independent of the analysed content. All other plug-ins, referred to here as extraction plug-ins, receive an input image – either grey or colour. They all run in parallel and return a list of ROIs which are weighted by the Classification Plug-In (see Figure 7-10). Currently, there exist two extraction plug-ins, the Visual Attention Plug-In and the Backprojection Plug-In, which will be explained in Sections 7.3.1 and 7.3.2.

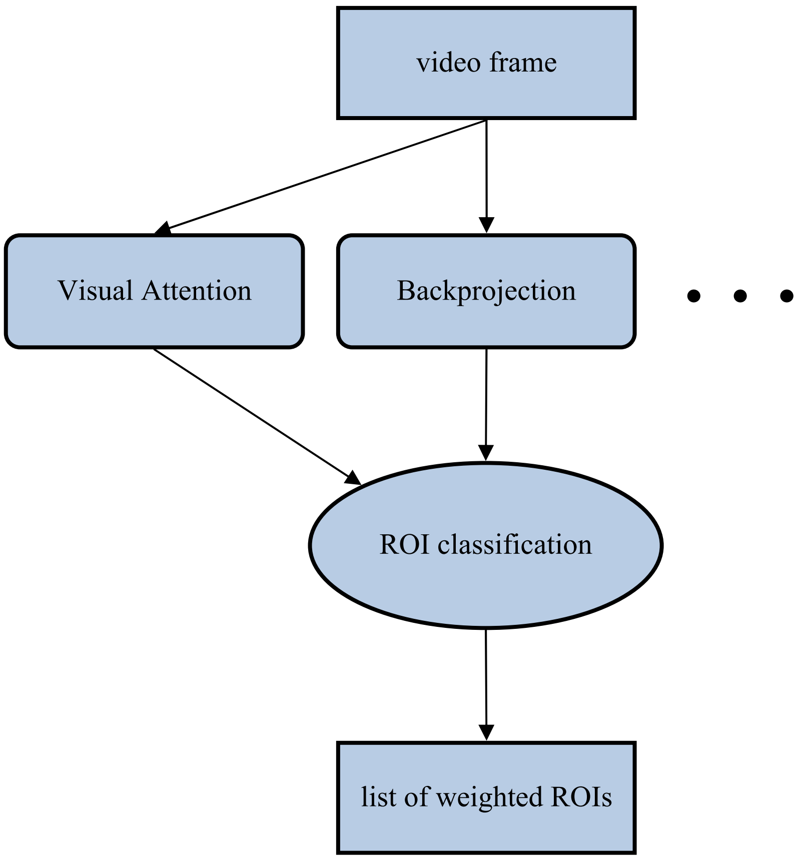


Figure 7-9: Flow chart for analysing video content by extraction plug-ins. Extraction plug-ins can be used in any combination and as required. Returned ROIs are collected and weighted by the Classification Plug-In. Finally, a list of weighted ROIs is returned for further processing.

ROIs are described by their x- and y-position (centre position), their width and height, a corresponding frame number and a weight value which specifies the contextual importance of a ROI (see Figure 7-10).

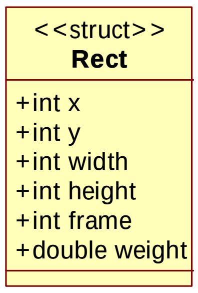


Figure 7-10: Structure of a ROI including its x- and y-centre position, width and height, frame number as well as its weight according to its computed contextual importance.

7.3.1 Visual attention plug-in

The Visual Attention Plug-In consists of three sub-modules: the Still Image Saliency Module, the Motion Saliency Module and the Segmentation Module. The Visual Attention Plug-In is a general approach combining saliencies of still as well as moving pictures. Despite its generality, due to different possible parameter settings of each sub-module the Visual Attention Plug-In can still be modified and hence adapted to content properties.

Firstly, the Still Image Saliency Module is presented. Afterwards, the Motion Saliency Module is discussed. Both modules analyse the video content independently and can run in parallel. Results of both modules are fused into one final weighted saliency map which is explained in Chapter 7.3.1.3. Finally, the Segmentation Module is explained, which combines related salient areas by bounding rectangles and represents those in a list of ROIs. This whole process is depicted in Figure 7-11.

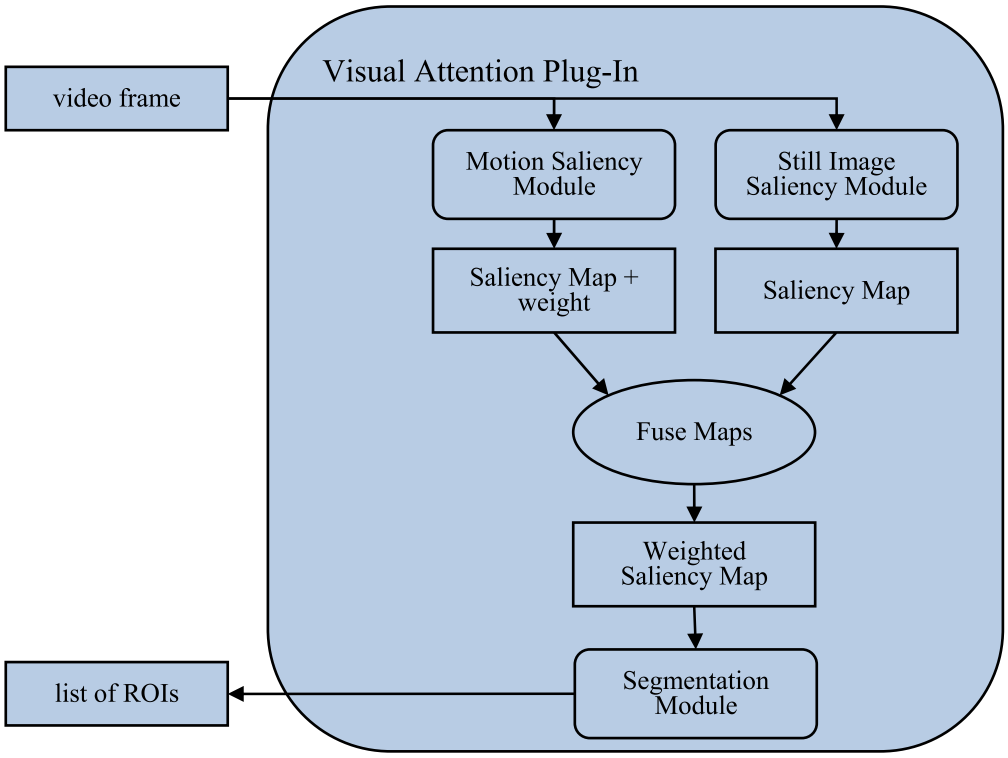


Figure 7-11: Flow chart of the Visual Attention Plug-In. An incoming grey image is independently analysed for motion and still image saliency. Information from both extractions is combined into a single weighted saliency map and ROIs are defined by a segmentation process applied on the saliency map.

7.3.1.1 ***Still image saliency module***

In Section 3.5, two visual attention systems have been introduced. The attention system by Itti is motivated by the human perception to detect features in a visual scene. In broadcast productions, images contain not just these features, but also contextual information in the form of design elements. Some image compositions have been introduced in Chapter 4. They are used by cameramen to attract a viewer's attention. Therefore, top-down information is already available in video broadcast productions. To find an attention system that best meets the requirements of this application, the one by Itti (see Section 3.5.1) has been compared to the one by Hou (see Section 3.5.2). Both are available as a MATLAB implementation, where the former is available including a GUI at (Bernhardt-Walther, 2010) and the latter simply consists of five lines of code, available at (Hou, Spectral Residual, 2009). Typical still images of sports productions were used to investigate their suitability for this application.

In sport applications, an attention model has to deal with dazzling colours, which may not be of contextual importance, e.g. advertisement banners or colour markings. Comparing the two attention models, Itti's model is more sensitive to dazzling colours because it uses a colour map. Spectral residual relies on grey images only and hence is less susceptible to this colour information (see Figure 7-12).

Especially for individual sports, objects are mostly focused by the cameraman, and in case of fast movement the difference between foreground and background can be recognized due to motion blur. Whereas Spectral Residual responds strongly to these image properties, the attention model by Itti does not consider such depth information.

Based on the outcomes of the evaluations, the Spectral Residual approach has been selected. It best meets the requirements of the system and in addition has the highest computational efficiency. It should be mentioned that this is not a general assessment, but one that is specific to the demands of this particular application.

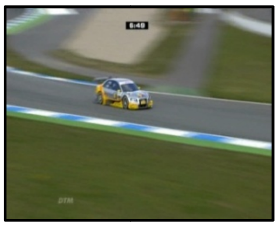
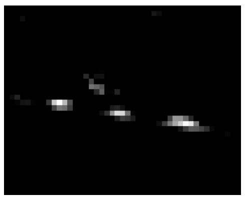
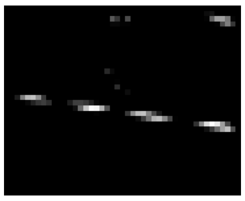
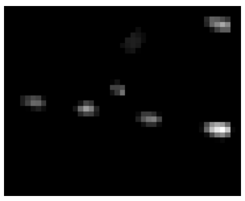
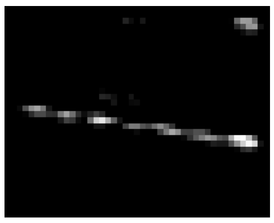
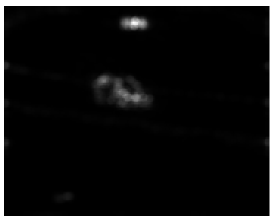
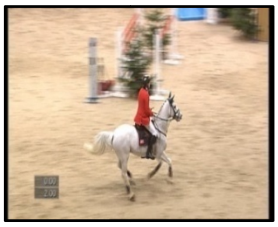
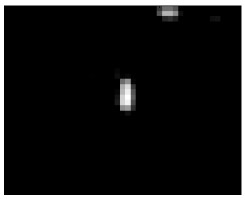
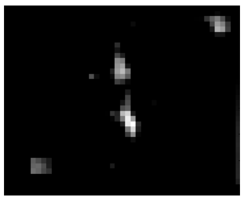
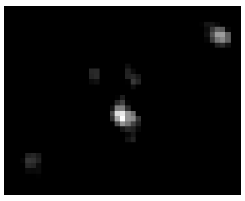
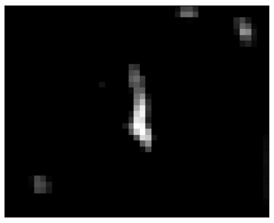
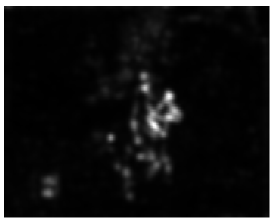
Input Image	Itti		Hou
	Conspicuity Maps	Saliency Map	
	Colour Map  Intensity Map  Orientation Map 		
	Colour Map  Intensity Map  Orientation Map 		

Figure 7-12: Excerpt of typical sports examples taken from a test carried out to compare Itti's and Hou's visual attention models.

Implementation of Spectral Residual

With the aid of the OpenCV library, the Spectral Residual approach has been ported from MATLAB to C++. According to Chapter 3.5.2, the incoming grey image is first scaled down to remove image details and to support the saliency of related areas. Here, a scale factor of 0.25 is used by default. Afterwards, a FFT is applied and the real- and imaginary parts are converted to magnitude and phase. After taking the logarithm of the magnitude image, the image is doubled and either of these is averaged by a 3x3 average filter. To compute the Spectral Residual, the averaged image is subtracted from the non-averaged image. This step splits redundancy and deviations from the norm (cf. 3.5.2) according to Equation (3.20). Then the image is re-transformed by an IFFT. The whole process is depicted in Figure 7-13.

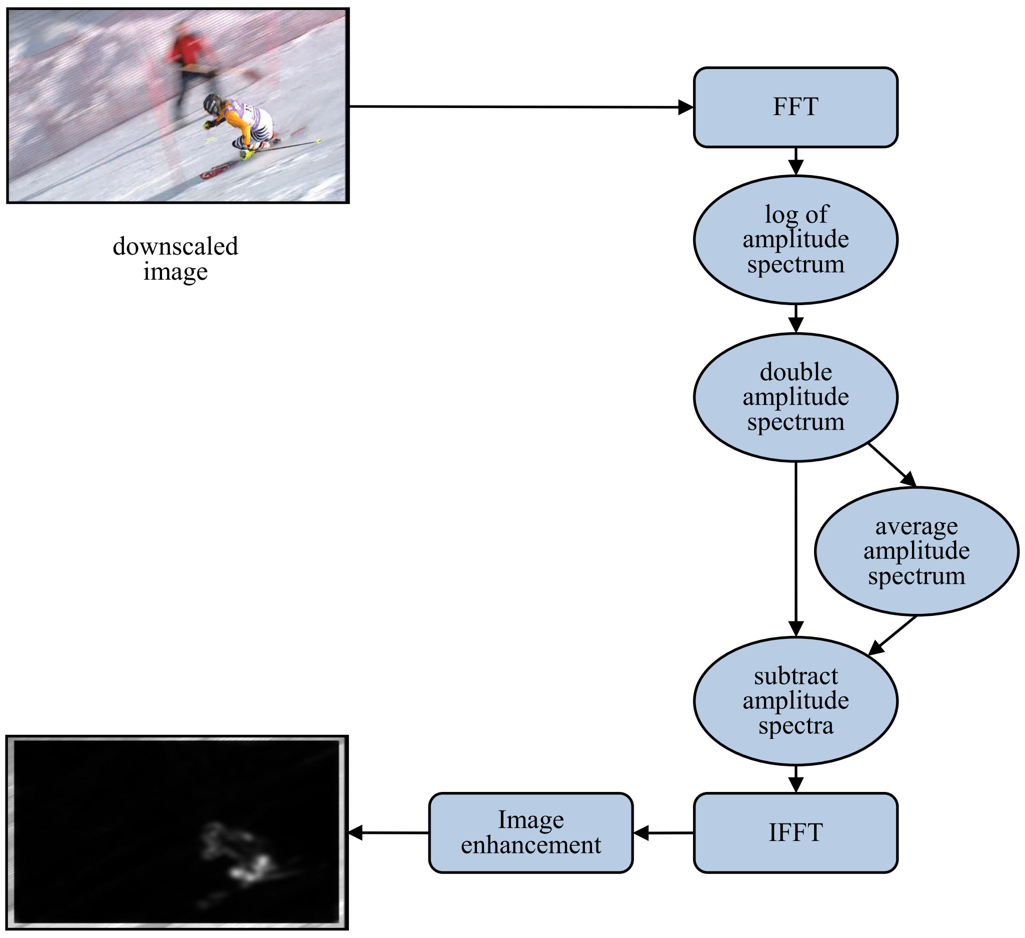

downscaled image
FFT
log of amplitude spectrum
double amplitude spectrum
average amplitude spectrum
subtract amplitude spectra
IFFT
Image enhancement

Figure 7-13: Flow chart of the Still Image Saliency Residual implementation (Spectral Residual).

The last processing step in Figure 7-13 is for image enhancement. This implies a gamma correction of 2.0 and a 3x3 Gaussian filter. Afterwards, a normalisation of the lowest and highest intensity value to the full range of 0-255 is applied.

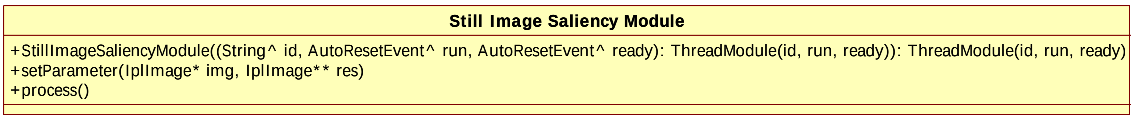
Parameter Settings and Class Structure

The Still Image Saliency Module allows the adaptation of three parameters. They influence the scaling of the input image and the kernel size of the average filter and of the Gaussian filter. As tests have shown, one setting delivers results that fit best under most conditions. These configuration parameters are defined as the default setting (see Table 7-1).

Table 7-1: Settings for the Still Image Saliency Module.

	Setting: default
scaling factor	0.25
kernel size of average filter	3x3
kernel size of Gaussian filter	3x3

The Still Image Saliency Module receives an 8 bit grey image and instantiates a 32 bit grey image that contains the saliency map. Figure 7-14 depicts the class structure of the module.



```
Still Image Saliency Module
+StillImageSaliencyModule((String^ id, AutoResetEvent^ run, AutoResetEvent^ ready): ThreadModule(id, run, ready)): ThreadModule(id, run, ready)
+setParameter(IplImage* img, IplImage** res)
+process()
```

Figure 7-14: Class structure of the Still Image Saliency Module.

7.3.1.2 *Motion saliency module*

Obviously, most sport productions contain camera motion and object motion. For example, individual sport is often shot by keeping the object of interest focused. In that case, the most interesting part of the image is the object that is kept at a rather static point and does not move much at all.

As long as objects are small compared to the whole video image it is assumed that background motion corresponds to camera motion. Knowing the camera motion, objects are detected in the Motion Saliency Module by subtracting consecutive video frames, where one of the two frames is warped by the computed camera motion. As a result, two images shot at different times get coincident backgrounds.

The motion vector field is computed by the OpenCV optical flow implementation of Lucas & Kanade (cf. Chapter 2.5.1.3). This method has been chosen for two main reasons. Firstly, the gradient based approach delivers fewer but much more reliable vectors than block matching (cf. 2.5.1.1 and 2.5.1.3). Hence, subsequent processing operations require less computing and filtering effort. Secondly, the Lucas & Kanade (Lucas & Kanade, 1981) approach directly delivers vectors that are described in the space-time domain in a very efficient way. Even if phase correlation might be the more robust motion detection, it does not deliver motion direction information (see 2.5.1.2). Because of that, computational effort can become quite high by combining phase correlation with additional techniques for assigning a motion offset to a direction.

Due to 25 fps/interlace in TV productions, the frame rate is comparably high. Therefore, global motion between two frames is relatively small. Assuming that radial distortions of lenses are small as well, the computation of 2D affine homography is absolutely adequate. To remove outliers from the data set, the Least Median of Squares approach (LMedS) has been chosen (see Chapter 2.5.2.1). Compared to RANSAC, LMedS does not need any threshold setting. As it can be assumed that generally the ratio of objects to background is less than 1:1, LMedS is absolutely sufficient. The decision was also made against M-estimators, because LMedS does not weight outliers by an appropriate influence function as M-estimators does, but completely removes them from the data set.

For the computation of the best fit affine mapping, the robust, non-linear homography estimation library *homest* has been chosen. It has been implemented in C/C++ by Manolis Lourakis (Lourakis, 2009). The approach of this implementation is according to the global motion estimation described in Chapter 2.5.2. It first selects a set of three random candidates and calculates the related residual of each affine model.

To select candidates that contain global information, a technique is applied that avoids sampling vectors lying close to each other, because a global affine transformation computed from closely located vectors is highly unstable. This method is described in (Zhang, Deriche, Faugeras, & Luong, 1995) and is based on a bucketing technique. For this, the minimal and maximal coordinates of matched points in an image are computed. The area that is spanned by these values is divided into $b \times b$ buckets (see Figure 7-15). To each bucket, a set of matched points is attached. Buckets without matches are removed. Subsamples are now randomly selected from three different buckets.

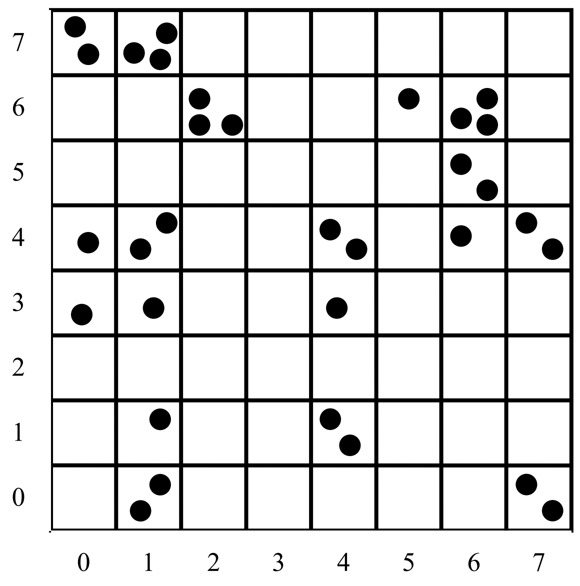


Figure 7-15: Division of matched coordinates in an image into 8×8 buckets.

In summary, the algorithm implemented in *homest* can be described as follows:

1. Randomly choose three corresponding points x_i and x_i' from the computed motion vector field according to the bucketing technique.
2. Compute the median value of squared residuals r_i for each affine model.
3. Choose the model with minimal error and remove motion vectors whose residual is above the median of squared residuals.
4. Refine the model parameters H by applying least-squares on defined inliers.

After computing H for two consecutive video frames, H is applied on the first image. In the next step, both images are subtracted from each other to blank coincident background and brighten up objects whose movement behaviours differ from camera motion (see Figure 7-16).

The last processing step of the Motion Saliency Module includes the same image enhancement as for the Still Image Saliency Module. A gamma correction of $\gamma = 2.0$ first tries to separate noise from important information. Afterwards, a Gaussian filter is applied with a kernel size of 9x9. This comparably large kernel size has been chosen due to the fact that rather than the whole object, the offset of a moving object causes bright areas in a difference image. Therefore, the filtering is used to blur these areas and bring separated blobs closer together. Finally, a normalisation of the maximum and minimum intensity values to the full intensity range is applied. The flow chart of the Motion Saliency Module is depicted in Figure 7-16.

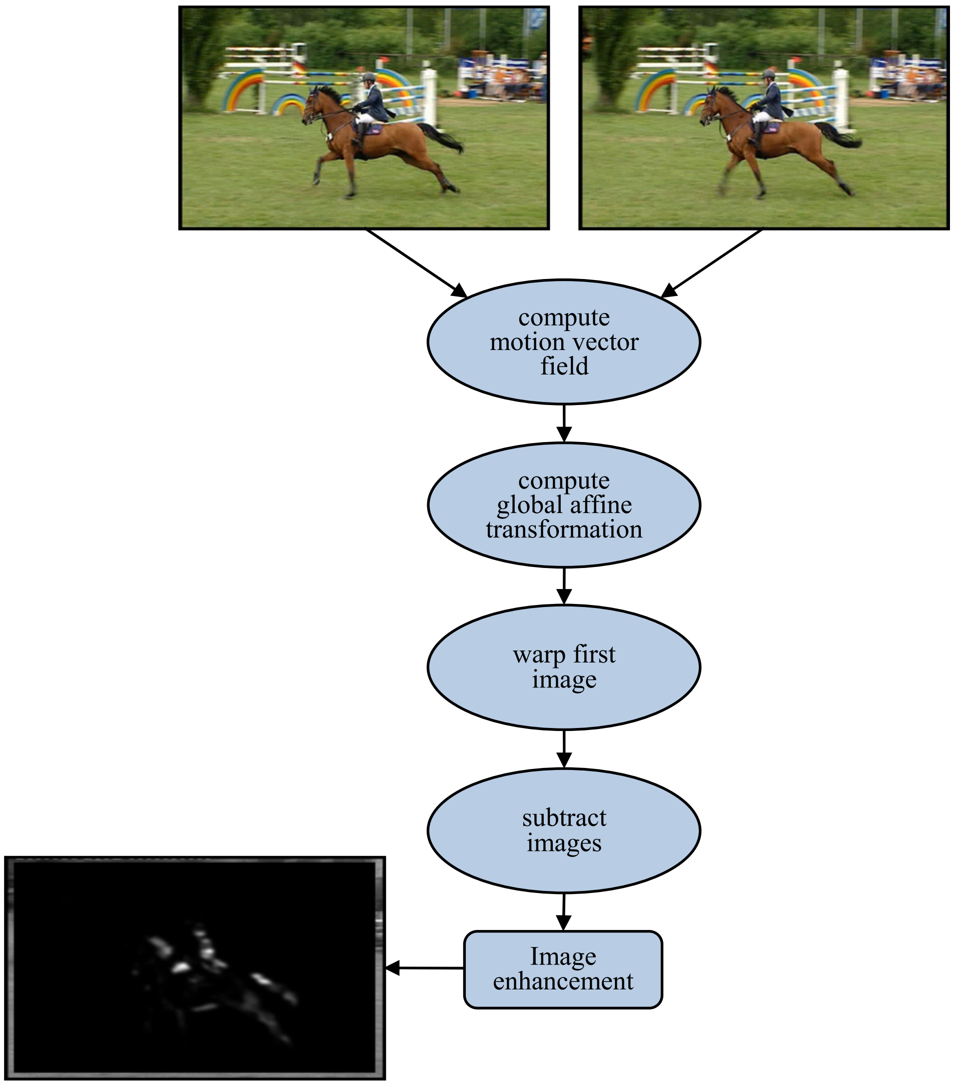

compute motion vector field
compute global affine transformation
warp first image
subtract images
Image enhancement

Figure 7-16: Flow chart for the computation of difference image by compensating global motion.

So far, it is assumed that outliers are removed from the computation of the affine homography. Despite this, there is still no information about the accuracy of H, which in turn gives information about the quality of the difference image. This information can be given by the root median squared error ($RMedSE$) of the Euclidean error between the first and second image:

$$RMedSe = \sqrt{\mathrm{med}[d(x_i{}', Hx_i)^2]} = \mathrm{med}[d(x_i{}', Hx_i)] \tag{7.1}$$

where $d(x, y)$ describes the Euclidean distance between x and y and med is the median value of the term computed in square brackets for the entire data set. It has to be mentioned that Equation (7.1) can be used here because an affine transformation between both images plays a symmetric role, i.e. $d(x_i', Hx_i) = d(x_i, H^{-1}x_i')$. In case of fitting a complete planar homography, H is no longer symmetric and more complex error functions have to be considered. For further information on alternative error functions, the author refers to (Hartley & Zisserman, 2008).

As the calculation of optical flow by Lucas & Kanade (Lucas & Kanade, 1981) requires uniquely identified points, the number of motion vectors can vary dependent on the strength of found corners. This means that the number of good features to track might decrease if corners are not sufficiently pronounced in an image. To limit the total number of found points, a predefined number of strongest corners can be specified. In this implementation, the number of features can be limited by the ratio between image resolution and a variable denominator:

$$T_{features} = \frac{image\ width\ \times image\ height}{n} \tag{7.2}$$

where $T_{features}$ and n are natural numbers with $1 < n < image\ width \times image\ height$. n is an adaptable parameter which is set to $n = 130$ by default. A decreasing number of motion vectors decreases the statistical significance for global motion estimation. For example, if $RMedSE$ is sufficiently small, the global motion estimation can still be poor due to a much too small total number of found features. Combining the number of found features with the $RMedSE$ provides a good possibility of evaluating the accuracy of the expected global affine transformation. This accuracy is expressed by a weighting factor w, with $w \in$ [0 ... 1], which is computed as follows:

$$w = \left(\frac{f}{T_{features}}\right)^4 \cdot \left(1 - \frac{RMedSe}{T_{RMedSe}}\right)^2 \tag{7.3}$$

where f is the number of found features and T_{RmedSe} is a threshold that limits the maximum allowed $RMedSE$. This equation, as well as T_{RmedSe}, has been heuristically evaluated, where $T_{RmedSe} = 1.0\ px$ by default. The weighting factor is returned by the Motion Saliency Module together with the motion map. As will be seen in the next section, w is used to weight results from the Still Image Saliency Module and from the Motion Saliency Module, before they are combined.

