Adaptation of Images and Videos for Different Screen Sizes

Inauguraldissertation zur Erlangung
des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

Dipl.-Wirt.-Inf. Johannes Kieß
aus Schwäbisch Hall

Mannheim, 2014

Dekan:	Prof. Dr. Heinz Jürgen Müller, Universität Mannheim
Referent:	Prof. Dr. Wolfgang Effelsberg, Universität Mannheim
Korreferent:	Prof. Dr. Ralf Steinmetz, Technische Universität Darmstadt

Tag der mündlichen Prüfung: 03.Juni 2014

Abstract

With the increasing popularity of smartphones and similar mobile devices, the demand for media to consume on the go rises. As most images and videos today are captured with HD or even higher resolutions, there is a need to adapt them in a content-aware fashion before they can be watched comfortably on screens with small sizes and varying aspect ratios. This process is called *retargeting*. Most distortions during this process are caused by a change of the aspect ratio. Thus, retargeting mainly focuses on adapting the aspect ratio of a video while the rest can be scaled uniformly.

The main objective of this dissertation is to contribute to the modern image and video retargeting, especially regarding the potential of the seam carving operator [2]. There are still unsolved problems in this research field that should be addressed in order to improve the quality of the results or speed up the performance of the retargeting process. This dissertation presents novel algorithms that are able to retarget images, videos and stereoscopic videos while dealing with problems like the preservation of straight lines or the reduction of the required memory space and computation time. Additionally, a GPU implementation is used to achieve the retargeting of videos in real-time. Furthermore, an enhancement of face detection is presented which is able to distinguish between faces that are important for the retargeting and faces that are not. Results show that the developed techniques are suitable for the desired scenarios.

Zusammenfassung

Durch die zunehmende Popularität von Smartphones und ähnlichen mobilen Geräten steigt die Nachfrage nach Inhalten, die unterwegs konsumiert werden können stetig. Da die meisten Bilder und Videos heutzutage in HD oder höheren Auflösungen aufgezeichnet werden, müssen sie mit Bezug auf den Inhalt angepasst werden, bevor sie komfortabel auf Bildschirmen mit kleinen Größen und verschiedenen Seitenverhältnissen betrachtet werden können. Diesen Vorgang nennt man *Retargeting*. Die meisten Verzerrungen während dieses Prozesses entstehen durch eine Veränderung des Seitenverhältnisses. Deshalb fokussiert sich Retargeting hauptsächlich auf die Anpassung des Seitenverhältnisses, während der Rest des Bildes oder Videos gleichmäßig skaliert werden kann.

Das Hauptanliegen dieser Dissertation ist es, einen Beitrag zu der modernen Anpassung von Bildern und Videos zu leisten, besonders mit Hinblick auf das Potential des Seam-Carving-Operators [2]. In diesem Forschungsbereich gibt es noch immer ungelöste Probleme, welche adressiert werden sollten, um die Qualität der Ergebnisse zu steigern und den Prozess der Anpassung schneller zu machen. Diese Dissertation präsentiert neuartige Algorithmen, die Bilder, Videos und stereoskopische Videos anpassen können, während sie mit Problemen wie beispielsweise der Bewahrung gerader Linien oder der Verringerung der Rechenzeit umgehen. Zusätzlich wird die GPU verwendet, um eine Anpassung von Videos in Echtzeit zu erreichen. Des Weiteren wird eine verbesserte Gesichtserkennung präsentiert, die unterscheiden kann, ob ein Gesicht für die Anpassung wichtig ist oder nicht.

Danksagung

Mein Dank gilt in erster Linie meinem verehrten Doktorvater Prof. Wolfgang Effelsberg, der mir die Möglichkeit zur Promotion an seinem Lehrstuhl gegeben hat. Er hat mich stets mit wertvollen Anregungen unterstützt und persönlich gefördert. Ebenfalls hervorheben möchte ich PD Dr. Stephan Kopf, der schon beim Anfertigen der Diplomarbeit mein Betreuer war und der mich auch während meiner ganzen Promotion begleitet hat. Ohne seinen hilfreichen akademischen Rat wäre diese Arbeit nicht entstanden.

Des Weiteren möchte ich mich bei meinen Kollegen, insbesondere Benni und Tonio, bedanken, die immer für ein sehr angenehmes Arbeitsklima gesorgt haben.

Eine herausragende Stellung nehmen meine Frau sowie meine Familie ein, ohne deren liebevolle Unterstützung diese Arbeit nicht zu dem geworden wäre, was sie heute ist. Sie hatten jederzeit ein offenes Ohr für mich und ich konnte mich immer bedingungslos auf sie verlassen. Ihnen gilt mein besonderer Dank.

Contents

List of Figures

List of Tables

Nomenclature

AG	Attention Group
AO	Attention Object
BDW	Bi-Directional Warping
CIE	Commission Internationale de l'Eclairage
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DCD	Dominant Color Descriptor
DTW	Dynamic Time Warping
fps	Frames Per Second
FSCAV	Fast Seam Carving for the Adaptation of Videos
GPU	Graphics Processing Unit
HADP	Hierarchically Accelerated Dynamic Programming
HD	High Definition, usually referring to a resolution of 1920×1080 pixels
ID	Identifier
IMED	Image Euclidean Distance
IV	Importance Value
MPS	Minimal Perceptible Size
MPT	Minimal Perceptible Time
MSER	Maximally Stable Extremal Region
OpenCV	Open Source Computer Vision Library
PAL	Phase Alternating Line, usually referring to a resolution of 720×576 pixels
PC	Personal Computer
RAM	Random Access Memory
RANSAC	Random Sample Consensus
RF	Retarget Factor

ROI	Region Of Interest
SDK	Software Developement Kit
SFW	Stereo Frame-Wise
SIFT	Scale-Invariant Feature Transform
SSD	Sum-of-Squared-Differences
SUSAN	Smallest Univalue Segment Assimilating Nucleus
SV	Stereo Video

CHAPTER 1

Introduction

This chapter provides a motivation and a brief history of *media retargeting*, the process of adapting an image or a video to fit a display with a different aspect ratio and a lower or higher resolution. The motivation is then followed by a detailed overview of the work that has been done in this dissertation.

1.1 Motivation

With the invention of the Kinetograph by Thomas Edison and William Dickson in November 1890, US cinema was born. It was the first motor-powered camera that was able to capture movement and create a film in the way we know it today. The movies were filmed on $35mm$ film, and sprocket holes were utilized to advance the film. Edison and Dickson used an aspect ratio of $4 : 3$ ($1.3\bar{3} : 1$) for their movies, which mimics the human eyesight visual angle of $4 : 3.075$. This aspect ratio became the universal standard for cinema movies until the $1950s$. The same aspect ratio was chosen for early television, as it matched the standard used in cinema movies and also was square enough to be conveniently displayed on round cathode-ray tubes (CRTs). These monitors need the square-like aspect ratio for a lag-free image formation in both directions and were the only ones that could be produced in mass production at the time.

As the number of viewers in the cinemas declined around the $1950s$ due to the huge success of home television, the studios in Hollywood looked for ways to set the cinema experience

apart from television. Two of the most notable things developed in this effort were *stereoscopic movies* (3D movies) and *widescreen movies*.

Stereoscopic movies are interesting because the technique at the time prevented these kind of movies from being a hit in the $1950s$. Nonetheless, it is again used since about 2009 with increasing success to set cinema apart from television. This time, the technique is much more advanced, and a lot of problems like headaches caused by a parallax mismatch between the eyes and the cameras are no longer there.

The second developement are widescreen movies produced with a aspect ratio of $16 : 9$ ($1.7\bar{7} : 1$). With this new aspect ratio, there was a visible distinction between content that was produced for cinema and content produced for television. Also, when cinema movies were later shown on television, they had to be retargeted.

In order to retarget widescreen movies for television, *cropping* and *letterboxing* were commonly used. Cropping simply cuts content from the borders of the frames. This could be done automatically by simply removing content equally from both sides. If it was done manually, an editor had to determine the best positions of a cropping window with the target aspect ratio for each shot. The drawback of cropping is a guaranteed content loss. Letterboxing shows the whole frame but adds black bars above and under the image to fill the missing space between the different aspect ratios. This method was often used and gave cinema movies in television a signature look. Although letterboxing shows all of the original frame content, details might get too small to recognize. Nowadays, modern flatscreen televisions have displays in the widescreen format ($16 : 9$) as a standard. Also, many digital programs produce and transmit content in the same format. Therefore, the classical use case of retargeting for showing cinema movies on television is no longer there.

But retargeting is not dead. In fact, it is even more important than ever. There are now a multitude of heterogeneous devices for capturing images and videos as well as for displaying them like tablets or smartphones. Additionally, the Internet makes a large amount of media data available for people to watch at home on their PCs or on the way on their mobile devices. Most of these displays have different resolutions and aspect ratios, leading to a new need of retargeting methods (see Figure 1.1 [1]).

The first automatic retargeting techniques that were proposed were used to create thumbnails of large image libraries or to adapt widescreen movies to the standard television resolution and aspect ratio [81, 52]. Mostly cropping was used in these algorithms as the *retargeting*

[1] `http://opensignal.com/reports/fragmentation.php`

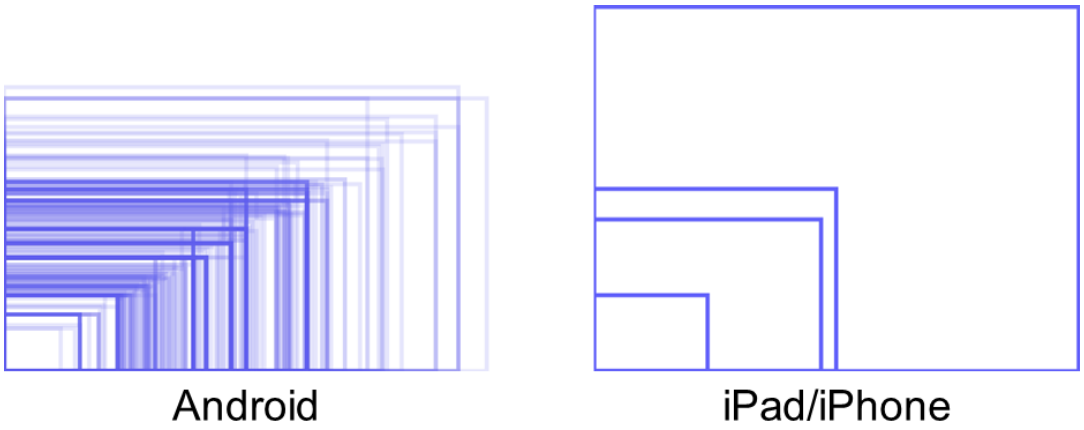


Figure 1.1: *Display resolutions of modern smartphones and tablets.*

operator[2]. With the current boom of mobile handheld devices mentioned above, the adaptation to their displays is the goal of more recent work. When simple scaling or cropping is used to resize an image or a video to a lower resolution or to a different aspect ratio, obvious distortions may occur. This includes stretched faces, missing content, or the whole media looking completely unproportional. Solely employing elemental retargeting techniques, which treat all image regions equally, does not lead to satisfying results. Important objects in the images like faces or text should be preserved. Algorithms that focus on this aspect are called content-aware algorithms.

