This is a preliminary version of an article published by
S. Kopf, T. Haenselmann, J. Kiess, B. Guthier, W. Effelsberg:
Algorithms for video retargeting.
Multimedia Tools and Applications, Special Issue: Hot Research Topics in Multimedia,
pp. 819-861, Vol. 51 (2), Springer Netherlands, 2011.

Algorithms for Video Retargeting

Stephan Kopf · Thomas Haenselmann · Johannes Kiess · Benjamin Guthier · Wolfgang Effelsberg

Abstract The visualization of high resolution video on small mobile devices is still a great challenge today. Most critical are the limited display resolution and different aspect ratios of handheld mobile devices. So far, there is no retargeting algorithm available that guarantees good results for all videos. We introduce a new video retargeting approach that reduces the resolution while preserving as much of the relevant content as possible. A central component of the system selects the most suitable algorithm to adapt a given shot. We have implemented two retargeting algorithms: a region of interest (*ROI*) based technique, and a fast implementation of seam carving for size adaptation of videos (*FSCAV*). The ROI-based retargeting detects important regions like faces, objects, text, and contrast-based saliency regions. A rectangular window within the larger frame is selected that defines the visible area of the target video. If several relevant regions are detected, an artificial camera motion (pan, tilt, or zoom) may change the selected view within a shot. For seam carving, we present two extensions: The first reduces the distortion of straight lines (lines may become curved or disconnected); the second avoids jitter in the target video, limits the large memory requirements and computational effort of seam carving, and makes it applicable to video retargeting. In addition, we present a heuristic that estimates the visual quality of the target video. If the quality drops below a threshold, the ROI-based retargeting is used for this shot. User evaluations confirm a very high visual quality of our approach.

Keywords video retargeting · video adaptation · seam carving · region of interest · contrast-based saliency

1 Introduction

Since their introduction, TV specifications like PAL or NTSC have only been subject to little change. They define the frame rates, the number of lines an image consists of and how color and audio are coded on an analog channel. It might be due to their analog nature that these specifications did not change significantly for about half a century. They define

Stephan Kopf, Thomas Haenselmann, Johannes Kiess, Benjamin Guthier, Wolfgang Effelsberg
Department of Computer Science IV, University of Mannheim, Mannheim, Germany
Tel.: +49-(0)621-181-2600
Fax: +49-(0)621-181-2601
E-mail: {kopf|haenselmann|kiess|guthier|effelsberg}@informatik.uni-mannheim.de

an aspect ratio of 4:3 which means that the screen is 4 units wide and 3 units high. In the early nineties, an extended analog format of a 16:9 aspect ratio was introduced as PALplus. Though the format adapts much better to the human field of vision which is significantly wider than high, the new 16:9 format was adopted by consumers only slowly and coexisted with 4:3 for at least another decade. Only the latest move to flat panel displays and digital transmission yielded a breakthrough for the wider 16:9 format. In addition, the dominance of digital over analog technology changed the settings in which video content is consumed. While the typical situation was watching TV in the living room for a long time, within the last decade it increasingly moved to the desktop or laptop, and even the cell-phone display, recently. In contrast to the limited number of TV standards, a large number of cinematic formats coexist, most of which have much wider ratios between 1.85:1 and 2.40:1.

Video retargeting aims at bringing video content which has been produced in a particular format and for a specific platform to another one. The problem is probably best known from the attempt to show films which have been produced in cinematic formats on a standard 4:3 TV screen: The traditional means of doing this conversion were limited and mostly unsatisfactory. The content provider can choose to shrink the movie until its width complies with the width of the TV screen. However, then the movie's height can no more cover the 4:3 area so that large parts of the screen remain empty. If we want to adapt 16:9 to the PAL standard, we need to divide both sides by four which yields 16/4:9/4 = 4:2.25. This wastes 25% of the channel's capacity resulting in a lower resolution and noticeably less details. As cinema movies are usually even wider than 16:9, in practice even more information is lost.

An alternative solution is to adapt the low height of the wide screen format to the TV's height which can be done by dividing each side by 3. 16/3:9/3 = 5 1/3:3. In this case, the movie is too wide so that 25% of the content can not be seen. As a compromise, a factor between 3 and 4 can be chosen which yields a moderate loss of content and a less severe loss of channel utilization. Depending on the film, most TV stations are slightly biased towards adapting the height because most of the relevant content takes place in the middle of the screen, the sides are of lower importance and can thus be cropped to a certain degree.

We have not yet discussed the most obvious solution, namely scaling the content horizontally and vertically by different factors. Obviously, dividing the width by 4 and the height by 3 would convert 16:9 into 4:3. However, humans are very sensitive to changes in the aspect ratio, particularly when observing other humans or well-known objects. While the perception of landscape and architecture is less sensitive to stretching or shrinking, changes of the aspect ratio of 5% and less can already result in an unnatural appearance of faces.

The modern 16:9 TV sets have the opposite problem, namely to adapt the 4:3 content. An early solution was in fact to ignore correct proportions and to stretch the image horizontally in order to meet the available width, yielding fairly broad faces and actors. Then, manufacturers became aware of the fact that the important content usually shows up in the middle of the screen. As a consequence they began to leave the middle part undistorted, stretching the image increasingly towards the left and right border. As long as actors stay centered in the middle, they appear mostly natural while the periphery of the image which is often more abstract and of lower importance is stretched a lot. Nevertheless, the overall appearance can benefit from this scheme. Though it is of static nature, this solution makes a first attempt to adapt the video in a content-based way.

Another need for the adaption of video content comes from the fact, that users increasingly view content on mobile devises with small displays. Displaying a soccer field does not make much sense on the tiny display of a cell-phone. If compromises need to be made in terms of what to display, then the respective clipping area should at least display the most relevant content like actors' faces, the ball in a soccer match, text, etc. This is why

the next generation of video adaption approaches tried to find the relevant content and defined a *minimal enclosing rectangle* around this content. Note that these rectangles can also move over the original full video over time or can zoom in and out. This can be considered a secondary video panning and zooming over the original content. As compared to static clippings, these dynamic ones have much better potential to adapt to the relevant content. However, the challenge lies in the definition of what is considered important in a video.

The latest generation of image and video adaption approaches like *seam carving* avoid this problem. Basically, the technique has been known from temporal audio compression for many years, but its extension to 2D or 3D signals, i.e. images or videos, which is also technically more demanding, has only been done recently. The so-called time-scale compression in audio can be used, e.g., to play a voice recording back faster than real-time without altering its pitch. One of the techniques employed is to cut out parts of the audio with the least amount of energy like the little pauses a speaker makes or slow on- or offsets of a sound that can be shortened. Similarly, in video retargeting, rather than deciding which interesting content to preserve, it is decided which irrelevant one to delete. As a rule of thumb, low entropy is little meaningful for humans as well. A part of an image which contains no edges often also contains little room for interpretation and can thus be deleted more easily in order to shrink an image. If it needs to be expanded, pixels in these areas can be repeated without making the changes too obvious. A technical advantage is also that the parts of the image with the lowest entropy can be found optimally, e.g., by means of dynamic programming. In contrast, finding the most relevant parts depends on a sometimes debatable definition of relevance and even different human observers might come to differing results. However, within this article we will also show how the combination of the content-based adaptation and the more objective seam carving can lead to even better results.

The article is organized as follows: In the next Chapter we will introduce different video retargeting techniques. Chapter 3 provides an overview of the video retargeting system and describes details of the video analysis steps. The information about shot boundaries, camera parameters, faces, text, objects, and saliency regions is the basis for the two video retargeting algorithms described in Chapter 4 and 5. Chapter 6 focuses on the automatic selection of a suitable retargeting algorithm, and the different retargeting algorithms are evaluated in Chapter 7. The last chapter concludes the article and provides some outlooks.

2 Related Work

2.1 Standardization and Classification of Video Retargeting Techniques

The semantic description of the content of videos is part of the *MPEG–7* and *MPEG–21* standards [30, 32], and metadata is specified for the personalization and the adaptation of videos [79, 80]. *MPEG–7* also encompasses a description language for the easy exchange of multimedia documents. Additionally, the network-based access to multimedia documents from arbitrary devices is supported (*universal multimedia access*) [4, 57, 82]. Rules to transcoding videos, a user history and individual user preferences (*user preference description*) can be stored and used for automatic adaptation.

Additional metadata is integrated into the *MPEG–21* standard. A description of the user's device (*usage environment description*) is available, characterizing the display, hardware and configuration of a handheld mobile device [61]. Techniques like the modeling of user queries and user preferences are also part of MPEG–21. Individual adaptation algorithms can be defined in MPEG–21 for digital elements (*digital item adaptation*) [31].

Approaches for the adaptation of videos can be classified according to the following categories:

- Location (server, proxy, client),
- Technical features (hardware, software, network capacity),
- Type of adaptation (static, dynamic),
- Layered video encoding (temporal / spatial scaling),
- Transcoding (with / without semantics), and
- Retargeting methods (discrete, continuous, hybrid).

A typical classification scheme distinguishes between *server-* [28, 57, 60], *proxy-* [24, 55] or *client*-based [45] adaptations. A bottleneck in the server-based approach is the CPU, especially if many clients connect to the server simultaneously. On the other hand, the client-based approach is unsuitable for full resolution videos due to the large amount of data to be sent to the client for adaptation [8].

Technical features of a handheld mobile device like hardware and software configuration define the specific parameters for video adaptation [20]. Another classification criterion is the *type of adaptation*. If the complexity of the algorithm is low, an adapted video can be computed in real-time (dynamic adaptation). Otherwise, precomputed adapted versions of a video are stored on the server for later usage without any further processing.

Temporal and spatial scaling is supported in several video standards (e.g., MPEG–2), where different quality levels can be stored in one video stream. The *base layer* uses a minimum amount of network and CPU resources. Additional *enhancement layers* improve the quality of the video step-by-step.

A *transcoding* algorithm modifies the format-specific parameters of a video, for example, the color depth, resolution, frame rate, or bit rate. Semantic transcoding considers the visual content of the video and tries to preserve the semantic content in the adapted video [62]. Many efficient *transcoding algorithms* were developed for the adaptation of videos [81]. Missing software decoders or insufficient hardware support on handheld mobile devices necessitate a change of the video codec [2, 5, 11, 46]. The transcoding of videos should keep the computational effort as low as possible and reuse data from complex calculations like estimation of motion vectors [74]. Most systems propose efficient algorithms to transcode videos but do not consider the semantic content of the videos [27, 59, 76].

The *retargeting methods* – classified as discrete, continuous, or hybrid techniques – are the focus of the following sections. We will present two retargeting methods in detail in Chapter 4 and 5.

2.2 Image Retargeting

Resizing of images is a popular research topic at the moment with a lot of different adaptation techniques available. However, all of these techniques use the same basic operations and either remove pixels from an image (*discrete methods*), merge the pixels of an image (*continuous methods*) or combine these two operations (*hybrid methods*).

2.2.1 Discrete Methods

Discrete methods resize an image by the removal of pixels. *Cropping* is a typical example for algorithms in this category which reduces the image size by removing image content from the borders of an image.

An automatic algorithm which uses cropping is presented by Suh et al. [77]. The important regions of an image, the so called *Regions of Interest* (ROI), are detected by either a saliency map [33] or face detection [70, 71]. The first approach is applicable to all kinds of images while the second one is restricted to images with humans.

A variation of the cropping approach is formulated by Santella et al. [68]. They detect the *Attention Objects* (AO) of an image by using eye movement tracking and combining it with segmentation. In a first step, the image is segmented into different regions. After that, the algorithm combines the fixations from the eye tracking with the region information to find out where the viewer looked. Subsequently, the AOs are identified by the *lazy snapping* approach [47]. The algorithm then crops the image according to these objects and a few basic rules of composition.

Liu et al. [52] improve upon the idea of cropping and introduce automatic image browsing, where a cropping rectangle is moved over the source image to show the interesting regions one after another. A saliency map is used to find the AOs of an image. Then, nearby objects are grouped together to form ROIs. These ROIs are zoomed in or out to fit a rectangle of the target size. A path is calculated which determines the order and the length in which the ROIs are shown. The newly introduced *minimal perceptible time* represents the time needed for a user to fully perceive the shown image. Finally, the shortest path between the centers of two ROIs is used to move the cropping rectangle between them.

Another browsing approach is presented by Liu et al. [53]. Regions are detected in the image with image segmentation based on spatial connectivity and color features of the pixels. An importance factor for each region is computed according to a position factor and the sum of its contrast set against the other regions. The order in which the regions are shown is determined by their average importance value, starting with the highest.

In previous work, we have presented an adaptation technique for small handheld mobile devices where we put the focus on an efficient implementation [42]. Cropping is used in a first step to remove border regions which usually contain little relevant semantic content (e.g., sky or grass). Attention objects like faces, text regions and contrast-based saliency maps are identified in a second step and combined into a ROI. We proposed a semi-automatic approach where users can mark obvious errors in the adapted images.