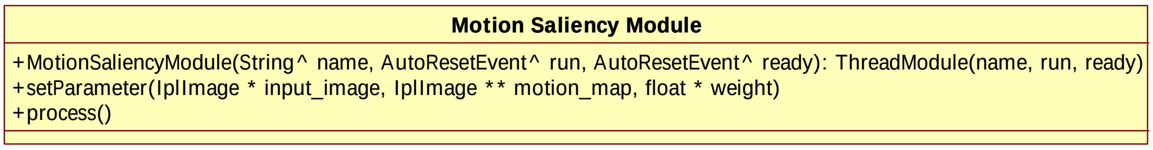
Parameter settings for Motion Saliency Module

The Motion Saliency Module defines eight adaptable parameters that are assigned to specified content properties (see Table 7-2). The "scale factor" specifies the image scale factor for the motion estimation. Half of image size (scale factor = 0.5) provides a good compromise between accuracy and computational effort. Therefore, this value has been chosen for all property values. "Maximal pyramid" level allows a search for corresponding points not only at one image level, but over several scales. A search on multiple pyramid levels provides an improvement for detecting greater pixel offsets. Because of that, more pyramid levels are used with increasing motion. The "optical flow window" parameter defines the number of neighboured pixels that are considered for optical flow computation on each pyramid level. As described in Section 2.5.1, the consideration of a pixel's neighbourhood is necessary to reduce the aperture problem. For all motion types, a neighbourhood of 3x3 is absolutely adequate. "Epsilon" specifies a required accuracy for the iterative optical flow computation. This value has been chosen sufficiently small for a robust estimation. "Maximal iterations" defines the maximal number of iterations to compute the optical flow criteria. "Inlier percentage" specifies the percentage of inliers for the Least Median of Squares computation (cf. Section 2.5.2). The parameters in the last two rows of Table 7-2 allow the adaptation of the weighting factor w according to Equation (7.3).

Table 7-2: Settings for the Motion Saliency Module.

	Setting 1	**Setting 2**	**Setting 3**
	Motion: slow	Motion: medium	Motion: fast
scale factor	0.5	0.5	0.5
maximal pyramid level	0	3	5
optical flow window	3x3	3x3	3x3
epsilon	0.01	0.01	0.01
maximal iterations	5	5	5
inlier percentage	0.5	0.5	0.5
denominator (n) to compute $T_{features}$	130	130	130
$RMedSe$ threshold (T_{RMedSe})	1.0	1.0	1.0

The Motion Saliency Module receives an 8 bit grey image and instantiates a 32 bit grey image that contains the motion saliency map. The pointer *weight* allows access to the calculated weighting value w according to Equation (7.3). Figure 7-17 depicts the class structure of the module.



```
Motion Saliency Module
+MotionSaliencyModule(String^ name, AutoResetEvent^ run, AutoResetEvent^ ready): ThreadModule(name, run, ready)
+setParameter(IplImage * input_image, IplImage ** motion_map, float * weight)
+process()
```

Figure 7-17: Class structure of the Motion Saliency Module.

7.3.1.3 ***Map fusion***

The fusion of the still image saliency map and the motion saliency map is done on the level of the Visual Attention Plug-In, which receives the results of both sub-modules. Additionally, the already mentioned weighting value w which is returned by the Motion Saliency Module is used for combining both maps.

For summing up both maps, the motion saliency map M is multiplied by w and the still image saliency map S is multiplied by $1 - w$, where w is calculated according to Equation (7.3):

$$R = \frac{1}{2}(w \cdot M + (1 - w) \cdot S) \tag{7.4}$$

where R is the resulting saliency map. Figure 7-18 shows an example for summing up the maps weighted by w.

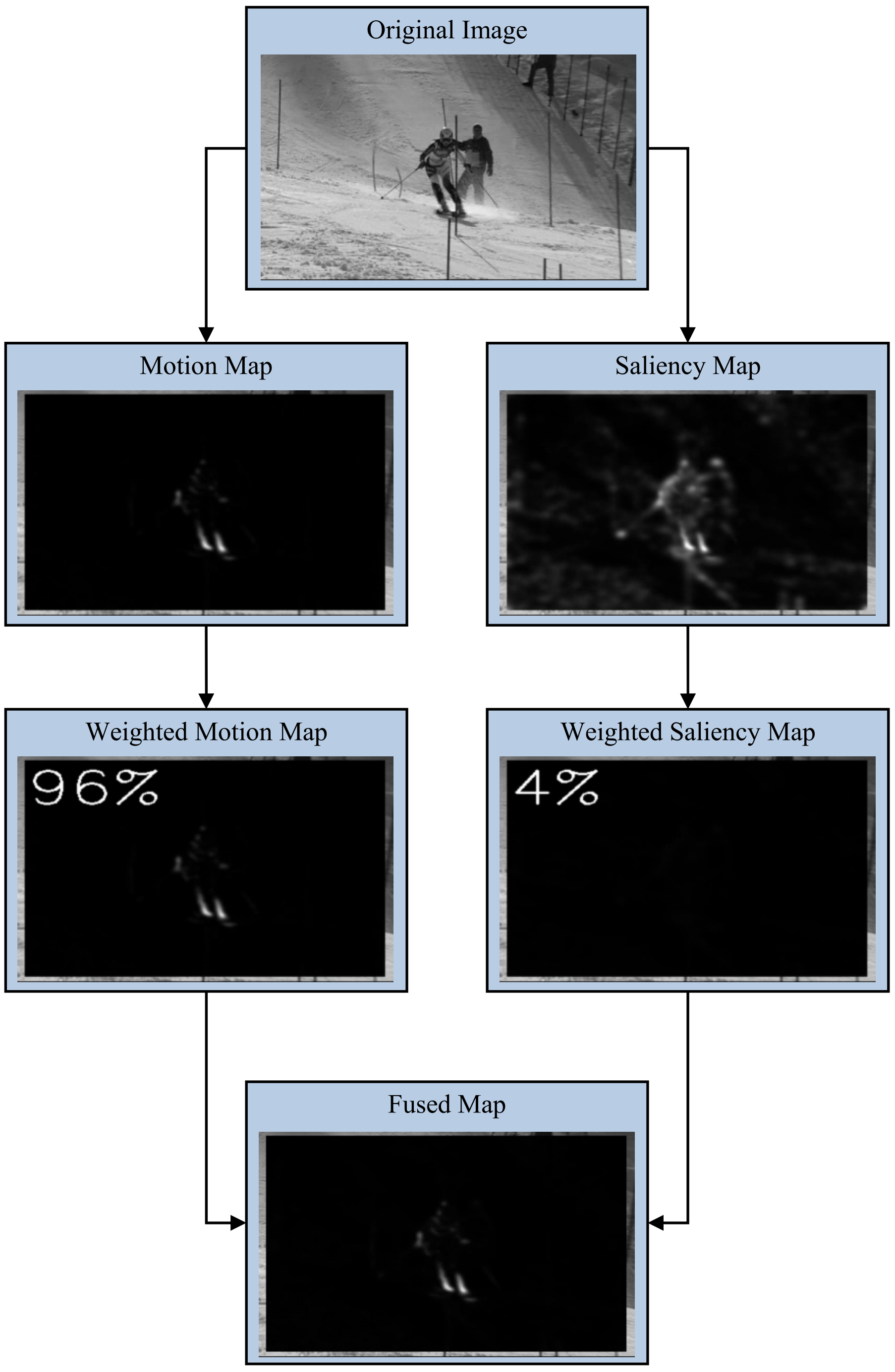


Figure 7-18: Example of combining motion saliency and still image saliency by weighting factor *w*.

7.3.1.4 *Segmentation module*

Segmentation is a crucial task in image processing. In this implementation, segmentation is used to make a decision regarding which areas in the saliency map correspond to foreground and which areas correspond to background. Furthermore, the segmentation process defines foreground areas by bounding rectangles that enclose related regions. From there on, ROIs are simply specified by their corner coordinates. This process of getting from image level to a more abstract level is described in the following.

Before the final implementation is explained, an alternative segmentation process that has been used is first briefly introduced. This approach makes use of the region labelling method *flood fill* (see Chapter 2.6.2). This operation starts at a specific point (*seed point*) and adds neighboured pixels as long as the decrease of intensity is below a predefined threshold. The seed point represents the pixel with the highest intensity in the image. As soon as the first region has been labelled, this region is suppressed and a new seed point is searched.

This approach has been rejected for several reasons. Firstly, it is assumed that the most salient pixels are those with the highest intensity. This is correct by definition, because saliency mainly arises from local contrasts in the input image. This means that, for example, on a green background, soccer players with red shirts are less salient than soccer players with white shirts. Searching for contextually important objects instead of salient regions, this approach delivers non-satisfying results, because less salient regions, i.e. the players with red shirts, might get lost due to thresholding. Additionally, this method can be computationally quite costly, because it iteratively labels regions by suppressing a found region and starting a new search. Another disadvantage is that flood fill requires some threshold settings which might end up giving greatly varying results. Even if this approach is more appropriate for saliency detection, it does not consider contextual importance. For this reason, an alternative approach has been chosen.

As already mentioned, saliency is not equivalent to contextual importance. Therefore, a more robust segmentation method has been chosen that first separates foreground and background by local binarisation (see Chapter 2.6.1). The advantage is that a hard decision is made on local conditions instead of globally set thresholds like maximal number of labels. Afterwards, an average filter with adaptable kernel size is applied to bring together separated regions which correspond to the same object. This approach has been preferred over morphological operations because morphological operations can change the shape of an area; especially in the case of several iterations this effect is not negligible. As this might result in pumping ROIs over time, a simple blurring has been chosen to keep the shapes of regions more regular. Additionally, the shape and size of the filter kernel strongly supports specific object shapes. For example, vertically oriented objects can be supported by vertically oriented filter kernels, whereas horizontally oriented objects are weakened.

In the next step, contours are detected and combined by edge linking as described in Chapter 2.6.3. This rather simple approach is much faster than flood fill – especially for an increasing number of salient areas – and it retains as much information as possible from the saliency map. Finally, bounding rectangles are computed for linked edges and their coordinates are represented in the ROI structure according to Figure 7-10. The whole segmentation process is depicted in Figure 7-19.

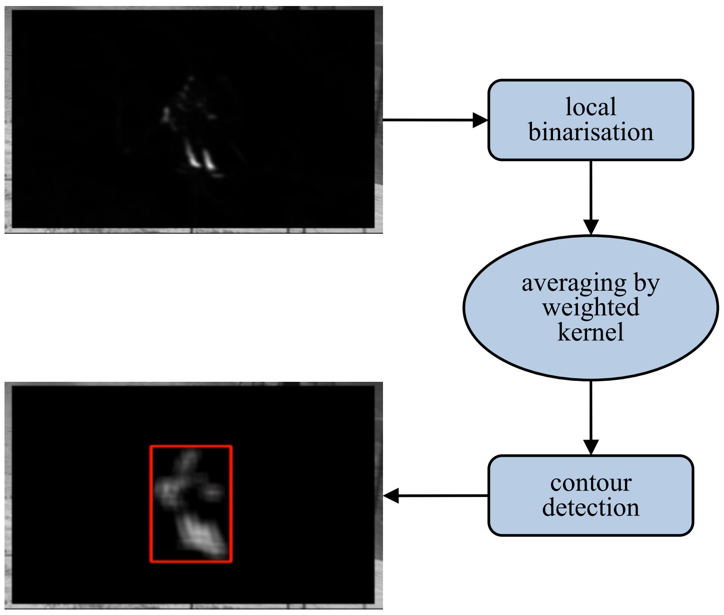


Figure 7-19: Flow chart of segmentation module for the example of Figure 7-18.

Parameter Settings and Class Structure

Parameters that are modifiable for the segmentation module influence only the kernel size and aspect ratio of the blur filter. As previously mentioned, this filter intends to merge parts that belong to the same object. To support specific shapes of objects that fit into a searched pattern, e.g. vertically oriented objects for soccer, different kernel shapes and sizes can be defined by the parameter settings of the segmentation module. Additionally, the size of each blur mask can be influenced by the “mask size multiplier”. This value allows the adaptation of the kernel size, dependent on the number of objects which are present. Here, it is assumed that content with multiple objects is shot in such a way that objects have a smaller size compared to content containing a single object. The Boolean value “binarise” allows switching off of the binarisation step in the Segmentation Module. As this is not necessary for the Visual Attention Plug-In, some other extraction modules might deliver an already binarised image. In this case, an additional binarisation can cause undesired results for segmentation.

Table 7-3: Settings for the Segmentation Module.

	Default	Setting 1	Setting 2	Setting 3	Setting 4	Setting 5
		Object appearance: solo	Object appearance: team	Object orientation: vertical	Object orientation: horizontal	Object orientation: default
blur mask width	7	-	-	3	3	7
blur mask height	7	-	-	9	9	7
mask size multiplier	1	2	1	-	-	-
binarise	true	true	true	true	true	true

The Segmentation Module receives an 8 bit grey image and returns a list of rectangles that were computed by the segmentation process. Figure 7-20 depicts the class structure of the module.

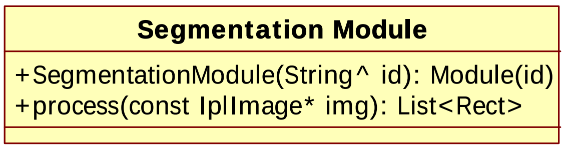

Segmentation Module
+SegmentationModule(String^ id): Module(id)
+process(const IplImage* img): List<Rect>

Figure 7-20: Class structure of the Segmentation Module.

7.3.2 Backprojection plug-in

The Backprojection Plug-In is a simple approach which allows the detection of plain backgrounds in a video image as well as objects which are on this area. This plug-in is intended to be used for sports content which takes place on more or less plain pitches. Making use of this plug-in in addition to the Visual Attention Plug-In allows a more reliable statement about possible objects of interest. As optimally two ROIs are estimated for each object (one by each extraction plug-in), it is desirable to reduce the number of ROIs to a representative single ROI in a last processing step. In this way, ambiguous statements for further processing can be avoided. This reduction is done by a clustering method (intra frame clustering), described in Section 7.3.4.2.

The backprojection method (histogram backprojection) has been proposed by Swain and Ballard in (Swain & Ballard, 1990). Originally, histogram backprojection was used to locate colours in an image which belong to a known object that has to be found. For this, the histogram M of a sample image which contains the desired colour pattern is computed. Additionally, the histogram I of the image to be analysed is determined. For each bin j of both histograms, a third histogram R is computed which is the ratio of M divided by I:

$$R_j = \frac{M_j}{I_j} \tag{7.5}$$

The histogram backprojection is finally estimated by mapping each three-dimensional colour value $c(x, y)$ to a histogram bin:

$$b_{x,y} = min\left(R_{h(c(x,y))}, 1\right) \tag{7.6}$$

where b is the backprojected image and $h(c)$ is a function that maps a colour value to a histogram bin.

Here, the hue channel of a colour image is used for histogram backprojection as it is intended to detect chromatically plain backgrounds. For this, a colour image is first converted from RGB to HSV (Hue, Saturation, Value) colour space. Afterwards, saturation and the achromatic part of HSV are rejected. For the sample image M, a hue histogram is computed with a sufficiently large bin size of 22° to cover a large colour range. This results in 16 bins for the hue range of 360°.

As described above, the histogram of the sample image is backprojected on the input image, which results in a binary image, where white represents colours that match the histogram bin of the sample image and black represents no matches (see Figure 7-21).

In the next step, the pitch position and shape are detected on a lower scale. For this, the binary image is scaled down by a factor of four to remove image details. To further suppress details and support large areas in the image, a median filter with a kernel size of a quarter of the image width is applied. For this process, available methods from OpenCV are used.

To extract possible players moving on the pitch, the binary image is up-scaled to the original size and is inverted, so that the pitch now has the colour black and other areas are white. This binary pitch template is subtracted from the backprojected image. As a result, possible players positioned on the pitch remain.

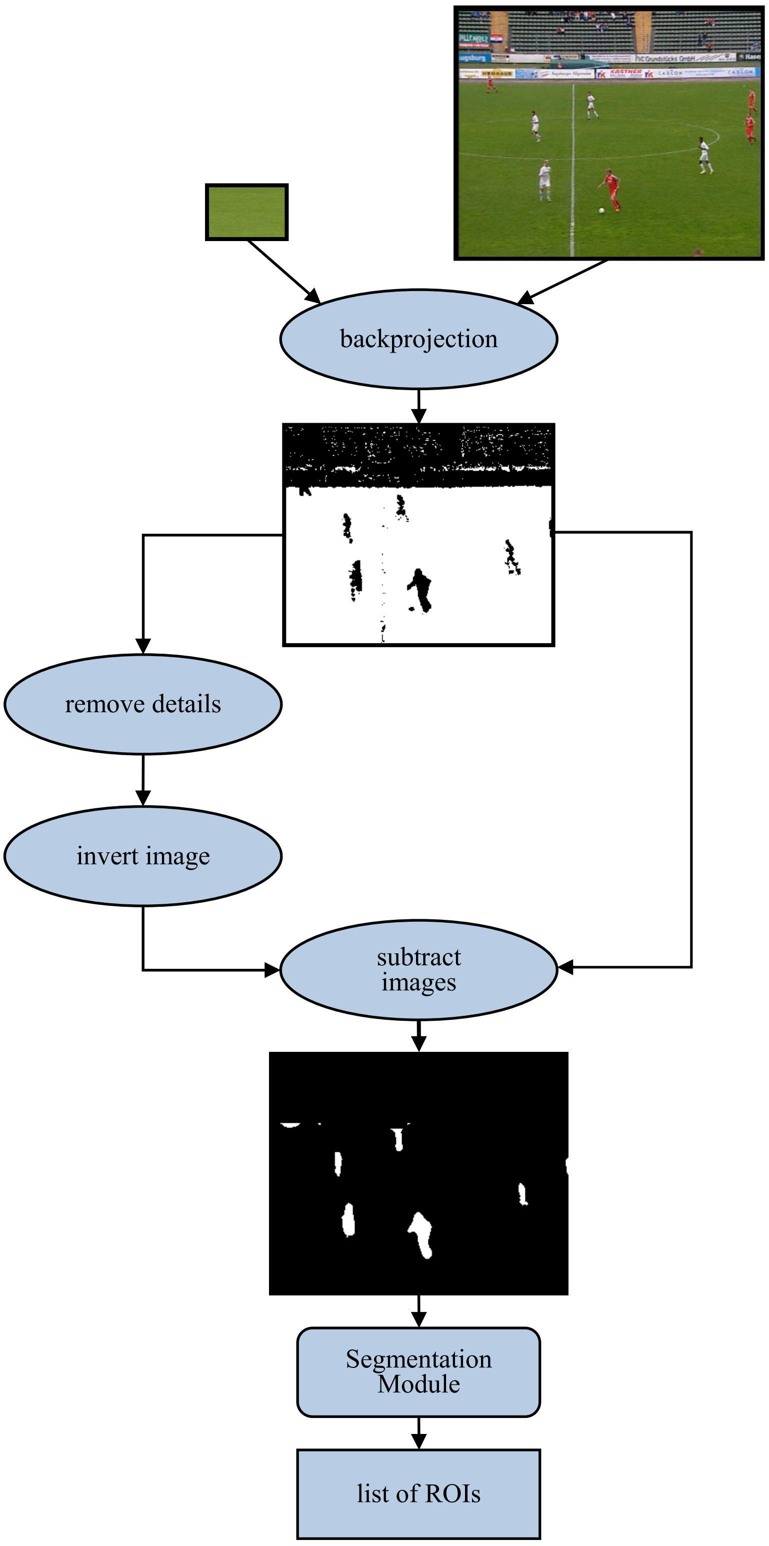


Figure 7-21: Flow chart of the Backprojection Plug-In. In the final step, objects are segmented by another instance of the Segmentation Module.

Obviously, a drawback is that players who are off the pitch are not or only partly detected. It is assumed that such effects can be compensated by ROIs estimated from the Visual Attention Plug-In.

In the last step, objects are segmented by another instance of the Segmentation Module, as described in Chapter 7.3.1.4.

The Backprojection Plug-In provides just a basic implementation within this work. For future work, it is desirable to detect achromatic pitches as well, for example by analysing grey images instead of the hue channel. Additionally, updating a backprojected histogram would increase the reliability for different types of pitches. For this, the sample image can be used as the initialisation for the updating process.

7.3.2.1 *Parameter settings and class structure*

As the Backprojection Plug-In currently provides just a basic implementation, only a default setting is defined at the moment. The "Sample ID" defines the sample image to compute the sample histogram. The "number of bins" value allows an adaptation of the bin size. To control the image scaling factor for the pitch mask computation, the "image scale factor" can be modified.

Table 7-4: Settings for the Backprojection Plug-In.

	Default
	Background colour: green
sample ID	0
number of bins	16
image scale factor	4

For the Segmentation Module instance, the same parameters are chosen as for the Visual Attention Plug-In. The only difference is, as the Backprojection Plug-In delivers a binarised image, that the binarisation in the Segmentation Module is switched off by setting "binarise" to false.

Table 7-5: Settings for the Segmentation Module.

	Default	Setting 1	Setting 2	Setting 3	Setting 4	Setting 5
		Object appearance: solo	Object appearance: team	Object orientation: vertical	Object orientation: horizontal	Object orientation: default
blur mask width	7	-	-	3	3	7
blur mask height	7	-	-	9	9	7
mask size multiplier	1	2	1	-	-	-
binarise	false	false	false	false	false	false

The Backprojection Plug-In receives an 8 bit colour image (RGB) and returns a list of rectangles. The sample image can be specified by the corresponding ID in the constructor.

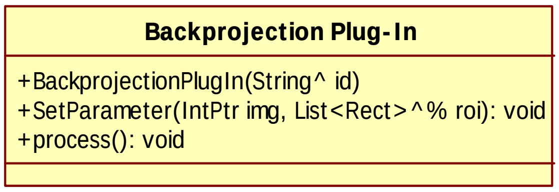

Backprojection Plug-In
+BackprojectionPlugIn(String^ id)
+SetParameter(IntPtr img, List<Rect>^% roi): void
+process(): void

Figure 7-22: Class structure of the Backprojection Plug-In.

7.3.3 Classification plug-in

As the system is a plug-in system, several plug-ins can be used to analyse different features in the same incoming video image. Due to this fact, the number of ROIs, i.e. possible hypotheses of contextually relevant regions, can become quite high. On one hand, this gives a high chance of detecting content related information. On the other hand, it is more important to separate relevant from irrelevant ROIs. The Classification Plug-In fulfils this purpose by weighting ROIs dependent on the given context, i.e. the genre information from the metadata. For this, three ROI features are considered: *shape*, *size* and *position*.

The idea of this plug-in is not to find a decision boundary between possible classes such as, for example, *k-nearest-neighbour* does (cf. Chapter 2.7.2). Within this work, it is of interest how ROIs fit into a single predefined class and leave the final decision to the cropping module. Hence, it is more a weighting of ROIs than a classifying.

The weighting used here is motivated by the *Bayes classifier*. Its main difference to the *Bayes classifier,* as described in Chapter 2.7.1, is that just a single class c is specified manually rather than multiple classes that have been estimated by training data. The class c specifies a special type of genre and the features $\vec{f_i}(s)$ represent shape, size and position. Due to the fact that just one class is defined, no prior probability is required here. To express the probability $P\left(\vec{f_i}(s)|s = c\right)$ of each feature as a weighting factor, every normal distribution is normalised to a range from 0 to 1. This allows the same influence of each feature on the total weight value w_{total}:

$$w_{total} = \mathcal{N}\left(P\left(f_{shape}|s = c\right)\right) \cdot \mathcal{N}\left(P\left(f_{position}|s = c\right)\right) \cdot \mathcal{N}\left(P\left(f_{size}|s = c\right)\right) \qquad (7.7)$$

where $\mathcal{N}(\cdot)$ is the normalisation operator. A threshold t_{total} that defines a minimal allowed weight value can be defined to reject ROIs from further processing.

7.3.3.1 ***The feature shape***

The feature *shape* allows weighting of ROIs by their orientation. For this, a ROI's aspect ratio is calculated and mapped to a function that describes the desired aspect ratio distribution. In terms of better handling, aspect ratios are converted to a fixed range from 0 to 2:

$$r = \begin{cases} \dfrac{height}{width} & \text{for } width \geq height \\ 2 - \dfrac{width}{height} & \text{for } height > width \end{cases} \qquad (7.8)$$

where $r \in [0 \dots 2]$.

To compute the probability of a ROI's aspect ratio P_{shape}, a normal distribution can be defined with a mean value of $\mu \epsilon [0 \dots 2]$ and an arbitrary standard deviation σ:

$$P(f_{shape}|s = c) = \frac{1}{\sigma \cdot \sqrt{2\pi}} e^{\left(-0.5 \cdot \left(\frac{r-\mu}{\sigma}\right)^2\right)} \tag{7.9}$$

Figure 7-23 illustrates the probability distribution for an ROI where its aspect ratio is expected to be approximately $\mu = 1.5$ (two times higher than wide). This simple weighting already allows suppressing ROIs that have an unlikely aspect ratio for a given video content.

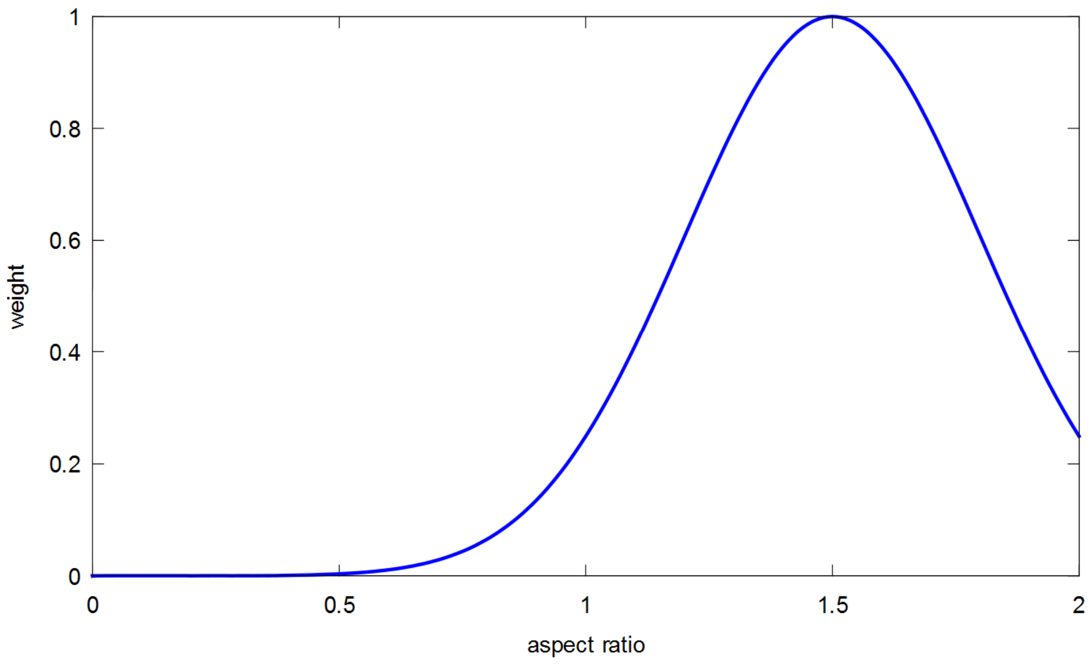


Figure 7-23: Probability distribution for an ROI with expected aspect ratio of $\mu = 1.5$ and $\sigma = 0.3$. According to Equation (7.9), a mean value of $\mu = 1.5$ represents an ROI that is two times higher than wide. The normal distribution $P(f_{shape}|s = c)$ is normalised to a range from 0 to 1 by the normalisation operator $\mathcal{N}(\cdot)$.

7.3.3.2 *The feature position*

The weight for the feature position is obtained by a two-dimensional normal distribution which spans a bell curve over the image (see Figure 7-24). For each pixel position in the image, the probability $P(f_{position}|s=c)$ is computed as follows:

$$P(f_{position}|s=c) = \frac{1}{2\pi \cdot \sigma_x \cdot \sigma_y} e^{\left(-0.5\cdot\left(\left(\frac{x-\mu_x}{\sigma_x}\right)^2 + \left(\frac{y-\mu_y}{\sigma_y}\right)^2\right)\right)} \tag{7.10}$$

where μ_x, μ_y are the mean values and σ_x, σ_y are the standard deviations of the two-dimensional normal distribution. By default, the mean values are set to the image centre and $\sigma_x = \frac{\mu_x}{2}$, $\sigma_y = \frac{\mu_y}{2}$.