In general, retargeting tries to identify the important parts of an image or video and tries to preserve these regions with as little changes as possible. This is normally done through an importance function which determines an importance value for each pixel based on several low-level features like contrast and color or high-level features like faces. Then, a retargeting operator is used to fit the image or frame to the target dimensions. While important areas often remain unchanged during the process, unimportant areas get distorted or removed. This is not visible to the viewer in the ideal case as the important areas are drawing the attention and thereby masking the artifacts that might occur elsewhere.

The two most prominent content-aware retargeting techniques are *seam carving* and *warping*. Seam carving was first introduced by Avidan and Shamir [2] and became quite popular after its publication. In the algorithm, paths of connected pixels are searched that reach from top to bottom or from left to right of the image. These paths are called seams, and each of the seams reduces or enlarges the size of an image by 1 if it is removed or duplicated. The earliest version of warping was proposed by Liu *et al.* [51] for the non-photorealistic warping

[2]This term always refers to a single retargeting method like scaling or cropping, and never to a combination of those.

of images. Their idea was further enhanced by Gal *et al.* [23] for 2D texture mapping. With this technique, images can be mapped to different 2D shapes like a flag or the side surfaces of 3D shapes, for instance a pyramid. This was one of the first warping methods to achieve photo-realistic results. Wolf *et al.* [99] then used the idea from [23] and introduced the first warping technique as it is used today. The basic idea is to lay a mesh over the source image and then map it to the target size by deforming the cells of the mesh. This equals non-uniform scaling.

Retargeting is a popular research topic, and a large number of interesting techniques have been proposed. They all have in common that each method is well-suited for some scenarios while it is not the optimal technique in others. The reason for this can be the quality of the results or the performance of the algorithm in different situations, like retargeting videos as a whole or as a stream in real time.

In the case of quality, each method has inherent shortcomings as reducing the width or height cannot be done without loss of information [27]. For instance, if cropping is used, parts of the source image are completely discarded. Also, images can not be enlarged with cropping. When warping is used instead, the information of the pixels with low importance is spread over the other pixels, but objects or parts of the background may get squeezed. A limitation specific to seam carving is the observation that in most cases seam carving produces unpleasant artifacts when it is used to reduce the width to below one half of the original size.

There are still limitations that nearly all techniques share. Objects including structured textures or straight lines are difficult to preserve, as they are mostly not the dominant parts of an image or video [96, 24, 50]. They can often be found in the background on buildings or as markings on a street. Another typical problem for retargeting are very complex scenes with a large amount of objects and few homogeneous areas [95, 103]. Retargeting also heavily depends on the importance calculation that is done as the first step. The main focus for the preservation normally lies on faces and whole persons if there are any, as they usually draw the most attention from the viewer. However, not all faces are equally important, for instance a face of a basketball player compared to a face in the crowd behind him. In videos, motion is really important, and an object that moves on the screen is usually interesting to a viewer. The benefit of such objects is that they are able to mask distortions in homogeneous areas or regions that are not so visually dominant. This principle is used by the warping techniques that do not discard information.

In terms of performance of the algorithms, a trade-off must be made between the quality of the results and the computation time that is needed [87]. For instance, an optimization over

an entire video sequence results in a highly complex problem with at least three dimensions (width, height and time). One way to reduce complexity is to use a heuristic and accept the fact that the result is not globally optimal. Another possibility is to look for ways of parallelizing individual steps of an algorithm and to use the GPU for faster processing.

The above mentioned problems and limiations led to the research questions that are formulated in the following section.

1.2 Outline

The main objective of this dissertation is to contribute to the modern image and video retargeting, especially regarding the potential of the seam carving operator [2]. Like mentioned in the previous section, there are still unsolved problems in this research field that should be adressed in order to improve the quality of the results or speed up the performance of the retargeting process. This led to the following research questions that guided this dissertation:

- How can straight lines or regular patterns be better preserved during seam carving?
- Can seam carving for videos be done with a more efficient processing step than the graph-cut based approach presented by the original authors [68]?
- Is the method of seam carving usable to retarget stereoscopic videos?
- Can the quality of the results of seam carving be improved via a combination with other retargeting operators?
- Is a differentiation between the faces that are found through face detection able to further improve the results of the retargeting?
- Is it possible to achieve real-time video retargeting by using the *graphics processing unit* (GPU) instead of the CPU?

These questions are answered in detail in the subsequent chapters of this dissertation. The outline of the chapters is as follows.

Chapter 2 thoroughly explains the basics terms and the general workflow of retargeting. Also, it gives an comprehensive overview of the field of image and video retargeting and describes the state-of-the-art algorithms. Especially, seam carving gained a lot of attention after its first publication in 2007 [2] and is used in many algorithms. In this chapter, the most influential retargeting algorithms are presented, and the different retargeting operators are described.

Chapter 3 covers improvements and enhancements of the seam carving algorithm [2]. Based on the first three research questions presented above, this chapter is divided into three parts. The first part (3.1) solves a common problem of the seam carving operator for images: seam carving achieves a high adaptation quality for landscape images, but if an image depicts objects with straight lines or regular patterns like buildings, the visual quality of the adapted images is much lower. Errors caused by seam carving are especially obvious if straight lines become curved or disconnected. Our solution is an additional energy criterion based on line detection which prevents the seams from crossing a straight line in adjacent pixel positions. In the second part (3.2), seam carving is used to retarget videos. Although there is already an approach by the original authors for videos [68], their algorithm is very complex and has a long processing time as it uses a global optimization over all frames. In contrast, we also take all frames into account but formulate the retargeting problem with less complexity. Our algorithm uses image registration techniques to align all frames of a shot and create a so-called background image. On this image, the seam carving operator for images can be used to find seams very efficiently. These seams are then tracked back to the individual frames. In comparision to [68], our new algorithm is significantly faster. Part three of the Chapter (3.3) expands seam carving for the retargeting of stereoscopic videos. To our knowledge, there is currently no other algorithm capable of retargeting this kind of video. For our method, we assume that the left and the right view of a video are given. In our approach, seams are searched in the left view and the disparity map [34] – the mapping between pixels in the left and the right frame – simultaneously to preserve the depth information as well as possible. For temporal consistency, the seams from the previous frame are used as a reference for searching seams in the current frame.

In Chapter 4, the novel SeamCrop approach is presented. This technique combines *seam carving* with *cropping* in order to overcome the limitations of the individual operators. The chapter is divided into four parts and answers the remaining research questions presented above. In the first part (4.1), the basic idea of SeamCrop is presented and used to retarget images. As cropping is still a valid retargeting option [69] and has the advantage of not producing artifacts besides cut-off objects, it is chosen as the main operator. First, seams are removed carefully until a dynamic energy threshold is reached to prevent the creation of visible artifacts. Then, a cropping window is selected in the image that has the smallest possible window size without having the removed energy rise above a second dynamic threshold. As the number of removed seams and the size of the cropping window are not fix, the process is repeated iteratively until the target size is reached. The second part of the Chapter (4.2)

introduces an enhancement to the use of face detection in the importance calculation of an image. Faces draw a lot of attention from the viewers, but the level of relevance may be different for different faces depending on the size, the location, or whether a face is in focus or not. Therefore, a novel algorithm which distinguishes in-focus and out-of-focus faces is presented. A face detector with multiple cascades is used first to locate face regions initially. Then, the ratio of strong edges in each face region is analyzed to classify out-of-focus faces. Finally, the GrabCut algorithm [67] is used to segment the faces and define binary face masks. These masks can then be used as an additional input to image retargeting algorithms. In the third part of the Chapter (4.3), the SeamCrop approach is extended to the retargeting of videos. The idea is to find an optimal cropping window and to include more useful content in it by additionally removing seams. We formulate the search for a cropping window in terms of a 2D rectangle representing the possible positions of the window over all frames. Finding the optimal position can be solved efficiently with dynamic programming. Seams are then searched frame-by-frame with the use of a new constraint that ensures temporal consistency without a global optimization. This leads to a fast processing time while maintaining comparable results to similar state-of-the-art algorithms. The fourth part of the Chapter (4.4) uses the GPU to speed up the processing time of the already efficient SeamCrop approach for videos. In contrast to the previous algorithm, the presented technique is optimized for parallel processes and a CUDA GPU implementation. The differences and the adjustments between the two versions are thoroughly discussed, and measurements of the efficiency are presented in a detailed performance test. In comparison to the CPU implementation, the computation time has been decreased up to 10.5 times.

Finally in Chapter 5, the dissertation is concluded and an outlook on future work is presented.

CHAPTER 2

Fundamentals and Related Work

In this Chapter, the fundamentals of image and video retargeting are discussed, and a detailed overview of the research field is given. Section 2.1 introduces basic terms and describes commonly used retargeting operators. Additionally, the basic workflow of most algorithms is explained, and two important optimization methods are shown. In Section 2.2, the most influental papers as well as the state-of-the-art algorithms are presented.

2.1 Fundamentals

This section introduces the concept of visual attention analysis, which is an important aspect in the retargeting of images and videos. Also, it presents some basic knowledge on media retargeting in general. Since many techniques work for images and videos alike, we only discuss their application to images for the sake of simplicity.

2.1.1 Visual Attention Analysis

The estimation of the *visual attention* (*importance, saliency*) of pixels in images is the first fundamental step of retargeting. Pixels, regions, or objects with high relevance to the understanding of the content must be preserved as well as possible.

Visual attention features can be classified into two categories: low level features and high level features. The first category represents features that catch the visual attention on a pixel level. Examples are the gradient energy used in [2] to find edges and contours, the motion

of pixels between two frames [10, 1], or color features [104, 57, 47, 36] derived from the CIE L*A*B or HSV color space which describe color and contrast as perceived by the human eye. Features in the second category include more complex regions of visual attention like faces [89], objects [88], people or text. By using several visual attention features in an image, an *importance map* (also called *saliency map*) is calculated.

Visual attention features can be detected automatically or with semi-automatic tools that allow the viewer to take part in the identification. In the following, the basics of the most commonly used automatic visual attention functions are explained.

The easiest way to detect salient areas in an image is to use the *gradient magnitude*. It measures the amount of change of intensity or color between pixels. This sharpens the contours of objects due to the contrast of the colors between the object and its surroundings. At the same time, homogeneous areas do not have much contrast but are normally not important for the image content. Most retargeting functions try to keep object contours intact as deformations in them are highly visible to the viewer. The gradient magnitude can be calculated from the adjacent pixels on top and bottom as well as to the left and right. As this function has a low complexity, it is well suited for algorithms that aim for a low processing time. However, it may lack the precision for certain retargeting algorithms when used without other importance functions.