The previous methods reduce the size of an image by cropping entire parts of the image, for instance at the borders. Seam carving, a technique introduced by Avidan and Shamir [1], instead removes connected paths of pixels within the image. As it is used as one of the video retargeting algorithms of our system, it is discussed in more detail in Chapter 5.

The loss of some image information is a major drawback of the discrete retargeting methods. Their use may result in partly visible or completely missing objects. The goal is therefore to remove the image parts containing the least valuable information.

2.2.2 Continuous Methods

Continuous methods retarget an image by merging adjacent pixels. A typical example is *scaling*, where pixels are merged uniformly to change the size of an image.

A non-linear approach is presented by Liu et al. [50]. This technique emphasizes the ROIs and uses higher scaling rates for the other parts. A combination of saliency map and object detection is used to identify a ROI. Two methods are used for the resizing: *Radial warping* or *Cartesian fisheye warping*. The first one places the ROI in the middle of the target image and scales the pixels according to their position between the ROI and the image border. The further away a pixel is from the center, the smaller it becomes. In the second

method, the image is segmented into nine regions, and the regions are scaled based on their position related to ROI.

Setlur et al. [72] try to preserve the ROIs in an image by *cutting* them out. ROIs are first detected by a combination of segmentation based on a saliency map and face detection. If the area containing all the ROIs fits into the target display, the image is cropped. Otherwise, the ROIs are removed from the image and the image background is repaired with the *inpainting technique* [26]. Then, the background is scaled to the target size and the ROIs are reinserted into the image. In case the removed objects are too large to fit in, they are scaled down in inverse proportion to their importance.

A technique not primarily designed for retargeting is introduced by Gal et al. [21]. This method is based on *warping* and maps each pixel with inhomogeneous 2D texture mapping to a new position in the target image. In the process, a 4-connected mesh with quad cells is deformed where some pixels are maintained while others are merged to a new target pixel. The relevance of each pixel is determined by a user generated importance map.

Another non-linear warping technique is presented by Wang et al. [84]. To guide the mapping, the image is again divided into small regions by overlaying a mesh with quad cells. A combination of a gradient and a saliency map is used to determine the importance of each cell. Important cells are scaled uniformly while the others are allowed to be distorted to fit into the target image. In order to prevent major distortions of important objects or structures, deformation energy and grid line bending energy are introduced. An optimization function is then minimized for the final mapping of the pixels.

Guo et al. [22] also use a mesh for their warping technique, but use triangular cells instead of quads. They define the retargeting as a *mesh parametrization problem* and connect the mesh with the underlying image structure. This structure is identified with saliency map and face detection. Important objects and structures are preserved following three constraints: the boundary constraint, the saliency constraint, and the structure constraint.

Global energy optimization is used in the approach presented by Ren et al. [63]. Instead of dividing the image with a mesh, each pixel is treated as an individual component. After the importance is determined with saliency and face detection, each pixel is assigned a new width and height. Finally, components in the same line/row are fused to form new components.

Continuous methods work very well if an image contains unimportant regions to shift the distortions to. If there are no such regions in an image, the result either becomes obviously distorted or resembles scaling. A major drawback of most methods in this category is their poor preservation of straight lines.

2.2.3 Hybrid Methods

Hybrid methods try to overcome the respective limitations of discrete and continuous methods by combining operators from both categories. A simple example would be to start cropping an image based on an importance map and then switch to scaling when a certain threshold is reached.

In a more complex manner, Rubinstein et al. [67] combine the three retargeting operators scaling, cropping and seam carving. To determine which operator to be used next, a new image similarity measurement called *bi-directional warping* is introduced. The retargeting problem is formulated in a *resizing space*, where the operators are combined in a multi-dimensional space. The order of the operators is found by dynamic programming and can either be a *regular path*, which means each operator is used exactly once in a certain order or a *mixed path*, where the operators are used as often as necessary with no particular order.

There are also some other combinations of these retargeting operators [12–14, 23, 29] which mainly differ in the importance measurement and the way they switch between the operators. While some rely on the importance measure for switching [12, 23, 29], Dong et al. present algorithms using an image distance function [13] or an operator cost-based approach [14]. The main difficulty when using hybrid methods is to find the optimal switching point between the operators as each viewer might prefer a different combination [67]. Also, the limitations of an individual operators might affect the subsequent operators.

2.3 Video Retargeting

Retargeting videos is a more complex problem than image retargeting due to the additional temporal dimension. The naive solution is using an image retargeting method on each frame individually. This may cause jitter and other obvious distortions. As the presented methods can be again categorized in discrete, continuous and hybrid methods, this section is structured in the same way as image retargeting.

2.3.1 Discrete Methods

Cropping is again a typical example of a discrete method. For instance, the left and the right borders of wide-screen cinema movies are simply cropped to adapt it to television. Another possibility is to move a cropping rectangle over the screen capturing the most important information. This is done by El-Alfy et al. [16] to process surveillance videos in real-time. The video stream is divided into overlapping segments and then computed independently. In each segment, saliency is used to identify the ROIs and a trajectory for the cropping rectangle is processed which maximizes the captured saliency.

A cropping rectangle is also used by Tao et al [78]. Under the assumption that moving foreground objects are the most interesting content for the viewer, they choose these objects as AOs. The algorithm then follows these objects with a cropping rectangle with the size of the target display.

Seam carving is as well a popular method for the resizing of videos. As mentioned in the image retargeting chapter, it will be described in more detail in a later subsection.

Analog to image retargeting, the major drawback of discrete methods for video retargeting are truncated or completely missing objects. Additionally, an important object can now move outside the screen when the algorithm is not able to catch up with the movement.

2.3.2 Continuous Methods

Scaling is once more an example for a continuous method. It is the only technique which can be applied to each frame independently without causing distortions due to its temporal constancy.

Wolf et al. [86] present a non-linear warping approach for videos which works in real-time. They assign an importance value to each pixel based on saliency and face detection and then map them into an image of the target size. The size and position of each pixel is generated according to their importance value, where unimportant pixels are fused with their neighbors and important pixels keep their size. For temporal coherence, each frame additionally considers its predecessor for the mapping of the pixels.

In contrast to the previous approach, the warping algorithm introduced by Wang et al. [85] does not work in real-time. It is an extension of the mesh-based scale-and-stretch

technique presented in [84]. First, estimated interframe camera motion is used to align consecutive frames of a video. An importance map is then generated via saliency that additionally takes the preceding and the succeeding frames into account for temporal coherence. Also, distinct moving areas of objects are detected and uniformly resized to preserve object motion. A global optimization function is minimized under spatial and temporal constraints for the final mapping of the pixels.

The warping technique by Krähenbühl et al. [44] works again in real-time and is designed for the retargeting of streaming videos. Instead of using a mesh to divide the frames into regions, the warp is computed per-pixel. The algorithm mixes automatically detected important regions with high level features selected by the producer. In order to preserve temporal coherence, the algorithm also integrates the information of a few additional frames into the warp. For the minimization of aliasing artifacts, a hardware accelerated *EWA* (elliptical weighted average) *splatting* [87] is finally performed on the output frames.

A completely different technique is presented by Cheng *et al.* [10]. It takes the idea from Setlur et al. [72] and extracts important objects from a video scene. The background is then repaired with inpainting [9] and rescaled to the target size before the selected objects are reintegrated into the video. Important objects are identified according to color, intensity and motion. Another interesting approach is introduced by Kim et al. [35], where the frames of a video are divided into stripes and then adaptively scaled based on an importance measure. The video is presented as a 3D video cube to assure temporal coherence and the retargeting is solved as a constrained optimization problem.

Continuous methods for video retargeting still have problems with the preservation of straight lines. Also temporal consistency cannot always be guaranteed which may cause jitter in the target video.

2.3.3 Hybrid Methods

The combination of two or more retargeting operators is also very common in video retargeting, where scaling is often added to an otherwise discrete method. For example, if the video contains several relevant regions, the algorithm probably decides to show a scaled version of the whole screen.

Such an approach can be seen in the algorithm proposed by Wang et al. [83]. Their goal is to adapt videos to various mobile devices. The ROI of a shot is detected and then shown in a cropping rectangle that covers it. Scaling is used to adapt the size of the cropping rectangle to the target display or show the whole screen of the original shot if there is no ROI identified. In this approach, the rectangle does not move within a shot.

Liu et al. [51] use a similar idea. A cropping rectangle is positioned in the frames based on an importance map. The size of the rectangle is again adjusted to contain the most important image regions and is additionally able to move over the screen. If no single region can be determined as the most interesting, the whole screen is shown. The importance is measured according to saliency, motion and a system of penalties, for example for the cropping of faces. Finally, the retargeting is solved through an optimization function.

Multi-scale trajectory optimization is proposed by Li et al. [48]. The algorithm uses the same basic idea as before but computes several trajectories for different sized crop rectangles. As the video is presented as a graph, a Max-Flow/Min-Cut algorithm is used to find these trajectories. Finally, the trajectories are ranked according to the importance captured in them and the best one is picked.

In our *mobile cinema* project [36], we have presented a video adaptation technique where we put the focus on the preservation of foreground objects like general objects, faces

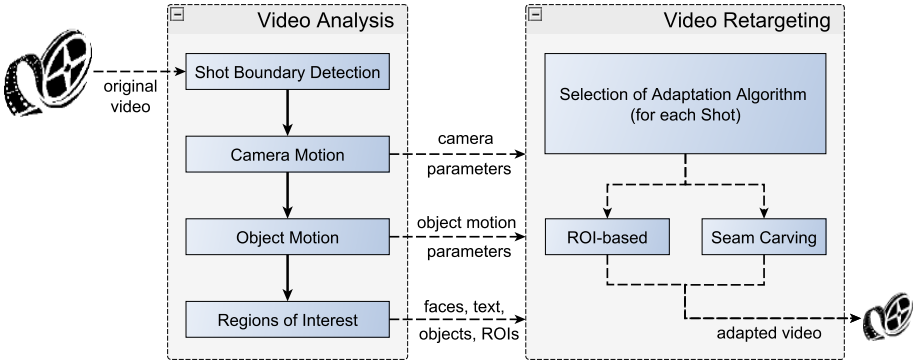

Fig. 1 Overview of the video retargeting system.

and superimposed text. The goal is the preservation of the aspect ratio, but also to maximize the visible information of these regions in the adapted video. Image regions are subject to constraints to preserve a *minimum perceptible size* as well as a *maximum reasonable size*. Solving an optimization problem (maximize the visible content) results in the parameters for scaling and cropping [41]. To increase the amount of visible content, the selected region may change over time by adding an artificial zoom or camera pan. An additional application was presented, that merges spatial and temporal adaptation techniques based on foreground objects to generate video summaries with limited screen resolutions [37].

Hybrid methods for video retargeting exhibit the same drawbacks as the respective methods for images. Additionally, the introduction of artificial pans or zooms into a video sequence might change the dramatic composition of a scene and may be an undesired effect.

As seen in this section, many video retargeting techniques are available today. Although some techniques usually achieve better results than others, the content of the video to be adapted has a great impact on the quality of the target video. In this article, we propose an adaptation scheme that allows the selection of a suitable retargeting algorithm on shot level. Each shot is analyzed, and based on the visual content, a specific adaptation technique is selected. In the following, we focus on two video retargeting techniques: a ROI-based algorithm and improved seam carving.

3 Video Analysis

3.1 System Overview

So far, there is no retargeting algorithm available that guarantees good results for all types of video. An algorithm may show excellent results for some videos, but is unsuitable for others. It follows that more than one technique should be considered for the adaptation of a video. The selection of a suitable one is then a nontrivial task. We aim to automatically derive characteristics from a video that allow the selection of a suitable adaptation algorithm. Fig. 1 gives an overview of our video retargeting system.

The selection of an adaptation algorithm is done for each shot (continuous camera recording). The analysis thus starts with shot boundary detection. Camera motion between adjacent frames and object motion are then estimated. Finally regions of interest (ROI) consisting of faces, text regions, moving objects, and salient features are detected.

The information calculated in the analysis step is used to select a suitable adaptation algorithm. As an example, cropping-based adaptation is unsuitable when important objects are located at image borders. Seam carved videos on the other hand typically exhibit major distortions in large objects or when the camera motion is orthogonal to the orientation of the seams. The features computed during the analysis is later reused to adapt each shot of the video. Finally, a low-resolution video is encoded and stored.

3.2 Shot Boundary Detection

Our shot boundary detection algorithm identifies hard cuts, fades and dissolves. Quantized color histograms and the sum of absolute differences of corresponding histogram bins are used to estimate the similarity of consecutive frames. A *hard cut* is detected if the histogram difference is significantly larger than the maximum histogram difference in the five-frame neighborhood of the analyzed frame.

Our detection of *fade-ins* and *fade-outs* computes the standard deviation of luminance values for each frame. If the standard deviation decreases from frame to frame and the final frames are close to monochrome frames, we classify the sequence as a fade-out. A *dissolve* has characteristics similar to those of a fade. The standard deviation of the pixel values in the middle of a dissolve is significantly lower than that at the beginning or end of it. We validate the detected cuts by incorporating edge information (*edge change fraction*) and camera motion [37].