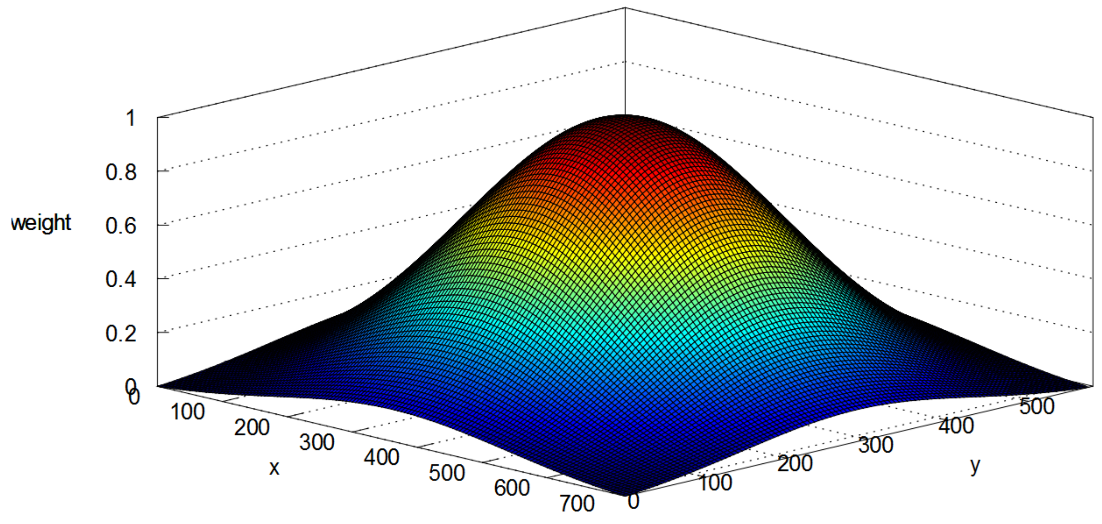


Figure 7-24: Two-dimensional normal distribution for computing the weight of a ROI's position. The probability values are normalised to a range from 0 to 1 by the operator $\mathcal{N}(\cdot)$, which results in the weight value. The image size is 720x576, the mean values are $\mu_x = 360$, $\mu_y = 288$ and the standard deviations are $\sigma_x = \frac{\mu_x}{2}$, $\sigma_y = \frac{\mu_y}{2}$.

7.3.3.3 *The feature size*

The weight of the feature size is calculated by the ratio between ROI size and image size. It is a binary value and is determined by a predefined threshold. Size ratios below the threshold are rejected whereas values greater than the threshold are weighted by 1.

Hence, the resulting weighting value is:

$$\mathcal{N}\big(P(f_{size}|s=c)\big) = \begin{cases} 1 & \text{for } r_{size} \geq t_{size} \\ 0 & \text{for } r_{size} < t_{size} \end{cases} \tag{7.11}$$

where $r_{size} = \frac{width_{ROI} \times height_{ROI}}{width_{image} \times height_{image}}$ and t_{size} is the threshold for the ROI to image ratio. t_{size} has been set sufficiently small to avoid the loss of possibly important ROIs ($t_{size} = 0.1$).

Parameter Settings and Class Structure

Settings for the Classification Plug-In mainly concern the adaptation of different normal distributions.

For the feature shape, the expected ROI aspect ratio can be set by the mean value and an expected standard deviation from that value. Assuming that ROIs tend to be vertically oriented, a negative mean value supports those aspect ratios. In turn, a positive mean value supports mainly horizontally oriented ROIs.

In the same way, the position of a ROI can be weighted by mean values for the x- and y-direction as well as the corresponding standard deviations. Increasing the standard deviations gives more weight to the border areas of an image.

The feature size simply defines a threshold which rejects ROIs that are below a ROI's-size-to-image-size ratio.

Finally, the threshold t_{total} allows the exclusion of ROIs from further processing whose weight value w_{total} is far below an expected weight. This value has been set to $t_{total} = 0.3$ by default. In the case where ROI properties are known in the analysed content, i.e. genre type information is available, the value is increased to $t_{total} = 0.5$. This higher value separates desired from undesired ROIs.

Table 7-6: Settings for the Classification Module.

	default	Setting 1	Setting 2	Setting 3	Setting 4
		Object appearance: solo	Object appearance: team	Object orientation: vertical	Object orientation: horizontal
mean value for normal distribution of feature shape (μ_{shape})	1.0	-	-	-0.7	0.7
divider for μ_{shape} to estimate standard deviation for feature shape (σ_{shape})	2.0	-	-	2.0	2.0
threshold for size feature (t_{size})	0.1	-	-	-	-
multiplier for image width to estimate mean value for normal distribution of position feature ($\mu_{position,x}$)	0.5	-	-	-	-
multiplier for image height to estimate mean value for normal distribution of position feature ($\mu_{position,y}$)	0.5	-	-	-	-
divider for $\mu_{position,x}$ to estimate standard deviation of feature shape ($\sigma_{position,x}$)	1.0	1.5	0.5	-	-
divider for $\mu_{position,y}$ to estimate standard deviation of feature shape ($\sigma_{position,y}$)	1.0	1.5	0.5	-	-
threshold for total probability (t_{total})	0.3	-	-	0.5	0.5

The Classification Plug-In receives a list which contains a further list of ROIs. Each outmost list element represents an extraction plug-in with its extracted ROIs in the inner linked list. For each ROI, the weight value is set by the computed weight value. Figure 7-25 depicts the class structure of the Plug-In.

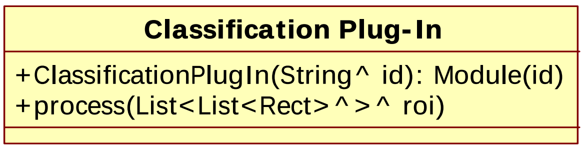

Classification Plug-In
+ClassificationPlugIn(String^ id): Module(id)
+process(List<List<Rect>^>^ roi)

Figure 7-25: Class structure of the Classification Plug-In.

7.3.4 Cropping plug-in

The Cropping Plug-In represents the final plug-in in the complete processing chain. It not only defines final cropping areas, but filters ROIs that move consistently over time. The filtering is done with the aid of a further sub-module, the Cluster Module, which groups corresponding ROIs across several frames.

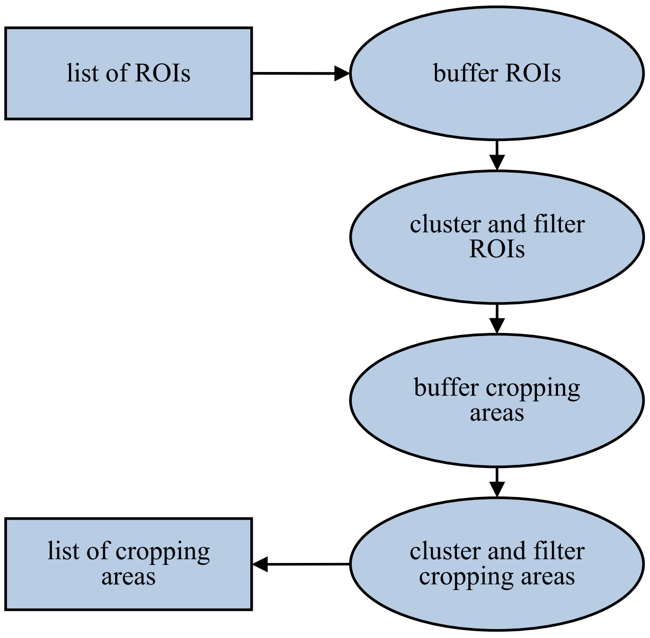


Figure 7-26: Flow chart for the Cropping Plug-In. ROIs and cropping areas are buffered and filtered over time to smooth their trajectories as well as to remove unreliable ROIs, i.e. cropping areas.

To provide buffered information, the Cropping Plug-In accumulates extracted ROIs in time slots of equal size (windows). After filtering ROIs, a cropping area per video image that best encloses high-weighted ROIs is defined. To avoid jittery movements, the cropping areas are buffered and filtered as well. These successive process steps of buffering and filtering are depicted in Figure 7-26. In the following section, the underlying class that manages the buffering of information for several frames is explained. Afterwards, the filtering and smoothing of ROIs and the cropping process are explained.

7.3.4.1 *The Window Manager*

The Window Manager class allows the accumulation of ROIs for a certain number of consecutive video frames (where ROIs can be either objects or cropping areas). The number of frames is fixed and cannot be changed during run time. To deal with shot boundaries that are annotated in the BMF-metadata, further sub-windows within a window are possible, where the first frame of each sub-window represents the first frame of a new shot.

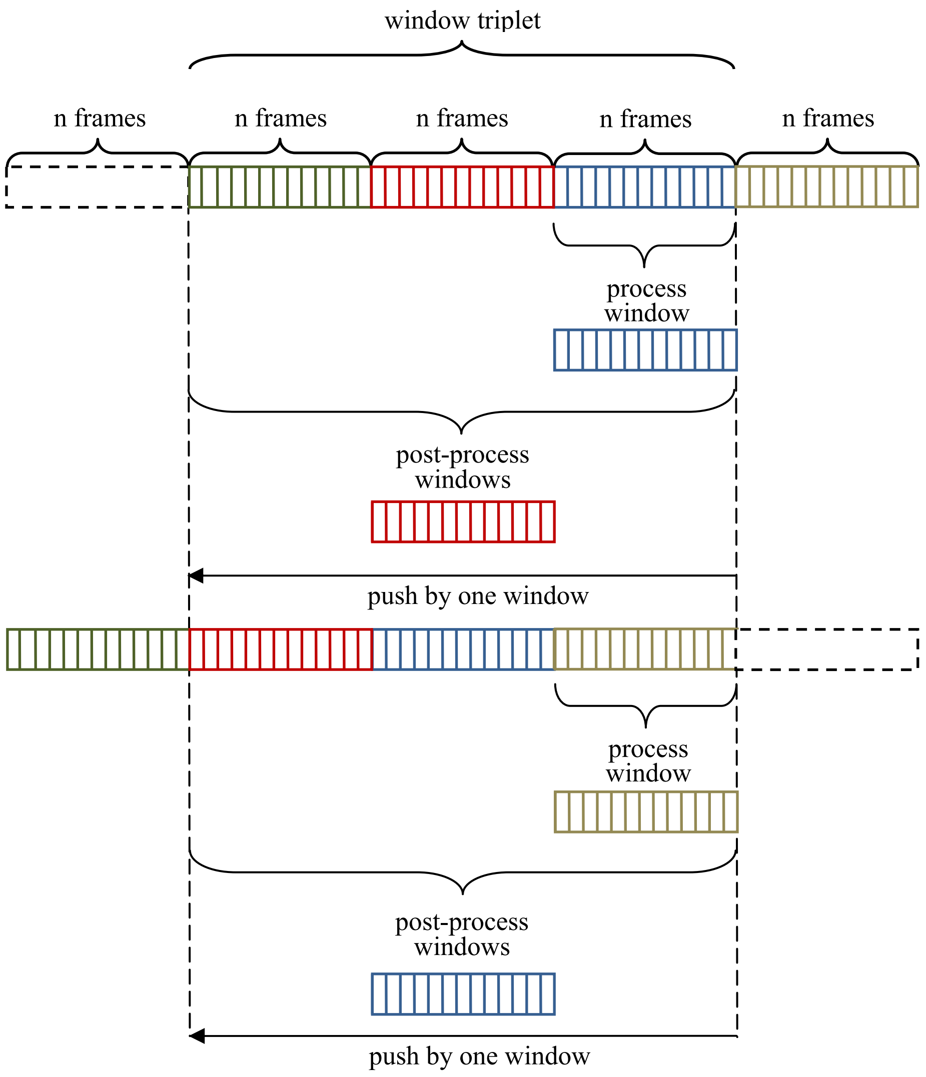
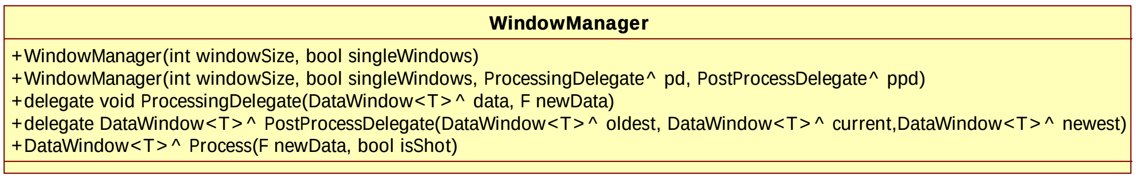


Figure 7-27: Processing of window triplets. Only the latest window in a triplet is processed. For post-processing a window triplet, only the centre window is redefined, whereas all three windows are considered for processing. After post-processing, each window is shifted by one and the new incoming window is attached.

The Window Manager is a generic frame buffer which allows any processing and post-processing of a single window or three consecutive windows (window triplet). In the latter case, the windows are pushed by one as soon as a new window is available, as in a FIFO (First In, First Out) queue (see Figure 7-27). After a new window has been received, it is processed immediately. Once a window triplet is completed, a post-processing on all three windows is applied. Which processing, i.e. post-processing is done, will be explained in the following sections. For the explanation of the Window Manager it is of interest only that such window processing can be done by any method as long as its function pointer is available and has the correct calling parameters.

The class structure of the Window Manager is depicted in Figure 7-28. It mainly consists of the constructors that allow referencing of the function pointer and setting the window size, as well as defining whether window triplets or single windows have to be processed. As soon as the function pointer is available (*ProcessingDelegate* and *PostProcessDelegate*), the Window Manager can be executed, whereby results of a window are returned when the post-processing step has been completed.



```
WindowManager
+WindowManager(int windowSize, bool singleWindows)
+WindowManager(int windowSize, bool singleWindows, ProcessingDelegate^ pd, PostProcessDelegate^ ppd)
+delegate void ProcessingDelegate(DataWindow<T>^ data, F newData)
+delegate DataWindow<T>^ PostProcessDelegate(DataWindow<T>^ oldest, DataWindow<T>^ current,DataWindow<T>^ newest)
+DataWindow<T>^ Process(F newData, bool isShot)
```

Figure 7-28: Class structure of the Window Manager.

7.3.4.2 ***Cluster module***

The Cluster Module is used to identify the behaviour of ROIs within a single time window. It provides processing methods to group corresponding ROIs and a post-processing method to filter ROIs within the estimated clusters. These methods are executed by the Window Manager. Processing and post-processing are applied on single windows only. Therefore, no window triplets are necessary for this operation.

First, the chosen clustering methods are justified and explained. Two types of clustering exist: the Inter-Frame Clustering and the Intra-Frame Clustering, which both apply the same similarity measure for different purposes. The Inter-Frame Clustering estimates trajectories of ROIs by grouping corresponding ROIs over time. The Intra-Frame Clustering fulfils the task of clustering ROIs which are returned by several extraction modules. This is the case when the Visual Attention Plug-In and the Backprojection Plug-In are used in parallel. In case that only one plug-in is used for extraction, the

Intra-Frame Clustering is not required. Afterwards, the post-processing, i.e. the filtering applied on the clusters is presented.

Inter-Frame Clustering

As already mentioned in Chapter 2.6.4, there mainly exist two types of clustering: *hierarchical clustering* and *partitioning*. Whereas the latter usually requires a predefined number of desired clusters, the former does not need such a target value. Due to the fact that the number of clusters cannot be determined in advance for this application, hierarchical clustering is chosen here. In turn, hierarchical clustering can be further split into two approaches: *divisive* and *agglomerative algorithms*. The divisive method starts with a single cluster containing the entire data set and partitions a cluster into two clusters step by step. Alternatively, an agglomerative algorithm functions as bottom-up process which starts with each individual item as an initial cluster. The main advantage of an agglomerative method is that clustering can be initialised without knowing the whole data set and clusters can be extended by new incoming data. Regarding this application, this means that as soon as information of the first frame within a time window is available, the clustering can be started and information from successive frames can be assigned directly. This type of procedure is much more efficient compared to the divisive method, as incoming data can be processed straight away. Due to this reason, the agglomerative clustering is the preferred one for this implementation.

A clustering process is started every new time window, or in case of shot boundaries, every new sub-window. Each ROI which has been extracted from the first available video frame in a time window serves as the initial cluster for the chosen agglomerative clustering method. With any time window that has been completed, the related clustering process is finalised.

As the intention of the Inter-Frame Clustering is to detect movement paths of ROIs, it can be assumed that a ROI lying on such a trajectory appears only once for each frame. This fact simplifies the clustering conditions (see Figure 7-29).

One of the most straightforward similarities measures of whether or not two ROIs of consecutive frames correspond to each other is the Euclidean Distance. As the distance alone does not provide information about the similarity of two ROIs' shapes, aspect ratios are additionally compared. This combination allows a significant predication for

ROI correspondences. Finally, a probability of belonging is computed by multiplying estimated probabilities for distance and shape:

$$P_{inter} = P_{distance} \cdot P_{shape} \tag{7.12}$$

Whether or not two ROIs correspond to each other is finally decided by a threshold for P_{inter}, which is set to $t_{inter} = 0.5$ by default.

Because the Inter-Frame Clustering compares temporally consecutive data, only the latest available ROI from previous frames in a cluster is of interest. This leads to a single-linkage (cf. Section 2.6.4.1) approach, with the difference that not the closest, but the latest ROI in a cluster is compared to the allocable ROI here. Obviously, an average-linkage or complete-linkage approach does not make sense for this application. The average-linkage would consider all ROIs in a cluster (centroid or medioid) to measure the similarity to the allocable ROI. As this includes ROIs from previous frames, past ROI positions or aspect ratios might have negative influences. In turn, the complete linkage uses the maximal distance from clustered ROIs to the allocable ROI as similarity measure, which is obviously not the information of interest either.

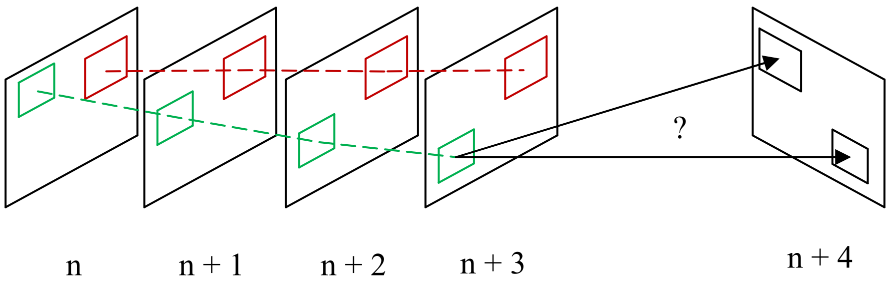


Figure 7-29: Single-linkage clustering applied on two ROI cluster (red and green rectangles) over time. Incoming data is compared with all latest ROIs of existing clusters.

The single-linkage clustering compares new incoming data with the latest ROIs of each existing cluster. To avoid wrong allocations, a cluster is compared with all new ROIs and, in turn, a possible cluster candidate is compared to each cluster to ensure that no cluster with higher correspondence exists. If a cluster with higher correspondence exists, the new ROI is assigned to this cluster. Once a ROI has been assigned to a cluster, it is removed from the list of incoming data. This process is repeated until exclusively ROIs with probabilities below the predefined threshold t_{inter} are left. For each of these remaining ROIs, a new cluster is created. The clustering process over time based on the single-linkage method is depicted in Figure 7-29.

Similarity measure by means of relative ROI distance

For the measure of the distance between two ROIs, not only the centre distance is used. Assuming that two relatively large ROIs overlap, their centre distance will be quite large whereas the actual ROI distance is zero. Because of this, the border to border distance is computed, which is the length of the line between border points which are collinear with both ROI centres. According to Figure 7-30 and with the aid of the intercept theorem, the border to border distance is computed by calculating the distance between border points D_1 and D_2.

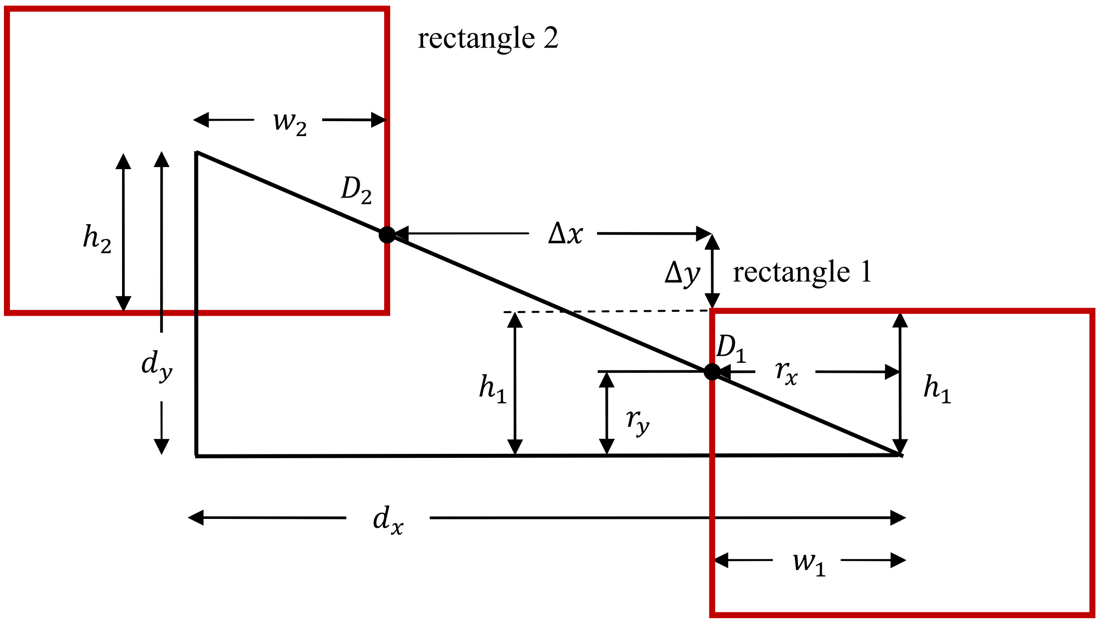


Figure 7-30: Relations between two ROIs for computing the border to border distance. The ratio between r_x and w_1 , i.e. w_2 describes whether the second rectangle is above/below or beside the first rectangle.

As the relations for the intercept theorem vary depending on ROI positioning (the second rectangle is on either the top/bottom/side of the first rectangle), the x- and y-border-distance $\Delta x, \Delta y$ between two ROIs is calculated in two steps:

1. **If $r_x \leq w_1$ (rectangle 2 lies above/below rectangle 1)**:

$$\Delta x_1 = |d_x - r_x|$$

$$\Delta y_1 = |d_y - h_1|$$

where $r_x = \frac{d_x}{d_y} \cdot h_1$

else:

$\Delta x_1 = |d_x - w_1|$

$\Delta y_1 = |d_y - r_y|$

where $r_y = \frac{d_x}{d_y} \cdot w_1$

2. **If $r_x \leq w_2$ (rectangle 1 lies above/below rectangle 2)**::

$\Delta x = |\Delta x_1 - r_x|$

$\Delta y = |\Delta y_1 - h_2|$

where $r_x = \frac{d_x}{d_y} \cdot h_2$

else:

$\Delta x = |\Delta x_1 - w_2|$

$\Delta y = |\Delta y_1 - r_y|$

where $r_y = \frac{d_x}{d_y} \cdot w_2$

For the computation of probability $P_{distance}$, the ROI x- and y-distance is set in relation to the width and height of each ROI. This gives information about the motion-offset of a ROI from one frame to the next depending on its own size. In the case that the ROIs overlap, either vertically or horizontal, the corresponding probability is set to 1. Finally, $P_{distance}$ is computed as follows:

$$P_{distance} = (P_x \cdot P_y)^n \tag{7.13}$$

where

$$P_x = \begin{cases} \frac{w_1 \cdot w_2}{\Delta x^2} & \text{for } \frac{w_1 \cdot w_2}{\Delta x^2} < 1.0 \\ 1.0 & \text{else} \end{cases}$$

$$P_y = \begin{cases} \frac{h_1 \cdot h_2}{\Delta x^2} & \text{for } \frac{h_1 \cdot h_2}{\Delta x^2} < 1.0 \\ 1.0 & \text{else} \end{cases}$$

and n is an exponent to control the weight of increasing distances. For this application, n is set to $n = 2$ to decrease the influence on the total probability by more distant ROIs.

Similarity measure by means of relative ROI aspect ratio

The probability P_{shape} for the relation of ROIs shape is calculated by a single probability, which compares each width and height of the ROIs:

$$P_{shape} = \left(\frac{MIN(w_1, w_2)}{MAX(w_1, w_2)} \cdot \frac{MIN(h_1, h_2)}{MAX(h_1, h_2)}\right)^m \tag{7.14}$$

where MIN and MAX return the minimum, i.e. maximum of two rectangles' width and height (cf. Figure 7-30). The exponent m weights the influence of P_{shape} on the total probability P_{inter}. It is set to $m = 1$ by default.

Intra-Frame Clustering

Intra-Frame clustering is only applied as pre-processing for Inter-Frame Clustering if more than one extraction plug-ins are used. The intention of this method is to reduce the number of ROIs in the case of several extraction plug-ins. In this way, ambiguities of ROIs can be avoided.

The similarity measure works in the same way as for the Inter-Frame Clustering. The difference is that $P_{distance}$ and P_{shape} are successively estimated. It is assumed that ROIs which correspond to the same object overlap. Therefore, P_{shape} is only computed when $P_{distance}$ is zero (ROIs overlap). Whether two ROIs belong to the same object or not ultimately depends entirely on P_{shape}. Hence, a threshold t_{intra} for P_{shape} is defined, which is set to $t_{intra} = 0.3$ by default.

At the moment, the Intra-Frame Clustering can treat results from only two extraction plug-ins. As it is planned to implement further extraction plug-ins in the future, this clustering method should be adapted. For the current implementation, the limitation to two extraction plug-ins is sufficient.

If two ROIs are grouped, only the one with the higher weight is kept to avoid changing of ROIs' shapes through averaging. To support the ROI which has been kept, its weight value is replaced by a new one which is calculated with the aid of the weight value of the rejected ROI:

$$w_{kept} = \begin{cases} \left(1 + {w_{rejected}}^k\right) \cdot w_{kept} & \text{for} \quad \left(1 + {w_{rejected}}^k\right) \cdot w_{kept} \leq 1.0 \\ 1.0 & \text{for} \quad \left(1 + {w_{rejected}}^k\right) \cdot w_{kept} > 1.0 \end{cases} \tag{7.15}$$

where w_{kept} is the weight value of the kept ROI, $w_{rejected}$ is the weight value of the ROI which has been rejected and k is an exponent which controls the influence of the rejected weight value on the kept one. By default, k has been set to $k = 3$.

Filtering ROIs in Clusters

Once a time window has been completed and clusters have been created, reliable ROIs are filtered and gaps within trajectories are closed. This requires a sufficient number of ROIs per cluster in order to evaluate their reliability. This can be ensured by total numbers which define the minimal required number of ROIs per cluster $t_{minNoROIs}$ and the maximal gap between two consecutive ROIs in a cluster $t_{maxGapROIs}$. Clusters with a size below $t_{minNoROIs}$ are removed and obviously missing ROIs in a cluster which are not more than $t_{maxGapROIs}$ are linearly interpolated by the adjacent ROIs. Figure 7-31 depicts the effect of clustering and filtering of ROIs over time.

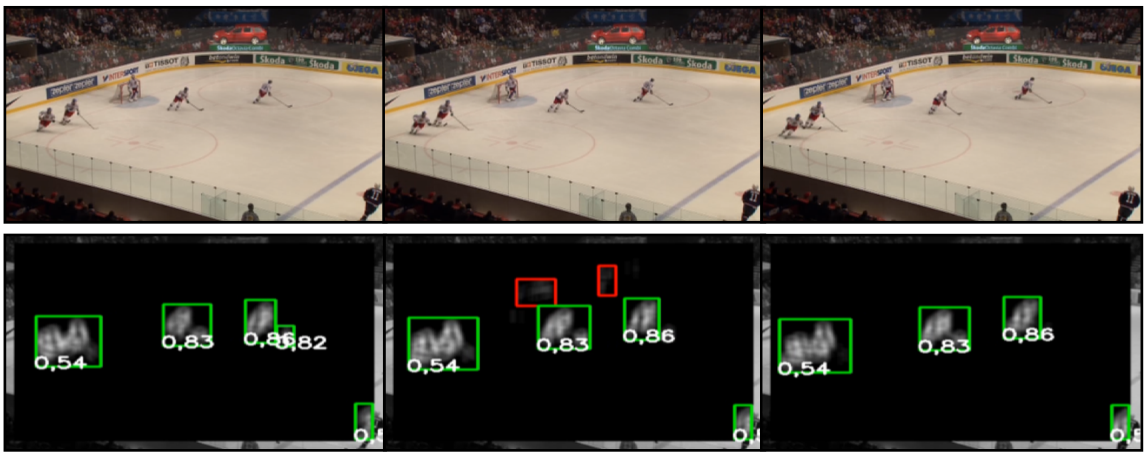


Figure 7-31: Example of filtering ROIs over time by clustering. Red rectangles represent ROIs which cannot be allocated to any cluster and are hence removed from further processing. Green rectangles represent ROIs which are consistent over time. The numbers on each filtered rectangle are the weighting factor w_{total} computed by the classification module (cf. Section 7.3.3).