Saliency maps are more complex than the gradient magnitude. They combine several features and also consider the area around a pixel for the calculation. The most prominent saliency function was introduced by Itti et al. [36] and builds the foundation of most of the recent saliency functions. The image is first scaled down into different resolutions (to a so-called *multi-resolution pyramid*). On each scale, feature filters for colors, intensity and orientation are used. This leads to feature maps for each filter on different scales. These maps are then combined to form one feature map for each feature. In the last step, these maps are again combined into the final saliency map.

Calculating a saliency map from several features on a multi-resolution pyramid is a computationally expensive operation. In order to speed this process up, Ma et al. presented a saliency map based on contrast and fuzzy growing [57]. The algorithm starts by producing a raw saliency map by comparing the color value of each pixel with the color values of the pixels around it. Depending on the application the saliency map is used for, the size of the area around the pixels can be varied. This is followed by a region-growing algorithm based on the fuzzy theory in order to identify salient regions. Two classes of areas are defined, one that includes important pixels and the other one with unimportant pixels. The algorithm then

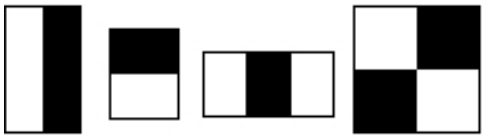


Figure 2.1: *Haar-like rectangular features used in the Viola and Jones framework (as presented in Viola et al. 2001).*

uses a probability model to allocate the pixels to the areas. Fuzzy logic is used as this theory effectively imitates human mental behavior.

Segmentation algorithms partition the pixels in an image into sets of pixels that share characteristics, like color or texture. There are a lot of different techniques that can be used to identify these regions, for instance clustering [82], region growing [84] or using histograms [9]. Segmentation can be used to supplement other visual attention functions like saliency maps and use the data from these functions as a foundation.

Faces draw a lot of the viewer's attention. Their detection is often used in conjunction with other energy functions. A popular *face detection* algorithm was introduced by Viola and Jones [89]. The algorithm uses rectangular features for image classification (see Figure 2.1). These classifiers are used on a greyscale version of the image to identify regions with 'darker' blocks adjacent to 'lighter' blocks like in the features. The features themselves have no fixed size and are used in different scales on the image. In order to increase accuracy without impeding performance, the classifiers are used in *cascades*. Cascades are sets of classifiers organized in a decision tree. When a first simple classifier returns a positive result, a second more complex classifier is used. This is repeated until all classifiers in the set are processed. A negative result from any classifier leads to the rejection of the area that is currently examined. This speeds up the detection as complex classifiers do not need to be used on all possible areas.

The visual attention functions presented up to this point work on images and frames of a video alike, but do not consider the temporal component of videos. Objects that have a different movement direction or speed than the background or other objects on the screen attract the viewer's attention as well. A simple approach for the creation of a *temporal saliency map* is to use the optical flow of a scene to identify pixels that move differently. Another possibility is to use feature points. The idea is to search for feature points via methods like the *Scale Invariant Feature Transform* (SIFT) [55] and use their correspondences for the creation of the temporal saliency map.

The algorithms presented in this dissertation aim for a fast processing time, therefore we chose to primarily use visual attention functions that are not very complex. Gradient magnitude

is used in all implementations, as it provides sufficient precision for our purpose. In some algorithms, this is combined with face detection and motion saliency.

For more details on visual attention and the human visual perception in general, please refer to the literature [22, 104, 56, 4].

2.1.2 Terms and Definitions

This subsection introduces commonly used terms and explains their role in the retargeting process.

The *importance value* **(IV)** measures the importance of a pixel in relation to the other pixels in an image. For each pixel, this value is derived from visual attention analysis and can then be used to describe entire objects or regions. Pixels with the highest IVs are the ones to be preserved in the retargeting process. Other authors may refer to the IV as *attention value* or *energy value*.

An *importance map* stores the calculated IVs for each pixel in an image, a frame or a whole video. This makes it possible to combine several visual attention functions for the computation of the IVs, for instance color features and face detection. In most cases, the values in the importance map are normalized to $[0, 1]$.

The terms *cost* and *energy* are usually used to describe a sum of IVs. For instance, if a column should be removed, the summed up IVs are the cost to remove this part of an image or video. Or when a global optimization wants to minimize the energy that is lost during retargeting, that means that the function tries to remove the pixels with the lowest IVs until the target size is reached.

A *region of interest* **(ROI)** describes a contiguous region in an image that draws the viewer's attention. Such a region does not necessarily represent an entire object, but an area with similar content like a segmented colorful region or a human face.

An *attention object* **(AO)** is an object within an image which is very likely to attract the viewers attention, like a person or a piece of text. A viewer will notice distortions or jitter in these areas first. An AO can consist of one or more ROIs. To determine the IV of such an

object, the average of the pixel IVs within the object is calculated. A conglomeration of AOs is sometimes called *attention group*.

A *crop rectangle* or *crop window* is a rectangular area in the source media that should be preserved while the surrounding pixels are dropped. Its size is chosen in relation to the target size and the ROIs. If it is used in a video sequence, size and position may change over time.

2.1.3 Operators

Different kinds of operators are used to retarget images and videos. This subsection explains the basic techniques.

Cropping describes a widely used technique where content is removed by discarding every pixel around a crop rectangle of the target size or aspect ratio. The goal is to include as much energy as possible in the rectangle. A widespread assumption is that some area around regions with high IVs should also be contained in the rectangle, as a tight crop does not look aesthetically pleasing. There are different methods to search for a crop rectangle, for instance using a greedy algorithm or doing a global optimization. A typical application is to crop the left and right borders of widescreen cinema movies for television. This kind of approach is unsuitable if relevant objects are located at the borders.

Browsing refers to a technique where at any given time only a rectangular part of the source is chosen and shown as the retargeted result. In contrast to cropping, the position of the rectangle is not fixed to one spot. If used on images, a crop rectangle is moved over the source image to show several interesting regions one after another. The result is an image sequence with the content around the crop rectangle being dropped. In videos, the rectangle may move over the screen during the course of a sequence.

Seam carving removes horizontal or vertical paths of contiguous pixels from within the image that will not be noticed by the viewer. These paths are called seams. The goal is to remove the seams that include the lowest overall energy of all possible paths. A typical example for a seam with low energy would be pixels depicting the sky from a landscape image, because this is a homogenous region where the pixels do not differ much and do not have much importance to the viewer.

Scaling is another widely used technique in retargeting. Every part of the source is shrunk or enlarged uniformly, regardless of the content. It is the default method that is for instance used when an image is viewed on the computer and the user wants to zoom in or out.

Warping is a technique where the source image is scaled non-uniformly to preserve the interesting regions. For example, in case of a size reduction, the distortions are distributed over the non-important regions by shrinking them more than the visually important ones. Typically, a triangle or a quad mesh is used to guide the scaling. The cells of the mesh vary in size, which also directly influences the performance of the algorithm. The lowest possible size is one pixel per cell.

Table 2.1 shows an overview of the methods for image and video retargeting presented in this Chapter. As additional information, the used retargeting operators and the year of publication are given for each method.

2.1.4 Basic Workflow

Nearly all algorithms presented in this Chapter follow a similar workflow that consists of two steps. In the first step, an importance map is generated via visual attention analysis methods like saliency detection [36]. The main goal is to preserve the regions with high importance as well as possible. Then, in the second step, an operator or a combination of operators is used to retarget the image. For images, this is pretty straightforward as there is no temporal component in the retargeting process.

These steps generally also apply to video retargeting, although there are some details that have to be considered, most notably the additional temporal component. When the algorithm does an optimization on the video as a whole, the steps are followed like before. As all frames are known during the process, temporal consistency is achieved by considering the movement of the pixels or whole objects between the frames. If the video is resized frame-by-frame, the two steps are usually repeated for each frame. Temporal consistency is harder to maintain in this case because there is only the information of the previous frames (or a few succeeding frames). All of the video algorithms that do not work on real-time video streams have in common that they split the video into individual shots first and then process each of them separately. For example, this splitting can be done by the algorithm of Zabih *et al.* [105].

Main Operator	Other operators	Type of Media	Year	Author
Cropping	-	Images	2003	Suh et al. [81]
	-	Images	2009	Kopf et al. [43]
	-	Images	2006	Santella et al. [73]
	-	Videos	2010	Carlier et al. [8]
	= Browsing	Images	2003	Liu et al. [54]
	= Browsing	Images	2007	Liu et al. [53]
	Scaling	Videos	2006	Liu et al. [52]
	Scaling	Videos	2010	Li et al. [49]
Seam Carving	-	Images	2007	Avidan and Shamir [2]
	-	Images and Videos	2008	Rubinstein et al. [68]
	-	Images	2012	Noh et al. [61]
	-	Videos	2009	Chiang et al. [13]
	-	Videos	2009/10	Han et al. [30, 29]
	Cropping, Scaling	Images and Videos	2009	Rubinstein et al. [70]
	Scaling	Images	2008	Hwang et al. [35]
	Scaling	Images	2009	Han et al. [31]
	Scaling	Images	2008/09	Dong et al. [16, 17]
	Cropping, Scaling	Images	2012	Dong et al. [15]
	-	Videos	2010	Grundmann et al. [24]
	-	Videos	2013	Yan et al. [102]
Warping	-	Images	2003	Liu et al. [51]
	-	Images	2006	Gal et al. [23]
	-	Images and Videos	2007	Wolf et al. [99]
	-	Images	2009	S.-F. Wang et al. [91]
	-	Videos	2011	S-F. Wang et al. [92]
	-	Images	2008	Y.-S. Wang et al. [96]
	-	Images	2009	Guo et al. [27]
	-	Images	2009	Ren et al. [65, 66]
	-	Images	2012	Zhang et al. [106]
	-	Videos	2009	Y.-S. Wang et al. [93]
	Cropping	Videos	2010	Y.-S. Wang et al. [95]
	Cropping	Videos	2011	Y.-S. Wang et al. [94]
	-	Videos	2009	Krähenbühl et al. [45]
	-	Videos	2013	Lin et al. [50]
	-	Videos	2008	Zhang et al. [107]
	-	Videos	2009	Kim et al. [42]
	-	Videos	2009	Shi et al. [79]
	-	Videos	2010	Yen et al. [103]
Cut out	Scaling	Images	2005	Setlur et al. [77]
Cut out	Scaling	Videos	2007	Cheng et al. [12]
Removing and Rearrangement	-	Images	2009	Pritch et al. [64]
Removing of pixels	Warping	Images	2010	Wu et al. [100]

Table 2.1: *Overview of earlier work on image and video retargeting*

This is a general explanation of the retargeting process, individual algorithms might vary in the execution of the steps, i.e., seam carving for images [2] repeats the steps for each row or column of the image.

2.1.5 Optimization Methods

In the following, we briefly discuss two optimization methods which are widely used by several retargeting operators. The general idea is to define an energy minimization problem to reduce the amount of relevant content that is removed and then use a suitable technique for computing a local or the global minimum.