3.3 Camera Motion Estimation

Image registration techniques can be used to track the image content in consecutive frames. We use the projective camera model which requires eight parameters to describe the motion of the camera in consecutive frames. The position of each pixel (x,y) in a frame i is transformed to a new position (x',y') in frame $i+1$:

$$x' = \frac{a_{11}x + a_{12}y + t_x}{p_x x + p_y y + 1}, \tag{1}$$

$$y' = \frac{a_{21}x + a_{22}y + t_y}{p_x x + p_y y + 1}. \tag{2}$$

Six parameters (a_{ij}, t_x, t_y) define affine motion and two parameters (p_x, p_y) a change of perspective. The parameters t_x and t_y correspond to a pan or tilt, and a_{ij} specify rotation and zoom.

To estimate the parameters of the model, point correspondences between two frames have to be established first. A large number of techniques has been proposed to identify characteristic feature points in images. Examples are corner detectors like Harris [25] or SUSAN [75], or detectors that are invariant to image transformations like SIFT [54] or SURF [3]. The latter detectors are especially advantageous in case of artificial distortions of frames [69]. Frame rates of at least 25 fps are typical in videos that were produced for cinema or television, leading to relatively little camera motion between consecutive frames. We thus do not require transformation invariant features. We chose the Harris detector with sub-pixel refinement for its good repeatability and accuracy [17].

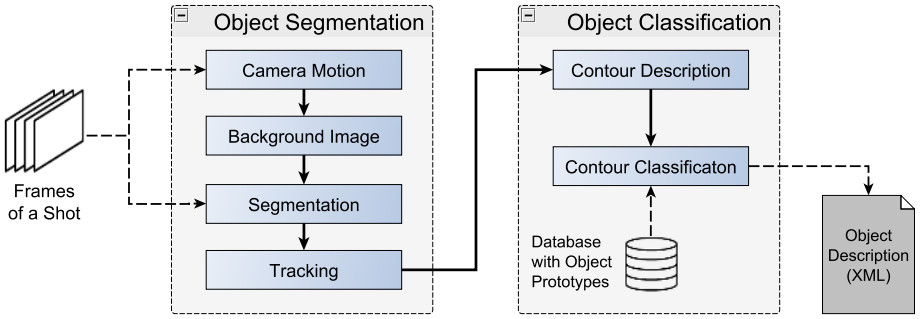

Fig. 2 Overview of the object recognition process.

A feature pair in two consecutive frames is considered for a correspondence check only if the spatial distance of the features is below a predefined threshold. The visual distance between two feature points is then defined as the sum of absolute differences in a local window of 8×8 pixels. A greedy algorithm selects corresponding feature points by choosing the two features with the smallest visual distance. The algorithm terminates if this distance exceeds a threshold.

Four feature point pairs between two frames are sufficient to calculate the eight parameters of the camera model. We use a robust algorithm for motion estimation to cope with false correspondences (outliers). One of the most popular techniques is the RANSAC algorithm [19]. A subset of four corresponding features is randomly drawn and the parameters of the camera model are calculated from these features. We classify features into inliers that fit to the model and outliers. The parameters with the largest number of inliers are kept. By repeating this step, the probability of the chosen features describing the camera motion correctly is very high.

3.4 Object Segmentation and Classification

Moving objects provide important semantic information. If the same moving object is visible in several frames of a shot, it should also be fully visible in the target video. The retargeting algorithm will assign a high priority to regions containing such objects.

Our object recognition algorithm consists of two components: a *segmentation module* and a *classification module*. This is depicted in Fig. 2.

The *object segmentation* starts by using the parameters of the camera motion estimation to construct a background image for the entire shot. The center frame of a shot is selected as a reference frame and all other frames are aligned to it. After alignment, the background objects are always located at the same absolute pixel position. A background image without moving foreground objects is constructed by applying a median filter to all pixels available at one position. Fig. 3 visualizes the construction of the background image.

Fig. 4 depicts two sample frames from a shot of a historical car race and the automatically constructed background image. The segmented and classified object (a car) is marked with a bounding rectangle. An object is segmented by comparing each transformed frame to the constructed background image. A pixel is characterized as object pixel if the absolute difference of aligned pixels exceeds a threshold. The segmented region is validated by tracking position, size, aspect ratio and direction of the motion. This helps to eliminate noise and

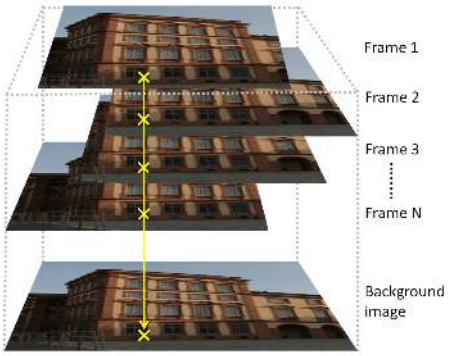

Fig. 3 The camera parameters are used to align the frames of a shot. A pixel of the background image is defined as median of all pixel values at this position.

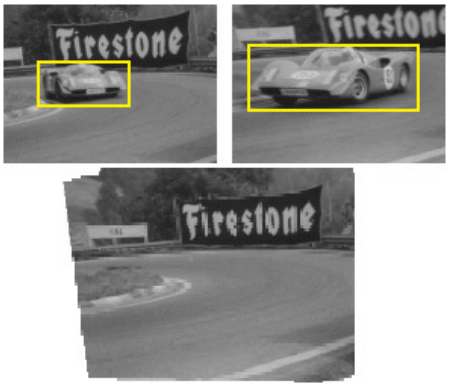

Fig. 4 Top: Bounding box of the segmented objects. Bottom: Example of a constructed background image.

remove segmentation errors. Only objects that can be tracked for several frames of the shot are kept for further processing.

The *object classification* module analyzes the segmented object masks. For each mask, the efficient curvature scale space (CSS) method is used [40, 64] to describe its shape. CSS is also used in MPEG-7. The CSS technique is based on curve evolution and provides a multi-scale representation of the curvature zero crossings of a closed planar curve [58].

The closed planar curve $\Gamma(u) = \{(x(u), y(u)) | u \in [0,1]\}$ represents the outer shape of the segmented object with the normalized arc length parameter u. The shape is iteratively deformed by a Gaussian kernel $g(u,\sigma)$ of width σ. The deformation of the curve is presented by:

$$\Gamma(u,\sigma) = \{(X(u,\sigma), Y(u,\sigma)) | u \in [0,1]\}. \tag{3}$$

$X(u,\sigma)$ and $Y(u,\sigma)$ denote the position of the curve pixels after convolution with $g(u,\sigma)$. The curvature κ of an evolved curve is defined as:

$$\kappa(u,\sigma) = \frac{X_u(u,\sigma) \cdot Y_{uu}(u,\sigma) - X_{uu}(u,\sigma) \cdot Y_u(u,\sigma)}{(X_u(u,\sigma)^2 + Y_u(u,\sigma)^2)^{3/2}}, \tag{4}$$

and requires the computation of the first and second derivatives of $X(u,\sigma)$ and $Y(u,\sigma)$. A CSS image $I(u,\sigma) = \{(u,\sigma) | \kappa(u,\sigma) = 0\}$ is a multi-scale representation of the inflection points of Γ. Significant peaks in the CSS images represent major concave segments of the shape. Fig. 5 shows an example of a CSS image. We use the triple (position, size, width) of the peaks as the feature points in the CSS image to describe a shape.

A major drawback of the standard CSS approach is the inadequate representation of convex segments of the shape. To overcome this problem, we use a two-step approach: First, we apply the standard CSS technique to get feature vectors that characterize concave parts of the shape. Next we create a second shape that is called *mapped shape* and provides additional features for the convex segments of the original shape. To create a mapped shape, we enclose the shape by a circle of radius R and identify the point P of the circle closest to each shape pixel. The shape pixel is mirrored along the tangent of the circle in P. This mapping can be understood as mirroring the shape along an enclosing circle. It guarantees

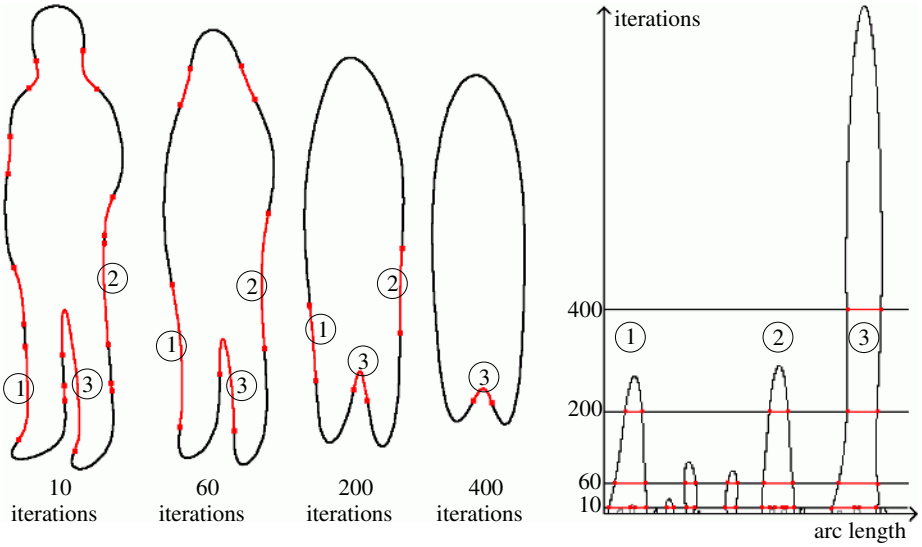

Fig. 5 Left: Smoothed shape after 10, 60, 200 and 400 iterations. Inflection points are marked with small dots. Right: CSS image of this shape. The numbered concave segments (red) create three significant peaks.

that strong convex segments of the original shape become concave segments of the mapped shape.

To create the mapped shape, each shape pixel at position u of the closed planar curve $(x(u),y(u))$ is mapped to a curve $(x'(u),y'(u))$. The center of the circle $(\bar{x},\bar{y})$ is calculated as the average position of shape pixels ($\bar{x} = 1/N\sum_{u=1}^{N} x(u),\ \ \bar{y} = 1/N\sum_{u=1}^{N} y(u)$), the radius $R = \max_u \left\{ \sqrt{(\bar{x}-x(u))^2+(\bar{y}-y(u))^2} \right\}$ is defined so that all shape pixels are located on or within the circle. The mapped curve is then calculated as follows:

$$D(u) = \sqrt{(\bar{x}-x(u))^2+(\bar{y}-y(u))^2}, \tag{5}$$

$$x'(u) = \frac{2R-D(u)}{D(u)} \cdot (x(u)-\bar{x})+\bar{x} \tag{6}$$

$$y'(u) = \frac{2R-D(u)}{D(u)} \cdot (y(u)-\bar{y})+\bar{y} \tag{7}$$

$D(u)$ is the distance between the center of the circle and the current shape pixel. If the position of a shape pixel and the center of the circle coincide, they cannot be mapped. If this is the case, the shape pixel will be interpolated from adjacent shape pixels of the mapped shape.

The *matching* compares the standard and the mapped CSS shape descriptors to pre-calculated object descriptions stored in a database. The results of the matching are aggregated for all frames of the shot. This adds robustness to the approach since unrecognizable single object views occurring in the video are insignificant compared to the entire shot. A detailed description and evaluation of the segmentation and classification algorithm can be found in [18, 38].

3.5 Face Detection

Persons are very important in most types of videos. Close-up views of the faces of main actors are important in feature films, whereas documentaries often feature athletes, politicians, etc. Face areas are one of the semantic features used for specifying interesting regions in a frame.

Rowley, Baluja and Kanade [65] have developed a famous, very reliable neural network face detector. It detects about 90 % of the frontal faces in a video. Non-face areas (i.e., false hits) are rare. We have implemented the face detector and trained the network with $7,500$ faces. We were able to reproduce the good detection results and extended the algorithm to detect slightly tilted faces (+/-30 degrees).

The faces are tracked within a shot. It allows us to find single skipped faces and removes most of the false hits (mis-classified face regions). For each detected face, its position and size in the next frame are estimated from the global camera motion. Tracking increases the reliability of the face detection algorithm with only a small increase of computation time.

3.6 Text Recognition

Superimposed text in videos has some useful properties:

- horizontal alignment,
- significant luminance difference between text and background,
- the character size is within a certain range,
- typically monochrome text,
- text is visible in consecutive frames,
- a horizontal or vertical motion of text is possible.

Our text detection algorithm identifies candidate text regions first and validates these regions in the following steps. It first searches the DCT coefficients of the macro-blocks for strong vertical, horizontal and diagonal frequencies. The bounding rectangle of connected blocks, that can be tracked through consecutive frames for at least one second, are marked as text area.

Next the exact boundaries of the text regions are detected. The gradient magnitude of each pixel is summed for each horizontal line of the bounding rectangle and a *horizontal projection profile* is calculated (see Fig. 6). Two significant peaks of the summed values indicate the exact position of the text line, e. g., the base and top line in a text with large capitals. The color of the detected text is used for validation. A detailed description of the text recognition algorithm can be found in [39].