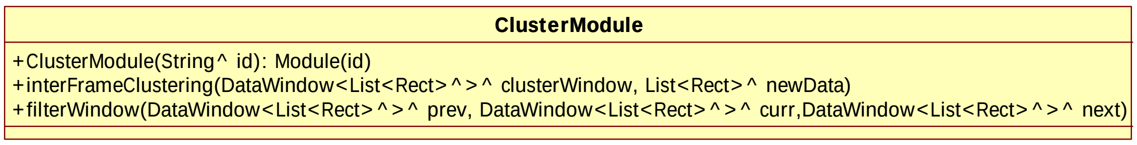
Parameter Settings and Class Structure

Currently, there exists one default setting for the Cluster Module, as these parameter settings provide a good intercept for most type of genres. Hence, this setting is loaded independently of given content properties.

Table 7-7: Settings for the Cluster Module.

	default
threshold for Inter-Frame Clustering (t_{inter})	0.5
threshold for Intra-Frame Clustering (t_{intra})	0.3
exponent to weight computed distance between ROIs (n)	2.0
exponent to weight computed shape similarity between two ROIs (m)	1.0
multiplier for window size to estimate minimal number of ROIs within one cluster ($t_{minNoROIs}$)	0.3
multiplier for window size to estimate maximal number of missing ROIs in a cluster ($t_{maxGapROIs}$)	0.05
exponent to support the kept ROI weight by the rejected one (k)	3.0

The functions *interFrameClustering* and *filterWindow* are referenced by a function pointer of the *WindowManager*. The *ClusterModule* receives the incoming data and assigns it to the existing cluster. In the case of the first frame in a time window, each ROI represents an initial cluster. The *filterWindow* function removes outliers and closes gaps in ROI trajectories as soon as a time window has been finalised. The Cluster Module inherits from the abstract class Module. Figure 7-32 depicts the class structure of the module.



```
ClusterModule
+ClusterModule(String^ id): Module(id)
+interFrameClustering(DataWindow<List<Rect>^>^ clusterWindow, List<Rect>^ newData)
+filterWindow(DataWindow<List<Rect>^>^ prev, DataWindow<List<Rect>^>^ curr,DataWindow<List<Rect>^>^ next)
```

Figure 7-32: Class structure of the Cluster Module.

7.3.4.3 *Defining the final cropping area*

For defining the final cropping area, the Cropping Plug-In is controlled by another instance of the Window Manager, which now buffers window triplets (cf. Chapter 7.3.4.1) instead of a single window.

The idea behind the cropping process is to define a moving scanning mask of fixed size that encloses as many high-weighted ROIs as possible. Fixing the size of the cropping area avoids the annoying effects of mixing camera motion – which is already part of the video – and dynamic zooming by the application.

The next section explains the processing of each window which is executed by the Window Manager. It firstly computes possible ROI combinations that fit into the scanning mask for every video frame. To avoid jittery movements of the cropping areas, the window is further filtered to remove outliers in a final step. The post-processing step is presented afterwards which recovers missing masks at outlier positions with the aid of the whole window triplet.

Computing ROI combinations

As already mentioned, the latest window in a window triplet is processed to define ROI combinations which best fit into a scanning mask. If a new window is available, all windows are shifted by one after the whole triplet has been post-processed (cf. Chapter 7.3.4.1).

For each frame in the latest window, possible ROI combinations are computed. To save computational effort, a level is defined l_{comb}, which limits the number of ROIs that form a combination for a single frame. After all combinations with a number of ROIs equal to or smaller than l_{comb} have been computed, they are stored in a list and the average weight for each combination is calculated. Afterwards, the list of combinations is sorted by the weights in descending order. This sorted list is processed, starting with the highest weight, to find a combination with an enclosing rectangle that fits into the scanning mask. Here, the first that comes along is chosen, as it is inevitably the highest weighted combination. In case that $l_{comb} > 1$, only combinations containing a number of l_{comb} ROIs are considered in the first pass. If no suitable enclosing rectangle has been found, combinations with $l_{comb} - 1$ ROIs are checked in a second pass. This iterative search is applied as long as an enclosing rectangle is found which fits into the scanning mask. In the worst case, either no ROI is found at all or no combinations fit into the scanning mask. In the first case, the image centre position is chosen. In the second case, the highest weighted single ROI is chosen, even if it exceeds the cropping size.

After ROI combinations have been determined, only the centre positions of them are of interest, as they are used as centre positions for the corresponding scanning mask. Even if the ROIs have been filtered in the Cluster Module, scanning masks might rapidly change their centre position from one frame to the next. Therefore, the window size has been set to 40 frames for a video with 25 fps which results in approximately 1½ seconds. Assuming that video content is not drastically moving within 1½ second, one representative scanning mask for the whole time window is still a sufficient time resolution. Therefore, it is not the intention to smooth the existing path of scanning mask movement, e.g. by a B-Spline, but rather to find the most representative scanning mask position within one time window. This is done with the aid of the median x- and y-centre position ($S_{rep}(x)$ and $S_{rep}(y)$) of scanning masks:

$$S_{rep}(x) = \underset{x}{\operatorname{med}} S_i(x) \tag{7.16}$$

$$S_{rep}(y) = \underset{y}{\operatorname{med}} S_i(y) \tag{7.17}$$

where $S_i(x)$ and $S_i(y)$ are x- and y-centre positions of scanning masks within a time window of n frames, with $i = 1, \dots, n$. For pan & scan mode, simply $S_i(x)$ or $S_i(y)$ is calculated, depending on whether a 4:3 video is converted to 16:9 or vice versa.

Except for the representative scanning mask position $S_{rep}(x)$, i.e. $S_{rep}(y)$, all other positions are removed. To re-obtain missing information, not just one time window is used, but the whole window triplet. For this, the representative scanning mask is assigned to the frame in the middle of a time window. Having a time window triplet, missing scanning mask positions are now interpolated linearly between the representative positions (see Figure 7-33).

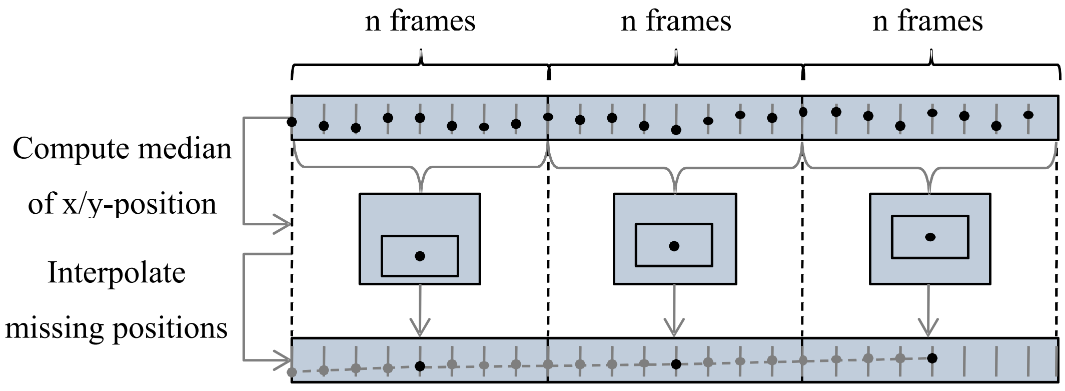


Figure 7-33: Temporal filtering and linear interpolation of scanning masks centre positions. For filtering, the median x- and y-centre-position within a time window are computed. Other positions are removed and missing cropping areas are linearly interpolated between windows.

In Figure 7-34, two defined cropping sizes applied on an ice hockey example by the Cropping Plug-In are depicted.

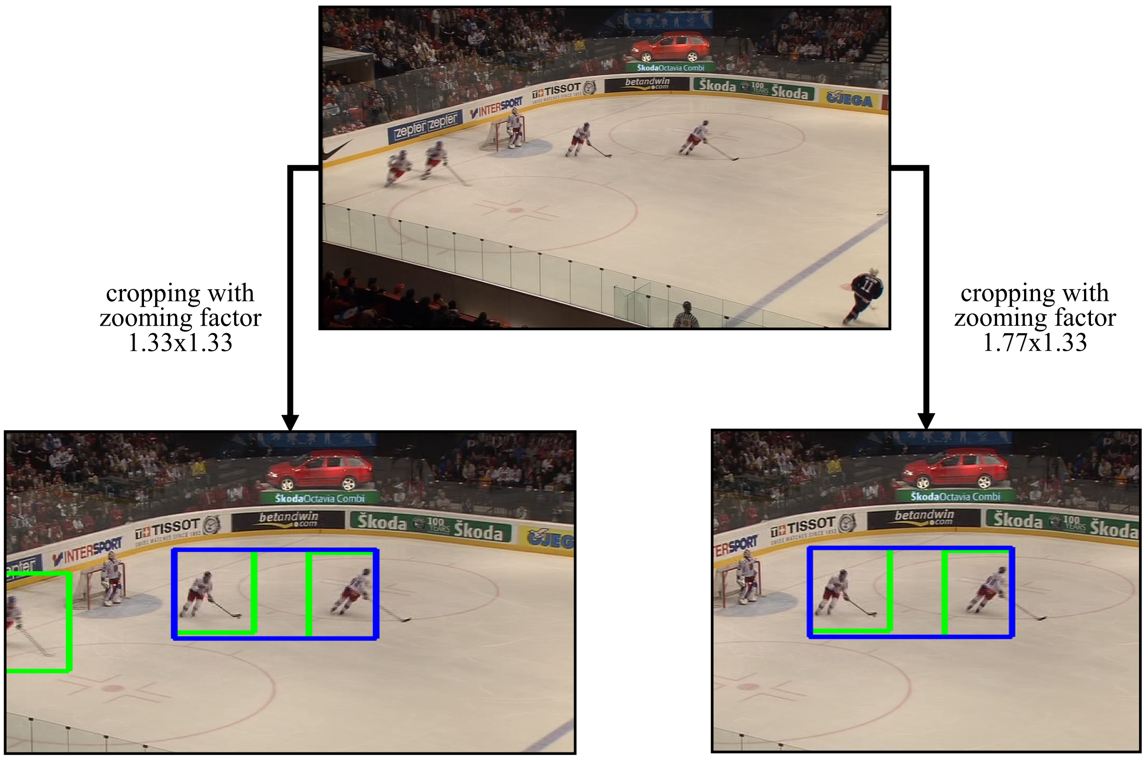


Figure 7-34: Example of two different cropping sizes applied on an ice hockey example with target aspect ratios of 16:9 (left) and 4:3 (right). Green rectangles are ROIs that are consistent over time and the blue rectangle describes the best combination of high-weighted ROIs that fit into the predefined cropping size.

Parameter Settings and Class Structure

The size of the scanning mask can be manually set in the XML-file for the Cropping Plug-In parameters. This is defined by a divisor for image width and height. The only parameter which is linked to video content properties is the number of ROIs which are used for computing ROI combinations. As in the case of single objects, e.g. for individual sports, no combinations are computed. In the case of multiple objects, combinations of two ROIs are computed. This value has been set relatively small, because a higher number of ROI combinations might result in misleading positions as long as the complete context is not understood, for example when the ball position in a soccer game is not known.

Table 7-8: Settings for the Cropping Plug-In.

	default	**Setting 1**	**Setting 2**
		object appearance: solo	object appearance: multiple
multiplier to estimate the buffer size through frames per second of the video	1.6	-	-
divisor which divides the image width to estimate the scanning mask width	1.33	-	-
divisor which divides the image height to estimate the scanning mask height	1.33	-	-
number of ROIs which define a combination	2	1	2

The Cropping Plug-In receives a list of ROIs for each frame and buffers processed ROIs internally. The final cropping area is returned through a single rectangle.

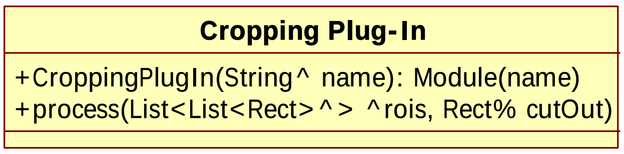

Cropping Plug-In
+CroppingPlugIn(String^ name): Module(name)
+process(List<List<Rect>^> ^rois, Rect% cutOut)

Figure 7-35: Class structure of the Cropping Plug-In.

7.4 Summary

This chapter presented the complete implementation. It first gave an overview of the system architecture which is based on the idea of a plug-in system. Each plug-in represents a component to either identify ROIs in a video, making use of low level methods (extraction plug-ins), or interpret those ROIs on a higher level. In a final step, ROIs are used to define cropping areas by focussing on the most attractive and contextually important regions.

There are currently two extraction plug-ins implemented, the Visual Attention Plug-In and the Backprojection Plug-In. The Visual Attention Plug-In relies on general assumptions about which features attract a viewer’s attention by combining the visual attention system of Hou (still image saliency) and motion information. The Backprojection Plug-In is aimed at specific types of genre where it is known that objects are moving on a plain background, for example a soccer game. Additionally knowing the approximate colour of the background, the feature extraction of this plug-in relies on important top-down information which guides the search for ROIs.

How top-down information can be fed into the system and how to link content specific components and settings based on this information has been presented in Chapter 7.2.2. The possibility to put content-related background knowledge into the system represents the advantage of this work compared to other approaches in the field of broadcast applications. The internal processing of such information relies on a description of each piece of video content by its properties. Those properties are not arbitrarily chosen, but represent the possible properties that can be interpreted by the plug-ins. These properties can be extended at any time, for example when new plug-ins are added. On the other hand, the system can analyse any type of content by running in a default mode.

In the following chapter, the proposed system has been applied on typical sports sequences to crop regions in a video by feeding the genre type through BMF metadata. Results have been compared to the same sequences cropped statically (cropping the centre position), cropped by a video editor from German public broadcasters and simply non-cropped versions by adding boxes (pillarbox or letterbox) if required.

8. Evaluation

The purpose of this evaluation is, on one hand, to make subjective assessments of videos which have been processed by the proposed system. On the other hand, it is of interest to investigate the necessity of adapting cropping area positions for sport applications dependent on the content.

For this, a subjective evaluation has been carried out which compared the output of the introduced application to results from manually and statically cropped sports videos. The manually cropped content has been prepared by a professional video editor from the Bayerischer Rundfunk[2]. The videos have been cropped at three different cropping levels. To get a comparison of the chosen cropping level compared to the non-cropped video, the simply scaled version of the source material was also part of the evaluation. All different cropping versions were finally scaled down to typical mobile TV resolutions.

Apart from evaluating different types of cropping and different cropping levels, the lines of vision of ten subjects watching sports videos have been measured. For the same sequences, ROIs have been extracted by the proposed application. The deviations between ROIs and gaze positions have been used to predict the precision of the system.

8.1 Subjective Evaluation

In this section, the chosen subjective assessment method is first introduced in Section 8.1.1. In the following Sections 8.1.2 - 8.1.6, the evaluation set-up, the material selection and the material preparation are presented. The chosen statistical method is justified for the given set-up in Section 8.1.6. Finally, results of the evaluation conclude this section.

8.1.1 Evaluation method

The Radio-communication Sector of the International Telecommunication Union (ITU-R) provides recommendations for a variety of subjective and objective quality assessments methods (List of ITU-R Recommendations and Reports, 2007). The most important recommendations for subjectively assessing the quality of television and multimedia pictures are ITU-R BT.500-11[3] (Rec. ITU-R BT.500-11, 2007) and ITU-R

[2] The Bayerischer Rundfunk is the Bavarian public broadcaster
[3] Methodology for the subjective assessment of the quality of television pictures

BT.1788[4] (Rec. ITU-R BT.1788, 2007), where the latter refers to the former. The main intention of such tests is to assess the quality of processed videos by subjects. Therefore, a sequence is processed in different ways and presented to the viewer. ITU-R BT.1788 distinguishes between the following subjective measurements procedures for assessing video quality in multimedia systems:

- Double-stimulus impairment scale (DSIS) method as described in Recommendation ITU-R BT.500.
- Double-stimulus continuous quality scale (DSCQS) method as described in Recommendation ITU-R BT.500.
- Single-stimulus (SS) methods as described in Recommendation ITU-R BT.500
- Stimulus-comparison (SC) methods as described in Recommendation ITU-R BT.500
- Single-stimulus continuous quality evaluation (SSCQE) method as described in Recommendation ITU-R BT.500
- Subjective Assessment of Multimedia VIdeo Quality (SAMVIQ)

According to ITU-T BT.500, double-stimulus methods allow the viewer to assess a test version in comparison to a reference version of a sequence. In turn, single-stimulus methods define that the viewer has to grade a single video without reference. SAMVIQ differs from these approaches as it allows the viewer directly to compare more than one version of a sequence to a defined reference version (multi-stimulus). Table 8-1 lists the previously mentioned methods with respect to their most relevant properties.

The multi-stimulus approach of SAMVIQ has the advantage that a subject can directly compare all processed versions of a sequence to a reference as often as he likes. Therefore, SAMVIQ offers a high reliability as the assessor is not forced to make a decision within a defined period (Kozamernik, Sunna, Wyckens, & Pettersen, 2005). As the methods defined in ITU-T BT.500 are mainly designed for video codec evaluations for TV, SAMVIQ is an appropriate method for combining different processing features in the context of multimedia, such as codec type, image format, bit-rate, temporal updating, zooming, etc. (Rec. ITU-R BT.1788, 2007).

[4] Methodology for the subjective assessment of video quality in multimedia applications

Due to its advantages over other methods defined in ITU-T BT.500 in the context of multimedia applications, SAMVIQ has been chosen as the subjective assessment method within this work.

As several cropped and non-cropped videos are compared in this test, a clear reference does not exist. Dependent on the defined reference and the questions asked to the subjects, results change significantly. Here, the statically cropped version has been chosen as a reference for two reasons. First, a direct assessment of the position of the cropping area is achieved. This statement is essential for this work as it indicates the necessity of adapting the cropping area intelligently instead of simply cropping the centre position. Second, the cropping level is assessed by rating the non-cropped version, which is just one among others. Thus, the rating of the cropping level is probably available as side information from the results. As this issue is not the main focus of this test, a general answer to this question is entirely satisfactory.

Table 8-1: Comparison of methods for subjective video quality assessment according to ITU-R recommendations (Rec. ITU-R BT.500-11, 2007; Rec. ITU-R BT.1788, 2007; Jumisko-Pyykkö & Strohmeier, 2008).

	DSIS	DSCQS	SSCQE	SS	SC	SAMVIQ
References	Explicit reference	Hidden reference	No reference	No reference	No reference	Explicit and hidden reference
Comparison	Double-stimulus	Double-stimulus	Single-stimulus	Single-stimulus	Stimulus-comparison	Multi-stimulus
Moment of rating	Re-trospective	Re-trospective	Con-tinuous	Re-trospective	Re-trospective	Re-trospective – rating can be adapted several times
Scale	5-grade impairment scale	5-point continuous scale	5-point continuou s scale	5-point continuous scale (or higher if required)	7-point comparison scale	5-point continuous scale
Length of stimuli	10 seconds	10 seconds	Long stimuli (>60 seconds) up to 20 minutes	10 seconds	10 seconds	Maximum 15 seconds

8.1.1.1 *The quality scale*

Even if SAMVIQ best meets the requirements of this evaluation, the recommended continuous quality scale from “bad” to “excellent” does not provide clear benchmarks to the subjects. Hence, there is a risk that results might scatter extremely. Additionally, it only allows a comparison to an absolute reference of best quality. Therefore, the scale has been replaced by the 7 point comparison scale according to the stimulus-comparison (see Table 8-1). This allows a direct comparison to the statically cropped version on a scale from "much worse" to "much better" (see Figure 8-1).

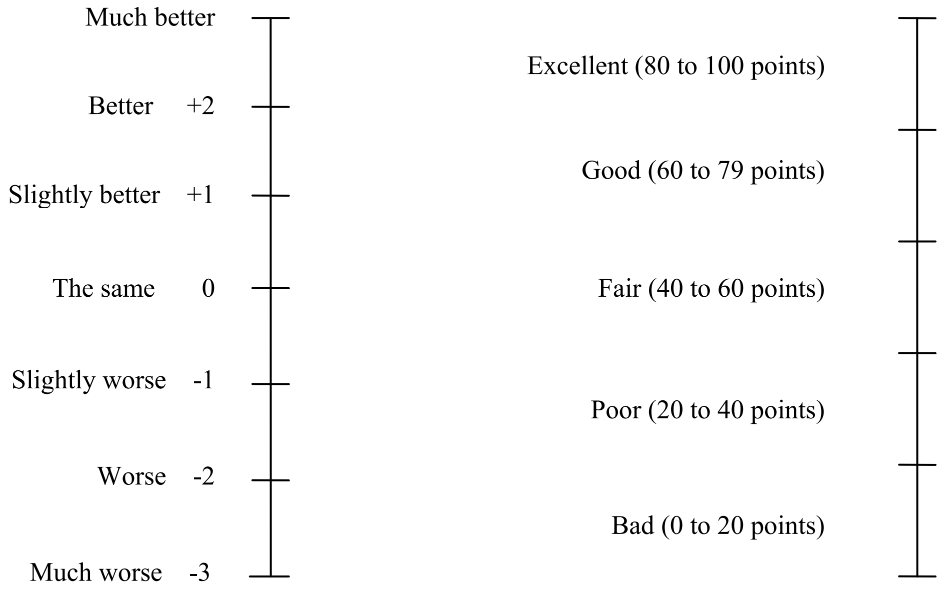


Figure 8-1: Comparison scale (left) and continuous quality scale (right) according to ITU-R BT.500-11 (Rec. ITU-R BT.500-11, 2007).

8.1.1.2 ***Attributes***

As an extension to the ITU-recommendation, attributes have been used which should reflect the reason for a subject's rating. Every time a subject assessed a video above or below zero, he had to specify an attribute which mainly influenced his decision. For this, the four following attributes have been used:

Motion: The motion of the video content is natural/unnatural and continuous/not continuous.

Sharpness: The video content has/does not have satisfying detail resolution.

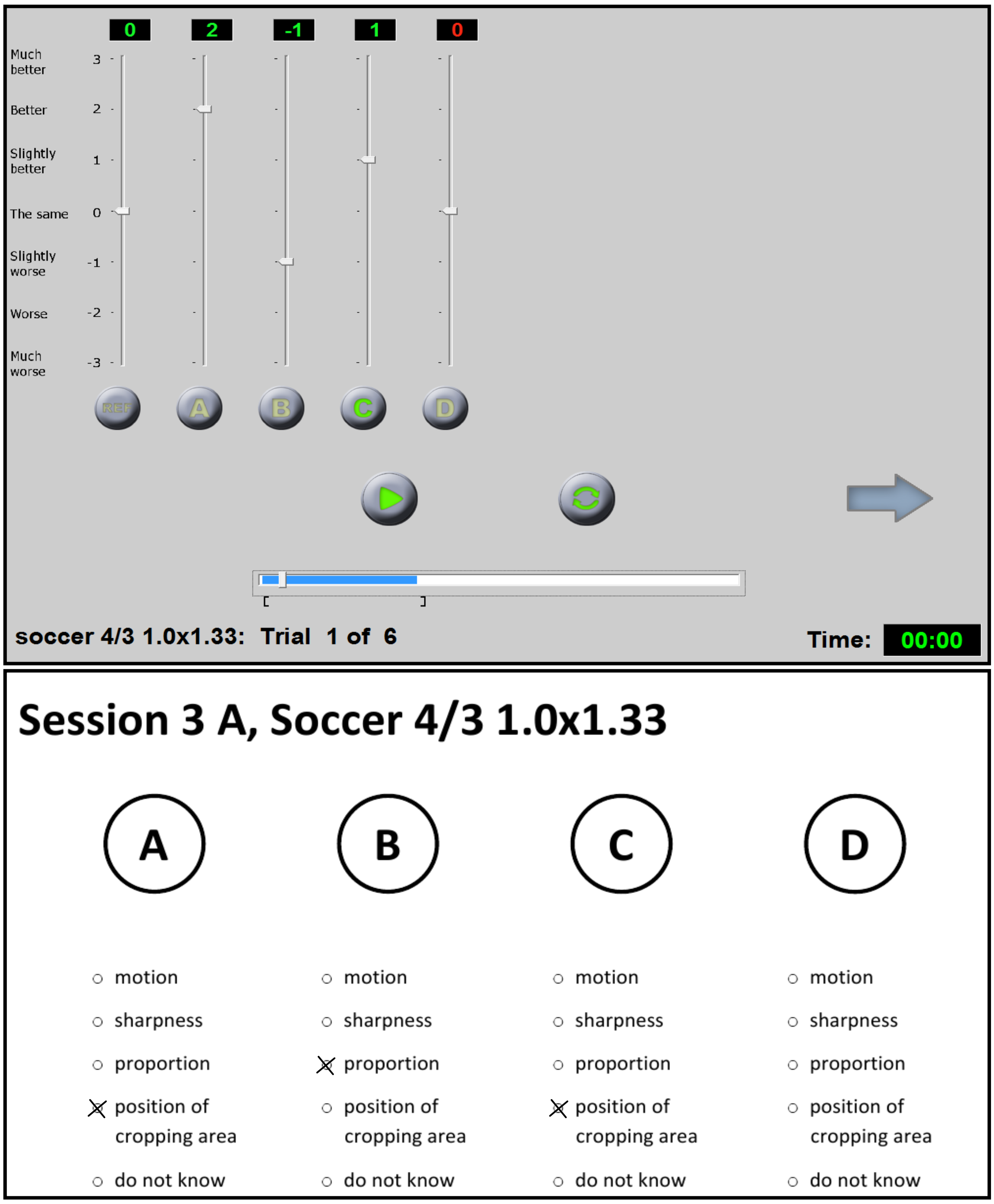
Proportions: The ratio between image size and image content is appropriate/not appropriate.

Position of the cropping area: The cropping area is well/poorly chosen and contains/misses most important elements.

If the subject was not able to justify his decision or the corresponding attribute was not listed, he had the possibility to specify "*do not know*".

8.1.2 Evaluation software

For the evaluation, the software "Suviq" (Subjective Video Quality) has been used which has been implemented at the IRT. The software meets the requirements of the SAMVIQ standard and is available as an executable file and source code. The source code has been slightly modified so that the scale is a comparison scale as required for this evaluation (see Figure 8-2).



Session 3 A, Soccer 4/3 1.0x1.33

A	B	C	D
○ motion	○ motion	○ motion	○ motion
○ sharpness	○ sharpness	○ sharpness	○ sharpness
○ proportion	[✓ proportion]	○ proportion	○ proportion
[✓ position of cropping area]	○ position of cropping area	[✓ position of cropping area]	○ position of cropping area
○ do not know	○ do not know	○ do not know	○ do not know

Figure 8-2: GUI of the slightly modified software Suviq (top) and attributes which have to be assigned to the according version A – D (bottom).

Here, the reference is the statically cropped video and the videos labelled A - D are the manually cropped, automatically cropped and simply scaled versions as well as the hidden reference in random order. For each trial, A - D are newly randomised so that the order of versions to be assessed changes. The software allows looping the selected sequences and switching between different versions while playing. To restrict the length of the loop, the brackets of the playbar can be moved. For each version A - D, the list of attributes was available on an extra sheet of paper (see Figure 8-2). The software is designed for dual screen mode. This means one monitor has been used for assessing and one monitor for rendering the video (see Figure 8-3).