Dynamic Programming is a technique that divides the complex optimization problem into small local problems that can easily be solved. A prominent example is the path finding of seam carving [2]. Instead of looking for all possible combinations of pixels over the whole image, the algorithm computes the optimal paths by going through each pixel in each row one after another. At each position, it determines its cheapest predecessor from three possible options of the row above. Then it adds the cost of the predecessor to the IV of the current position and uses this as information in the next row. When it has computed the last row, it just has to look for the position with the lowest cost; this is the end point of the globally optimal seam. Dynamic programming can be very efficient compared to an optimization over all pixels at once, but it is not applicable for all types of retargeting problems. An obvious case is warping, where all the cells of the mesh depend on each other during resizing.

Another idea is to interpret the pixels in an image or a video as a graph. The difference of the energy values between adjacent pixels (graph nodes) defines the weights of the edges. To solve the energy minimization problem, the maximum flow in a graph is computed which is equivalent to the minimal cut of the graph (*max-flow/min-cut theorem*) [62]. The idea is to add an artificial source node and a sink node to the graph and compute the maximum flow (capacity) that can be sent from source to sink. Optimization techniques which employ the max-flow/min-cut optimization are usually called *graph cuts*. Boykov and Kolmogorov [6] published a comparison of different max-flow/min-cut algorithms and provide a software library of these methods. Especially when using graph cuts for video retargeting, the memory usage quickly increases and makes this algorithm inapplicable in most scenarios. E.g., without using specific algorithms to reduce the total amount of memory, a PAL resolution video clip (720×576 pixels) with 150 frames would require more than 44 GByte of memory for optimization [43].

2.2 Retargeting Algorithms

The algorithms described here retarget images or videos to a different size or aspect ratio. Oftentimes, operators that were originally developed to adapt images are later extended to also be applicable to video. We thus interleave image and video techniques in our description to keep similar approaches close together in the text.

This section is organized into four broad classes of retargeting operators: cropping (Section 2.2.1) cuts off the least important parts of an image or a video frame. It is usually referred to as browsing, if the cropping window is moved over time to retain different aspects of an image in turn. Seam carving (Section 2.2.2) removes seams of uninteresting pixels inside an image. It is often combined with other techniques like scaling and cropping. Many papers have been published as follow-ups to the original seam carving, enhancing various aspects like image quality or run-time performance. There also exist seam carving approaches suitable for video. Warping (Section 2.2.3), of which scaling is a special case, shrinks parts of an image or video frame non-uniformly according to the image content. Regions that are considered to be important, for example faces or regions with strong motion, are typically shrunk less strongly than unimportant ones. This leads to distortions being in regions that are less visible for the viewer, for instance in a homogeneous region that depicts the sky. Original and unique approaches that do not fit into any of these categories are presented in the miscellaneous section (2.2.4). Retargeting techniques that use more than one operator to achieve their goal are categorized into the class of their dominant operator. For example, if seam carving is combined with scaling, this is described in the seam carving Section.

2.2.1 Cropping

Cropping refers to the definition of a rectangular image area that is kept unchanged while everything outside of it is discarded. A naïve approach to cropping is to keep the center part of the screen intact and cut off the rest. However, important information may be completely removed or objects may be cut. The main challenge of cropping thus lies in finding an ideal size and position of the cropping window.

A more sophisticated approach called browsing moves the cropping window over the screen. If this is applied to an image, a virtual video is created that highlights different aspects of the image one after another [54]. It can also be applied to a video in an effort to always keep the most interesting regions visible in the retargeted result.

Figure 2.2: *The face is detected and then the image is cropped to include the face and some space around it (from Suh et al. 2003).*

"Automatic Thumbnail Cropping and its Effectiveness"

Suh *et al.* [81] propose a basic cropping approach for the automatic generation of thumbnails. Important regions of the image are identified, and the rest of the image is cropped. The authors use two different approaches to identify the Region of Interest (ROI): The first one uses a saliency map [36] and is applicable to all kinds of images. The second one considers semantic information by using face detection and is thus restricted to images of humans.

The crop rectangle tries to satisfy two conflicting conditions: minimizing its size and keeping most of the important regions of the image. First, the cropping rectangle is initialized to contain the pixels with the highest saliency. A greedy algorithm is then used to expand the crop rectangle by incrementally including the next most salient point and a small rectangle around it. Adjacent pixels are added, because peaks often correspond to foreground objects that spread over several pixels. A saliency threshold is determined adaptively to stop the expansion when a large expansion of the crop rectangle would be required to add a small amount of saliency. This adaptive search is necessary because the most effective value varies from image to image.

This approach is a general method for thumbnail cropping and relies on low-level features exclusively. For human image thumbnails, the authors claim better recognizability if face detection is used. They apply CMU's on-line face detection [75, 74] to the image first. The crop rectangle is then chosen to contain all the detected faces (see Figure 2.2).

In the context of web page adaptation, Kopf *et al.* [43] extend the automatic cropping technique presented by Suh *et al.* [81]. To reduce the overall computational effort, border regions which do not contain relevant semantic content are identified and cropped beforehand. An importance map based on faces, text regions and contrast-based features is created. A software tool helps

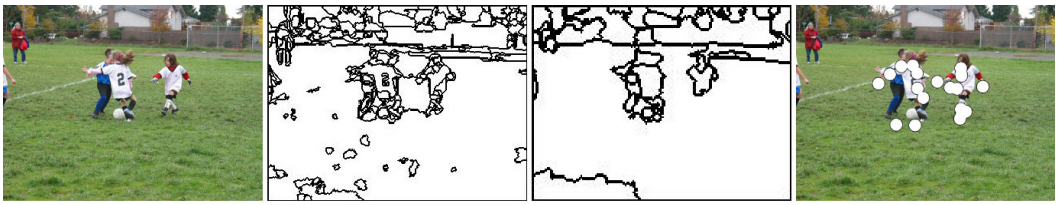


Figure 2.3: *(a) Original image, (b) Fine segmentation, (c) Coarse segmentation, (d) Fixation locations derived from eye tracking measurements (from Santella et al. 2006).*

users to validate the automatically extracted data and enables a preview of the adapted images by simulating the displays of several handheld mobile devices.

"Gaze-based Interaction for Semi-Automatic Photo Cropping"

As an alternative to the fully automatic cropping approaches using low level features and face detection, Santella *et al.* [73] introduce a semi-automatic image retargeting technique. Eye tracking is used to identify the attention objects (AO). AOs in conjunction with a few basic rules of composition are then used to crop the image.

A combination of segmentation and eye movement tracking is used to determine the IV of each pixel. The algorithm first segments the image at a fine scale (see Figure 2.3 (b)). This segmentation divides the image into regions with similar color. A soft assignment is made between the fixations of a viewer and nearby regions in order to find out where the viewer is looking (see Figure 2.3 (d)). Each fixation in a region makes a contribution to the IVs of its pixels which are averaged to one IV for the entire region. To prevent large background regions from gaining a lot of importance through a single fixation, the added IVs fall off sharply with the distance from the fixation point.

Foreground objects are identified by using the IVs to automatically guide the "lazy snapping" approach [48]. This approach is normally initalized manually by a partial hand labeling of an oversegmented image into foreground and background regions (Figure 2.3 (b) shows an oversegmented image). In this algorithm, the top 10% of the regions with the highest importance value are marked as foreground while the 50% with the least importance are marked as background. The rest of the regions are then labeled by the "lazy snapping" technique. Based on these labels, similar regions can be combined to get a coarser segmentation (see Figure 2.3 (c)).

As an AO like a face should not be cropped at its segmentation border, the high IV of this regions is extended a little over its border. This is achieved by averaging the importance map

with a dilated version of itself. The importance map is finally normalized to a maximum value of one.

The cropping task is then formulated as an optimization problem. Each potential crop is assigned an importance value (IV) that represents the amount of important information included in its result. Additionally, the segmentation and three basic rules of photography are taken into account: the entire subject and some context around it should be included, intersections between the edge of the crop and objects should be avoided, and the AOs in an image should be highlighted to increase recognizability.

The inclusion of the AOs is ensured by using the importance map. Penalties are given when the crop rectangle excludes these objects or cuts through them. To avoid cuts through background objects and to make sure that the cuts pass through homogeneous regions, it is counted how many segmentation borders are crossed by the crop rectangle. These crossings are also penalized. Finally, large crop rectangles are penalized to encourage maximizing the AOs in the final image.

Crowdsourced video retargeting suggested by Carlier *et al.* [8] also makes the user part of the identification process, like in [73]. Instead of eye tracking, a video player is offered that allows the viewers of a video to zoom and pan as they please. The positioning of the user's cropping window is tracked and used to generate ROIs. With this information, automatically resized versions can be computed where a cropping rectangle is moved over the input sequence including zooms and artificial cuts.

"Automatic Browsing of Large Pictures on Mobile Devices"

Automatic browsing is a method presented by Liu *et al.* [54]. It simulates the human browsing behavior when watching a large image by showing the interesting regions of the picture in a pan. There are two modes for the user to choose from: skimming mode, where as much information as possible is shown in a limited amount of time, and the perusing mode, where the objective is to minimize the time needed to show a certain percentage of the information contained in the image.

The algorithm itself is divided into several steps. First, the input image is analyzed and an importance map is generated. Heuristic rules are applied to create sets of attention groups (AG), and finally, the optimal path is generated.

Low and high level features like saliency or human faces are extracted to determine a set of AOs. The *minimal perceptible size* (MPS) represents the highest allowed amount of downscal-

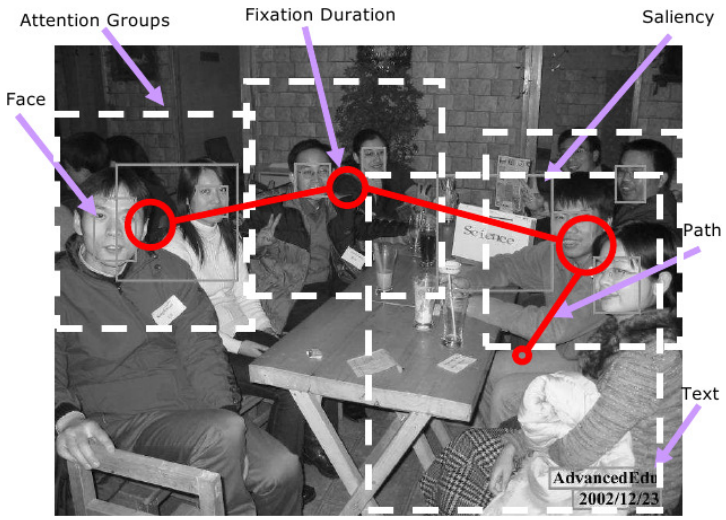


Figure 2.4: *The optimal browsing path (from Liu et al. 2003). The white dotted rectangles mark the positions and the size of the fixation window during the browsing.*

ing of an AO so that its details remain fully recognizable by the beholder. Similarly, a newly introduced *minimal perceptible time* (MPT) is a lower threshold for the fixation duration of an AO on screen to be perceptible by the viewer. The MPT is computed from saliency, detected faces and text. Large AOs are split if their MPS is bigger than the target screen size. This does not apply to detected faces since their MPS is usually smaller than the screen of the target device. Nearby AOs are then combined to AGs that fit into a screen. The importance value (IV) and minimal perceptible time (MPT) of an attention group (AG) are the sum of the individual IV and MPT values of its objects.