Fig. 6 Example of a horizontal projection profile to detect the boundaries of text lines.

Fig. 7 From left to right: contrast map, quantized contrast map, ROI, bounding box of ROI in original image.

3.7 Regions of Interest

We use the approach presented by Ma and Zhang [56] to calculate a contrast-based saliency map and estimate regions of interest (ROI). Their general assumption is that image regions that are relevant for an observer have a high contrast. The *contrast map* of an image of size $n \times m$ is defined as:

$$C_{i,j} = \sum_{q \in \Theta} d(p_{i,j}, q) \qquad \forall i = 0, \ldots, m-1, \quad j = 0, \ldots, n-1. \tag{8}$$

$C_{i,j}$ is the contrast value at pixel position (i, j), $p_{i,j}$ is the color of the current pixel, and $q \in \Theta$ is a pixel in the local neighborhood Θ of (i, j). We use the following color distance function d derived from the Gaussian distance:

$$d(x, y) = 1 - e^{\frac{-\|x,y\|}{2\sigma^2}}, \tag{9}$$

where $x = (x_0, x_1, x_2)$ and $y = (y_0, y_1, y_2)$ represent the color values of two pixels in the *LUV* space, and $\|x, y\| = \sqrt{(x_0 - y_0)^2 + (x_1 - y_1)^2 + (x_2 - y_2)^2}$ is the Euclidean distance between these color values. σ defines the sensitivity of the contrast map; the neighborhood Θ specifies whether the contrast map should highlight small details like differences of single pixels or larger regions. The contrast map is smoothed with a Gaussian kernel to reduce noise and to remove small holes in objects which makes clustering of high contrast values easier.

In the next step, the contrast values are quantized. Region growing is applied to the *quantized contrast map* to identify regions with high saliency. In many cases, the contrast map highlights the relevant regions and objects in an image well. This is sometimes compromised when applying the fuzzy growing technique presented in [56] to the contrast map. Instead of using fuzzy growing, we implemented a color quantization technique based on the Linde-Buzo-Gray (generalized Lloyd) algorithm [49]. The values of $C_{i,j}$ are mapped to a fixed number of new values which define the *codebook*. The values in the codebook should be selected, so that the average distortion caused by the mapping is minimized. To achieve this, the codebook must fulfill two constraints: A contrast value must be mapped to the nearest value in the codebook, and each codeword must be the centroid of contrast values mapped to it. By defining a fixed number of contrast values (the number of elements in the codebook), a local minimum is estimated in an iterative way. This defines the mapping of each pixel and results in a quantized contrast map.

Region growing is applied on the quantized contrast map to cluster similar contrast values into a region. Regions with high contrast values define a region of interest (ROI). A rectangular bounding box describes a ROI. Fig. 7 shows the steps to detect the region of interest.

4 ROI-based Video Retargeting

To achieve the spatial reduction it is possible, to either scale the frame, to crop parts or to combine both methods. An advantage of scaling is that all parts of the full frame are still visible. Relevant parts may be lost if a border is cropped. On the other hand it is possible that content in a scaled frame is downscaled beyond recognition This can happen with text that is scaled down too much. Scaling also reduces the recognizability of other content like people or objects and many details may be lost. Non-uniform (anisotropic) scaling with changes in the aspect ratio usually causes disturbing effects and reduces the perceived visual quality significantly.

The following ROI-based video retargeting approach guarantees a well-balanced trade-off between scaling and cropping: Identify the most relevant regions in each shot, crop borders and scale the cropped frames to the target resolution. Our algorithm utilizes the following four heuristics [41]:

- Regions with relevant semantic content should be clearly visible in the target video. A region should not be part of the adapted video if the semantic content is no longer recognizable due to its limited resolution. For instance, text in an image is worthless if the character size is too small to be readable, or if characters are truncated.
- Regions without expressive content should not be visible in the target video. Dark borders (e.g., black stripes in wide screen movies) or large monochrome regions (e.g., sky) adjoining an image border are typical irrelevant regions.
- The selected region (*selected window*) is scaled to the appropriate screen resolution. The aspect ratio of the selected window should be identical to the one of the target video.
- Maximize the visible information in each frame. An artificial camera motion is inserted if several important regions are detected. A continuous modification generates an artificial camera motion; a sudden shift of the selected window produces and artificial hard cut.

The principle behind our approach is to select a rectangular region (*window*) within the video and scale it down to its target size if necessary. The position of the selected region is not fixed in a shot, and even artificial camera motion within a larger frame of the original video is possible. If a person is shown in a shot, the algorithm might begin with the full video frame and then zoom to the person or focus on the face of the person. All the important details will still be visible in the zoomed version despite the lower resolution of the video.

Regions of interest created from high-level semantic information and contrast-based saliency maps are extracted to enable the identification of relevant regions first (see Chapter 3). Next, we estimate the position and size of the selected window within a frame, so that the visible information, i.e. text, faces, objects, and contrast-based saliency is maximized. To avoid jitter or jumps between consecutive frames the selected windows are aggregated at the shot level. The window may follow a continuous motion (pan/tilt) or scaling operation (zoom) within the larger frame of the original video. In the following, we present a detailed description of our approach.

4.1 Importance Value of the Selected Window

Each semantic feature (text, faces, people, and moving objects) and the contrast-based saliency region represent important information in our terms. In the following, we use the term *saliency region* to specify one of these regions. For each saliency region, an importance

value is defined that characterizes the relevance of the region. The *importance value of the selected window* is defined as the sum of importance values of its saliency regions. We aim to estimate the position and size of the selected window in a frame, such that the overall importance value of this window is maximized in the target video. We specify the following design goals:

- The importance value of the selected window should be maximized,
- up-sampling (enlargement) of saliency regions is not allowed, and
- the aspect ratio of the region must match the target video size.

We assume that the importance of a saliency region is proportional to the size of its bounding box. The size of the content in the target video depends on the scaling factor, which on the other hand depends on the size of the cropped border (a large border reduces the amount of scaling). If the size of a saliency region drops below a certain threshold, the information it contains is no longer relevant. E.g., text with a character size is smaller than a certain value becomes unreadable. If this is the case, the importance value of this text region is set to zero. A threshold defines the *minimum perceptible size* of each type of saliency region. In addition to a lower threshold, an upper threshold may be required which defines a *maximum reasonable size* of a saliency region. Very large text does not necessarily contain a large amount of information. The size of the characters should thus be kept within a certain range.

Based on these conditions, the *importance value* $V \in [0;1]$ of a text region with height H is defined as:

$$V = \begin{cases} \frac{H_{max}}{H} & H > H_{max}, \\ \frac{H}{H_{max}} & H_{min} \leq H \leq H_{max}, \\ 0 & H < H_{min}. \end{cases} \tag{10}$$

The threshold values for the minimum perceptible size H_{min} and the maximum reasonable size H_{max} define the minimum and the maximum character height. These values depend on user preferences and the screen used to watch a video.

If parts of saliency regions in the cropped frame are no longer visible (e. g., some characters in a line of text), the importance value of this region is ignored. This is considered during the parameter estimation of the selected window in the following steps (see Equ. 12).

The importance value of the other saliency regions (faces, people, objects, contrast-based ROI) also depends on their size. The heuristics are analogous although an upper threshold that limits the size of an area is not required, and the importance value is proportional to its size. Note that it is never allowed to upscale (zoom-in) the spatial resolution of a video. As with text areas, the importance value is also set to zero for partly visible saliency regions. Fig. 8 (left) shows an example containing three automatically detected saliency regions (contrast-based saliency was not used in this example).

4.2 Parameter Estimation of the Selected Window

The summarized importance value $V(W)$ of the selected window W aggregates the importance values V_i of all saliency regions S_i that are contained entirely in the selected window:

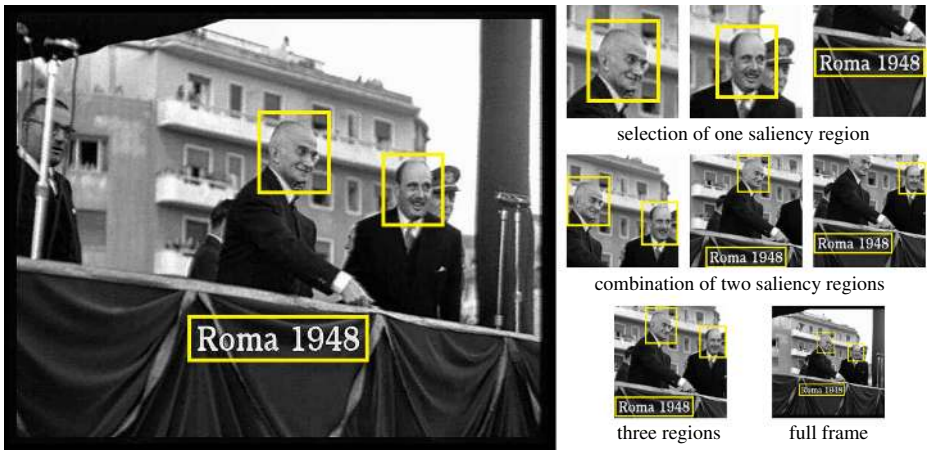

Fig. 8 Sample frame of a historical video with three automatically detected saliency regions. Eight possible combinations of these regions are analyzed in order to find the selected window that maximizes the importance value.

$$V(W) = \sum_i \alpha_i \cdot L_i(W) \cdot V_i, \tag{11}$$

$$L_i(W) = \begin{cases} 1 & S_i \in \mathrm{W}, \\ 0 & \text{else}. \end{cases} \tag{12}$$

The factors α_i are weights for certain types of saliency regions to model the increased importance of for example faces in an image. The binary variable $L_i(W)$ signalizes whether the saliency region i is entirely included in the selected window W. We have implemented a fast algorithm that calculates the maximum of $V(W)$ and detects the size and position of the optimal selected window W^*:

$$W^* = \arg\max_W \{V(W)\} \,. \tag{13}$$

Each border of the optimized selected window must match a border of at least one of the detected saliency regions so that this region is inside the selection. If the window includes only a part of a saliency region, the importance value of this region is set to zero and does not increase $V(W)$ anyway. On the other hand, a higher scaling factor is required if the size of the selected window increases, leading to a decrease of the importance values of all saliency regions within the window and $V(W)$ is not maximized.

Based on these observations, it is easy to calculate the best position and size of a selected window: All combinations of saliency regions are tested. The current selected window is defined as the bounding box of the chosen regions. The full frame is used if no saliency region is selected. The importance value $V(W)$ is calculated after scaling window to its target size; the position and size of the optimal selected window W^* and its value $V(W^*)$ are stored.

A large number of saliency regions entails a significant increase in the computational effort. The complexity to estimate the parameters of the selected window is $O(2^N)$, where N is the number of saliency regions in a frame. Since the number of saliency regions in

real video is usually rather low, this complexity does not constitute a problem. In the rare event that many saliency regions are detected in a frame, only the largest eight regions are considered. Usually, the aspect ratios of the selected window and the target video do not match. To fix this, the height or width of the selection is enlarged until the ratios fit.

An example for the combination of the saliency regions is shown on the right side of Fig. 8. In the top row, the selected windows contain one saliency region, the windows in the middle combine two, and the ones at the bottom all three saliency regions. Without cropping the borders (example on the lower right), it is difficult to read the text. The importance value of the selected window correlates well with the visual quality, and the combination of scaling and cropping produces significantly better results than scaling alone.

4.3 Aggregation of Selected Windows

Although we have calculated the region that maximizes the importance value of a frame, it is probably not the best selection for the target video. Independent selection is not suitable for videos because small variations of the size or position of a region between adjacent frames lead to a disturbing jitter effect.

Thus the size and position of the selected window are adjusted according to the neighboring frames. A linear function is fitted to the height of the windows for all frames in a shot. The width is then determined by the aspect ratio of the target video. The window position is treated similarly: The horizontal and vertical position values are approximated by two linear functions. We note that a similar recapturing is often done manually when wide-screen cinema films are edited for television.

4.4 Artificial Camera Operations

Cinematographic rules state that subsequent camera operations (e.g., a zoom-in followed by a pan followed by a zoom-out) should be used with great care. We establish the rule that two shots with camera operations should be separated by at least one shot with a static camera. There are two options when selecting the visual region for the target video: To choose a selected region that maximizes the importance value or to select the original frame. Both regions must be scaled to fit to the target resolution of the video.

We have included a third option that adds a virtual camera motion. We apply the following procedure if two selected windows of high relevance are detected: If the size and position of these regions do not change significantly within a shot, one window is selected for the first frame and the second window is used for the last frame of the current shot. Linear transitions considering the size and position of both selected windows are calculated if the distance between them is under a certain threshold. Otherwise, an artificial hart cut is inserted in the middle of the shot. This option will not be used if a camera operation was detected or artificially inserted in the previous shot.

The artificial camera motion can highlight different saliency regions over time, but also show more details within a large saliency region. Imagine a scene where the importance value of a face is similar to the one of the entire person. The virtual camera motion could create a shot that shows the entire person first followed by a zoom-in on the face.