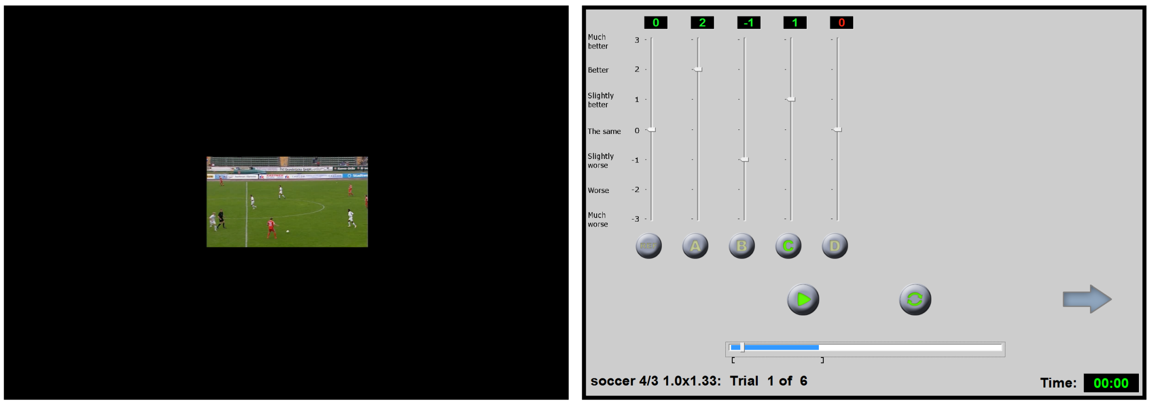


Figure 8-3: Illustration of dual screen mode. One monitor is for rendering the video (left) and the other serves as an assessment GUI (right).

8.1.3 Material selection and processing

The material which has been used for this evaluation was exclusively clean feed material from the archive of the Bayerischer Rundfunk. Other material, for example broadcasted material, is inacceptable as it can contain graphics which can be truncated by cropping. Additionally, the bitrate might be much too low for further processing which can cause heavy artefacts.

Table 8-2: Chosen sequences of team sports (soccer, ice hockey) and individual sports (skiing, show jumping) for this evaluation.

	Type of sport	Format	Duration
Sequence 1	Soccer	576*i*25, 16:9	15s
Sequence 2	Soccer	576*i*25, 4:3	18s
Sequence 3	Ice hockey	576*i*25, 16:9	18s
Sequence 4	Skiing	576*i*25, 16:9	12s
Sequence 5	Show jumping	576*i*25, 16:9	19s
Sequence 6	Show jumping	576*i*25, 4:3	17s

For the evaluation, six SDTV sports sequences from individual sports and team sports have been chosen (see Table 8-2).

8.1.3.1 ***Target formats***

The video formats presented to the subjects have been selected for typical resolutions used for mobile TV. It can be inferred from launched services and recommendations by mobile TV video encoder manufacturers (MAYAH Communications, 2006), that QVGA (320x240, 4:3) is currently the most common video resolution used for mobile TV. For touch screen devices such as Apple iPhone/iPod Touch, there is a trend to 16:9 formats for mobile devices as well. Therefore, in this evaluation 16:9 is considered as an additional aspect ratio. To have as similar viewing conditions as possible for the 4:3 and 16:9 format, a 16:9 format from the DVB-H specification (ETSI TS 102 005 V1.2.1, 2006) with a similar vertical resolution of 400x224 was selected.

8.1.3.2 ***Material preparation***

In the context of cropping video material for broadcast applications, no clear guidelines exist. Only format conversions from 16:9 to 4:3 or vice versa are recommended by using the pan & scan method (cf. Sections 4.4.2.1 and 4.4.2.2).

Here, the pan & scan approach has been chosen as the basis for two further cropping levels. In other words, the cropping mask size on the next higher level is defined by the pan & scan mask size applied on the previous cropping level. This results in relatively rough cropping levels with a highest cropping level of 1.77x1.33 for 16:9, or 1.33x1.77 for 4:3. As no specifications exist for how cropping levels should be applied on different types of content, this rough graduation should give a good indication of possible finer graduations. Figure 8-4 compares the sizes of cropping levels applied on the source material.

To avoid possible differences in quality, all videos have been passed through the same processing chain. For this, an application has been programmed which processes videos in the same way as the proposed cropping system. It receives a comma-separated value file (*.csv) which contains information about the cropping level and the centre position of the scanning mask for defined key frames. Scanning mask positions between key frames were interpolated linearly for each frame.

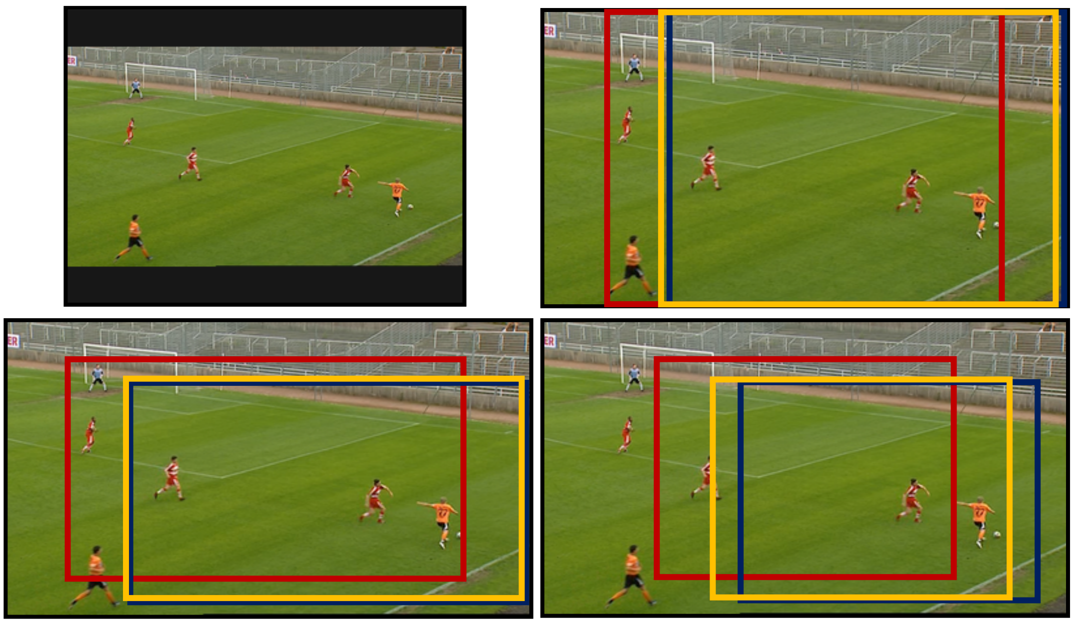


Figure 8-4: Illustration of different cropping levels applied on the example soccer with an aspect ratio of 16:9. The upper left image shows the non-cropped version with letterbox. The other images illustrate the positions of the cropping areas for the statically, i.e. centred cropped version (red bounding box), manually cropped version (blue bounding box) and automatically cropped version (yellow bounding box).The upper right image show results for a cropping level of 1.33x1.0. The bottom left image depicts positions of cropping areas for a cropping level of 1.33x1.33. The bottom right image show results for a cropping level of 1.77x1.33.

All video processing has been applied on uncompressed video (24 bit, RGB). The video decoding has been done by the free open source software VirtualDubMod (Virtual Dub Mod, 2003). As the sequences have been presented on a PC screen, they have been deinterlaced after decoding by the open source VirtualDubMod plug-in Yadif deinterlace algorithm (Balakhnin, 2009). Additionally, 8 pixels have been cut off on the right and left side to remove pixels which do not correspond to the active image area (cf. Section 2.1.2). Finally, the progressive and uncompressed video (576p25) data have been wrapped into an AVI container format.

Manual cropping has been done by a professional video editor of the Bayerischer Rundfunk at a typical editing desk which is part of the TV production workflow. The video-editing-software used was the AVID Media Composer. AVID allows setting key frames to define scanning mask positions. Missing positions can be calculated either by linear interpolation or by splines. The editor of the Bayerischer Rundfunk had chosen the linear interpolation as this is usually sufficient. Instead of exporting the cropped video clips, the positions for each key frame have been logged to an Open Media Framework file (OMF). OMF is a platform-independent file format supported by AVID to transfer digital media. This data has been read in an offline process and converted to .csv files. Finally, the video material has been cropped by the previously mentioned cropping application by parsing the data from the .csv file. This process allowed an accurate reproduction of the scanning mask positions set by the video editor without video quality differences to the other cropped versions.

The statically cropped version as well as the non-cropped version have both also been processed by the previously mentioned cropping application. For the statically cropped version, all videos have been cropped from the centre with the corresponding cropping level defined in the .csv file. In the same way, the non-cropped videos have been cropped from the centre; however, the cropping level has been set to 1.0x1.0 (no cropping). Although this step had no effect on the video content, it completely excluded any possible differences in the assessed videos due to processing.

The automatically cropped versions have been created by the system proposed in this work, where each cropping level has been set in the .xml file for the Cropping Plug-In.

After all videos were cropped, they were scaled to the target resolution (either 320x240 or 400x224) with the aid of VirtualDubMod. To keep the best possible quality of the videos, the Lanczos interpolation has been used (cf. Section 2.3.2.3).

8.1.4 Viewing conditions

As no literature is known which tackles the issue of common viewing distances for small displays, i.e. mobile devices, the viewing distance for this test represents a compromise between theoretically correct and commonly used viewing distances for mobile devices.

Assuming that the human eye has a point source acuity[5] of 1 minute of arc (Smith & Archison, 1997), the theoretically minimal viewing distance dependent of the vertical resolution of the video can be calculated. According to Figure 8-5, the following relation can be established:

$$\tan\left(\frac{\alpha}{2}\right) = \frac{\frac{1}{2} \cdot a}{d} \tag{8.1}$$

where d is the viewing distance, a is the distance between two pixels in vertical direction and α is the beam width at distance d between two pixels in vertical direction.

[5] Point source acuity is a measure of the ability to resolve two very close point sources (Smith & Archison, 1997).

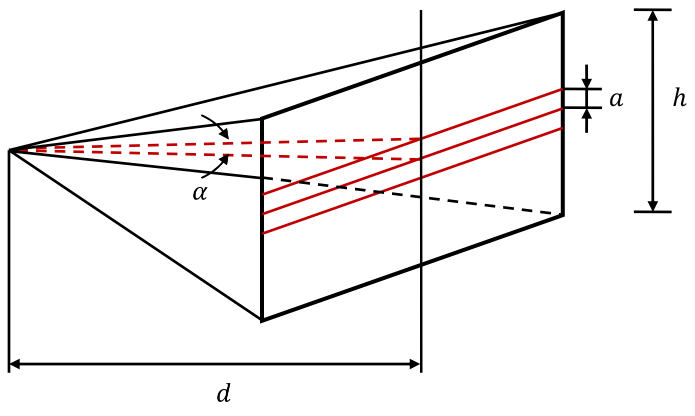


Figure 8-5: Illustration of relations between beam width and viewing distance for a video presented on a screen.

Furthermore, it can be assumed that the distance between two pixels in the vertical direction can be expressed by:

$$a = \frac{h}{n} \tag{8.2}$$

where h is the height of the video on screen and n is the vertical resolution of the video. Substituting a in Equation (8.1) and converting for $\frac{d}{h}$ results in:

$$\frac{d}{h} = \frac{1}{2 \cdot n \cdot \tan\left(\frac{\alpha}{2}\right)} \tag{8.3}$$

According to Equation (8.3), the optimal minimal viewing distances given in height units for 240 and 224 are:

$$\left(\frac{d}{h}\right)_{240} = \frac{1}{2 \cdot 240 \cdot \tan\left(\frac{1}{60° \cdot 2}\right)} = 14.34 \tag{8.4}$$

$$\left(\frac{d}{h}\right)_{224} = \frac{1}{2 \cdot 224 \cdot \tan\left(\frac{1}{60° \cdot 2}\right)} = 15.35 \tag{8.5}$$

These viewing distances are unrealistic in everyday life. From tests with persons watching videos on their mobile phones, 10 height units provide a good approximation.

As a compromise between theoretical and practical values, 12 height units have been chosen for this evaluation.

The evaluation took place at the IRT in a room specially designed for viewing tests. The light conditions within this room as well as luminance conditions of the PC screen were calibrated according to ITU-R BT.1788 (Rec. ITU-R BT.1788, 2007). The settings used for this evaluation are summarised in Table 8-3:

Table 8-3: Settings used for the evaluation according to ITU-R BT.1788 (Rec. ITU-R BT.1788, 2007).

Parameter	Setting
Viewing distance	Constrained: 12 H
Peak luminance of the screen	70-250 cd/m^2
Ratio of luminance of inactive screen to peak luminance	≤ 0.05
Ratio of the luminance of the screen, when displaying only black level in a completely dark room, to that corresponding to peak white	≤ 0.1
Ratio of luminance of background behind picture monitor to peak luminance of picture	≤ 0.2
Chromaticity of background	D_{65}
Background room illumination	$\leq$ 20 lux

Each subject was asked to lean his head against the headrest of a chair specially prepared for this evaluation to keep the viewing distance constant during the evaluation. Before the evaluation actually started, all subjects were familiarised with the test procedure by an introduction (see Appendix) and a training session. Figure 8-6 illustrates the conditions on-site.

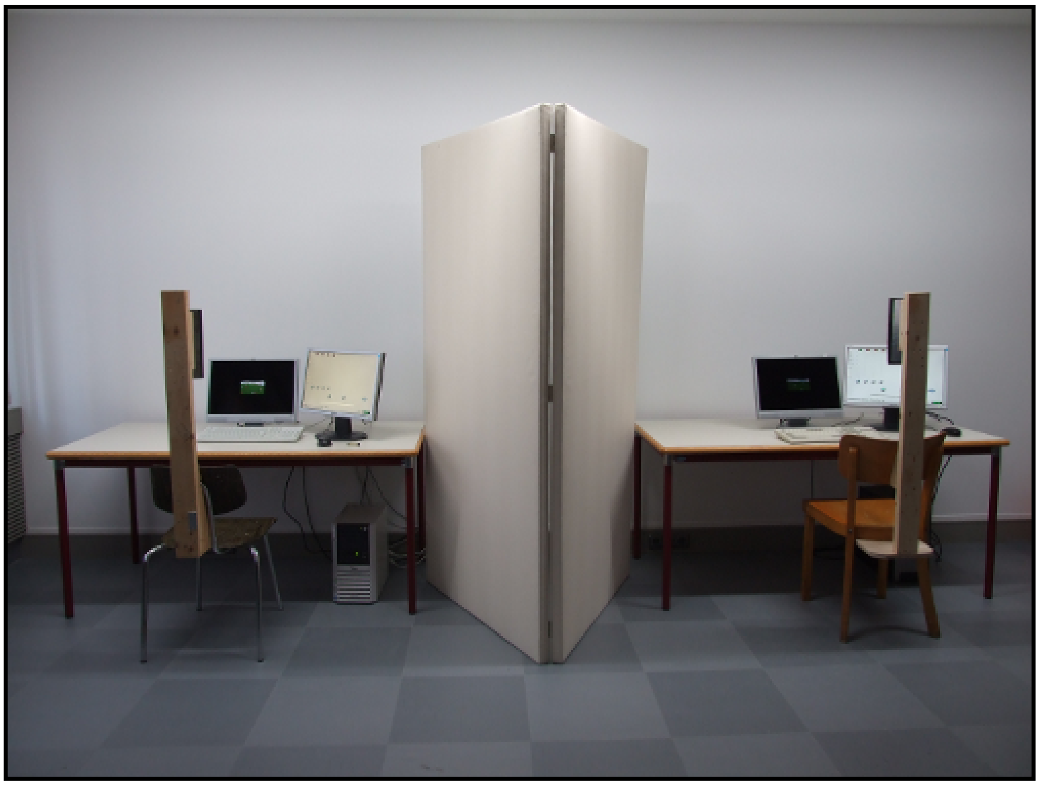


Figure 8-6: Viewing conditions on-site according to ITU-R BT.1788 (Rec. ITU-R BT.1788, 2007).

To avoid juddering of presented videos, a display has been used which allows a display refresh rate of 75 Hz (integral multiple of video frame rate). Table 8-4 lists the configurations of the multimedia systems depicted on Figure 8-6.

Table 8-4: Configurations of the multimedia systems used for the subjective evaluation test environment.

	PC1	**PC2**
Type of Display	HP W19b and PHILIPS 200WS	HP W19b and PHILIPS 200WS
Display size	19" and 20"	19" and 20"
Video display card	NVIDIA GeForce 7300 SE	NVIDIA GeForce 7300 SE
Model	Intel Core 2 Duo, 3.0 GHz, 3 GB RAM	Intel Core 2 DUO, 2.66 GHz, 2GB RAM,
Image information	320x240 and 400x224 native	320x240 and 400x224 native

8.1.5 Subjects

As recommended for SAMVIQ, 15 persons participated in the subjective test. The observers were exclusively employees at IRT. Eight subjects were experts and seven were non-experts, where non-experts are not directly concerned with picture quality as part of their normal work.

8.1.6 Statistical methods

Before introducing the chosen statistical method at the end of this section, a brief overview on most common statistical procedures is given. First, possible measurement scales are presented, which are the basis of any statistical evaluation. In Section 8.1.6.2, parametric statistics are compared to non-parametric statistics and conditions are defined for when to use which method.

8.1.6.1 ***Measurement scales***

A determining factor for the choice of an appropriate statistical procedure is which measurement scale has been used. A measurement scale allows the characterisation of variables to be measured. Measurement scales can be split into two groups: *non-metric scales* and *metric scales* (Frank & Todeschini, 1994).

Non-metric scales are used for estimating qualitative variables. The simplest example for a *non-metric scale* is a *nominal scale* which is the mathematically weakest scale for qualitative variables. It only defines a category or label without any order relation. A more significant *non-metric scale* than the *nominal scale* is the *ordinal scale*. The *ordinal scale* arranges categories in order, where the differences between ranks are more relative than quantitative.

Metric scales define a scale for quantitative variables. A *metric scale* where the starting point is not clearly defined, but the difference between each pair of adjacent values is defined, is called a *proportional scale*. In turn, a stronger *metric scale* where its starting point is well-defined is called a *ratio scale*. Examples of variables measured on ratio scales are weight and length.

8.1.6.2 ***Parametric versus non-parametric statistics***

Inferential statistical[6] procedures can be categorised as *parametric* and *non-parametric statistics* (Sheskin, 2007). *Parametric statistics* make an assumption of population parameters which characterise an underlying distribution. *Non-parametric statistics*, also referred to as *distribution/assumption free,* are not making an assumption about an underlying distribution in advance. Therefore, *non-parametric statistics* are applied when the underlying distribution is questionable.

As general rule, *non-parametric statistics* are applied if one or more assumptions for parametric tests are violated. According to (Corder & Foreman, 2009), parametric assumptions include data which:

- approximately resembles a known probability distribution (in most cases normal distribution)
- has respective populations of approximately equal variances
- consists of independent observations, except for paired values
- consists of values on a metric scale
- is adequate large

Table 8-5: Commonly used statistical tests for parametric statistics and their non-parametric counterparts

Type of test	Parametric Test	Non-Parametric Test
Comparing two related samples	Wilcoxon signed rank test	t-test for dependent samples
Comparing two unrelated samples	Mann-Whitney U-test	t-test for independent samples
Comparing three or more related samples	Friedman test	Repeated measures analysis of variance (ANOVA)
Comparing three or more unrelated samples	Kruskal-Wallis H-test	One-way analysis of variance (ANOVA)
Comparing two rank-ordered variables	Spearman rank-order correlation	Pearson product-moment correlation

[6] Inferential statistics or statistical induction are statistical methods which describe a model from random sampling.

For both types of statistics, *statistical tests* (*hypothesis tests*) exist which verify a hypothesis about parameters, a distribution or a goodness of fit (Frank & Todeschini, 1994). Most *non-parametric tests* are based on ranked data by ordering from the lowest to the highest value, where the median value is usually used to estimate the location of a population distribution. Table 8-5 gives an overview of the most important parametric and corresponding non-parametric tests.

8.1.6.3 ***Chosen statistical method***

Due to the fact that the comparison scale used in this evaluation is a non-metric scale, the chosen statistical method is a *non-parametric method*.

To estimate the location of population distribution, the *median* value (0.5-quantile) and *quartiles* (0.25-quantile and 0.75-quantile) are used. For the number of 15 subjects, this results in the following ranks:

$$Q_{.25} = (n+1) \cdot 0.25 = (15+1) \cdot 0.25 = 4 \tag{8.6}$$

$$Q_{.25} = (n+1) \cdot 0.5 = (15+1) \cdot 0.5 = 8 \tag{8.7}$$

$$Q_{.75} = (n+1) \cdot 0.75 = (15+1) \cdot 0.75 = 12 \tag{8.8}$$

For this evaluation it is of interest whether two cropping methods applied on a video have been assessed as significantly different or not. Therefore, two related samples are compared, which leads to the *Wilcoxon signed rank test* (cf. Table 8-5).

This non-parametric method tests whether paired observations X and Y with $X = X_1, \dots, X_n$ and $Y = Y_1, \dots, Y_n$ correspond to the same location of symmetric population.

Supposing that Z is continuous around a median θ, where $Z_i = X_i - Y_i$ for $i = 1, \dots, n$, the null hypothesis that X and Y correspond to the same location of a population is $H_0\colon \theta = 0$. The test verifies whether this hypothesis can be confirmed and, in turn, the alternative hypothesis $H_1\colon \theta \neq 0$ can be rejected. For this, $|Z|$ is first computed and sorted in ascending order. After ranking the absolute values of Z, the sum $\sum R_+$ is computed by ranks $|Z_i|$ which correspond to values $Z_i > 0$ and the sum $\sum R_-$ is computed by ranks $|Z_i|$ which correspond to values $Z_i < 0$. Identical absolute values are averaged before summing up. Difference values of $Z_i = 0$ are excluded from the ranked list.

Finally, H_0 is tested by a predefined significance level α. According to (Sachs & Reynarowych, 1984), this level is commonly set to $\alpha = 5\%$. H_0 is rejected when the value $\hat{R}$, which is the smaller value of $\sum R_-$ and $\sum R_+$, lies below the listed value of Table 8-6. This test is called *one-sided Wilcoxon signed rank test*. Table 8-6 is a short excerpt of values taken from (Sachs & Reynarowych, 1984) and (Cook, 2009).

For a number of test observations $n > 25$, it is assumed that $R(n,\alpha)$ follows a normal distribution and hence the following approximation can be applied:

$$R(n,\alpha) = \frac{n \cdot (n+1)}{4} - z \cdot \sqrt{\frac{1}{24} \cdot n \cdot (n+1) \cdot (2n+1)} \tag{8.9}$$

where z denotes the bounds of a normal distribution for a given α (Sachs & Reynarowych, 1984).

Table 8-6: Short excerpt on values for critical values for the Wilcoxon signed rank test from (Sachs & Reynarowych, 1984) and (Cook, 2009).

	One-sided			**One-sided**	
n	5%	1%	**n**	5%	1%
6	2	-	**21**	67	49
7	3	0	**22**	75	55
8	5	1	**23**	83	62
9	8	3	**24**	91	69
10	10	5	**25**	100	76
11	13	7	**26**	110	84
12	17	9	**27**	119	92
13	21	12	**28**	130	101
14	25	15	**29**	140	110
15	30	19	**30**	151	120
16	35	23	**31**	163	130
17	41	27	**32**	175	140
18	47	32	**33**	187	151
19	53	37	**34**	200	162
20	60	43	**35**	213	173

For the *Wilcoxon signed rank test* computation, the software WinSTAT (Fitch, 2010) has been employed. This software returns values for $\sum R_-$, $\sum R_+$, z and α. For consistency, the values of $\sum R_-$ and $\sum R_+$ are exclusively used to perform the one-sided test for all evaluations within this work.

For further information concerning the *Wilcoxon signed rank test*, the author refers to (Sheskin, 2007), (Sachs & Reynarowych, 1984) and (Kotz & Johnson, 1988).

8.1.7 Evaluation results

In this section, results of the subjective evaluation for each type of sport are presented. For statistical analysis, the previously mentioned methods of median and quartile, Wilcoxon-test and frequency distributions of attributes have been applied.

Each type of genre is treated in a separate section where the results of all cropping levels are plotted in a single figure. In the lower part of each figure, the frequency distribution of attributes to the respective assessments is presented graphically. For illustrative purposes, a distinction is made between attributes according to positive and negative ratings by splitting the frequency distribution. This allocation allows a better visualisation and exploration of assessments by subjects. The sum of attributes per assessment still results in 100%.

8.1.7.1 *Ice hockey 16:9*

Compared to the statically cropped version, clear differences to all other versions of the ice hockey sequence can be recognised (see Figure 8-7). For cropping level 1.33x1.0, the letterbox version has been assessed as slightly worse than the statically cropped version due to the attribute “proportions” (see Figure 8-7 below). This significant difference diminishes with higher cropping levels mainly due to the “position of the cropping area”, which indicates that a static cropping area seems no longer to capture the most relevant areas of the ice hockey sequence. This assumption is confirmed by the fact that with higher cropping levels the median values of the automatically and manually cropped versions tend to be slightly better than the static version. Additionally, the frequency distributions of chosen attributes clearly affirm this positive trend by the attribute “position of the cropping area”.

According to the Wilcoxon test, there is no significant difference between the automatically and statically cropped versions for all cropping levels (see Table 8-7).

The highest percentages of ratings which have negatively influenced the automatic version were caused by of the attribute “motion”. It shows that the subjects were mostly satisfied with the “position of the cropping area”, whereas the movement of the cropping area might have been slightly annoying.

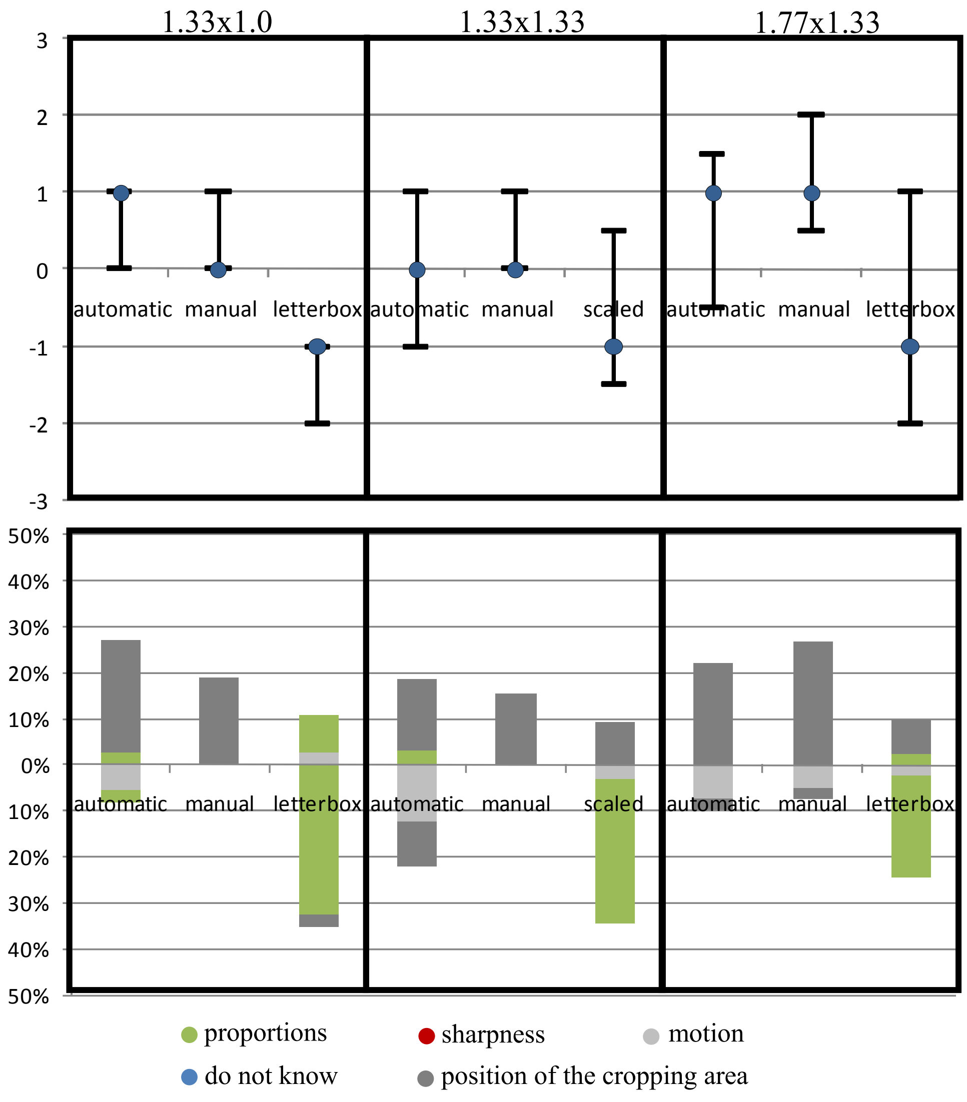


Figure 8-7: Median and quartiles for all cropping levels applied on the ice hockey sequence (top) and corresponding frequency of attributes (bottom).