The authors approximately model the human browsing behavior by employing two different states: the fixation state where an interesting region is examined and the shifting state where the attention moves towards the next region. The optimal shifting path is the shortest path between the centers of two AGs (see Figure 2.4).

In the skimming mode, the objective is to maximize the information fidelity shown to the user in a limited period of time. The problem is NP-hard, and the algorithm enumerates all possible paths. For the path generation, all AGs are sorted by their IV in decreasing order. The algorithm searches among all possible paths from the starting point and computes the total browsing time and the information fidelity. The browsing path with the highest information fidelity is chosen.

In the perusing mode, the goal is to minimize the time needed to show a certain percentage of information included in the image to the viewer. The searching technique used in skimming

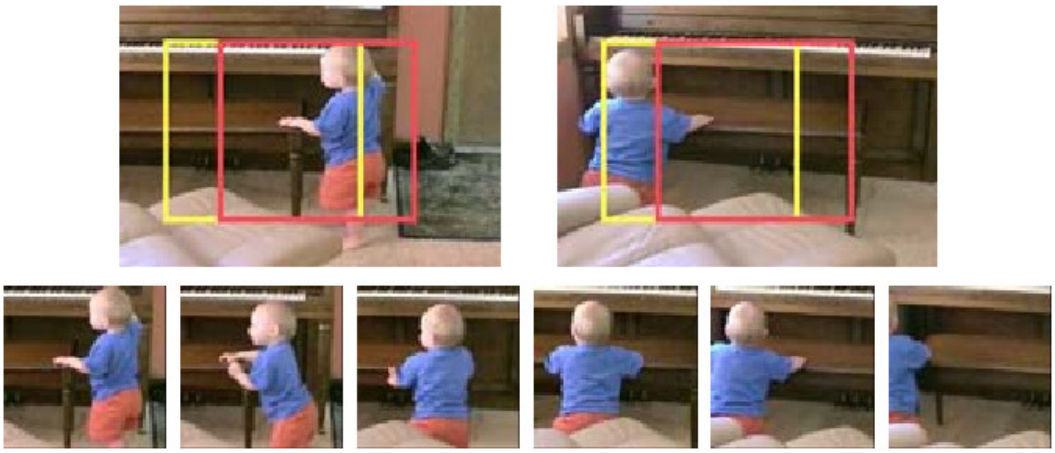


Figure 2.5: *Example of a virtual pan (from Liu et al. 2006). The crop rectangle starts at the position of the red rectangle and follows the baby up to the position of the yellow rectangle.*

mode is slightly adjusted to solve this task. First, the AGs are sorted by ascending MPT. The algorithm then searches among all possible paths from the starting point and calculates the total browsing time and the information fidelity. Finally, the path with the minimum time cost and an information fidelity above the predefined threshold is selected.

Liu *et al.* [53] introduce image browsing using a region-based visual attention analysis. Regions are identified with image segmentation which integrates the spatial connectivity and color features of the pixels to cluster them. Then, an importance value is generated for each region consisting of a position factor, and the sum of its contrast compared to other regions. The regions are ranked by their average importance value and then shown in descending order.

"Video Retargeting: Automating Pan and Scan"

Liu *et al.* [52] introduce a combination of scaling, cropping and browsing for video retargeting. The interesting regions are identified, and a movable crop rectangle is placed over the important regions in the original sequence. A penalty system is used to prevent the algorithm from cutting faces or use larger scale factors.

The basic idea of the algorithm is to define penalties for the loss of important information and minimize these costs. First, the video is segmented into shots. Then, in order to determine which information is relevant, an importance map is computed for each frame. This map combines a saliency map and face detection. Additionally, *motion contrast* is used as a new motion saliency measure. This measure represents the difference in motion between fore-

ground objects and the background. Therefore, the algorithm segments the frames into global background and foreground regions. The magnitude difference of the motion (optical flow) of a pixel compared to the background motion determines its saliency. To avoid noise, a Bilateral Filter [85] is used.

For retargeting, a rectangle in the source is chosen and scaled to the target size. Each possible rectangle is assigned the sum of a number of penalty criteria: the sum of the importance of the cropped pixels, large scale factors, a high aspect ratio change factor (when scaling), cutting off parts of faces, putting a face too close to the edge of a frame and the introduction of cuts or pans.

Although artificial cuts or pans are penalized, they are sometimes necessary to capture all the important information of a shot. The motion of the crop rectangle is limited to three shot types: a crop, a pan or a cut. When using a crop, the crop rectangle does not move and both object and camera movement of the source video are preserved. A brute-force search is used to find the optimal rectangle in this case.

The crop rectangle is able to do a pan, e.g., when an important object is moving across the screen (see Figure 2.5). To avoid introducing artifacts, only horizontal pans are allowed. Additionally, zooms are avoided and each shot is restricted to contain at most a single pan. In order to make the pan seem to be a part of the original shot, it is necessary to prevent instant acceleration. The crop rectangle has to accelerate slowly, get to a constant velocity and then slow down again before reaching the final position. A brute-force search is not efficient to find the optimal pan. Instead, the ideal solution is approximated by finding the optimal rectangle for each frame and then fitting an interpolation curve.

A cut divides the shot into two separate sub-shots. This can be necessary if important information is distributed over the whole screen and cannot be captured with a pan, e.g., during a dialog. Similar to the introduction of pans, performing a cut is restricted to one per shot. The resulting sub-shots must meet two restrictions to be valid: They must have a minimum length of 63 frames, and they must be sufficiently different (assessed by color histogram intersection). The sub-shots are then treated as individual shots, but are limited to being cropped in order to avoid motion matching issues. The penalty for cropped information in the sub-shots is discarded as some of this information will appear in the other shot.

The optimal cut is again approximated. The penalties for the crop rectangles of the left and the right sides are computed, and it is determined which side is shown first by exhaustively searching all orderings with possible rectangles from both sides to find the minimum.

Li *et al.* [49] retarget a video with multi-scale trajectory optimization, which is a combination of cropping and scaling. Each shot of the video sequence is represented as a graph and treated separately. The importance value is determined pixel-wise for each frame within a spatio-temporal neighborhood. Several trajectories with different crop rectangles are computed by a max-flow/min-cut algorithm and ranked by the importance captured in them. The one with the most importance is picked for the shot. It is notable that the size of the crop rectangle is fixed within each shot. Also, the search for the optimal trajectory is split up into two separate problems in the horizontal and vertical directions to speed up the process.

2.2.2 Seam Carving

Seam carving is an influential technique that was first published by Avidan and Shamir in 2007 [2]. Since then, it inspired a host of researchers to work on improvements of various aspects. It also represents the starting point of most of the algorithms presented in this dissertation. The original idea is to remove connected paths of pixels from inside an image in a way that the removal will be unnoticed. Seam carving has been extended to video, combined with other operators, and improved for a better performance in many follow-up papers. This Section presents a selection of them.

"Seam Carving for Content-Aware Image Resizing"

Seam Carving is a technique proposed by Avidan and Shamir [2] for content-aware resizing of images. The basic idea is to remove paths of low energy pixels (*seams*) from top to bottom or from left to right which are not so important for the understanding of the image content. The removal of each seam causes a reduction of the image size by one where vertical seams reduce the width and horizontal seams reduce the height. Figure 2.6 shows an example with vertical seams and the final image after their removal.

A *vertical seam* $\mathbf{s} = \{(x(i), i)\}_{i=1}^{H}$ in an image I with height H is subject to two conditions in order to be valid. First, the seam consists of one and only one pixel in each row, and second the horizontal distance between two adjacent seam pixels $|x(i) - x(i-1)|$ is not allowed to exceed a threshold of T. In case of $T = 1$, the seam pixels of vertical seams are vertically or diagonally connected (8-connected). These conditions lead to the following definition of a vertical seam:

$$\mathbf{s} = \{s_i\}_{i=1}^{H} = \{(x(i), i)\}_{i=1}^{H}, s.t. \forall i : |x(i) - x(i-1)| \leq T \tag{2.1}$$

Figure 2.6: *Left: Original image. Center: The vertical seams that have been found (red). Right: Image after the seams have been removed.*

The pixels of a seam are chosen based on an energy function e, whereas the authors suggest to use the e_1 error norm:

$$\mathbf{e_1}(\mathbf{I}) = |\frac{\partial}{\partial_x} I| + |\frac{\partial}{\partial_y} I| \tag{2.2}$$

Other recent work proposes to use wavelet decomposition [31] as the energy function. Additional information may modify the energy function, like a human attention model based on face detection and saliency [35].

Dynamic programming is used to identify the optimal seam s^*, which is defined as the seam with the minimum cost based on the energy function e. The cost of a seam is the sum of the energy values of all path pixels. To calculate these costs for a vertical seam, the cumulative minimum energy M for all pixels (i, j) is computed by traversing the image from the first row to the last row:

$$\mathbf{M}(\mathbf{i},\mathbf{j}) = e(i,j) + \min\left(M(i-1,j-1), M(i-1,j), M(i-1,j+1)\right) \tag{2.3}$$

The minimum value of M in the last row indicates the total cost of a seam with the lowest energy. By backtracking from this minimum, the path of the optimal seam is found. The computation of M for horizontal seams is equivalent.

In case of an enlargement of the size of the image I by k, the seams are duplicated instead of being removed. However, a duplicated seam is not removed from the image and therefore identified again as the optimal seam. This repetition of the same seam leads to a *"stretching artifact"*. To prevent these artifacts, the next k seams that would be removed are identified and put into an order starting with the seam with the lowest energy. The seams are then duplicated one after another.

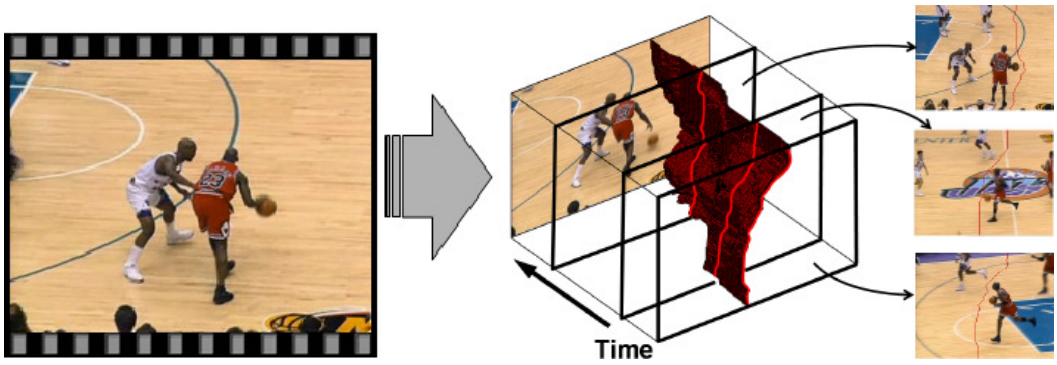


Figure 2.7: *Improved seam carving finds 2D manifolds in the 3D video cube (from Rubinstein et al. 2008, shortened). The seams in each frame are defined by the intersections of the manifold with the frames.*

If both dimensions of an image are to be changed, it is necessary to build the optimal order of vertical and horizontal seams. Therefore, the minimum energy of vertical and horizontal seams is compared, and the lower one is chosen. This is repeated, and an optimal order is built.