(a) (b) (c) (d)

Fig. 9 Original image (a). The optimal seams in the first iterations are marked in (b). Image adapted to 35 % of the original width based on seam carving (c) and scaling (d).

5 Improved Seam Carving for Video Retargeting

5.1 Seam Carving for Images

The seam carving technique was proposed by Avidan and Shamir [1] to enable a content-aware adaptation of the resolution of images. Their idea was to identify and remove a path (*seam*) of pixels with low relevance for the content of an image. Each removed seam causes a reduction of the image size by one, where *vertical seams* are used to change the width of an image and *horizontal seams* to change its height. Fig. 9 shows an example of vertical seams that are detected in each iteration. The optimal seams usually avoid foreground objects.

A *vertical seam* $\mathbf{s} = \{s_i\}_{i=1}^{H} = \{(x(i),i)\}_{i=1}^{H}$ of an image with height H is defined as a path of seam pixels s_i from top to bottom with the following constraints:

1. One and only one seam pixel is selected in each row.
2. The horizontal distance between two adjacent seam pixels $|x(i) - x(i-1)|$ does not exceed a threshold T. If $T = 1$, seam pixels of vertical seams are vertically or diagonally connected (8-connected).

These conditions define a *vertical seam*:

$$\mathbf{s} = \{s_i\}_{i=1}^{H} = \{(x(i),i)\}_{i=1}^{H}, \text{ s.t. } \forall i = 2 \ldots H : \; |x(i) - x(i-1)| \leq T. \tag{14}$$

A *horizontal seam* is an 8-connected path from left to right which enables the change of the height of an image. It is defined in a similar way by switching rows and columns of an image. An *energy function* defines which pixels should be selected for a seam. The optimal seam $\mathbf{s}^*$ is defined as seam with minimum cost based on an energy function E:

$$\mathbf{s}^* = \arg\min_{\mathbf{s}} E(\mathbf{s}). \tag{15}$$

Avidan and Shamir [1] have evaluated the performance of several energy functions and recommended using the gradient magnitude. A significant visual improvement can be achieved when *forward energy* is used as presented by Rubinstein et al. [66]. This new energy criterion looks ahead in time and considers the energy brought into the destination image by removing seams.

The *cost of a seam* $E(\mathbf{s})$ is defined as the sum of all absolute gradient values of path pixels. Dynamic programming is used to identify the optimal seam. Depending on the desired size of the adapted image, one optimal seam $\mathbf{s}^*$ is identified in each iteration. The path pixels

of the optimal seam are removed, and all pixels are shifted horizontally or vertically to fill the gap.

Compared to scaling, a major advantage of seam carving is the fact that changes of the aspect ratio are handled very well. The method also focuses on preserving the important content of the image. However, large objects like buildings lead to clearly visible artifacts.

5.2 Quality Improvements for Seam Carving

Errors caused by seam carving are especially disturbing when the original image includes regular patterns or straight lines [34]. These lines may become curved or disconnected in the target image. Fig. 10 shows an example of an image with straight lines which has been adapted through seam carving. The distortions on the wall are clearly visible. These distortions may occur if a seam crosses a non-vertical or non-horizontal line.

Fig. 10 Left: Original image. Right: Image adapted to 60% of the original width using seam carving (forward energy used). Straight lines become curved and cause clearly noticeable errors.

When, for instance, a vertical seam is removed from an image, all the pixels on the right side of the seam are moved one pixel to the left in order to close the gap. If this is repeated and several seams are crossing a straight line in adjacent intersection points, a distortion becomes visible when the pixels are moved up (see Fig. 11).

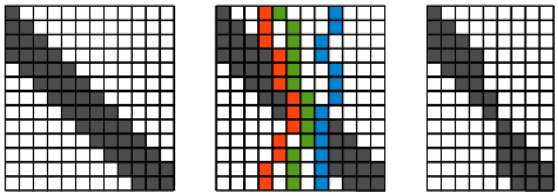

Fig. 11 a) A straight line. b) Three seams crossing the line. c) An obvious distortion occurs by removing adjacent seams.

In the following subsections, we introduce an improved version of the seam carving algorithm for images containing straight lines.

5.2.1 Workflow of the Algorithm

In most images, an optimal seam crosses one or more objects (e.g., buildings, streets, tree trunks, cars, street lights) and deforms straight lines. It is not our goal to shift an optimal seam so that the seam does not cross an object at all. By updating the energy map, we slightly adjust the paths of the optimal seams in the next iterations so that visible distortions are reduced. Slight changes to seam pixels ensure that optimal seams do not cross a straight line in *adjacent* pixel positions. If the intersection points are scattered all over the line, the distortion is distributed and becomes less obtrusive.

Our algorithm improves upon the regular seam carving algorithm by employing line detection. Each time a seam crosses a line, the energy in the local neighborhood of the intersection point is increased in order to reduce the probability that other seams cross the line in adjacent pixel positions. We apply the *Canny edge detector* [7] to identify significant edges. Afterwards, the edge pixels are transformed into *Hough space* [15] to locate pixels on straight lines. Edge pixels that are located on straight lines will be called *line pixels* in the following.

The only required change to seam carving lies in the computation of the energy map. For each selected optimal seam, the intersection points of optimal seams and line pixels are detected, and the energy map is modified in the local neighborhood of all these points. Fig. 12 gives an overview of the algorithm. The details of the line detection algorithm and the improvements of seam carving are presented in the following sections.

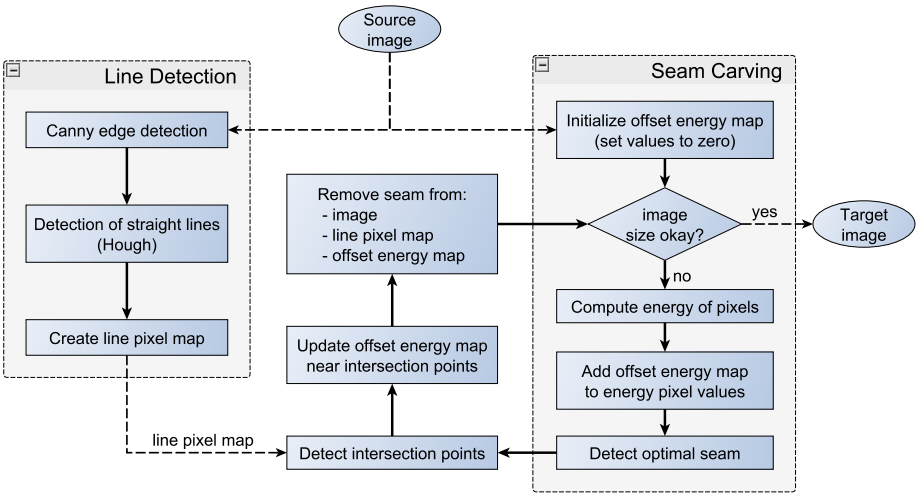

Fig. 12 Overview of the workflow of the improved seam carving algorithm

5.2.2 Detection of Straight Lines

Edges in the image are detected by the *Canny edge detector*. Edge pixels are transformed into *Hough space* next. Each point in Hough space corresponds to a straight line in the edge image. Only the most significant straight lines corresponding to Hough pixels that exceed a threshold are considered.

For each line candidate, the edge pixels coinciding with the line are counted. An edge pixel is considered as line pixel, if the distance between edge pixel and line is below a threshold, and if the line segment has a minimum length. Small gaps between valid line segments are filled up. Because the accuracy of the orientation of the detected lines is not sufficient, we use a gradient descent algorithm to optimize the parameters of a line by maximizing the total number of line pixels on each line.

5.2.3 Modification of the Energy Map

The seam carving algorithm starts by computing the gradient magnitude at each pixel to be used as its energy value. We also use forward energy as proposed by Rubinstein [66]. An additional energy map that adds an offset value to each gradient is initialized to zero. All seams are calculated using the energy values and the optimal seam is selected.

Along the path of the optimal seam, the intersection points of the seam and the detected lines are marked. The *offset energy map* is increased in the local neighborhood of these points. When calculating the optimal seam in the next iterations, the values stored in the offset energy map are added to the gradient magnitudes. This stimulates that seams avoid intersection points in the following iterations. Fig. 13 shows the modification of the energy map.

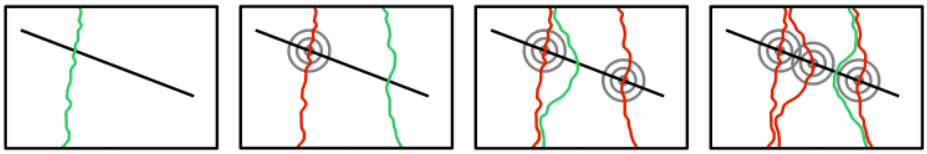

Fig. 13 Visualization of the basic idea of the modification of the energy map: The optimal seam (green) of one iteration may cross a straight line (black). After the removal of the seam (the removed seam is marked in red), the energy in the local neighborhood of the intersection point is increased. This decreases the probability that an optimal seam crosses these areas in the next iterations.

The intersection point of the offset energy map is increased by a value of 200, and adjacent pixels in a region of 7×7 pixels are increased according to a 2D Gaussian distribution. After the modification of the offset energy map, the pixels of the optimal seam are removed from the image, line pixel map, and offset energy map. The algorithm stops after a sufficient number of seams have been removed to reach the target image size.

5.3 Seam Carving for Videos

As Rubinstein et al. presented in [66], the quality of adapted videos is very low if seam carving is used on each frame separately. This leads to visible artifacts, and the video becomes blurred and shaky. These artifacts are caused by small differences in consecutive frames, like lighting changes, object or camera motion, noise and compression errors. The seam carving algorithm is very sensitive to such changes. Even with a static camera, small differences in pixel values lead to different optimal seams.

Rubinstein et al. [66, 73] proposed an extension of the seam carving technique for videos which removes 2D seam manifolds from 3D space-time volumes. They use graph cuts instead of dynamic programming (see Fig. 14). Whereas both are used to find the seams with

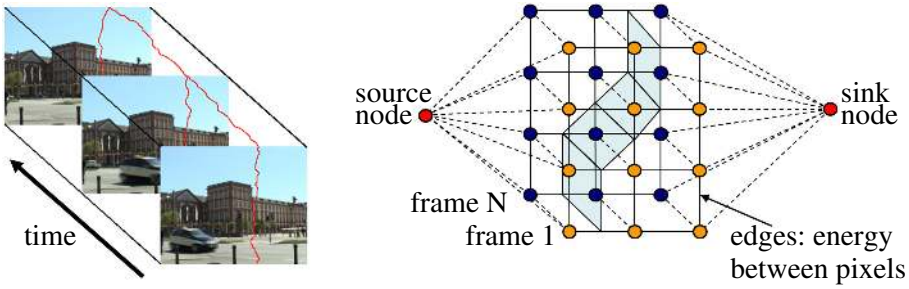

Fig. 14 Left: The frames of a shot define a 3D cube. Right: Each pixel of the cube is connected to other pixels in spatial and temporal dimension. The large number of edges in this graph cause the high computational effort of the graph cut algorithm.

low overall energy values, the latter one does not work for videos. A seam is given by a minimal cut in the graph and has to be monotonic and connected in order to be valid.

The major disadvantage of this extension to videos is its high computational complexity. The graph cut algorithm takes a lot of processing time and memory, making it unusable on HD or even PAL videos if all pixels of the video are considered as nodes in the graph. Even for short sequences, the total amount of data required for the graph cut approach cannot be handled in memory [43]. Therefore, the graph is approximated temporally and spatially, and refined in several iterations. In spite of this optimization, the time to adapt a video is still significantly higher compared to scaling, cropping or our *fast seam carving for videos* (*FSCAV* [43]) algorithm presented in the following section.

5.4 Fast Seam Carving for Videos (FSCAV)

Similar to the ROI-based video retargeting, the *FSCAV* algorithm operates on the level of shots. Even if we adapt full-length videos, the algorithm considers all shots separately. We put the focus on the following requirements:

1. An optimal seam should be *robust*, that is to say the removal of a seam should not lead to a blurred or shaky video.
2. To limit the computational effort and memory consumption, 1D seams should be calculated in 2D images instead of computing 2D seam manifolds in 3D space-time volumes.

We first assume, that the video to adapt contains no object motion: Changes are only caused by camera motion. The first requirement is valid if a pixel on the optimal seam represents the same visual content in all frames (we call it a *robust seam*). If a robust seam is deleted, the same object regions are removed in all frames, and the video does not become shaky. *FSCAV* tries to identify robust seams by estimating and compensating the camera motion between all frames.

Knowing the camera motion also makes it possible to adhere to the second requirement: Instead of searching 2D seam manifolds in a 3D cube, we compensate for the camera motion between consecutive frames. This enables the aggregation of pixel and energy values into a single image. The seams of the aggregated image are robust seams, because they can be mapped back into all frames of the video by applying the inverse camera motion. This guarantees that optimal seams describe the same image content in all frames. We present details of our *FSCAV* algorithm in the following and consider three major challenges:

Fig. 15 Top: Three sample frames of a video sequence with horizontal pan. Bottom: Background images constructed with 20 frames based on the median (left) and foreground objects (right).