Table 8-7: Values of the one sided Wilcoxon test applied on the ice hockey sequence.

	1.33x1.0		**1.33x1.33**		**1.77x1.33**	
	$\hat{R}$	different	$\hat{R}$	different	$\hat{R}$	different
static-automatic	$\hat{R} = 23 \geqq 21 = R(13,0.05)$	no	$\hat{R} = 42.5 \geqq 21 = R(13,0.05)$	no	$\hat{R} = 24.5 \geqq 21 = R(13,0.05)$	no
static-manual	$\hat{R} = 0 \leqq 3 = R(7,0.05)$	yes	$\hat{R} = 0 \leqq 0 = R(5,0.05)$	no	$\hat{R} = 18.5 \leqq 25 = R(14,0.05)$	yes
static-letterbox/ scaled	$\hat{R} = 14.5 \leqq 30 = R(15,0.05)$	yes	$\hat{R} = 26.5 \geqq 25 = R(14,0.05)$	no	$\hat{R} = 29.5 \geqq 25 = R(14,0.05)$	no
automatic-manual	$\hat{R} = 21 \geqq 8 = R(9,0.05)$	no	$\hat{R} = 22 \geqq 17 = R(12,0.05)$	no	$\hat{R} = 36.5 \geqq 25 = R(14,0.05)$	no
automatic-letterbox/ scaled	$\hat{R} = 14 \leqq 30 = R(15,0.05)$	yes	$\hat{R} = 26 \geqq 21 = R(13,0.05)$	no	$\hat{R} = 26 \geqq 25 = R(14,0.05)$	no
letterbox/ scaled-manual	$\hat{R} = 7 \leqq 30 = R(15,0.05)$	yes	$\hat{R} = 19.5 \leqq 25 = R(14,0.05)$	yes	$\hat{R} = 15.5 \leqq 21 = R(13,0.05)$	yes

8.1.7.2 *Soccer 16:9*

For all versions of the soccer 16:9 sequence, the automatically and manually cropped versions are preferred to the statically cropped version, which preference gets more significant with higher cropping levels. Clearly, this positive trend is attributable to the "position of the cropping area". Only for the highest cropping level, the manually cropped version has been significantly preferred to the automatic version because of the "position of the cropping area".

The slightly worse assessments of the non-cropped version seem to be caused by "proportions". In turn, the non-cropped version gets more support with higher cropping levels which in the end is reflected by the "position of the cropping area". Again, this indicates that the static version obviously does not deliver satisfying results because of losing important image regions by not adapting the cropping area to the content.

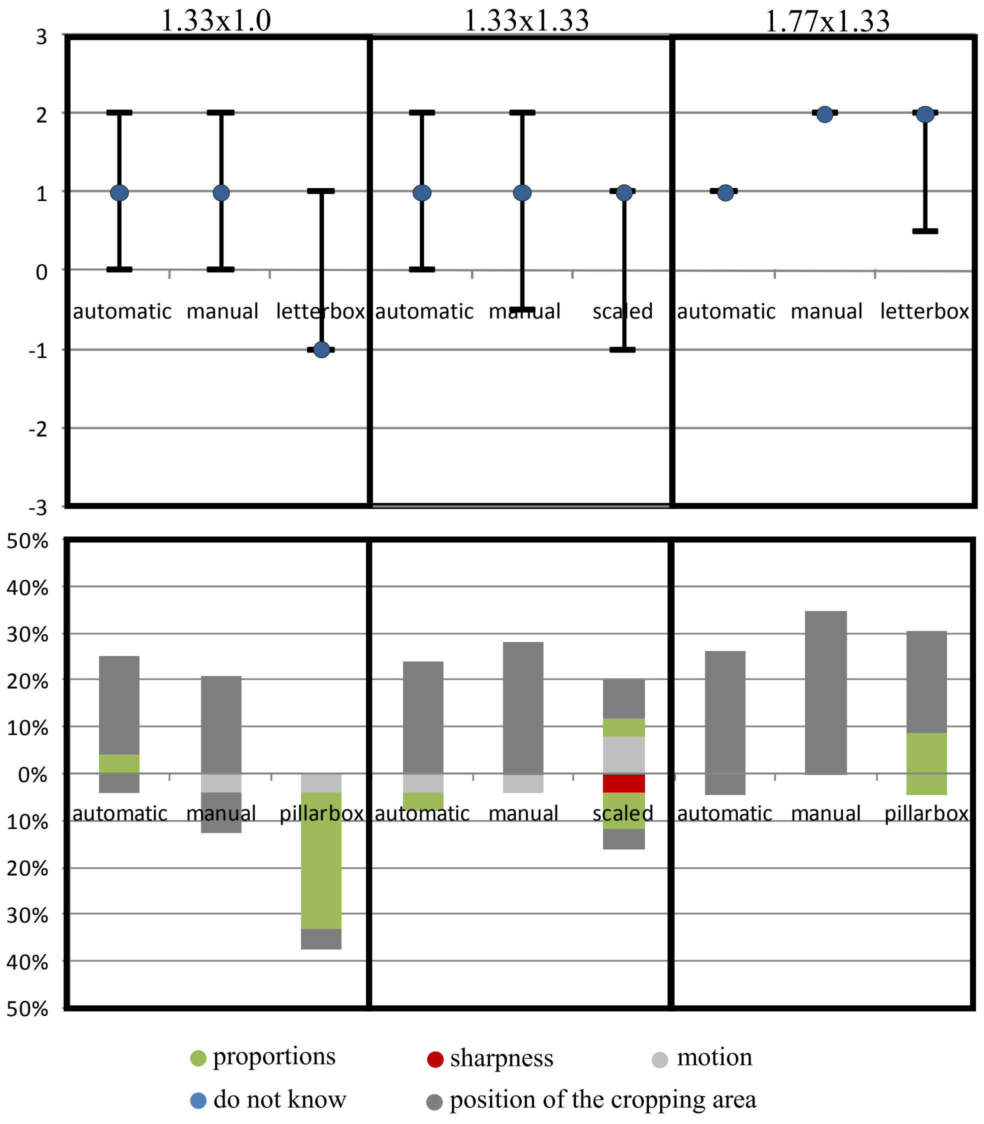


Figure 8-8: Median and quartiles for all cropping levels applied on the soccer 16:9 sequence (top) and corresponding frequency of attributes (bottom).

Table 8-8: Values of the one sided Wilcoxon test applied on soccer 16:9 sequence.

	1.33x1.0		**1.33x1.33**		**1.77x1.33**	
	$\boldsymbol{\hat{R}}$	different	$\boldsymbol{\hat{R}}$	different	$\boldsymbol{\hat{R}}$	different
static-automatic	$\hat{R} = 3.5 \leqq 13 = R(11{,}0.05)$	yes	$\hat{R} = 12 \leqq 13 = R(11{,}0.05)$	yes	$\hat{R} = 11 \leqq 21 = R(13{,}0.05)$	yes
static-manual	$\hat{R} = 18.5 \leqq 21 = R(13{,}0.05)$	yes	$\hat{R} = 27 \geqq 25 = R(14{,}0.05)$	no	$\hat{R} = 0 \leqq 25 = R(14{,}0.05)$	yes
static-letterbox/ scaled	$\hat{R} = 37.5 \geqq 30 = R(15{,}0.05)$	no	$\hat{R} = 39 \geqq 30 = R(15{,}0.05)$	no	$\hat{R} = 21.5 \leqq 25 = R(14{,}0.05)$	yes

	1.33x1.0		**1.33x1.33**		**1.77x1.33**	
	$\hat{R}$	different	$\hat{R}$	different	$\hat{R}$	different
automatic-manual	$\hat{R} = 5 \geqq 2 = R(6,0.05)$	no	$\hat{R} = 25.5 \geqq 13 = R(11,0.05)$	no	$\hat{R} = 6 \leqq 25 = R(14,0.05)$	yes
automatic-letterbox/ scaled	$\hat{R} = 7.5 \leqq 17 = R(12,0.05)$	yes	$\hat{R} = 21.5 \geqq 13 = R(11,0.05)$	no	$\hat{R} = 42 \geqq 21 = R(13,0.05)$	no
letterbox/ scaled-manual	$\hat{R} = 12 \leqq 21 = R(13,0.05)$	yes	$\hat{R} = 30.5 \geqq 17 = R(12,0.05)$	no	$\hat{R} = 13 \leqq 17 = R(12,0.05)$	yes

8.1.7.3 *Soccer 4:3*

This sequence represents an example where the cropping algorithm makes a wrong decision for a short time and loses the actual region of interest. This is reflected in the results for the cropping level 1.33x1.33. In turn, for the highest cropping level, this brief mistake seems to be less annoying than no adaptation of the cropping area at all. Hence, the "position of the cropping area" of the automatically cropped version is slightly favoured to the position of the statically cropped area.

The manually cropped version significantly stands out with higher cropping level. Finally, the manual version was clearly preferred to the static version due to the "position of the cropping area".

For the highest cropping level, the assessments of the pillarbox version strongly scatter. The frequency distribution of attributes shows that this is caused by ratings against the pillarbox version because of the attribute "proportions". On the other hand, the "position of the cropping area" has mainly been chosen by the other part of subjects, which means that the cropping area of the static version does not properly fit.

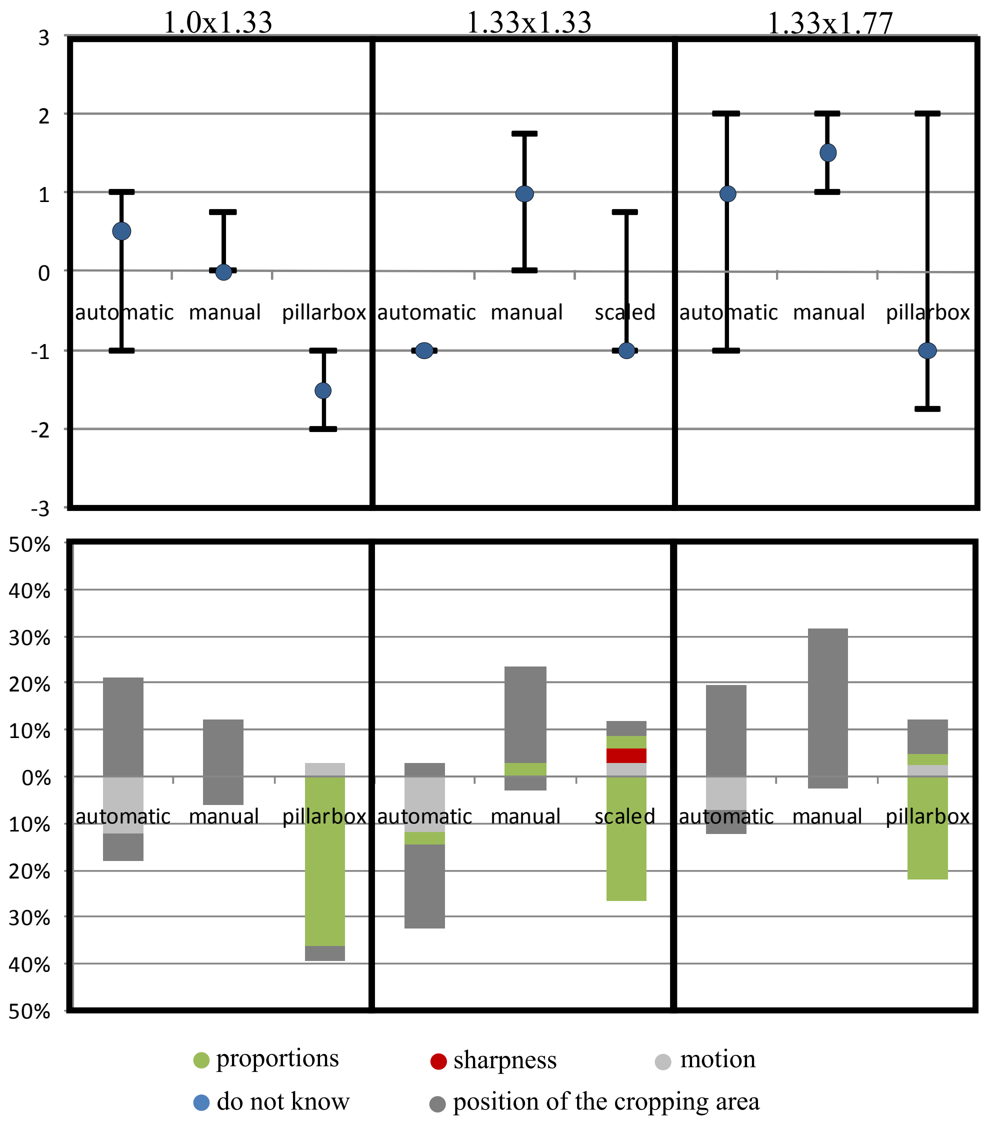


Figure 8-9: Median and quartiles for all cropping levels applied on the soccer 4:3 sequence (top) and corresponding frequency of attributes (bottom).

Table 8-9: Values of the one sided Wilcoxon test applied on soccer 4:3 sequence.

	1.0x1.33		**1.33x1.33**		**1.33x1.77**	
	$\hat{R}$	different	$\hat{R}$	different	$\hat{R}$	different
static-automatic	$\hat{R} = 42.5 \geqq 21 = R(13,0.05)$	no	$\hat{R} = 5 \leqq 17 = R(12,0.05)$	yes	$\hat{R} = 37 \geqq 21 = R(13,0.05)$	no
static-manual	$\hat{R} = 6 \geqq 2 = R(6,0.05)$	no	$\hat{R} = 3 \leqq 8 = R(9,0.05)$	yes	$\hat{R} = 10 \leqq 25 = R(14,0.05)$	yes
static-letterbox/ scaled	$\hat{R} = 4 \leqq 25 = R(14,0.05)$	yes	$\hat{R} = 32 \geqq 21 = R(13,0.05)$	no	$\hat{R} = 49.5 \geqq 25 = R(14,0.05)$	no

	1.0x1.33		1.33x1.33		1.33x1.77	
	$\boldsymbol{\hat{R}}$	different	$\boldsymbol{\hat{R}}$	different	$\boldsymbol{\hat{R}}$	different
automatic-manual	$\hat{R} = 33.5 \geqq 17 = R(12{,}0.05)$	no	$\hat{R} = 0 \leqq 10 = R(10{,}0.05)$	yes	$\hat{R} = 25 \geqq 21 = R(13{,}0.05)$	no
automatic-letterbox/scaled	$\hat{R} = 0 \leqq 10 = R(10{,}0.05)$	yes	$\hat{R} = 4.5 \geqq 2 = R(6{,}0.05)$	no	$\hat{R} = 2 = 2 = R(6{,}0.05)$	no
letterbox/scaled-manual	$\hat{R} = 0 \leqq 13 = R(11{,}0.05)$	yes	$\hat{R} = 14 \leqq 17 = R(12{,}0.05)$	yes	$\hat{R} = 15.5 \leqq 21 = R(13{,}0.05)$	yes

8.1.7.4 ***Show jumping 16:9***

For the show jumping example with an aspect ratio of 16:9, the adapted versions have not been preferred most of the time. Just at the highest cropping level, the manually and automatically prepared versions show better results than the static version. According to the Wilcoxon test, all three versions compared to the statically cropped version were equally assessed at this level.

It has to be mentioned that these results not really give an answer to the question whether there is a demand for cropping or not. As the subjects were asked to compare in relation to the static version, it cannot be found out if no cropping or cropping has been preferred at the highest cropping level of this example. It can just be stated that all versions have been favoured in relation to the statically cropped version due to the "position of the cropping area", respectively "proportions".

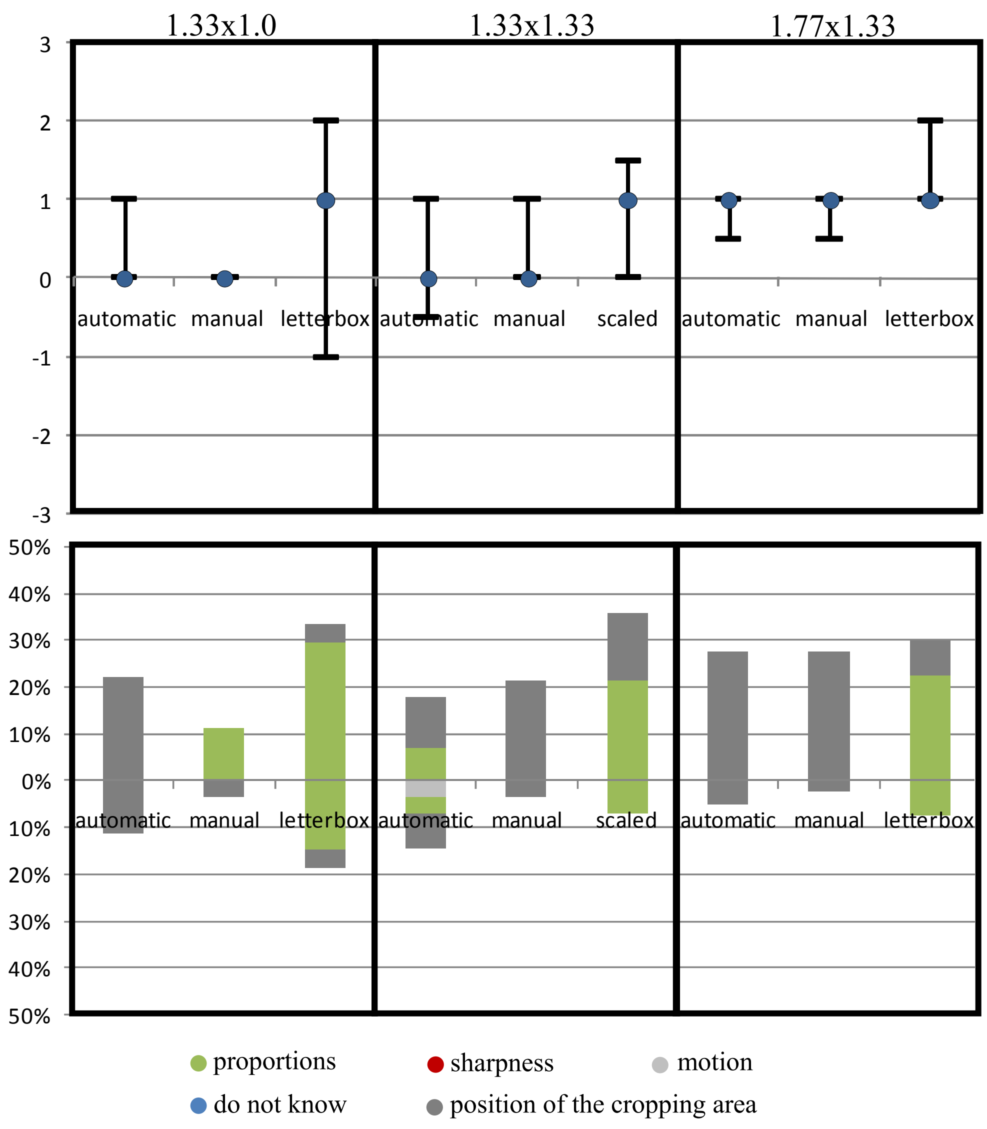


Figure 8-10: Median and quartiles for all cropping levels applied on the sequence show jumping 16:9 (top) and corresponding frequency of attributes (bottom).

Table 8-10: Values of the one sided Wilcoxon test applied on sequence show jumping 16:9.

	1.0x1.33		**1.33x1.33**		**1.33x1.77**	
	$\hat{R}$	different	$\hat{R}$	different	$\hat{R}$	different
static-automatic	$\hat{R} = 15 \geqq 8 = R(9,0.05)$	no	$\hat{R} = 20.5 \geqq 8 = R(9,0.05)$	no	$\hat{R} = 11 \leqq 21 = R(13,0.05)$	yes
static-manual	$\hat{R} = 1.5 \geqq 0 = R(1,0.05)$	no	$\hat{R} = 3.5 \geqq 3 = R(7,0.05)$	no	$\hat{R} = 5 \leqq 17 = R(12,0.05)$	yes
static-letterbox/scaled	$\hat{R} = 37.5 \geqq 30 = R(15,0.05)$	no	$\hat{R} = 9 \leqq 17 = R(12,0.05)$	yes	$\hat{R} = 26 \leqq 30 = R(15,0.05)$	yes
automatic-manual	$\hat{R} = 15 \geqq 8 = R(9,0.05)$	no	$\hat{R} = 23 \geqq 13 = R(11,0.05)$	no	$\hat{R} = 12.5 \geqq 3 = R(7,0.05)$	no

	1.0x1.33		1.33x1.33		1.33x1.77	
	$\hat{R}$	different	$\hat{R}$	different	$\hat{R}$	different
automatic-letterbox/ scaled	$\hat{R} = 39.5 \geqq 25 = R(14,0.05)$	no	$\hat{R} = 14.5 \leqq 17 = R(12,0.05)$	yes	$\hat{R} = 34 \geqq 17 = R(12,0.05)$	no
letterbox/ scaled-manual	$\hat{R} = 32 \geqq 25 = R(14,0.05)$	no	$\hat{R} = 20 \geqq 13 = R(11,0.05)$	no	$\hat{R} = 39 \geqq 21 = R(13,0.05)$	no

8.1.7.5 *Show jumping 4:3*

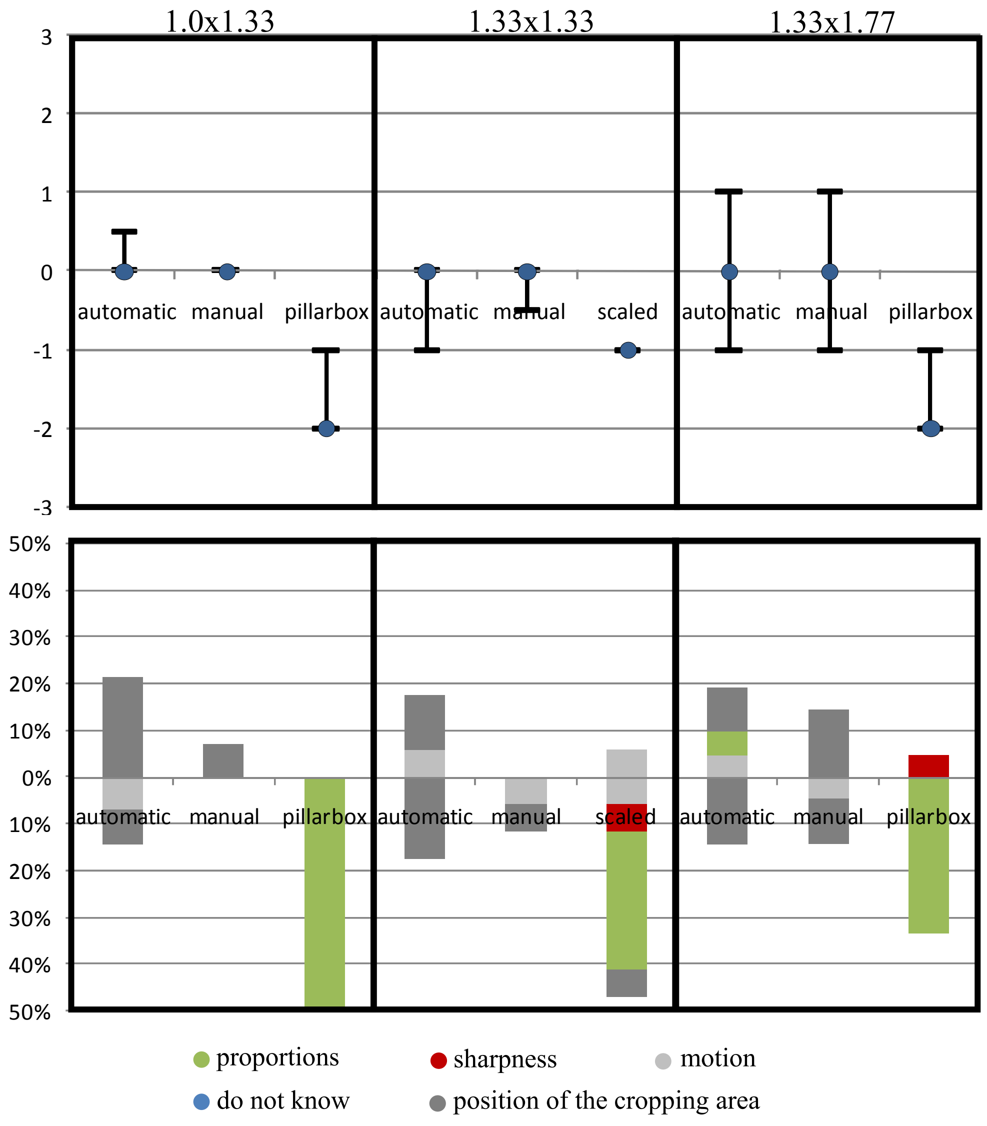


Figure 8-11: Median and quartiles for all cropping levels applied on the sequence show jumping 4:3 (top) and corresponding frequency of attributes (bottom).

Compared to the previous example of show jumping, this sequence interestingly shows nearly the opposite results especially for the non-cropped version. Obviously, this seems to be related to the aspect ratio of the video sequence. As the source material is 4:3, most cropped versions result in an aspect ratio of 16:9. Taking the previous show jumping example into account, it can be said that there seem to be a preference of the aspect ratio 16:9 for show jumping. The high number of the attribute "proportion" confirms this assumption that aspect ratio might be the reason for this result. In turn, all cropping versions do not significantly differ according to the Wilcoxon test.

Table 8-11: Values of the one sided Wilcoxon test applied on sequence show jumping 4:3.

	1.0x1.33		**1.33x1.33**		**1.33x1.77**	
	$\hat{R}$	different	$\hat{R}$	different	$\hat{R}$	different
static-automatic	$\hat{R} = 12 \geqq 3 = R(7,0.05)$	no	$\hat{R} = 13.5 \geqq 5 = R(8,0.05)$	no	$\hat{R} = 30 \geqq 17 = R(12,0.05)$	no
static-manual	$\hat{R} = 0 = 0 = R(1,0.05)$	no	$\hat{R} = 0 = 0 = R(4,0.05)$	no	$\hat{R} = 30.5 \geqq 13 = R(11,0.05)$	no
static-letterbox/scaled	$\hat{R} = 0 \leqq 30 = R(15,0.05)$	yes	$\hat{R} = 13 \leqq 30 = R(15,0.05)$	yes	$\hat{R} = 8.5 \leqq 25 = R(14,0.05)$	yes
automatic-manual	$\hat{R} = 10.5 \geqq 2 = R(6,0.05)$	no	$\hat{R} = 27.5 \geqq 13 = R(11,0.05)$	no	$\hat{R} = 35.5 \geqq 17 = R(12,0.05)$	no
automatic-letterbox/scaled	$\hat{R} = 0 \leqq 25 = R(14,0.05)$	yes	$\hat{R} = 0 \leqq 8 = R(9,0.05)$	yes	$\hat{R} = 3.5 \leqq 21 = R(13,0.05)$	yes
letterbox/scaled-manual	$\hat{R} = 0 \leqq 30 = R(15,0.05)$	yes	$\hat{R} = 17.5 \leqq 21 = R(13,0.05)$	yes	$\hat{R} = 7 \leqq 21 = R(13,0.05)$	yes

8.1.7.6 *Skiing 16:9*

This example show strong scattering results. The main reason is possibly a shaking camera at the end of the sequence. Nevertheless, the results show similar trends as the previous examples. The higher the cropping level, the more the non-adapted cropping area is rated worse compared to the other versions. Again, the main attributes used are the “position of the cropping area” and “proportions”.