"Improved Seam Carving for Video Retargeting"

Rubinstein *et al.* [68] extend the original seam carving approach to the retargeting of videos. Because of the complexity involved in analyzing the three-dimensional space-time cube in video retargeting, dynamic programming is no longer an option to solve the minimization. Thus, graph cuts are proposed as a replacement. Also, a new importance criterion is introduced.

Seam carving can be formulated as a minimum cost graph cut problem. In the following, it is assumed that *vertical seams* are searched in an image. A grid-like graph is constructed where each node represents a pixel and is connected to each of its neighboring pixels in the space-time cube. All the pixels in the leftmost and rightmost columns are connected to two virtual terminal nodes S (source) and T (sink) with infinite weight arcs.

The optimal seam is defined as the $S/Tcut$ (or simply cut) with the minimum costs among all valid cuts. A cut partitions a graph into two disjoint subsets. The cost of a cut is the sum of the costs of the crossed arcs between the virtual terminal nodes. The pixels included in a seam are the ones on the left side of the cut. There are two additional constraints that must be satisfied in order to get a valid seam: Monotonicity and connectivity. The first constraint ensures that a seam includes exactly one pixel in each row, while the second constraint provides connectedness of the seam pixels.

In the original seam carving approach, the seams with the lowest importance value are removed. However, this may lead to distortions and artifacts, because it ignores the new edges that are brought into the image between previously not adjacent pixels as a result of pixel removal. The new importance criterion proposed by the authors considers the expected results and determines the importance value of each pixel from these new edges. This new importance criterion can also be used in dynamic programming for the retargeting of images.

To extend this to videos, time is added to the graph as a third dimension. The virtual terminal nodes are connected to all left and right columns of frames, as well as the normal nodes to their left and right neighbors (from the previous and the following frame, respectively). The graph construction includes the same restrictions as before to assure that a cut will be monotonic and connected. The result is a 2D manifold in a 3D space-time volume that assigns each frame a 1D seam (see Figure 2.7).

Noh *et al.* [61] extend seam carving [2] for a better preservation of regular shapes. The algorithm uses a new energy criterion based on gradient differences. This criterion uses the differences in gradient orientation and magnitude before and after a seam is removed, similar to forward energy [68]. This minimizes the differences between the pixels that become adjacent after the removal of a seam.

Seam carving [2, 68] is the basis of a highly parallelizable real-time content-aware video retargeting technique proposed by Chiang *et al.* [13]. In the retargeting process, the video is computed frame by frame, which makes the technique also suitable for images. A frame is enhanced with a newly introduced just-noticeable-distortion (JND) model, and then an importance map is computed by applying an edge detection filter. JND is composed of several image features like gradient, entropy, visual saliency, segmentation, etc. Also, a multiseam search scheme is presented which can find multiple seams without the need to recompute the importance map after each seam by backtracking and splitting found seam paths. The seams are searched sequentially, frame by frame. For temporal coherence, the seams need to be connected which leads to them forming a continuous surface in the 3D video cube (similar to the manifold shown in Figure 2.7). The algorithm is intended for an implementation on the GPU.

Han *et al.* [30, 29] extend the seam carving technique by Rubinstein *et al.* [68] to simultaneously find multiple 3D surfaces in a 4D graph by global optimization via S/T cuts. This 4D graph is composed of the dimensions of the 3D video cube (width, height and time) and

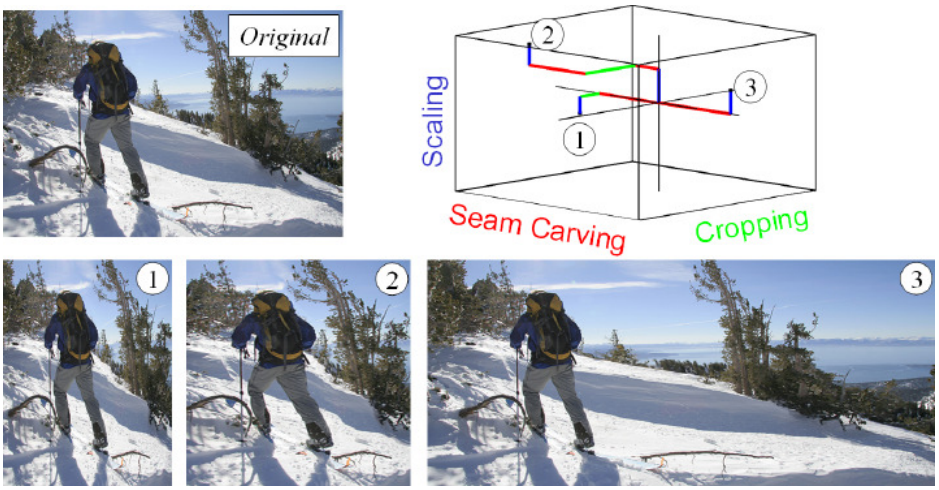


Figure 2.8: *Three possible results of multi-operator retargeting by using different paths in the resizing space (from Rubinstein et al. 2009).*

a fourth dimension that corresponds to a set of sub-graphs, each containing a video cube with the previous sizes. Each sub-graph is used to find one 3D surface.
Compared to seam carving, the algorithm in [30] searches for the global optimum of k seams instead of finding the optimal seam iteratively k-times. Image gradient and optical flow are used as measurements of visual attention, but may be replaced or extended with any visual attention model. Slight enhancements are presented in [29], where additional constraints for region smoothness and seam shape are incorporated to further improve the results.

"Multi-operator Media Retargeting"

Rubinstein *et al.* [70] proposed multi-operator media retargeting, which is a combination of three different resizing operators: bi-cubic scaling, cropping and seam carving. An image similarity measurement called *bi-directional warping* is introduced to determine in which order the operators are optimally used. This order is found via dynamic programming in the *resizing space*, which is defined as a multi-dimensional space to combine the operators. Each dimension equals one operator in either width or height direction. With three operators in the algorithm, this leads to a 6-dimensional resizing space.

The retargeting operations have to be *discrete* and *separable* (in dimension). Therefore, they are restricted to the atomic operation of reducing the image width or height by one at a time. As an example, if an image is to be reduced in width by k, then the retargeting operators are used k times. As a special case, k successive scaling operations by one pixel are combined into one scaling operation of k pixels to avoid blur.

The atomic operations are used in the resizing space, where each operation is defined by two dimensions – width and height, and a step in a direction equals a retargeting operation of one pixel in width or height. Each path in this space corresponds to a combination of the retargeting operators to reach the target size (see Figure 2.8).

A global similarity measure called *Bi-Directional Warping* (BDW) is used to determine the optimal path in the resizing space. It is based on *Dynamic Time Warping* (DTW) [72], which is an algorithm for measuring the similarity between two one-dimensional signals by non-linearly warping one signal onto the other. The constraints of DTW - the first and last elements of the signals must be mapped to each other, all elements have to be used in the warp, and the warp has to be *monotonic*, which means no backward matching - are a little relaxed for BDW. It is no longer necessary to match the first and the last elements of the signals, and it is also allowed to introduce gaps in the warp. Because the authors only want a single match that minimizes the warping cost, one-to-many matchings from the source image to the target image are not permitted while many-to-one matchings are still allowed.

The paths in the resizing space are categorized as either *mixed paths* or *regular paths*. In a mixed path, the order of the operators and the number of times they are used is not known ahead of time. They can be freely combined and used as often as necessary to achieve the target size. In a regular path, the combination of the operators is chosen before the retargeting is started (for instance, begin with scaling, then seam carving and finally cropping). The algorithm chooses how often the individual operators are used. Dynamic programming is used to find the optimal paths in both of these optimization problems.

The multi-operator algorithm can also be extended to the retargeting of video. In this case, the algorithm finds the best *regular* paths for the key-frames and interpolates the number of times each operator is used in each intermediate frame.

Seam carving is currently very popular, and many researchers introduce hybrid methods to combine it with other retargeting operators. Hwang *et al.* [35] extend the visual attention analysis by adding a saliency map and face detection. Also, they introduce a switching scheme which changes the resizing operator to scaling once the importance value for the minimal seam is above a threshold.

Han *et al.* [31] propose to base the visual attention identification on wavelet decomposition. They use the difference of the minimum seam compared to the maximum seam in relation to the average importance value to determine the switching point to scaling.

Several methods combining multiple operators are introduced by Dong *et al.* [16, 17, 15]. When joining seam carving and scaling, they suggest to alternate between these operators adaptively. This switching can be based on the changes of the importance value between two seams that should be removed [16]. The resulting scaling number is relative to the seam importance and the number of seams that have been removed. This alternating is repeated until the target size is reached. Also, Hierarchically Accelerated Dynamic Programming (HADP) [78] is suggested by the same authors for speeding up the seam carving algorithm.

Alternating can also be done by optimizing an image distance function [17]. This function consists of a patch-based approach called *Image Euclidean Distance* (IMED) [90], *Dominant Color Descriptor* (DCD) [58], similarity and seam importance variation. The algorithm starts with seam carving and scales a version of the image to the target size after each removed seam. The resized version with the minimum distance is used as the final result. Both presented methods are also able to enlarge images.

In an operator cost-based approach, scaling, cropping and seam carving are combined [15]. By statistical analysis of the operator costs, an operator for each resizing step is chosen. Dependent on image energy and DCD, these cost functions evaluate how much damage each operator causes to the current image. User studies are conducted to analyze the users' compromise tendency when an operator has to damage the original image.

"Discontinuous Seam-Carving for Video Retargeting"

One of the key aspects in previous seam carving approaches is the condition that the pixels of a seam must be connected. Grundmann *et al.* [24] present an algorithm which allows the seams to be spatially and temporally discontinuous. The authors suggest an appearance-based temporal coherence measure that allows the algorithm to work on a frame-by-frame basis and lets the seams be disconnected in the temporal dimension (in contrast to the surfaces found in [68]). This measure is also used in a modified version in the spatial domain. As the algorithm computes the resizing frame by frame, it is suitable for the retargeting of long sequences or streams.

The importance value for each pixel in a frame is composed of a spatial cost, a temporal cost and the saliency cost of removing that pixel from the frame. With these importance maps, seams can be calculated for each frame separately using dynamic programming [2]. In the following, the removal of vertical seams is discussed.