1. How can the algorithm avoid that an optimal seam removes too many pixels from moving foreground objects?
2. What happens if a seam is not visible in all frames of a sequence (e.g., in case of a camera pan)?
3. How many seams can be removed from a video without visibly impairing its quality?

5.4.1 Aggregation of frames

The camera motion parameters estimated during the video analysis step are used to build a *background image* (see Section 3). By applying a median filter, moving foreground objects are no longer visible in the background image. They are not considered at all. To reduce deformations in moving objects, we use segmented object masks derived in the video analysis step: All object pixels are copied into the background image which is used for the identification of optimal seams. An energy map based on the gradient magnitude of each pixel is calculated for this background image. Fig. 15 shows an example of a background image and an image with foreground objects.

5.4.2 Identification of robust seams

To identify *robust seams*, we apply the seam carving algorithm to the energy map of the background image (including objects) and obtain a list of suitable seams. Iteratively, the seams are mapped to the individual frames of a shot by applying the camera motion model with inverse parameters. However, the direct utilization of the mapped seams is problematic as will be explained soon.

The characteristics *visibility* and *completeness* of a seam are introduced to validate whether a mapped seam is suitable or not. A seam is *visible* if it is included in all frames of the shot. A seam is *complete* if a pixel is assigned in each row (vertical seam) or in each column (horizontal seam). This corresponds to the first constraint of the definition of a seam. We classify a seam as *robust seam* if it is both *visible* and *complete* in all frames.

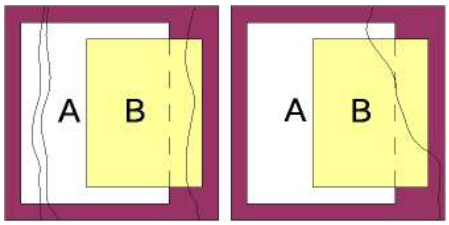

Fig. 16 Example of seams that are *not visible*. Left: The seams are located in frame *A* or *B*, but not in both frames. Right: The vertical seam does not pass through frame *A* or *B* from top to bottom.

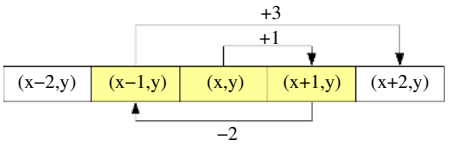

Fig. 17 Example of the search for unoccupied pixels in case of a horizontal seam: If the position of a robust seam pixel is already occupied (yellow pixels) by another seam, a new and unused position for this pixel is found.

- *Visible seams:* In case of camera motion like a pan or tilt, it may occur that the image content covered by a seam is *not visible* in all frames. Fig. 16 (left) shows such an example, where two seams are mapped to only frame *A* while another is only visible in frame *B*. In Fig. 16 (right), it is not possible to map the seam of the background image into frame *A* or *B* because it does not pass through the frame from top to bottom. Seams that are not visible are ignored to avoid shaky videos.
- *Occupied pixels:* Another typical problem is caused by rounding errors or inexact parameters of the camera model. In this case, two different seam pixels of the background image may be mapped to the same pixel in a frame. Replacing the previous pixel or removing the new pixel would lead to gaps in seams and should be avoided. Fig. 17 illustrates our approach to detect the next unoccupied pixel position. Because a pixel (x,y) is already occupied, the adjacent pixel $(x+1,y)$ and $(x-1,y)$ are checked next. The search is implemented by using a counter which changes its sign and increases its absolute value by 1 in each step.
- *Incomplete seams (gaps in seams):* In case of a camera zoom, it is possible that horizontal or vertical gaps appear in seams due to the mapping from the static background image. These gaps appear periodically and cause frayed edges in the adapted frames. The gaps are filled by interpolating seam pixels.
- *Fast camera motion:* Some camera operations like long camera pans are a great challenge for this algorithm. Robust seams cannot exist if the first and the last frame of the shot do not share any visual content. To avoid this problem, the sequence is split in the middle if the total number of robust seams is insufficient to achieve the desired resolution of the video. Both video segments are then processed separately. We split the sequence so that the video segments overlap by 0.5 seconds. The overlap is necessary to fade from the first to the second segment and to reduce the visible error at the transition of the segments. Each segment is split recursively until a sufficient number of robust seams are detected.

6 Selection of the Retargeting Algorithm

Comparing the two adaptation algorithms that were introduced, major differences become obvious. The ROI-based technique always selects a window within each frame. Good visual results can be achieved in target videos when saliency regions are close together. On the other hand, the full frame is scaled or relevant parts are cropped, if saliency regions are

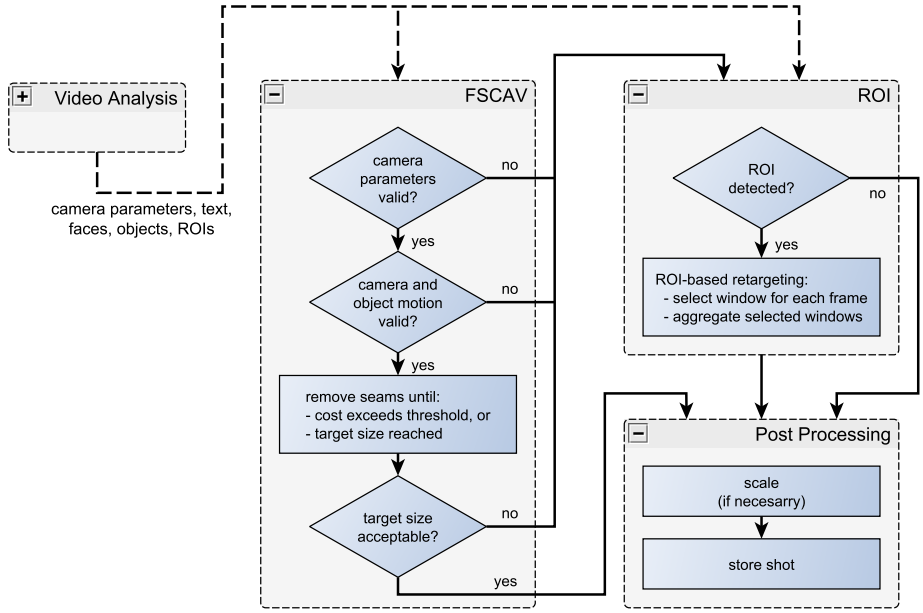

Fig. 18 Selection of a suitable video retargeting algorithm.

located at image borders. In this case, seam carving usually improves the quality of the target videos significantly.

Fig. 18 gives an overview of our approach to select a suitable adaptation algorithm. After the video analysis step, seam carving is applied to a shot. A heuristic measures the visual quality of the target video. In case of insufficient quality, ROI-based retargeting is applied instead.

The main question we investigate is how much removal of the next seam reduces the image quality. Although shaky videos were not observed, other obvious image errors may occur by deformations of foreground objects. From empirical observations we derived three heuristics to identify typical errors.

The first heuristic validates the *parameters of the camera model* to guarantee a correct background image. Errors can occur in large foreground objects or when an insufficient number of characteristic features is found in the image background. It is checked whether the parameters are within a characteristic interval or whether a parameter changes considerably between adjacent frames. The binary variable $C_{i,i+1}$ the correctness of the parameters of the camera model $(t_x, t_y, a_{i,j}, p_x, p_y)$ between frames i and $i+1$:

$$C_{i,i+1} = \begin{cases} 1 \text{ if } & |\frac{t_x}{W}|, |\frac{t_y}{H}| \leq 0.05 \ \wedge \\ & \text{rotation angle} \leq 5 \text{ degrees} \ \wedge \\ & \text{scaling factor} \leq 4 \text{ percent} \ \wedge \\ & |p_x|, |p_y| \leq 10^{-5} \\ 0 & \text{else.} \end{cases} \tag{16}$$

The thresholds were estimated empirically; W and H are the width and height of a frame. We switch to the ROI-based video retargeting technique if the parameters of at least one pair of adjacent frames are invalid.

The second heuristic checks the *direction of camera and object motion*. The visual quality of the target video highly depends on the direction of motion: If the camera or objects move in parallel to the seams, the quality is usually very good. On the other hand, when objects move orthogonally to the direction of a seam, different parts of the object are removed in each frame, and the visual quality drops significantly. This will be further discussed in the evaluation section of this article. A similar phenomenon occurs with orthogonal camera motion: The number of robust seams that are detected in all frames of a shot are very low. The shot is split in the middle, and each segment is processed separately. Although a fade reduces segment transition effects, a quality degradation is still visible. These observations lead to the following rules:

1. Seam carving should not be applied if the average number of seams an object moves through exceeds a threshold.
2. In case of orthogonal camera motion, a shot is (iteratively) split into shorter segments. If the length of a segment drops below a threshold, the ROI-based retargeting technique is used instead.

The third heuristic analyzes the *quality of a frame* after the removal of the next optimal seam (this step is repeated until the target size is reached). We assume that that pixels from relevant objects have been deleted and errors become obvious when seams with high cost values are removed. No more seams are deleted if the cost $E(s_i^*)$ of the optimal seam in iteration i exceed a threshold. The threshold T_S depends on the cost of the seams in the original image:

$$T_S = \alpha \cdot E(s_1^*) + (1-\alpha) \cdot E(s_1^{MAX}) \tag{17}$$

$E(s_1^{MAX})$ is the maximum cost of a seam in this iteration. The costs are weighted by a parameter α. After the removal of the first i seams ($E(s_i^*) \leq T_S < E(s_{i+1}^*)$), the image size is validated. The shot is scaled if the required scaling and especially the anisotropic scaling is within user defined parameters.

7 Evaluation

This section presents user evaluations and experimental results for the proposed retargeting algorithms. We examine the detection of saliency regions and ROI-based retargeting first, and then focus on the fast seam carving technique.

7.1 ROI-based Image Retargeting

Opposed to watching videos, users looking at a static image for some time makes it easier to recognize errors in the spatial domain. In addition, we did not only apply the fully automatic algorithms to detect ROIs, but also developed a software tool that enables users to modify, delete, or add saliency regions in a convenient way. This makes it possible to evaluate the ROI-based retargeting algorithm without considering erroneous saliency regions from the analysis step. The saliency regions identified by this semi-automatic approach, are assumed to always be correct.

We have carried out two user studies to evaluate the quality of the ROI-based adaptation system. Questions regarding software ergonomics, quality of the adapted content, and

perceived benefit of the adaptation were asked. The general feedback to the implemented adaptation technique and the software tool is positive. Especially the combination of automatic and semi-automatic algorithms and the ease of correcting the automatically generated metadata is a great benefit of this system.

We simulated 20 different handheld mobile devices with resolutions ranging from 160×160 to 640×480 pixels. In a first test, ten users evaluated the quality of the automatic adaptation algorithm without semi-automatic refinement. Each person examined images adapted to randomly selected devices. The users had to grade the quality on a scale from 5.0 (excellent) to 1.0 (insufficient). The average value for the automatic adaptation of images is 3.6. Two users classified the results barely adequate. The reason for this low rating was the fact that the quality of some adapted images was very poor especially if important objects were missing or if parts of them had been cropped.

Each test user had to correct or add the automatically detected saliency regions in a second evaluation. The adapted images were presented again, and the quality was considered to be much higher this time (average value of 4.1). Some users mentioned the necessity of a minimum border around each saliency region (especially for faces). This observation was incorporated into ROI-based video retargeting. The image quality is acceptable for two users; the other users grade it much better (between very good and excellent).

7.2 ROI-based Video Retargeting

The most critical aspect of the ROI-based video retargeting is the reliable detection of saliency regions. Missing or incorrectly classified regions cause missing or truncated text, objects, people, or faces in the target video.

The *shot boundary detection* is quite robust: For most video genres, both precision and recall exceed 95 percent on average. This value slightly drops for advertisements, music clips, and some scenes in action movies. The estimation of the *camera motion* is sufficiently precise. Very large objects (e.g., a close-up view of a face) make it impossible to estimate the camera parameters correctly. Using Equ. 16, all invalid camera parameters are detected in the test sequences.

The detection of *moving objects* is much more fault-prone. The recognition rate in shots with one car or one person is acceptable (about 40 %). It is much lower for planes or boats due to the changing background (water) or the low number of edges in the background. The probability of a correct object classification decreases significantly if:

- more than one object moves in the shot,
- the object is very large,
- the background is blurred or noisy,
- luminance changes within the shot, or
- the object is partially occluded.

Our *face detection* system locates more than 90 % of the frontal faces with a minimum height of at least 25 pixels. The detection of *contrast-based saliency regions* works very well in most cases. A disadvantage of this algorithm is the high sensitivity to threshold values. Minor changes of the parameters cause significantly smaller or larger regions.