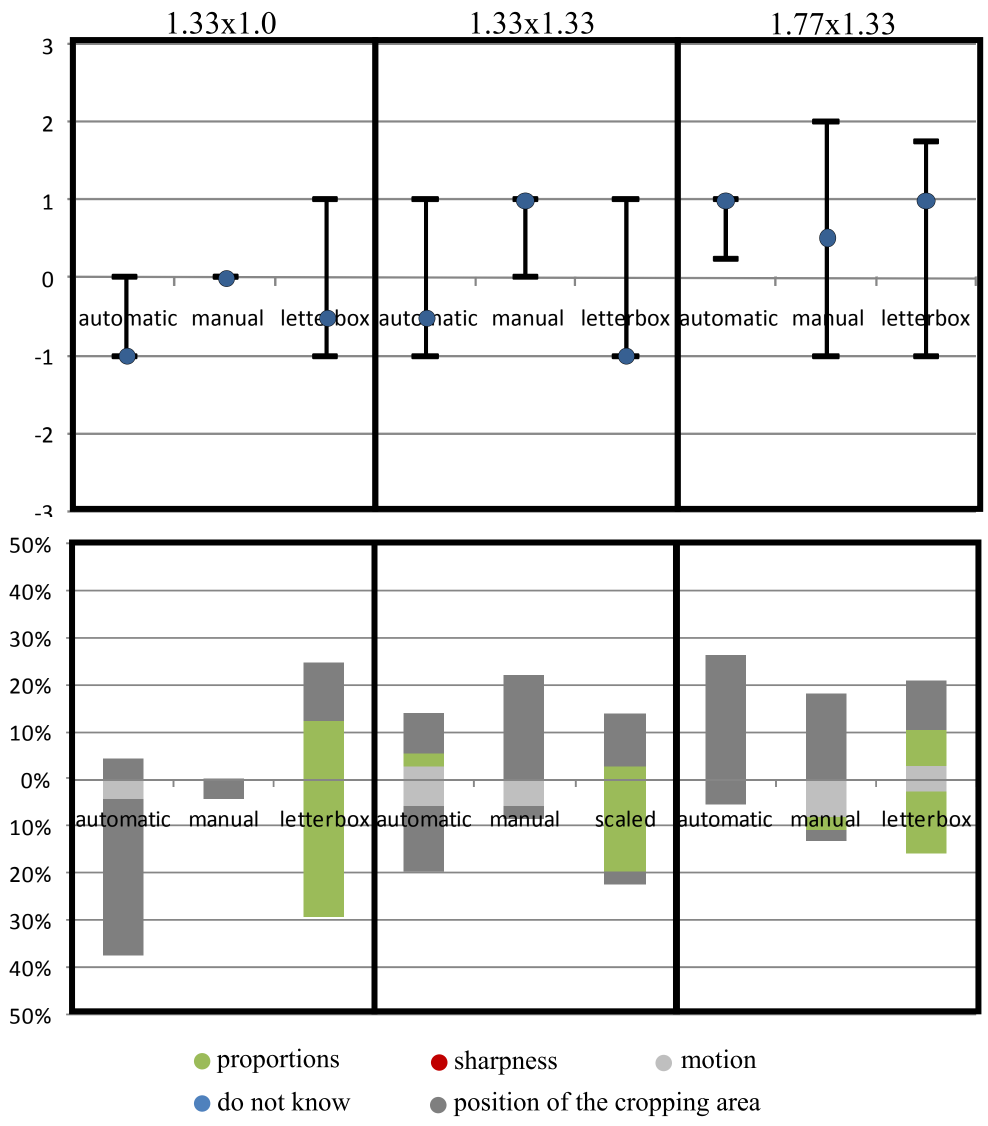


Figure 8-12: Median and quartiles for all cropping levels applied on the skiing 16:9 sequence (top) and corresponding frequency of attributes (bottom).

Table 8-12: Values of the one sided Wilcoxon test applied on the skiing sequence.

	1.33x1.0		1.33x1.33		1.77x1.33	
	$\hat{R}$	different	$\hat{R}$	different	$\hat{R}$	different
static-automatic	$\hat{R} = 5 \leqq 10 = R(10,0.05)$	yes	$\hat{R} = 33.5 \geqq 17 = R(12,0.05)$	no	$\hat{R} = 12 \leqq 17 = R(12,0.05)$	yes
static-manual	$\hat{R} = 0 = 0 = R(1,0.05)$	no	$\hat{R} = 17 \geqq 13 = R(11,0.05)$	no	$\hat{R} = 31 \geqq 17 = R(12,0.05)$	no
static-letterbox/ scaled	$\hat{R} = 41.5 \geqq 21 = R(13,0.05)$	no	$\hat{R} = 44 \geqq 21 = R(13,0.05)$	no	$\hat{R} = 41 \leqq 25 = R(14,0.05)$	no
automatic-manual	$\hat{R} = 9.5 \leqq 10 = R(10,0.05)$	yes	$\hat{R} = 8 \geqq 5 = R(8,0.05)$	no	$\hat{R} = 26 \geqq 13 = R(11,0.05)$	no
automatic-letterbox/ scaled	$\hat{R} = 13 \geqq 10 = R(10,0.05)$	no	$\hat{R} = 24.5 \geqq 10 = R(10,0.05)$	yes	$\hat{R} = 20 \geqq 10 = R(10,0.05)$	no
letterbox/ scaled-manual	$\hat{R} = 33 \geqq 17 = R(12,0.05)$	no	$\hat{R} = 18.5 \geqq 13 = R(11,0.05)$	no	$\hat{R} = 33 \geqq 13 = R(11,0.05)$	no

8.1.7.7 *Combined results*

As a summary of all individual results, this section combines all examples into one representation. 4:3 and 16:9 are presented separately as the previous evaluations have shown quite different results for both aspect ratios.

16:9 content

For cropping level 1.33x1.0, the letterbox version tends to be slightly worse than the statically cropped version mainly due to the attribute "proportions". This difference decreases with higher cropping levels. For the highest cropping level, the letterbox version is slightly better than the statically cropped version, which is mainly due to the "position of the cropping area". This suggests that the static cropping area seems no longer to enclose the most relevant areas of the sequences.

According to the Wilcoxon test, there is a slight preference for automatically and statically cropped versions for all cropping levels (see Figure 8-13). Additionally, automatically and manually prepared versions show very similar trends. For the highest cropping level, this positive tendency becomes significant. It shows that the subjects were more satisfied with the adapted cropping versions because of the "position of the cropping area". This confirms the fact that with higher cropping levels a statically cropped version is no longer sufficient most of the time.

This evaluation does not directly give an answer to the question of whether there is a demand for cropping. As the subjects were asked to compare the sequences in relation to the static version, it is not answered if adapted versions or the non-cropped version were preferred. It can only be stated that all versions have been favoured in relation to the statically cropped version due to the attributes "position of the cropping area" and "proportions".

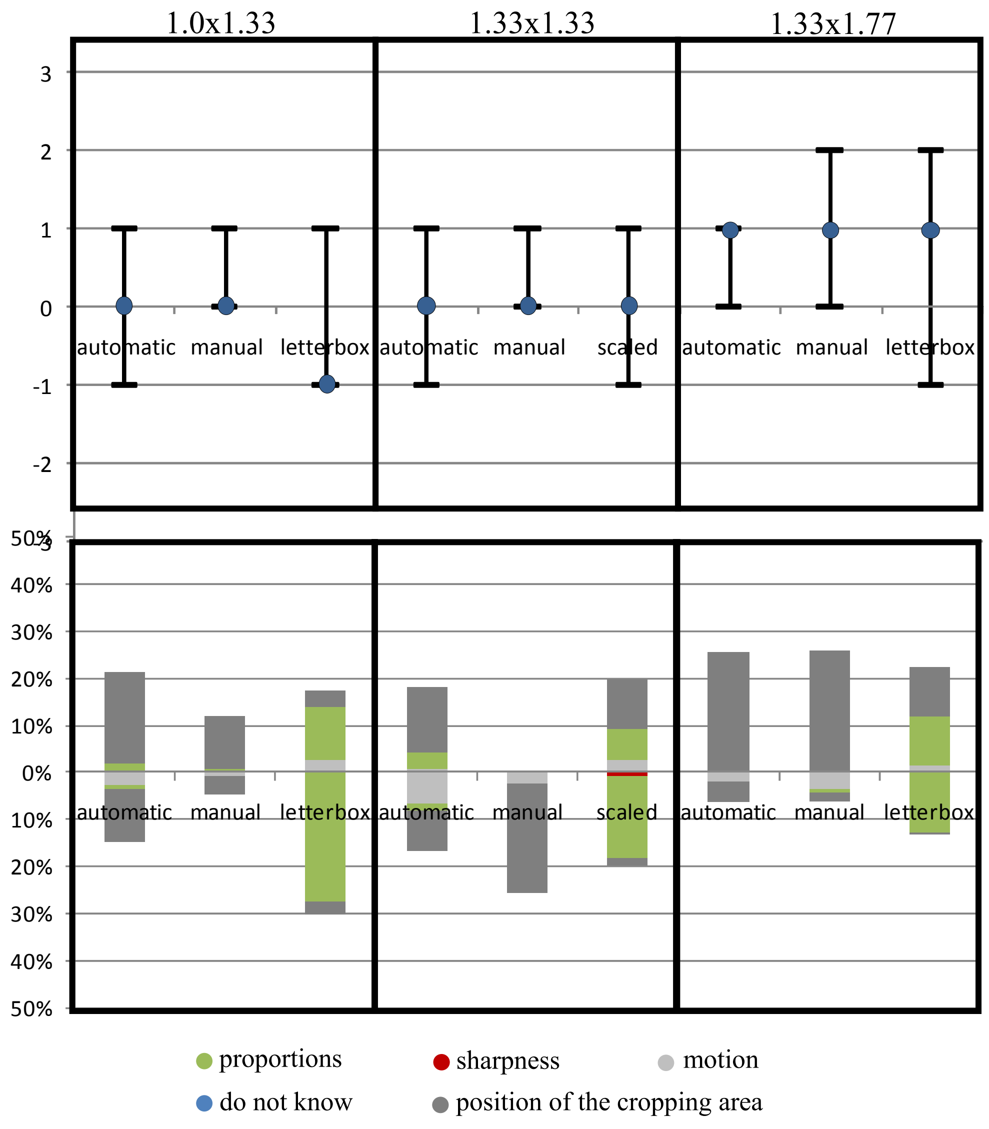


Figure 8-13: Median and quartiles for all cropping levels applied on all 16:9 sequences (top) and corresponding frequency of attributes (bottom).

Table 8-13: Values of the one-sided Wilcoxon test applied on 16/9 sequences.

	1.33x1.0		**1.33x1.33**		**1.77x1.33**	
	$\hat{R}$	different	$\hat{R}$	different	$\hat{R}$	different
static-automatic	$\hat{R} = 322 \leqq 336 = R(43,0.05)$	yes	$\hat{R} = 447 \geqq 407 = R(47,0.05)$	no	$\hat{R} = 225.5 \leqq 486 = R(51,0.05)$	yes
static-manual	$\hat{R} = 56.5 \leqq 83 = R(23,0.05)$	yes	$\hat{R} = 148.5 \leqq 241 = R(37,0.05)$	yes	$\hat{R} = 208.5 \leqq 507 = R(52,0.05)$	yes
static-pillarbox/ scaled	$\hat{R} = 694 \geqq 642 = R(58,0.05)$	no	$\hat{R} = 631 \geqq 550 = R(54,0.05)$	no	$\hat{R} = 602 \leqq 618 = R(57,0.05)$	yes
automatic-manual	$\hat{R} = 297.5 \geqq 200 = R(34,0.05)$	no	$\hat{R} = 418 \leqq 426 = R(48,0.05)$	yes	$\hat{R} = 389.5 \geqq 389 = R(46,0.05)$	no
automatic-pillarbox/ scaled	$\hat{R} = 441 \leqq 486 = R(51,0.05)$	yes	$\hat{R} = 525.5 \geqq 389 = R(46,0.05)$	no	$\hat{R} = 492 \geqq 446 = R(49,0.05)$	no
pillarbox/ scaled-manual	$\hat{R} = 476 \leqq 550 = R(54,0.05)$	yes	$\hat{R} = 418 \leqq 426 = R(48,0.05)$	yes	$\hat{R} = 398.5 \leqq 426 = R(48,0.05)$	yes

4:3 content

For all cropping levels, the pillarbox respectively scaled version has been assessed as significantly worse than the statically cropped version. Obviously, this is due to the attribute “proportions”. Compared to the results for 16:9 content, subjects seem to be more satisfied with the statically cropped version even at the highest cropping level.

The automatically and manually adapted versions show similar trends as for the 16:9 content. Especially for the highest cropping level, both versions tend to be slightly better than the statically cropped version. Again, this can be confirmed by the attribute “position of the cropping area”.

For the cropping level 1.33x1.33, the automatically cropped version tends to be slightly worse. Obviously, this results from the soccer example already discussed in Section 8.1.7.3, where the algorithm made a false detection for a short time.

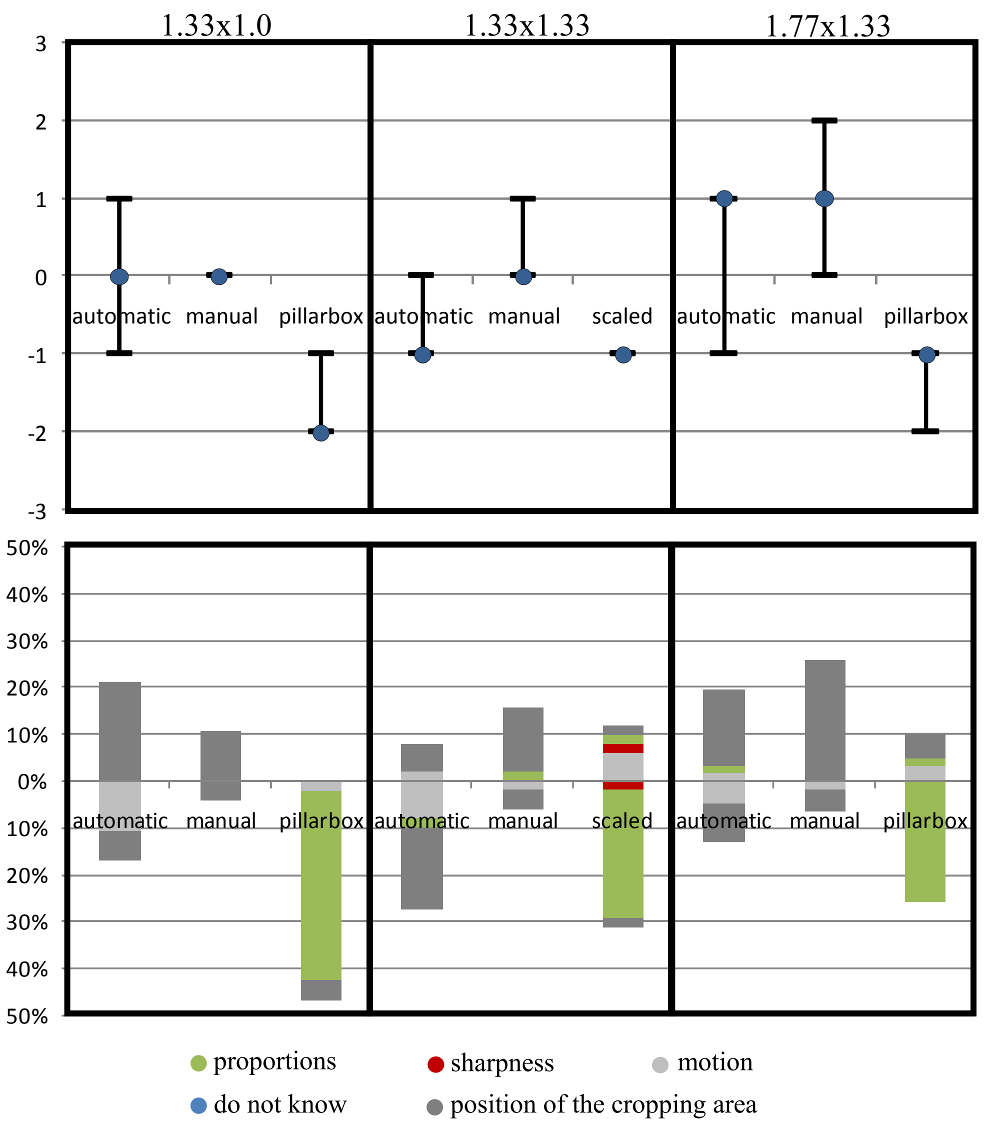


Figure 8-14: Median and quartiles for all cropping levels applied on all 4:3 sequences (top) and corresponding frequency of attributes (bottom).

Table 8-14: Values of the one sided Wilcoxon test applied on 4/3 sequences.

	1.0x1.33		**1.33x1.33**		**1.33x1.77**	
	$\hat{\boldsymbol{R}}$	different	$\hat{\boldsymbol{R}}$	different	$\hat{\boldsymbol{R}}$	different
static-automatic	$\hat{R} = 95.5 \geqq 60 = R(20{,}0.05)$	no	$\hat{R} = 36 \leqq 60 = R(20{,}0.05)$	yes	$\hat{R} = 125.5 \geqq 100 = R(25{,}0.05)$	no
static-manual	$\hat{R} = 7 \geqq 3 = R(7{,}0.05)$	no	$\hat{R} = 66 \geqq 21 = R(13{,}0.05)$	no	$\hat{R} = 72 \leqq 100 = R(25{,}0.05)$	no
static-pillarbox/ scaled	$\hat{R} = 7.5 \leqq 140 = R(29{,}0.05)$	yes	$\hat{R} = 93 \leqq 130 = R(28{,}0.05)$	yes	$\hat{R} = 111.5 \leqq 130 = R(28{,}0.05)$	yes

	1.0x1.33		1.33x1.33		1.33x1.77	
	$\hat{R}$	different	$\hat{R}$	different	$\hat{R}$	different
automatic-manual	$\hat{R} = 77 \geqq 47 = R(18{,}0.05)$	no	$\hat{R} = 50 \leqq 67 = R(21{,}0.05)$	yes	$\hat{R} = 129.5 \geqq 100 = R(25{,}0.05)$	yes
automatic-pillarbox/ scaled	$\hat{R} = 0 \leqq 91 = R(24{,}0.05)$	yes	$\hat{R} = 48 \geqq 30 = R(15{,}0.05)$	no	$\hat{R} = 10 \leqq 47 = R(18{,}0.05)$	yes
pillarbox/ scaled-manual	$\hat{R} = 0 \leqq 110 = R(26{,}0.05)$	yes	$\hat{R} = 66 \leqq 100 = R(25{,}0.05)$	no	$\hat{R} = 38.5 \leqq 110 = R(26{,}0.05)$	yes

8.2 Gaze Tracking

Within this section, results of gaze tracking measurements are presented. For this, all the video examples which were chosen for the subjective evaluation have been presented again to ten of the fifteen subjects. The estimated gaze tracking data has been compared to the ROI positions determined by the proposed system. Besides this, the algorithm was launched with and without optimised parameter settings to demonstrate the advantage of genre information fed to the system.

8.2.1 Gaze measurement set-up

The gaze positions of subjects have been measured with the aid of the open source gaze tracking software from the IT University of Copenhagen (ITU Gaze Tracker, 2010). For the set-up, two infrared webcams were deployed. One was attached to a specially prepared chair to film the eye of the subject (see Figure 8-15). The other served exclusively as an infrared light source which was positioned on the table in front of the subject. The infrared LEDs of the camera fixed at the chair were turned off. Infrared light has been used for a clearer distinction between the pupil and other parts of the face.

Additionally, the reflection of the LED in the eye has been taken into account (making use of this reflection is called “glint mode” in the ITU-software) to estimate a geometric relationship between LED, eye reflection and monitor. To calibrate the parameters for this relationship, moving dots were presented on the screen to the subject and the subject was asked to blink as little as possible as well as not to move his head during and after calibration. Once calibrated, all sequences were presented to the subjects successively.

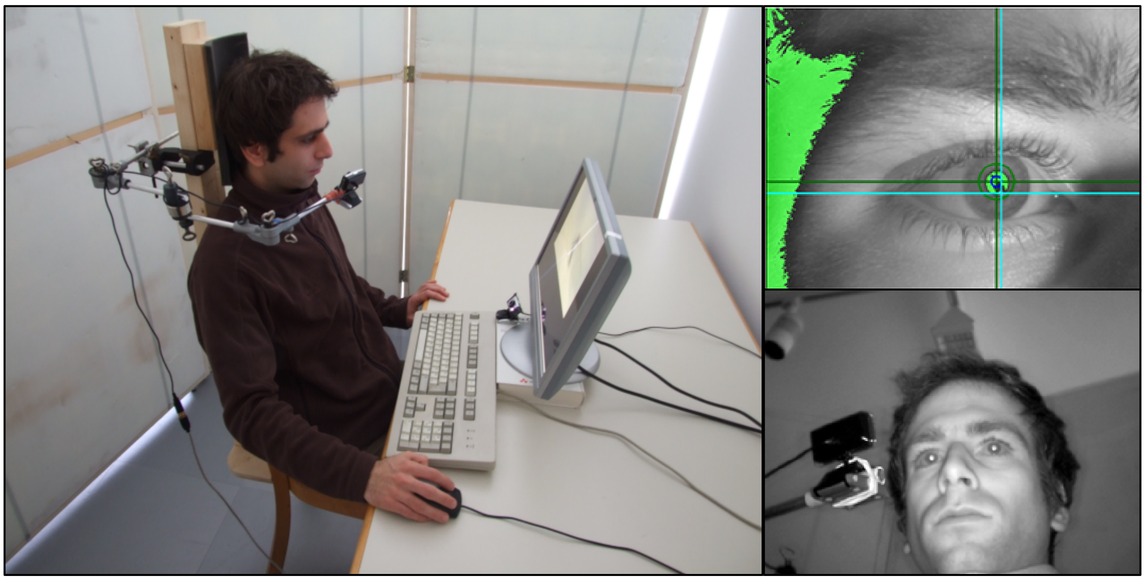


Figure 8-15: Set-up for gaze measurement. The camera fixed on the chair was used for glint detection (cyan cross hairs, top right) and pupil detection (green cross hairs, top right). The infrared light projected on the subject was emitted by the camera on the table (bottom right). This camera was intended merely to provide a point light source by its LEDs rather than to film the scene.

8.2.2 Determination of gaze to ROI deviation

In addition to the ITU gaze tracker, the set-up was extended by a self-programmed media player which renders videos and synchronises the gaze position data received by the ITU software via UDP. The player converted the measured screen position to the corresponding gaze position in the video and logged this data to a simple text file containing frame number and pixel position in the video.

After measuring the gaze position, the player illustrated the logged data in the form of a red dot (see Figure 8-16).

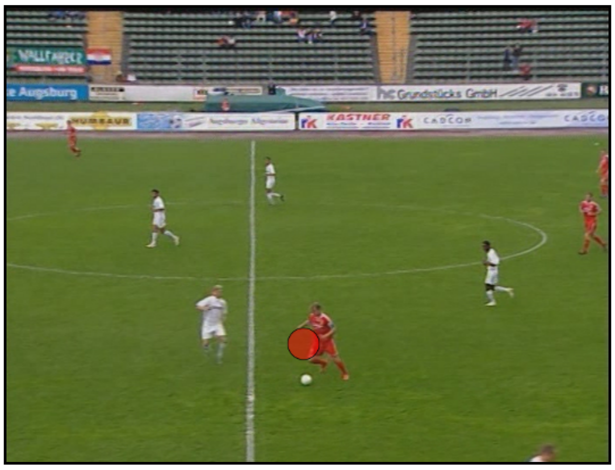


Figure 8-16: Illustration of measured gaze position with the use of a red dot.

In an offline process, the Euclidean distances between measured gaze positions and estimated ROI positions have been computed for each video frame. The ROI positions were extracted by the proposed system in the same way as for the estimation of cropping areas for the subjective evaluation. As the gaze tracker sends approximately four position coordinates per second, the median of the Euclidean distances was calculated over a period of 20 video frames.

8.2.3 Scatter plots

The measured data for all types of sports are combined into a single scatter plot for 16:9 and 4:3 content. It should be mentioned that gaze tracking delivers only approximate position data and is still quite error-prone. For this, the scatter plots serve to give an impression of the system’s reliability rather than to supply accurate measurement data.

The coordinate system for the scatter plots is centred, so that the closer the plotted data points are to the coordinate origin, the smaller are the Euclidean distances between the gaze position and ROI position.

The outer rectangle drawn in the coordinate system represents the original resolution of the video and the inner rectangle corresponds to the range of the highest zoom level (1.77x1.33, respectively 1.33x1.77) applied for the subjective evaluation. The dashed lines symbolise the corresponding pan & scan mask.

8.2.3.1 *Scatter plots for 16:9 content*

The presented 16:9 content contained the examples of soccer, ice hockey, show jumping and skiing. The parameter settings for ROI extraction were the optimised ones for each content as well as the default setting for the system. Looking at the results, both measurements show an accumulation around the coordinate centre (see Figure 8-17). In addition, the scatter plot for optimised parameters (top) clearly shows less scattering and a promoting of concentration around the origin. The increased spread scattering within the plot for default parameters (bottom) is mainly due to sports with multiple objects, as these are the more critical ones in defining ROIs.

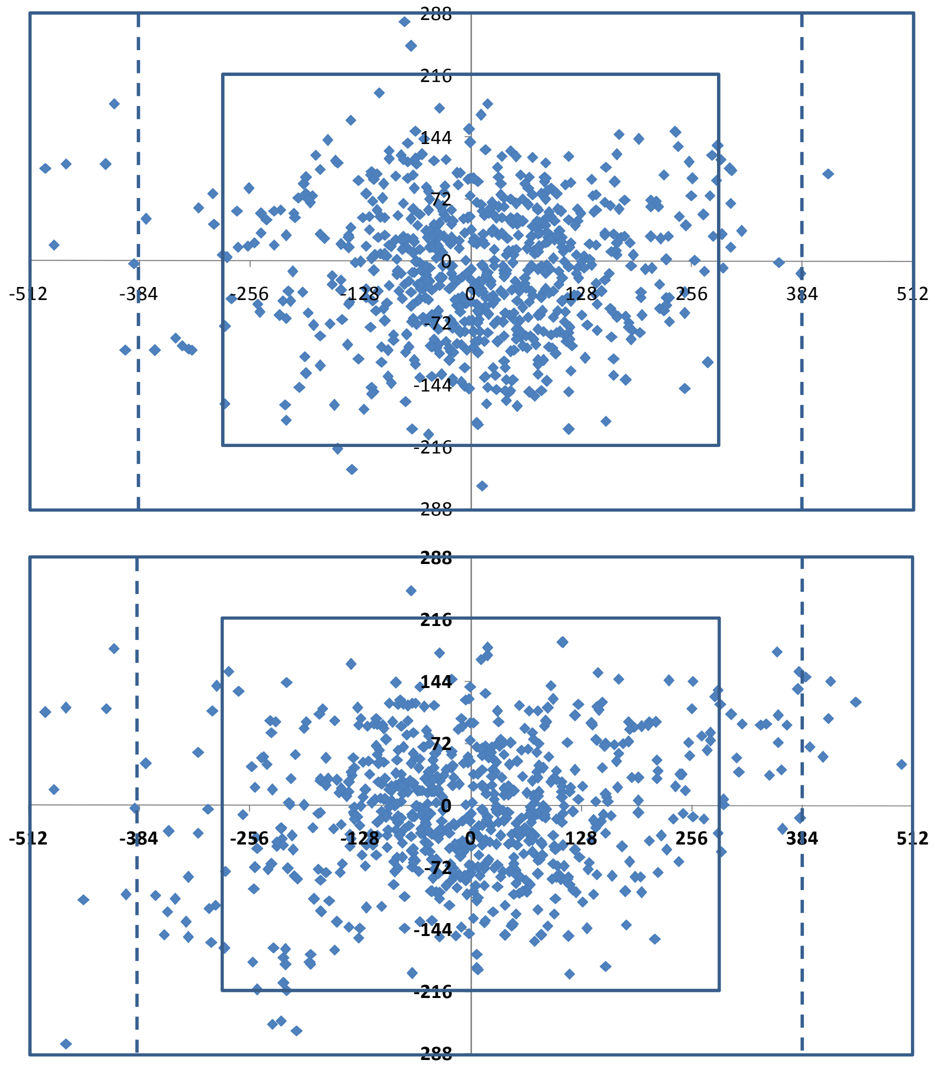


Figure 8-17: Scatter plot of deviation between measured gaze position and extracted ROI position with optimised parameters (top) and default parameters (bottom). The inner rectangle corresponds to the range of the highest zoom level and the dashed lines depict the pan & scan range for 16:9 content.

8.2.3.2 *Scatter plots for 4:3 content*

The 4:3 content presented to the subjects consisted of the soccer and show jumping examples. Again, the default parameters caused a much higher scattering which is the result of detecting false ROI positions. The scatter plot for optimised parameters shows that deviations are mainly accumulated within the highest zoom level range. This indicates that the algorithm is almost always pointing to the regions of interest, which could be identified by the individual viewer.