Figure 2.9: *Example of a temporally discontinuous seam (from Grundmann et al. 2010). In the left image, the traced x-t slice of the running person in the frames on the right is shown. The red seams are produced by seam carving with 2D surfaces in the video cube and intersect with the traced person. In comparison, the green seams do not have to be temporally connected and can jump behind the person without distorting it.*

For the computation of the temporal costs in a frame, the seam from the previous frame is used as a starting point. For each pixel, a comparison is made between the current frame with this pixel removed and the identical frame without the seam pixel in the same row. The difference is measured with the *sum-of-squared-differences* (SSD) and used as the temporal cost. This makes it possible for seams to avoid colliding with a moving object that crosses the screen, as shown in Figure 2.9.

Seams are also allowed to be discontinuous in the spatial domain. In a similar manner to the measure of temporal coherence, a pixel considers all pixels in the row above as potential predecessors of the seam. Since the optimal predecessor usually lies within ~ 15 pixels, the search window is limited to reduce the complexity and to implicitly enforce that seams only do piecewise jumps.

Additionally, a saliency measure is presented which averages the saliency over spatio-temporal regions found by video segementation [25]. If automatic methods for the saliency measure fail, users can manually highlight salient regions.

Yan *et al.* [102] present a similar approach as [24] which uses seam carving [2, 68] on a frame-by-frame basis. Their goal is to provide a framework which enables every seam carving method for images to be used on videos. First, importance maps are calculated in each frame based on gradient information by a standard Sobel operator and a saliency map. For temporal coherence, a vertical seam is divided into several parts, each covering about the same amount of rows (horizontal seams work similarily). In the middle of each part is a so-called key point which is used to modify the importance map of the following frame. In each frame after the first, a search area is defined around the key points of the according seam from the previous

frame. Then, all pixels in the search area and an equally large area around them are matched against the key point with a same sized area around it for the best fit based on the sum of absolute differences. The importance map is adjusted by lowering the energy of matching pixels and raising the other pixels, forming a low-energy path in the importance map. This map can be used with any seam carving implementation to retarget a frame independently from the other frames.

2.2.3 Warping

Warping is a specialized form of scaling. While scaling resizes the entire image uniformly by the same scale factor, warping is content-adaptive and determines scaling factors in a more fine-grained way. In general, a stronger scale factor is applied to regions that are not the focus of a viewer's attention while the outstanding ones are only modified as little as possible. For this purpose, the image is often subdivided into a mesh grid with triangular or quadrangular cells. Typically there are more degrees of freedom than just a scale factor for each cell, i.e., the cells are allowed to be deformed in the retargeting process.

When applying warping to the frames of a video, waving may arise as an undesirable artifact. The papers on video warping thus focus on image stability and run-time performance.

"Automatic Image Retargeting with Fisheye-View Warping"

Liu *et al.* [51] developed a non-photo-realistic fisheye-view warping method which emphasizes the Region of Interest (ROI) and scales down the less important parts of an image. Objects closer to the center appear larger than those further away. This is either done by radial warping or Cartesian warping. The former scales down each pixel in relation to its place between the ROI and the border while the latter divides the image into nine regions and scales them down according to their position.

In the algorithm, a single minimal rectangular ROI is searched that covers the important objects. To measure the importance, an importance map is created consisting of a saliency map computed using Ma's contrast-based method [57] and face detection with the Adaboost method [88]. To reduce the computational effort, a greedy algorithm is used to grow the ROI from an initial candidate ROI (found e.g. by face detection) until sufficient importance is covered.

As preserving the ROI is the main objective, its aspect ratio is not changed during retargeting. Also, the ROI is scaled down until it takes up 70% or less of the target screen. It will

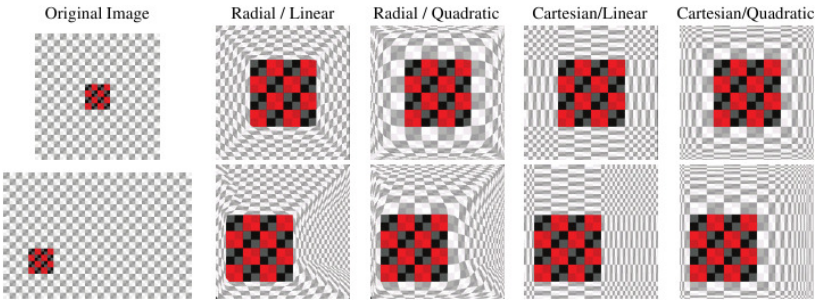


Figure 2.10: *Comparison of radial warping and Cartesian warping (from Liu et al. 2003). The red and black squares mark the ROI.*

not be enlarged in any case. The new position in the target image is determined so that the proportions of the border remain constant. Subsequently, the rest of the image is scaled down by either radial warping or Cartesian warping (see Figure 2.10).

Radial warping is centered around the center of the ROI. The scaling function is a two-dimensional curve. The area within the ROI is linearly interpolated, while a quadratic Bézier curve is used for the rest. In this way, each pixel in the source image is mapped into the target image in relation to its position between the center and the border. The further an object is away from the center, the smaller it will appear in the result.

Cartesian warping divides the image into a 3 x 3 grid with the ROI as the center cell. It is called Cartesian because it applies the fisheye warp scaling function in each cartesian dimension. Each grid cell is scaled independently to fit into its target location.

Another early version of a warping-based retargeting technique is proposed by Gal *et al.* [23] for 2D texture mapping. With this technique, images can be mapped to different 2D shapes or the sides of 3D shapes, for instance a flag or a pyramid. They use inhomogeneous mapping guided by a user-generated importance map to deform a grid representing the source image. This was one of the first warping methods to achieve photo-realistic results.

"Non-homogeneous Content-driven Video-Retargeting"

Wolf *et al.* [99] consider the problem of retargeting videos to different aspect ratios and sizes. The method is an asymmetric scaling function where the interesting regions are shrunk less than the other regions in order to retain the original size of important objects if possible. The algorithm is intended for adapting video streams. It works in real-time on low resolution videos.

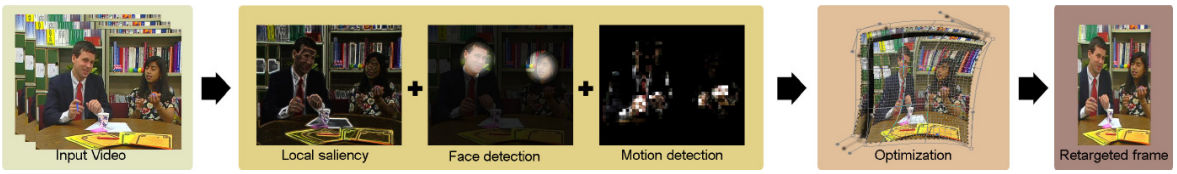


Figure 2.11: *Overview of the video warping algorithm by Wolf et al. (from Wolf et al. 2007). First, the importance value of each pixel is computed. Then, the image is mapped to the new size by nonuniform scaling.*

The algorithm retargets the individual frames of a video by mapping the pixels of the source frame into the target frame with the desired size. Pixels in the less interesting regions are blended with each other in the process. Also, each frame is computed separately with only the computed mapping of the predecessor given.

First, an importance value consisting of saliency, face detection and motion detection is computed and assigned to each pixel. Then, the pixels of the source frame are mapped to the target frame by computing a new location (see the overview in Figure 2.11). This mapping is formulated as a sparse linear system of equations that is solved in a least squares manner. The x and y variables are computed separately using the same linear method. In the following, only the computation of the y variable is considered.

The mapping is constrained by four conditions. Each pixel has to have a fixed distance to its left and right neighbors, they need to be mapped to a position similar to their upper and lower neighbors, the mapping must be similar to the mapping of the same pixel one frame before, and the new locations must fit into the retargeted frame. The importance value determines whether a pixel is important and is mapped far from its upper and lower neighbors, or is unimportant and is blended with them. In this way, the important regions maintain their original size if possible.

S.-F. Wang *et al.* [91] extend the warping technique for videos introduced by Wolf et al. [99] and use it for image retargeting. They add line detection for an improved preservation of image structure and use subsampling for increasing the performance of the algorithm.

S.-F. Wang *et al.* [92] also present an algorithm that preserves structured objects through the introduction of block structure energy. This technique uses salient edges found by the color tensor [97] and the Harris corner detector [32] to put bounding boxes around those objects and to scale them uniformly in the resizing process. Additionally, an optimal compressability rate is computed based on the gradient magnitude and orientation distribution of the image. This rate indicates how well the other areas in the image can be utilized for retargeting. According

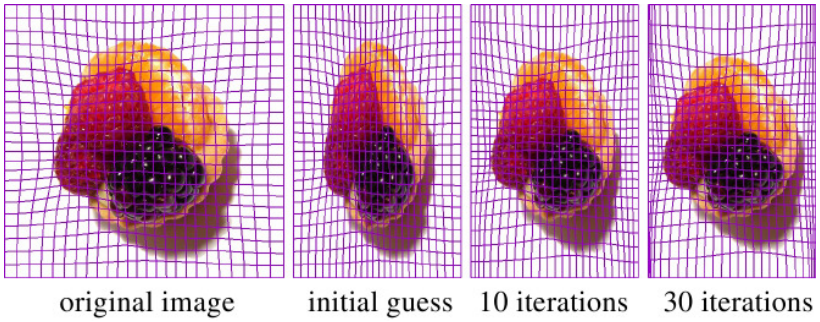


Figure 2.12: *In the original image, the mesh is adapted to the image content based on a significance map. The initial guess is generated by a homogeneous scale (from Y.-S. Wang et al. 2008)*

to the compressability rate, the image is warped to the optimal size that meets the target aspect ratio and then scaled uniformly to the target size.

"Optimized scale-and-stretch for image resizing"

Another warping approach that generates images that look more realistic is proposed by Y.-S. Wang *et al.* [96]. The image is partitioned into small regions, and an optimal scaling factor is determined iteratively for each of the regions. This allocation is achieved by placing a mesh over the source image. An importance map marks the important regions of the image. These regions are scaled uniformly while the less important regions are allowed to be squeezed or stretched, as distortions will not be so obvious in these areas (see Figure 2.12).

The algorithm represents the image as a mesh consisting of quads with the vertices in the upper left and the lower right determining the size. In order to keep the image rectangular, the boundary vertices are constrained to run along their respective boundaries. In order to retarget the image, the lower right vertex is moved to determine the new size, and a deformed mesh geometry is searched which transforms each quad according to its importance. The importance of each quad is measured as the average of the importance values of its pixels which in turn are computed from a gradient and a saliency map. The resulting values are normalized and used as weights later on.

The optimization considers the deformation of the quads and the bending of the grid lines. For the quads, a shape distortion energy is computed which represents the difference between the deformed quad and a uniformly scaled version of the original quad. This distortion energy is combined with the importance value as a weight, since distortions in quads with lesser importance are more easily tolerated than in quads with high importance. The bending of the

grid lines is computed as the difference between the edge length ratio before and after the deformation.