7.3 Retargeting by Seam Carving

7.3.1 Quality Improvements

In the following, we briefly evaluate our enhanced seam carving algorithm and compare the quality of the adapted images to the quality of regular seam carving with forward energy. In the case of landscape images where no straight lines are detected, traditional seam carving and our enhanced approach lead to identical results. However if the image contains diagonal lines in particular, the quality achieved by our version is significantly better. This can be seen in Fig. 19. The red ellipses mark distortions introduced by regular seam carving.

Fig. 19 Left: Original. Center: Seam carving with forward energy. The red ellipses mark warped lines. Right: Our result. The images have been reduced to 60% of their original width.

7.3.2 Fast Seam Carving for Videos (FSCAV)

To evaluate the quality of the *FSCAV* algorithm, we compare videos adapted by scaling, cropping and *FSCAV*. 45 video sequences (single shots) from television, the Web or recorded with a HD camcorder have been selected. The resolution of the sequences varies between PAL (720×576 pixel, 25 fps) and HD resolution (1920×1080 pixel, 25 fps).

Due to the fact that *FSCAV* is particularly sensitive to object motion, the video sequences have been grouped into 5 categories: *static* (no camera motion, average number of object pixels is less than 1 percent of all pixels), *camera motion only*, *low object motion* (1-10 percent object pixels), *high object motion* ($>$ 10 percent), and *very large objects* occupying at least 50 percent of the height or width of an image. Camera motion occurs in all but the static sequences. Table 1 gives an overview of the sequences in each category.

Table 1 Overview of test sequences

Type of Sequence	Number of Videos Sequences	Length [frames]
Static	5	40 – 120
Camera motion only	12	60 – 250
Low object motion	15	50 – 500
High object motion	11	90 – 260
Very large objects	2	100 – 250

One video of each category was chosen for the evaluation. Ten students between 21 and 24 years evaluated the test series by watching the original video sequence first and then the three adapted versions. The order of the adapted sequences was unknown to the users and changed with each test series. The width of each video was reduced by 45 percent (to 400×568 pixels from PAL resolution). In this evaluation, we did not consider failings caused by an incorrect detection of saliency regions. Therefore, we manually defined the optimal border to be used for cropping. The subjects filled out a questionnaire and answered the following questions:

- How well are details preserved?
- Which kind of disturbing effects did you recognize?
- Which visual errors did you recognize?
- What is your overall impression of the adapted video?

Answers were given on a scale from 5 (excellent) to 1 (insufficient), and additional comments were collected. Another task was to sort the adapted sequences of each test series by quality.

Cropping always leads to a loss of relevant content in the adapted video and achieved the worst results in the evaluation (*sufficient*). Judging the quality of *scaling* and *FSCAV* is difficult: Although the average quality of *FSCAV* is better than scaling – between *good* and *very good* – the grades of the individual sequences differ by a lot. *FSCAV* is significantly better when object and camera motion are relatively low (*static*, *camera motion only*, and *small object motion*) and ranges between *excellent* and *very good*.

The visual quality of sequences with high camera or object motion depends on the direction of the motion: The quality of *FSCAV* is *very good* when the camera or the objects move in parallel to the seams and decreases with orthogonal motion. The best case occurs

when the seams are uniformly distributed causing an effect similar to scaling (see Fig. 20 (d)). If many seams gather in a small area, a moving object is significantly warped in this region. This reduces the perceived average quality to *sufficient*. The quality is *insufficient* when large objects cover a major part of a frame. Test persons ranked these sequences even lower than cropped videos.

All conclusions drawn from the user evaluation were considered for the selection of a suitable retargeting algorithm. First, the average visual quality of *FSCAV* is better compared to scaling or cropping, therefore *FSCAV* is used as default algorithm. Only when *FSCAV* fails, the ROI based method is chosen.

Fig. 20 shows sample video frames from each category and their adapted versions. In a static sequence (a), *FSCAV* preserves the windows and the tree much better compared to the scaled frame. In (b) and (c), scaling causes higher distortions of the building and the yellow boats. Many people cross the image in sequence (d). When we consider the background and the people, the differences of scaling and seam carving are very low. Major differences only appear in the yellow flowers. Despite the deformation of the person, the quality of the scaled video is higher in sequence (e). The tram crosses the full image and is heavily deformed over time by the removal of seams.

7.3.3 Seam Carving based on Graph Cut versus FSCAV

This part of the evaluation compares the two seam carving techniques (*graph cut* presented in [66] and *FSCAV*). Their visual differences are rather low. In many situations, no difference can be seen if only one frame is observed. In videos without object motion (*static* or *camera motion*) the quality of *FSCAV* adapted videos is slightly better. This is because the seams calculated by graph cut change from frame to frame and introduce a small amount of shakiness, which is most obvious in background objects.

The *graph cut* technique generates videos of higher quality if small objects move in parallel to the seams (seams usually avoid the foreground objects). In case of *fast moving objects* or *very large objects*, the visual quality achieved by both adaptation techniques is much lower. Temporal connectedness of 2D seam manifolds leads to a moving of seam pixels by at most one pixel per frame. Even slow moving objects that move orthogonally to the seams will eventually cross them and cause artifacts comparable to *FSCAV*.

Another aspect is the *computational effort* and *working memory* required by both algorithms. *FSCAV* is separated into a *video analysis* phase and an *adaptation* phase. As result of the analysis phase, an image with global seams and parameters of the camera model is stored. The global seams are mapped to each frame in the adaptation phase. The adaptation is efficient and can be handled in near real-time on a standard personal computer if decoding and re-encoding of the adapted video stream is not considered.

The memory requirements of *FSCAV* are directly linked to the maximum number of frames which are loaded into memory at one time. Sequences in PAL resolution are processed entirely in memory. HD resolution sequences are decoded on the fly to reduce the total amount of memory to less than 200 MB. As a side effect, run-time increases due to hard disk access. The memory requirements of the *graph cut* algorithm are very high making this technique only applicable to low resolution videos. Each pixel of the video sequence is represented as a node in a 3D spatio-temporal cube and several edges are connected to each node.

We analyze the memory requirements and the computational effort of the different algorithms for three test sequences (low resolution, PAL resolution, and HD resolution). In our implementation, we used the *max-flow* algorithm presented by Boykov and Kolmogoro [6].

Fig. 20 Top to bottom: Sample sequences from the 5 categories: static (a), camera motion only (b), small object motion (c), high object and camera motion (d), and large objects (e). The width of each video is reduced by 45 percent.

Table 2 lists the measurements of the run-time and the (theoretical) memory requirements on a standard personal computer (Athlon 64 Dual Core, 2.4 GHz, 2 GB RAM). PAL and HD sequences could not be adapted by the original graph cut algorithm on our system due to the required amount of memory. The numbers in brackets show the theoretical requirements of the max-flow algorithm for these sequences. Both memory requirement and computational effort of the *FSCAV* algorithm are much lower.

A hierarchical approximation was proposed by Rubinstein et al. [66] to reduce the memory requirements when detecting seam manifolds. We implemented a hierarchical graph cut algorithm which reduces the resolution of a video to the low resolution video defined above. The spatial and temporal resolutions of HD videos are reduced by a factor of 16 and 4 respectively. Seam manifolds in the smallest video cube are detected and mapped to the next level. A 16 pixel wide video slice is selected in the next hierarchy level so that the cut is located at its center. The width of the slice is set to 8 and 4 pixels in the last 2 iterations. The major disadvantage of using a hierarchical approximation is the fact that a seam may be trapped in a local minimum and detection of the optimal cut is no longer guaranteed.

Table 2 Run-time and the memory requirements

	Low resolution 120×68 50 frames	*PAL* 720×576 150 frames	*HD* 1920×1080 200 frames
Crop	$<$1 s	5 s	32 s
Scale	$<$1 s	6 s	36 s
FSCAV			
- Video Analysis	14 s	8 min	51 min
- Retargeting	1 s	11 s	83 s
Graph Cuts	17 min / 290 MB	— / (44 GB)	— / (292 GB)
Hierarchical Graph Cuts	—	49 min / 530 MB	123 min / 820 MB

7.3.4 Limitations of FSCAV

The calculation of the camera parameters fails in sequences with fast camera motion, large foreground objects, missing feature points in background objects, or dolly shots. For instance, it was not possible to create a background image in several shots of a soccer game, and the sequence could not be processed with the *FSCAV* algorithm. In all these cases the ROI-based retargeting is automatically used.

The quality of the target videos processed with the ROI-based algorithm is much better when objects move orthogonally to the direction of the seams. Just like seam carving for images, *FSCAV* has problems with objects covering a large part of the screen, and some content may make perceptible errors very likely. The detection and preservation of straight lines improves the visual quality significantly.

Despite these shortcomings, the visual quality achieved by seam carving is higher in most cases. If the most critical shots are processed with an alternative technique, the overall visual quality of the target videos improves further.

8 Conclusions and Outlook

We presented our video retargeting system to adapt videos to the limited screen resolution of handheld mobile devices. Each shot is processed separately. According to camera and object motion, the most suitable algorithm – *FSCAV* or ROI-based retargeting – is used. New heuristics for seam carving were presented that identify obvious visual errors caused by the removal of seams. The major advantage of our fast seam carving algorithm compared to previous approaches is its increased efficiency. In addition, we presented our seam carving version with enhanced preservation of regular objects and straight lines. If a seam crosses a line, adjacent energy values are increased in order to prevent the following seams from crossing the line in the same place again. User evaluations indicated a high quality of the target videos.

In future work we plan to include additional video retargeting techniques like image warping. A reliable heuristic to describe whether a retargeting technique is suitable or not for a given video sequence is a tool we would also like to develop.

Acknowledgements The authors acknowledge the financial support granted by the *Deutsche Forschungsgemeinschaft* (DFG). We would like to thank the following flickr.com users for providing their images via the creative commons license: teoruiz (bridge.jpg), the_tahoe_guy (road.jpg) and digital_cat (construction_site.jpg). We thank Instituto Luce for providing historical films within the European research project *ECHO*. Furthermore, we would like to thank Sabine Olawsky for the development of the contrast-based saliency detection.

References

1. Avidan S, Shamir A (2007) Seam carving for content-aware image resizing. ACM Transactions on Graphics, SIGGRAPH 2007 26(3)
2. Bai B, Harms J (2005) A multiview video transcoder. In: Proceedings of the 13th annual ACM international conference on Multimedia, ACM Press, pp 503–506
3. Bay H, Ess A, Tuytelaars T, Gool LV (2008) SURF: Speeded Up Robust Features. Computer Vision and Image Understanding (CVIU) 110(3):246–359
4. Beek P, Smith JR, Ebrahimi T, Suzuki T, Askelof J (2003) Metadata-driven multimedia access. In: IEEE Signal Processing Magazine, IEEE Computer Society Press, vol 20(2), pp 40–52
5. Bjrk N, Christopoulos C (2000) Video transcoding for universal multimedia access. In: Proceedings of the 2000 ACM workshops on Multimedia, ACM Press, pp 75–79
6. Boykov Y, Kolmogorov V (2004) An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 26(9), pp 1124–1137
7. Canny JF (1986) A computational approach to edge detection. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Computer Society Press, vol 8(6), pp 679–698
8. Cardellini V, Yu P, Huang Y (2000) Collaborative proxy system for distributed web content transcoding. In: Proceedings of 9th International ACM Conference on Information and Knowledge Management, ACM Press, pp 520–527
9. Cheng WH, Hsieh CW, Lin SK, Wang CW, Wu JL (2005) Robust algorithm for exemplar-based image inpainting. In: The International Conference on Computer Graphics, Imaging and Vision, IEEE Press, pp 64 – 69