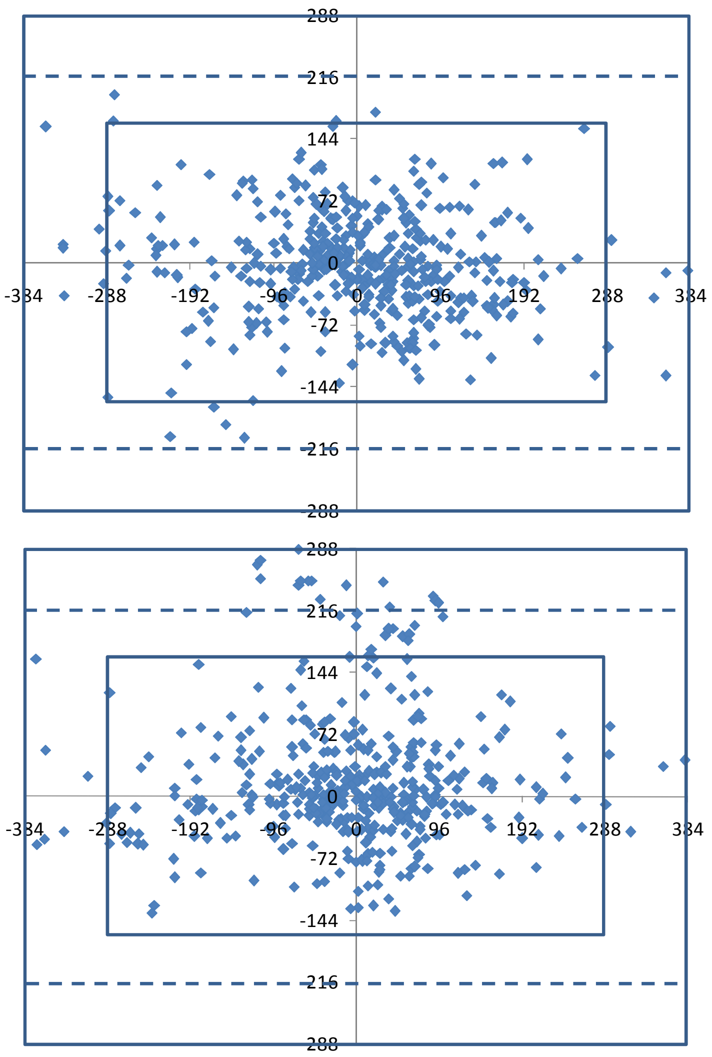


Figure 8-18: Scatter plot of deviation between measured gaze position and extracted ROI position with optimised parameters (top) and default parameters (bottom). The inner rectangle corresponds to the range of the highest zoom level and the dashed lines depict the pan & scan range for 4:3 content.

9. Conclusion

9.1 Summary

The presented system follows a new approach to optimising the extraction of regions of interest (ROIs) from video content. The main concept is to exploit production metadata and image compositions to use this information as prior knowledge for the extraction of ROIs. The ROI candidates are filtered by several processes on different levels to estimate contextually important regions. Based on the computed weights for each ROI, the system is able to define a final cropping area that encloses as much important image information as possible. The application can be adapted to any type of content or it can run in a default mode. For the purposes of this work, the detection of cropping areas has been specified for different types of sports content.

The system architecture is based on the concept of a plug-in system. The plug-ins represent components which either detect ROIs by low level feature extraction (extraction plug-ins) or interpret the contextual importance of extracted ROIs on a higher level. The extraction plug-ins that have been implemented are the Visual Attention Plug-In and the Backprojection Plug-In. The former relies on assumptions motivated by visual attention models which mainly detect bottom-up features in a visual scene that attract a viewer's attention. The plug-in combines the visual attention system by Hou (Hou & Zhang, Saliency Detection: A Spectral Residual Approach, 2007) and a method that detects object motion by compensating for camera movements in image sequences. The Backprojection Plug-In has been developed for specific types of content for which it is known that objects are moving on a plain background, as happens, for example, in a soccer game. By pre-loading a rough colour histogram of the expected background, the extraction is guided by valuable top-down information. The plug-in architecture allows for the flexible extension of the system by additional plug-ins.

The amount of production metadata that is fed into the application has been kept low in order to ensure that the required effort for content annotation is at a reasonable level. The exploited information includes the type of genre, which can be easily annotated by an editor on site or in advance of a production. Additionally, shot boundary information is assumed to be available in the descriptive data which can, for example, be estimated by recording the switch between cameras that are on air.

The proposed application has been used for a subjective evaluation including sequences from four different types of sports, of which two were available in two different aspect ratios (16:9 and 4:3). The aim of the test was to examine the reliability of the system as well as to compare different types of cropping approaches for content viewed on small displays. Additionally, three different cropping levels were used to assess the acceptance of each cropping level applied to SDTV production material. For this, statically cropped, manually cropped, and non-cropped versions as well as a version automatically cropped by the system have been compared and assessed by 15 subjects. For each assessment, the subject had to specify the attribute which justified his decision. Results of the subjective evaluation have shown that with higher cropping levels, statically cropped versions are less satisfying than those with adaptive cropping masks (manually and automatically cropped versions). Despite a few rare cases, the automatically cropped version was perceived to be worse than the manually cropped version.

In addition to the subjective evaluation, a gaze tracking has been carried out to compare a subject's line of vision with the positions of ROIs extracted by the proposed system. Results of this test should demonstrate the reliability of the system, independently of cropping. For this, ten subjects had to watch sports material in a normal way while their line of vision was logged. The deviations of each gaze position from the ROIs extracted by the system were computed and visualised in scatter plots. Extracted ROIs were determined by optimised plug-in parameters and additionally by default parameters. The scatter plots show that the system almost always points at the region which was the focus of the viewer's attention.

With HDTV penetrating the market quickly, a wider range of display resolutions needs to be considered for broadcast productions. This means that the required cropping ratio increases, which also has an effect on the amount of work for a video editor compared to SDTV productions. This indicates the necessity of a system which identifies possible regions of interest automatically. In turn, the system does not have the complete contextual knowledge and hence the selection of the specific cropping area should still be in the hands of the video editor. Therefore, the proposed solution tackling the problem of automatic cropping and scaling should be seen as a supporting tool for video editors, for example by suggesting possible cropping areas. Such a solution can provide an improvement of the production work flow.

9.2 Strengths and Limitations

The detection of contextual importance in TV sports productions by the proposed system is highly motivated by the image compositions of cameramen and contextual saliency. As this system is less biologically motivated, it concentrates and focuses on features in videos which are caused by typical TV productions. The system architecture has been chosen in such a way that extensions and modifications can be made anytime and as required. The possibility of mixing several extraction modules and weighting their results with subsequent classification provides a high adaptability to specific content. Clear tendencies in results from the subjective evaluation and the gaze tracking prove the success of this approach. Nevertheless, more extraction modules would be desirable for a more flexible analysis of different content, which will be discussed in the next section.

Although the results show quite high system accuracy, human intervention is still necessary for TV productions. Only the combination of human and machine provides a fast and reliable solution. While the application might identify important regions faster than humans, it can still make wrong decisions.

As mentioned previously, the evaluation did not directly answer the question of the necessity of cropping at all. Within this work, only the cropping of the system has been compared to other types of cropping. Additionally, HDTV content has not been taken into account for the evaluation as those productions are still rare. Nevertheless, it would be desirable to answer questions about the system’s reliability on HDTV content in subsequent investigations.

9.3 Future Work

As already mentioned, some more extraction modules are desirable which were not implemented within this work. For example, object tracking could deliver important top-down information in addition to the Visual Attention Plug-In. Especially in the case of content with a single object, this method could be easily initialised by the weight values of ROIs. On the other hand, for content with multiple objects, object tracking might get out of control with an increasing number of objects. In such cases, additional information, for example by the Backprojection Plug-In, might be preferable.

Other high level information could be gained by the detection of faces in video images. The presence of clearly identified faces can deliver important contextual information.

To reduce abrupt motions of the cropping mask, a smoothing by splines instead of simple linear interpolation is quite promising. It gives a more natural impression of motion and avoids fast changes of movement directions. This complaint can be observed in the results of the subjective evaluation as some subjects have devalued the automatically cropped version due to the attribute "motion" (cf. results of ice hockey, soccer and skiing sequences).

10. References

Arthur, C. (2007, August 2). *Television is a turnoff for mobile users*. Retrieved June 26, 2010, from The Guardian: http://www.guardian.co.uk/technology/2007/aug/02/guardianweeklytechnologysection.mobilephones

Avidan, S., & Shamir, A. (2007). Seam carving for content-aware image resizing. *ACM Transactions on Graphics , 26*.

Balakhnin, A. (2009). *Yadif deinterlace algorithm*. Retrieved October 10, 2009, from http://avisynth.org.ru/yadif/yadif.html

Bernhardt-Walther, D. (2010). *Saliency Toolbox*. Retrieved January 20, 2010, from http://www.saliencytoolbox.net/index.html

BMF – Broadcast Metadata exchange Format. (2007). *01.00.00*. Munich: Institut fuer Rundfunktechnik.

Bovik, A. C. (2009). *The Essential Guide to Video Processing.* San Diego: Elsevier.

Brigham, E. O. (1988). *Fast Fourier Transform and Its Applications.* Prentice-Hall.

Burger, W., & Burge, M. J. (2008). *Digital Image Processing - An Algorithm Introduction Using Java* (1 ed.). New York: Springer Science and Business Media.

Chen, L., Xie, X., Fan, X., Ma, W.-Y., Zhang, H.-J., & Zhou, H. (2002). A visual attention model for adapting images on small displays. (M. Research, Ed.) 353 - 364.

Chen, L., Xie, X., Fan, X., Ma, W.-Y., Zhang, H.-J., & Zhou, H. (2003). Visual attention based image browsing on mobile devices. *Proceedings of the 2003 International Conference on Multimedia and Expo , 2*, pp. 53 - 56.

Cheng, W.-H., Chu, W.-T., & Wu, J.-L. (2005). A Visual Attention Based Region-of-Interest Determination Framework for Video Sequences. *E88-D*, 1578-1586.

Clements B., R. D. (1980). *Photographic Composition.* (V. N. Reinhold, Ed.) New York.

Conway. (2002). Neural mechanisms of color vision: double-opponent cells in the visual cortex. 12. (Springer, Ed.)

Cook, A. (2009). *Critical values for the Wilcoxon signed rank test.* Retrieved November 6, 2009, from National University of Singapoore: http://courses.nus.edu.sg/course/stacar/internet/st2238/handouts/signedrank_table.pdf

Corder, G. W., & Foreman, D. I. (2009). *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach.* Hoboken: John Wiley & Sons.

Dearden, A., Demiris, Y., & Grau, O. (2006). Tracking football player movement from a single moving camera using particle filters. *Proceedings of the 3rd European Conference on Visual Media Production* , pp. 29-37.

Deselaers, T., Dreuw, P., & Ney, H. (2008). Pan, Zoom, Scan – Time-Coherent, Trained Automatic Video Cropping. *IEEE Conference on Computer Vision and Pattern Recognition* .

Doczi, G. (1981). *The power of limits: proportional harmonies in nature, art, and architecture.* Boalder: Shambhala Publications.

EBU Technical Recommendation R95-2000 . (2000). *Television Production for 16:9 Widescreen: Safe Areas* . Geneva: European Broadcasting Union.

Erhardt, A. (2008). *Einführung in die Digitale Bildverarbeitung - Grundlagen, Systeme und Anwendungen.* Wiesbaden: Vieweg+Teubner.

ETSI TS 102 005 V1.2.1. (2006). *Digital Video Broadcasting (DVB); Specification for the use of Video and Audio Coding in DVB services delivered directly over IP protocols* . Sophia Antipolis: Eurpean Telecommunication Standard Institute.

Fasulo, D. (1999). *An Analysis of Recent Work on Clustering Algorithms.* Seattle: University of Washington.

Feininger, A. (1965). *The Complete Photographer.* New Jersey: Prentice Hall.

FFmpeg. (2010). Retrieved August 19, 2009, from http://ffmpeg.org/

Figueroa, P., Leite, N., Barros, R., Cohen, I., & Medoini, G. (2004). Tracking soccer players using the graph representation. *Proceedings of the 17th International Conference on Pattern Recognition , 4*, pp. 787-790.

Fisher, R. (2007). *CVonline*. Retrieved August 3, 2009, from http://homepages.inf.ed.ac.uk/rbf/CVonline/unfolded.htm

Fitch, R. K. (2010). *WinSTAT*. (R. Fitch Software) Retrieved Janaury 20, 2010, from http://winstat.de/default.htm

Forsyth, D. A., & Ponce, J. (2003). *Computer Vision - A Modern Approach.* (Prentice-Hall, Ed.) New Jersey.

Frank, I. E., & Todeschini, R. (1994). *The data analysis book.* Amsterdam: Elsevier Science.

Freund, Y., & Schapire, R. E. (1999). Large Margin Classification Using the Perceptron Algorithm. In *Machine Learning* (Vol. 37, pp. 277-296). Springer.

Frintrop, S. (2006). VOCUS: A Visual Attention System for Object Detection and Goal-Directed Search. *Lecture Notes in Artificial Intelligence , 3899*.

Gonzales, R., & Woods, R. (2002). *Digital Image Processing.* New Jersey: Prentice-Hall.

Hartley, R., & Zisserman, A. (2008). *Multiple View Geometry in Computer Vision* (2 ed.). Cambridge: Cambridge University Press.

High Definition (HD) - Image Formats for Television Production. (2004). Geneva: European Broadcasting Union.

Hollander, M., & Wolfe, D. A. (1968). *Nonparametric satistical methods.* New York: Wiley.

Horn, B., & Schunck, B. (1981). Determining optical flow. *Artificial Intelligence , 17* , pp. 185-203.

Horton, I. (2008). *Ivor Horton's Beginning Visual C++ 2008.* Indianapolis: Wiley.

Hou, X. (2009). *Spectral Residual*. Retrieved September 1, 2009, from http://www.its.caltech.edu/~xhou/projects/spectralResidual/spectralresidual.html

Hou, X., & Zhang, L. (2007). Saliency Detection: A Spectral Residual Approach. *IEEE Conference on Computer Vision and Pattern Recognition* .

Itti L., K. C. (1998). A Model of Saliency-based Visual Attention for Rapid Scene Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence , 20*, pp. 1254-1259.

Itti, L. (2010). *iLab Neuromorphic Vision C++ Toolkit*. Retrieved December 29, 2009, from http://ilab.usc.edu/toolkit/

Itti, L. (2000). Models of Bottom-Up and Top-Down Visual Attention. California Institute of Technology.

ITU Gaze Tracker. (2010). Retrieved January 13, 2010, from http://www.gazegroup.org/downloads/23-gazetracker

Jack, K. (2001). *Video Demystified - A Handbook for the Digital Engineer.* Eagle Rock: LLH Technology.

Jacobson, R. (2000). *Information design.* MIT Press.

Jain, A., Murty, M., & Flynn, P. (1999). Data Clustering: A Review. *ACM Computing Surveys* , pp. 264-323.

Jumisko-Pyykkö, S., & Strohmeier, D. (2008). *Report on research methodologies for the experiments.* Retrieved November 4, 2009, from Mobile 3DTV: http://sp.cs.tut.fi/mobile3dtv/results/tech/D4.2_Mobile3dtv_v2.0.pdf

Kalman, D. (1996). A Singularly Valuable Decomposition: The SVD of a Matrix. *The College Mathematics Journal , 27* (1).

Kerr, D. A. (2009). *Chrominance Subsampling in Digital Images.* Retrieved January 23, 2010, from http://dougkerr.net/Pumpkin/articles/Subsampling.pdf

Knee, M. (2008). Aspect Processing: The Shape of Things to Come by Mike Knee of Snell & Wilcox. *International Broadcast Conference* .

Koch, C., & Poggio, T. (1999). Predicting the visual world: silence is golden. *Nature Neuroscience , 2* (1).

Kopf, S., Haenselmann, T., Farin, D., & Effelsberg, W. (2004). Automatic Generation of Summaries for the Web. *Storage and Retrieval Methods and Applications for Multimedia* .

Kotz, S., & Johnson, N. L. (1988). *Encyclopedia of stastical sciences* (Vol. 9). New York: John Wiley & Sons.

Kozamernik, F., Sunna, P., Wyckens, E., & Pettersen, D. I. (2005). *Subjective Quality of Internet Video Codecs - Phase 2 Evaluations using SAMVIQ.* Geneva: European Boadcast Union.

Laurent, I. (2000). Models of Bottom-Up and Top-Down Visual Attention. California Institute of Technology.

Le Meur, O., Cloarec, S., & Guillotel, P. (2008). Automatic content repurposing for mobile applications. *International Broadcast Conference* .

Le Meur, O., Le Callet, P., & Barba, D. (2007). Predicting visual fixations on video based on low-level visual features. *Vision Research , 47*, pp. 2483-2498.

(2007). *List of ITU-R Recommendations and Reports.* Geneva: International Telecommunication Union.

Lloyd, E., Maclean, R., & Stirling, A. (2006). *Mobile TV — results from the BT Movio DAB-IP pilot in London.* Retrieved July 29, 2010, from European Broadcast Union: http://www.ebu.ch/fr/technical/trev/trev_306-movio.pdf

Lourakis, M. (2009). *homest: A C/C++ Library for Robust, Non-linear Homography Estimation*. Retrieved September 5, 2009, from http://www.ics.forth.gr/~lourakis/homest/

Lucas, B. D., & Kanade, T. (1981). An Iterative Image Registration Technique with an Application to Stereo Vision. *Proceedings of Imaging Understanding Workshop* .

Ma, Y., Soatto, S., Kosecka, J., & Sastry, S. S. (2004). *An Invitation to 3-D Vision.* New York: Springer Verlag.

MAYAH Communications. (2006). *MAYAH Communications - IO [io] 8000 / 8001 User Guide.* Retrieved November 10, 2007, from http://www.mayah.com/products/io-8000ad-support.html

Numata, M., Senoo, H., & Shishikui, Y. (2008). Proposal for a context-aware broadcasting service "AdapTV" - A video trimming system for soccer games to fit the display resolution. *Broadcast Asia 2008 International Conference* .

Paragios, N., Chen, Y., & Faugeras, O. (2005). *Handbook of Mathematical Models in Computer Vision.* New York: Springer-Verlag.

Poynton, C. (2003). *Digital video and HDTV* . San Francisco: Morgan Kaufmann.

Rec. ITU-R BT.1788. (2007). *Methodology for the subjective assessment of video quality* . Geneva: International Telecommunication Union.

Rec. ITU-R BT.500-11. (2007). *Methodology for the subjective assessment of the quality of television pictures* . Geneva: International Telecommunication Union.

Rec. ITU-R BT.601-5. (1995). *Studio encoding parameters of digital television for standard 4:3 and widescreen* . Geneva: International Telecommunication Union.

Rec. ITU-R BT.709-3. (1998). *Parameter values for the HDTV standards for production and international programme exchange* . Geneva: International Telecommunication Union.

Rousseeuw, P., & Leroy, A. (2003). *Robust Regression and Outlier Detection.* New Jersey: John Wiley & Sons.

Sachs, L., & Reynarowych, Z. (1984). *Applied Statistics: A Handbook of Techniques.* New York: Springer Verlag.

Schölkopf, B., Burges, C. J., & Smola, A. J. (1999). *Advances in Kernel Methods: Support Vector Learning.* Cambridge: MIT Press.

Sheskin, D. J. (2007). *Handbook of Parametric and Nonparametric Statistical Procedures* (4 ed.). Boca Raton: Taylor & Francis Group.

Smith, G., & Archison, D. A. (1997). *The eye and visual optical iinstruments.* Cambridge: Cambridge University Press.

Swain, M. J., & Ballard, D. H. (1990). Indexing via Color Histograms. *Third International Conference on Computer Vision* .

Technical Guidelines for the production of Television Programmes for ARD, ZDF and ORF. (2006). Munich: Institut fuer Rundfunktechnik.

Toennies, K. D. (2005). *Grundagen der Bildverarbeitung.* Munich: Pearson Studium.

Treisman A., G. G. (1980). A feature-Integration Theory of Attention. *Cognitive Psychology , 12*, pp. 97-136.

Treisman, A. (1986). Features and Objects in Visual Processing. *255* , 114-125. (S. American, Ed.)

Tsotsos J. K., e. (1995). Modeling visual attention via selective tuning. *Artificial Intelligence , 78*, pp. 507-545.

ViBE Mobile TV Encoder Data Sheet. (2009). Retrieved December 29, 2009, from Grass Valley: http://www.grassvalley.com/assets/media/1989/CDT-3004D-6.pdf

Virtual Dub Mod. (2003). Retrieved August 10, 2009, from http://virtualdubmod.sourceforge.net/

Weber. (1990). *Sehen, gestalten und fotografieren.* Basel: Birkhäuser Verlag AG.

Willow Garage. (2010). Retrieved January 19, 2010, from http://opencv.willowgarage.com/wiki/#Welcome.2BAC8-Introduction.WhatisOpenCV.3F

Wolfe, J. M. (1994). Guided Search 2.0. *Psychonomic Bulletin & Review , 1*, pp. 202-238.

Zaller, J. (2007). *Snell & Wilcox Helios: Optimizes Video for Distribution to Mobile TV Devices*. Retrieved December 29, 2009, from http://broadcastengineering.com/RF/broadcasting_snell_wilcoxs_helios/index.html

Zhang, H.-J. (2002). A model of motion attention for video. *International Conference on Image Processing* .

Zhang, Z. (1997). Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting. *Image and Vision Computing Journal , 15* (1).

Zhang, Z., Deriche, R., Faugeras, O., & Luong, Q. (1995). A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry. *Artificial Intelligence* (78).

11. Abbreviations

ARD	-	Arbeitsgemeinschaft der öffentlich-rechtlichen Rundfunk-anstalten der Bundesrepublik Deutschland (German public broadcaster)
AVI	-	Audio Video Interleave
BMF	-	Broadcast Metadata Exchange Format
CLI	-	Common Language Infrastructure
CLR	-	Common Language Runtime
DAR	-	Display Aspect Ratio
DC	-	Direct Current
DFT	-	Discrete Fourier Transform
DSCQS	-	Double-Stimulus Continuous Quality-Scale
DSIS	-	Double Stimulus Impairment Scale
EBU	-	European Broadcasting Union
FFT	-	Fast Fourier Transform
FIFO	-	First In, First Out
FIT	-	Feature-Integration Theory
GC	-	Garbage Collection
GUI	-	Graphical User Interface
HDTV	-	High Definition Television
HSV	-	Hue, Saturation, Value
IDE	-	Integrated Development Environment
IFFT	-	Inverse Fast Fourier Transform
IRT	-	Institut fuer Rundfunktechnik (research centre of the German broadcasters)
ISDB-T	-	Integrated Services Digital Broadcasting - Terrestrial
ITU	-	International Telecommunication Union
ITU	-	IT University of Copenhagen
LED	-	Light Emitting Diode
LMedS	-	Least Median of Squares
LQS	-	Least Quantile of Squares

MMV	-	Multimedia and Visualization
MPEG	-	Moving Picture Experts Group
MXF	-	Material eXchange Format
NHK	-	Nippon Hōsō Kyōkai (Japan Broadcasting Corporation)
OMF	-	Open Media Framework
OpenCV	-	Open Source Computer Vision
ORF	-	Österreichischer Rundfunk (Austrian Broadcasting)
PAR	-	Pixel Aspect Ratio
RANSAC	-	RANdom SAmple Consensus
RGB	-	Red, Green, Blue
RMedSE	-	Root Median Squared Error
ROI	-	Region Of Interest
RT	-	Response Time
SAD	-	Sum of Absolute Difference
SAMVIQ	-	Subjective Quality of Internet Video Codecs
SAR	-	Sample Aspect Ratio
SC	-	Stimulus Comparison
SDTV	-	Standard Definition Television
SS	-	Single Stimulus
SSCQE	-	Single Stimulus Continuous Quality Evaluation
SSD	-	Sum of Squared Difference
Suviq	-	Subjective Video Quality
T-DMB	-	Terrestrial - Digital Multimedia Broadcasting
TV	-	Television
UDP	-	User Datagram Protocol
UUID	-	Universally Unique Identifier
VOCUS	-	Visual Object detection with a CompUtational attention System
XML	-	Extensible Markup Language
ZDF	-	Zweites Deutsches Fernsehen (German public broadcaster)

12. Appendix

Name	Session Number	Date	Signature
	Training		
	Session 1 A		
	Session 2 A		
	Session 3 A		

<u>Introduction</u>

You are participating in an evaluation for assessing video sequences which have been prepared for mobile TV. The objective is to assess the subjective impression of the composition of the chosen cropping area. The rating should not be done concerning video quality differences between the four shown sequences and should not be related to the image size.

The sequences to be assessed are three different levels of cropping with aspect ratios of 16:9 and 4:3. Each cropping level has been created by three different cropping methods. An additional sequence is the non-cropped original version. Please lean your head during evaluation against the headrest attached to the chair.

The subjective impression should be assessed in relation to a reference on a seven grade scale from “much better” to “much worse”. The reference should serve as benchmark, not as an absolute reference. If you have no preference at all, please assess the sequences as “the same”.

For every unequal assessment, an attribute must be checked on the attached paper sheet if one of those attributes has influenced your decision. In the case that no attribute is listed on the attached, please check “do not know”.

Each sequence must be completely watched at least once. The order of the sequences to be played can be chosen arbitrarily. An automatic looping of the sequences can be done by pressing 🔄 (“loop button”). Please use the slide control to assess the quality of each sequence. Choose a value on the seven grade scale that best suits your opinion. After you have watched and assessed all sequences, the next trial can be started by pressing ➡ (“continue-button”).

Explanation of attributes:

- Motion: The motion of the video content is natural/unnatural and continuous/not continuous.
- Sharpness: The video content has/does not have satisfying detail resolution.
- Proportions: The ratio between image size and image content is appropriate/not appropriate.
- Position of the cropping area: The cropping area is well/poorly chosen and contains/misses most important elements.

12.1 Published work based on the thesis

Publication	Contributions to the Thesis
Deigmoeller, J., Itagaki, T., Stoll, G. (2008). An approach for an intelligent crop and scale application to adapt video for mobile TV. *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting.*	Chapter 1 (Section 1.1); Chapter 5; Chapter 6 (Section 6.1.1)
Deigmoeller, J., Itagaki, T., Stoll, G., Just, N. (2010). An approach to intelligently crop and scale video for broadcast applications. *Proceedings of the 2010 ACM Symposium on Applied Computing.*	Chapter 1 (Section 1.1); Chapter 2 (Section 2.5.2); Chapter 3 (Section 3.5), Chapter 5; Chapter 6 (Section 6.1.1); Chapter 7 (Section 7.2.1, 7.2.2, 7.3.1, 7.3.3, and 7.3.4); Chapter 8 (Section 8.2.2, and 8.2.3)); Chapter 9 (Section 9.1)
Deigmoeller, J., Itagaki, T., Stoll, G., Just, N. (2010). A context-based approach to crop and scale video for broadcast applications. *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting.*	Chapter 1 (Section 1.1); Chapter 2 (Section 2.5.2); Chapter 3 (Section 3.5); Chapter 5; Chapter 6 (Section 6.1.1); Chapter 7 (Section 7.2.1, 7.2.2, 7.3.1, 7.3.2, 7.3.3, and 7.3.4); Chapter 8 (Section 8.1.1, 8.1.3, 8.1.5, and 8.1.7); Chapter 9 (Section 9.1)
Deigmoeller, J., Itagaki, T., Stoll, G., Just, N. (2010). Contextual Cropping and Scaling of TV Productions. *Multimedia Tools and Applications (MTAP) Special Issue on ACM SAC'10 MMV Track.* (selected from the MMV track of the 2010 ACM Symposium on Applied Computing, currently in submission)	Chapter 1 (Section 1.1); Chapter 2 (Section 2.5.2); Chapter 3 (Section 3.5); Chapter 5; Chapter 6 (Section 6.1.1); Chapter 7 (Section 7.2.1, 7.2.2, 7.3.1, 7.3.2, 7.3.3, and 7.3.4); Chapter 8 (Section 8.1.1, 8.1.3, 8.1.5, 8.1.7, and 8.2.3); Chapter 9 (Section 9.1)