10. Cheng WH, Wang CW, Wu JL (2007) Video adaptation for small display based on content recomposition. Circuits and Systems for Video Technology, IEEE Transactions on 17(1):43 –58
11. Curran K, Annesley S (2005) Transcoding media for bandwidth constrained mobile devices. In: International Journal of Network Management, John Wiley & Sons, Inc., vol 15(2), pp 75–88
12. Dong W, Paul JC (2008) Adaptive content aware image resizing. Eurographics 2009 28(2)
13. Dong W, Zhou N, Paul JC, Zhang X (2009) Optimized image resizing using seam carving and scaling. ACM Transactions on Graphics 28(5):1–10
14. Dong W, Bao G, Zhang X, Paul JC (2010) Interactive multi-operator image resizing and evaluation. Journal of Computer Science and Technology 25(2)
15. Duda RO, Hart PE (1972) Use of the hough transformation to detect lines and curves in pictures. Communications of the ACM 15(1):11–15
16. El-Alfy H, Jacobs D, Davis L (2007) Multi-scale video cropping. ACM international conference on Multimedia pp 97–106
17. Farin D (2005) Automatic video segmentation employing object/camera modeling. PhD thesis, Technische Universiteit Eindhoven, Einhoven, The Netherlands
18. Farin D, Haenselmann T, Kopf S, Kühne G, Effelsberg W (2003) Segmentation and classification of moving video objects. In: Furht B, Marques O (eds) Handbook of Video Databases: Design and Applications, Internet and Communications Series, vol 8, CRC Press, Boca Raton, FL, USA, pp 561–591
19. Fischler M, Bolles R (1981) Random sample concensus: A paradigm for model fitting with applications to image analysis and automated cartography. In: Communications ACM, ACM Press, vol 24(6), pp 381–395
20. Fox A, Gribble S, Chawathe Y, Brewer E (1998) Adapting to network and client variation using infrastructural proxies: Lessons and perspectives. In: IEEE Personal Communication, IEEE Computer Society Press, vol 5(4), pp 10–19
21. Gal R, Sorkine O, Cohen-Or D (2006) Feature-aware texturing. In: Proceedings of Eurographics Symposium on Rendering, pp 297–303
22. Guo Y, Liu F, Zhou ZH, Gleicher M (2009) Image retargeting using mesh parameterization. IEEE Transactions on Multimedia 11(5):856–867
23. Han JW, Choi KS, Wang TS, Cheon SH, Ko SJ (2009) Improved seam carving using a modified energy function based on wavelet decomposition. In: IEEE 13th International Symposium on Consumer Electronics, pp 38 –41
24. Han R, Bhagwat P, LaMaire R, Mummert T, Perret V, Rubas J (1998) Dynamic adaptation in an image transcoding proxy for mobile WWW browsing. In: IEEE Personal Communication, IEEE Computer Society Press, vol 5(6), pp 8–17
25. Harris C, Stephens M (1988) A combined corner and edge detector. In: Proceedings of Alvey Vision Conference, pp 147–151
26. Harrison P (2001) A non-hierarchical procedure for re-synthesis of complex textures. The 9-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision pp 190–197
27. Hjelsvold R, Vdaygiri S, Leaute Y (2001) Web–based personalization and management of interactive video. In: Proceedings of the 10th international conference on World Wide Web, pp 129–139
28. Hossain M, Rahman A, Saddik A (2004) A framework for repurposing multimedia content. In: Proceedings of the Canadian Conference on Electrical and Computer Engineering, IEEE Computer Society Press, pp 971–974

29. Hwang DS, Chien SY (2008) Content-aware image resizing using perceptual seam carving with human attention model. In: IEEE International Conference on Multimedia and Expo, pp 1029–1032
30. ISO/IEC (2002) Information technology – multimedia content description interface (MPEG-7) – Part 8: Extraction and use of MPEG-7 descriptions. Tech. Rep. TR 15938-8, ISO/IEC
31. ISO/IEC (2003) MPEG-21 multimedia framework – Part 7: Digital item adaptation (final committee draft). Tech. Rep. N 5845, ISO/IEC
32. ISO/IEC (2004) Information technology – multimedia framework (MPEG-21) – Part 1: Vision, technologies and strategy. Tech. Rep. TR 21000-1, ISO/IEC
33. Itti L, Koch C, Niebur E (1999) A model of saliency-based visual attention for rapid scene analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(11):1254–1259
34. Kiess J, Kopf S, Guthier B, Effelsberg W (2010) Seam carving with improved edge preservation. In: Proceedings of IS&T/SPIE conference on Multimedia on Mobile Devices, vol 7542
35. Kim JS, Kim JH, Kim CS (2009) Adaptive image and video retargeting technique based on fourier analysis. In: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, IEEE, pp 1730–1737
36. Kopf S, Effelsberg W (2008) Mobile cinema: Canonical processes for video adaptation. In: Multimedia Systems, Springer, vol 14(6), pp 369–375
37. Kopf S, Haenselmann T, Farin D, Effelsberg W (2004) Automatic generation of summaries for the Web. In: Proceedings of IS&T/SPIE conference on Storage and Retrieval for Media Databases, vol 5307, pp 417–428
38. Kopf S, Haenselmann T, Effelsberg W (2005) Enhancing curvature scale space features for robust shape classification. In: Proceedings of IEEE International Conference on Multimedia and Expo (ICME), IEEE Computer Society Press, pp 478–481
39. Kopf S, Haenselmann T, Effelsberg W (2005) Robust character recognition in low-resolution images and videos. Tech. Rep. TR-05-002, Department of Mathematics and Computer Science, University of Mannheim, Germany
40. Kopf S, Haenselmann T, Effelsberg W (2005) Shape-based posture and gesture recognition in videos. In: Proceedings of IS&T/SPIE conference on Storage and Retrieval Methods and Applications for Multimedia, vol 5682, pp 114–124
41. Kopf S, Lampi F, King T, Effelsberg W (2006) Automatic scaling and cropping of videos for devices with limited screen resolution. In: Proceedings of the 14th ACM international conference on Multimedia, ACM Press, pp 957–958
42. Kopf S, Guthier B, Lemelson H, Effelsberg W (2009) Adaptation of web pages and images for mobile applications. In: Proceedings of IS&T/SPIE conference on Multimedia on Mobile Devices, vol 7256, pp 72,560C–72,560C–12
43. Kopf S, Kiess J, Lemelson H, Effelsberg W (2009) FSCAV: Fast seam carving for size adaptation of videos. In: Proceedings of the 17th ACM international conference on Multimedia, ACM, New York, NY, USA, pp 321–330
44. Krähenbühl P, Lang M, Hornung A, Gross M (2009) A system for retargeting of streaming video. In: ACM SIGGRAPH Asia, ACM, New York, NY, USA, pp 1–10
45. Lei Z, Georganas ND (2001) Context-based media adaptation in pervasive computing. In: Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering, IEEE Computer Society Press, vol 2, pp 913–918
46. Lei Z, Georganas ND (2002) Rate adaptation transcoding for precoded video streams. In: Proceedings of the 10th ACM international conference on Multimedia, ACM Press,

pp 127–136
47. Li Y, Sun J, Tang CK, Shum HY (2004) Lazy snapping. ACM Transactions on Graphics (TOG) 23(3):303–308
48. Li Y, Tian Y, Yang J, Duan LY, Gao W (2010) Video retargeting with multi-scale trajectory optimization. In: Proceedings of the international conference on Multimedia information retrieval, ACM, New York, NY, USA, pp 45–54
49. Linde Y, Buzo A, Gray R (1980) An algorithm for vector quantizer design. IEEE Transactions on Communications 28(1):84 – 95
50. Liu F, Gleicher M (2003) Automatic image retargeting with fisheye-view warping. In: Proceedings of the 16th annual ACM symposium on User interface software and technology, pp 153–162
51. Liu F, Gleicher M (2006) Video retargeting: Automating pan and scan. ACM international conference on Multimedia pp 241–250
52. Liu H, Xie X, Ma WY, Zhang HJ (2003) Automatic browsing of large pictures on mobile devices. ACM international conference on Multimedia pp 148–155
53. Liu H, Jiang S, Huang Q, Xu C, Gao W (2007) Region-based visual attention analysis with its application in image browsing on small displays. In: Proceedings of the 15th international conference on Multimedia, pp 305–308
54. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. In: International Journal of Computer Vision, Kluwer Academic Publishers, vol 60(2), pp 91–110
55. Lum W, Lau F (2002) A context-aware decision engine for content adaptation. In: IEEE Pervasive Computing, IEEE Computer Society Press, vol 1(3), pp 41–49
56. Ma YF, Zhang HJ (2003) Contrast-based image attention analysis by using fuzzy growing. In: Proceedings of the 11th ACM international conference on Multimedia, ACM Press, pp 374–381
57. Mohan R, Smith J, Li C (1999) Adapting multimedia internet content for universal access. In: IEEE Transactions on Multimedia, IEEE Computer Society Press, vol 1(1), pp 104–114
58. Mokhtarian F, Bober M (2003) Curvature Scale Space Representation: Theory, Applications, and MPEG-7 Standardization (Computational Imaging and Vision, 25). Kluwer Academic Publishers, Dordrecht, The Netherlands
59. Nepal S, Srinivasan U (2003) DAVE: A system for quality driven adaptive video delivery. In: Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval, ACM Press, pp 223–230
60. Noble B, Satyanarayanan M, Narayanan D, Tilton JE, Flinn J, , RWalker K (1997) Agile application-aware adaptation for mobility. In: Proceedings of the 16th Symposium on Operating System Principles, pp 276–287
61. Nurnett I (2003) MPEG-21: Goals and archievments. In: IEEE Multimedia, IEEE Computer Society Press, vol 10(6), pp 60–70
62. Obrenovic Z, Starcevic D, Selic B (2004) A model-driven approach to content repurposing. In: IEEE Multimedia, IEEE Computer Society Press, vol 11(1), pp 62–71
63. Ren T, Liu Y, Wu G (2009) Image retargeting based on global energy optimization. In: Proceedings of the 2009 IEEE international conference on Multimedia and Expo, IEEE Press, Piscataway, NJ, USA, pp 406–409
64. Richter S, Kühne G, Schuster O (2001) Contour-based classification of video objects. In: Proceedings of IS&T/SPIE conference on Storage and Retrieval for Media Databases, vol 4315, pp 608–618
65. Rowley HA, Baluja S, Kanade T (1998) Neural network-based face detection. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Computer Society

Press, vol 20(1), pp 23–38
66. Rubinstein M, Avidan S, Shamir A (2008) Improved seam carving for video retargeting. ACM Transactions on Graphics, SIGGRAPH 2008 27(3)
67. Rubinstein M, Shamir A, Avidan S (2009) Multi-operator media retargeting. ACM Transactions on Graphics, SIGGRAPH 2009 28(3):1–11
68. Santella A, Agrawala M, DeCarlo D, Salesin D, Cohen M (2006) Gaze-based interaction for semi-automatic photo cropping. ACM Conference on Human Factors in Computing Systems pp 771–780
69. Schaber P, Kopf S, Thorwirth N, Effelsberg W (2010) Semi-automatic registration of videos for improved watermark detection. In: ACM SIGMM conference on Multimedia Systems, ACM, New York, NY, USA, pp 23–34
70. Schneiderman H (2010) Face detection demonstration. Tech. rep., Robotics Institute, Carnegie Mellon University, http://www.vasc.ri.cmu.edu/cgi-bin/demos/findface.cgi
71. Schneiderman H, Kanade T (2000) A statistical model for 3D object detection applied to faces and cars. In: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society Press
72. Setlur V, Takagi S, Raskar R, Gleicher M, Gooch B (2005) Automatic image retargeting. Proceedings of the 4th international conference on Mobile and Ubiquitous Multimedia pp 247–250
73. Shamir A, Avidan S (2009) Seam carving for media retargeting. Communications of the ACM 52(1):77–85
74. Shanableh T, Ghanbari M (2000) Heterogeneous video transcoding to lower spatio-temporal resolution and different encoding formats. In: IEEE Transactions on Multimedia, IEEE Computer Society Press, vol 2(2), pp 101–110
75. Smith SM, Brady JM (1997) SUSAN – new approach to low level image processing. In: International Journal of Computer Vision (IJCV), vol 23(1), pp 45 – 78
76. Steiger O, Ebrahimi T, Sanjuan D (2003) MPEG-based personalized content delivery. In: Proceedings of IEEE International Conference on Image Processing (ICIP), IEEE Computer Society Press, vol 3, pp 45–48
77. Suh B, Ling H, Bederson B, Jacobs D (2003) Automatic thumbnail cropping and its effectiveness. Proceedings of the 16th annual ACM symposium on User interface software and technology pp 95–104
78. Tao C, Jia J, Sun H (2007) Active window oriented dynamic video retargeting. Proceedings of the Workshop on Dynamical Vision
79. Tseng B, Lin CY, Smith JR (2004) Using MPEG-7 and MPEG-21 for personalizing video. In: IEEE Multimedia, IEEE Computer Society Press, vol 11(1), pp 42–52
80. Vetro A (2004) MPEG-21 digital item adaptation: Enabling universal multimedia access. In: IEEE Multimedia, IEEE Computer Society Press, vol 11(1), pp 84–87
81. Vetro A, Chrisopoulos C, Sun H (2003) Video transcoding architectures and techniques: An overview. In: IEEE Signal Processing Magazine, IEEE Computer Society Press, vol 20(2), pp 18–29
82. Vetro A, Christopoulos T, Ebrahimi T (2003) Special issue on universal multimedia access. In: IEEE Signal Processing Magazine, IEEE Computer Society Press, vol 20(2), pp 69–79
83. Wang J, Reinders M, Lagendijk R, Lindenberg J, Kankanhalli M (2004) Video content presentation on tiny devices. IEEE International Conference on Multimedia and Expo pp 1711–1714
84. Wang YS, Tai CL, Sorkine O, Lee TY (2008) Optimized scale-and-stretch for image resizing. ACM Transactions on Graphics 27(5):1–8

85. Wang YS, Fu H, Sorkine O, Lee TY, Seidel HP (2009) Motion-aware temporal coherence for video resizing. ACM Transactions on Graphics 28(5)
86. Wolf L, Guttmann M, Cohen-Or D (2007) Non-homogeneous content-driven video-retargeting. In: Proceedings of the Eleventh IEEE International Conference on Computer Vision
87. Zwicker M, Pfister H, van Baar J, Gross M (2002) EWA splatting. IEEE Transactions on Visualization and Computer Graphics 8(3):223–238