The algorithm computes the retargeted image iteratively by minimizing quad deformation and grid line bending. It starts from an initial guess and continues the alternating steps until all the vertex movements are smaller than a threshold. Additionally it is assumed that prominent objects span several adjacent quads. Therefore, it is proposed to smooth the difference of the scaling factors of adjacent quads in each iteration.

A good initial guess is important as it reduces the number of iterations needed to find a nearly optimal solution to the minimization problem. A suitable guess can be obtained by a uniform scaling of the original mesh (see the initial guess in Figure 2.12). Another possibility is to have a user reduce the size of the image step by step in real-time by continuously manipulating the handle vertex in the lower right corner. In this case, the first retargeting is achieved as described before and the following retargetings are done by using the previous result as the initial guess. This is effective since the deformation is continuous and the optimal solutions for the following frames are similar to each other.

To further enhance the quality of the results, the initial uniform grid mesh can be adapted to the importance map. In this case, shape and size of the quads are slightly deformed to attract more vertices to the important regions (see the original image in Figure 2.12). The mesh connectivity has to be kept intact in the process.

"Image Retargeting Using Mesh Parametrization"

Guo *et al.* [27] also use a mesh to warp the image to the target size. However, they use a triangular mesh that is consistent with the underlying image structure, and solve the retargeting as a constrained mesh parametrization problem.

The algorithm starts by computing a contrast-based saliency map as proposed by Ma *et al.* [57]. A normalized importance value is assigned to each pixel. Additionally, the Viola-Jones face detector [88] is employed and enhanced by a method developed by the authors that uses a detected face to estimate the rest of the body of the person. The importance values in the areas of a human body are automatically set to the highest value of 1.

In order to create the mesh, feature points are detected in the image. First, the image boundary is discretized, and all boundary points are used as features. Then, an edge detection operator (e.g., Canny [7]) is used to detect the edges in the image. A distance threshold helps filtering the pixels as not all edge pixels are used as feature points. Additionally, some auxiliary

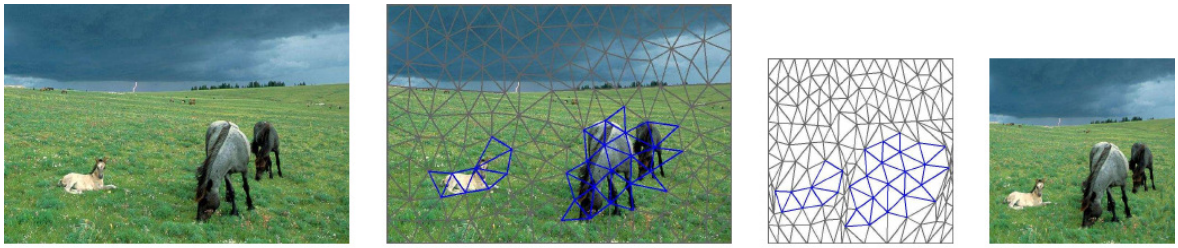


Figure 2.13: *(a) Original image, (b) Image with the overlaid mesh. Adjacent triangles with a saliency value above a certain threshold are clustered into objects (marked in blue) (c) The resized mesh, (d) The resized image (from Guo et al. 2009, shortened).*

points are added to keep uniformity in the point density. Lastly, the constrained Delaunay triangulation algorithm [76] is used to create a feature-consistent mesh.

This mesh is then associated with the previously computed importance map. The importance value assigned to a triangle in the mesh is the average of the importance values of all pixels in the triangle. Three constraints are defined and incorporated into the mesh parametrization process to preserve important parts of the image: the boundary, saliency and structure constraint.

The boundary constraint maps the four corresponding corner points of the mesh from the original image onto the corners of the target image size. Parametrization is used to find the other boundary points of the image border, where only one coordinate for each point has to be computed because they are located on a line. Salient objects are preserved and emphasized by the saliency constraint. Adjacent triangles above a certain importance threshold are clustered into an object (see Figure 2.13 (b)). To avoid distortion in the object, only a scaling transformation is allowed on those triangles. Also, the relative scales of the objects are exaggerated in contrast to the other regions in order to emphasize these objects (see Figure 2.13 (c) and (d)).

In the structure constraint, strong edge segments are maintained. They are detected with the Hough transformation and then filtered with a threshold to include only segments of a certain length. These segments are discretized into points before the mesh is constructed with the Delaunay triangulation. This ensures that the strong edges correspond to mesh edges.

Mesh parametrization is traditionally used to find a correspondance between an unorganised 3D mesh and a 2D mesh on a plane. The authors adapted the parametrization to use it for a 2D-to-2D optimization, as images have only two dimensions. The target mesh is calculated by minimizing a constrained energy function using Newton's method [63]. With this method,

a series of sparse linear equations are solved iteratively. The initial solutions of the equations are found by scaling the input image uniformly.

Another retargeting approach is presented by Ren *et al.* [65]. They formulate the retargeting as a global optimization problem, which is solved by linear programming. After calculating the importance value for each pixel from saliency and face detection, each pixel is treated as a separate unit with a flexible size. After the optimization, the sizes of the units are adjusted, and units in the same row/column are combined to generate new units with the size of a pixel. The result is a warped version of the original image.

Ren *et al.* [66] also proposed a warping-based technique that formulates the retargeting problem as an relocation problem of the mesh vertices. Curve-edge trapezoid meshes are used to represent the image. Two energy maps, an importance and a sensitivity map, are generated to emphasize important content as well as to prevent visual distortions. The importance map uses saliency and face detection, the sensitivity map uses a weighted gradient map that was introduced by Y.-S. Wang *et al.* [96]. Then, the regions are warped by optimally relocating the mesh vertices under the energy constraints.

Zhang *et al.* [106] think that texture regions in an image should be taken more into account during warping. Their algorithm starts by segmenting the image using the mean-shift segmentation technique [14]. Then, a saliency map is computed for each pixel, and the Gabor-filtering-based texture descriptor [101] is used to identify texture regions. This additional information is used during the warping step to better preserve these regions.

"Motion-Aware Temporal Coherence for Video Resizing"

Another warping-based approach for the retargeting of videos is suggested by Y.-S. Wang *et al.* [93]. It builds on the mesh-based scale-and-stretch technique introduced by the same author [96] and aligns consecutive frames of a video by estimating interframe camera motion to retain camera motion in the resized video (see Figure 2.14). Additionally, object motion is preserved by detecting distinct moving objects in multiple frames and resizing them uniformly. The algorithm does not work in real-time.

The goal of the algorithm is to achieve temporal coherence in the retargeting process, i.e., the prevention of motion artifacts like flickering and waving. The importance map is thus enhanced by additional constraints in the optimization step to prevent artifacts introduced by

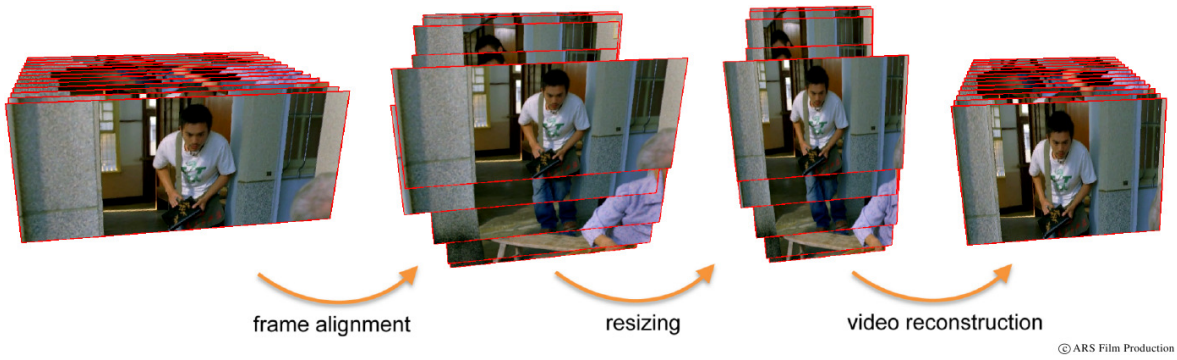


Figure 2.14: *Overview of the steps taken in 'Motion-Aware Temporal Coherence for Video Retargeting' (from Y.-S. Wang et al. 2009). The frames are aligned and then resized to the target size.*

motion. There are two types of motion with a different nature in a video: camera motion, bringing a global visual effect to an entire scene, and object motion which only affects a part of a scene and is independent of the camera movement.

To preserve the camera motion, the consecutive frames of a video are aligned by estimating the interframe camera motion. A restricted model consisting of scaling and translation is used as it is more robust than a full 2D affine transformation. The feature points of each frame are detected by SIFT [55], and the feature correspondence is robustly extracted using the RANSAC algorithm [21]. Since there generally is no single frame that shares enough background with all the other frames of a video, no single fixed reference frame is used. The frames are aligned to a common camera coordinate system instead. The motion is preserved by constraining relative positions of every pair of consecutive grid meshes.

Once the frames are aligned, an importance map is computed by multiplication of saliency and gradient magnitude, like the one they used in [96]. To ensure temporal coherence of the importance map, it is blended with the importance map of several aligned frames. The IV of a pixel is defined as the weighted maximum importance at a given pixel among the aligned frames. Like this, salient information in temporal and spatial contexts can be detected.

A defined motion saliency map is explicitly not included in the computation of the importance map as it is sufficient for the warping algorithm to generate object motion by subtracting camera motion from the original video. An image mosaic [83] is built as a background scene image for each frame, and a motion saliency map is generated from the RGB color difference between an aligned frame and its background image. To preserve the detected motion, all identified moving areas of a dynamic object are resized in a consistent manner. This is achieved by adding additional constraints to the optimization.

Figure 2.15: *The regions between the red lines are defined as critical regions and should be preserved in the retargeting process (from Y.-S. Wang et al. 2010). These regions may differ in size between the frames and the content outside of them is allowed to be discarded.*

The optimization is solved as a nonlinear least-squares problem where the non-linearity is caused by the constraints that penalize grid line bending. In order to reduce the processing time of the algorithm, a long video (of a single scene) is divided into short clips with some slightly overlapping frames. The resizing problem is then solved on these individual clips, and additional temporal coherence constraints are added to the overlapping frames.

"Motion-based Video Retargeting with Optimized Crop-and-Warp"

In a novel approach, Y.-S. Wang *et al.* [95] combine cropping with warping and distribute the distortions in both spatial and temporal dimensions. They define critical regions in each frame that must be preserved while the rest of the frame is allowed to be cropped. One of the main contributions of this technique is to give up the notion that all important content has to be preserved in each frame of a sequence.

Furthermore, the authors believe that motion and temporal dynamics are the core considerations when retargeting videos. This leads to the assumption that the problem of video retargeting can not be solved by simply expanding image-based algorithms with temporal constraints.

The proposed algorithm starts with the definition of critical regions in the frames of the video sequence. A critical region in a frame contains either content which has just entered or is about to dissappear as well as actively moving foreground objects. When reducing the width, these regions are defined by the leftmost and the rightmost columns that include pixels of this important content (see Figure 2.15). In the following description of the algorithm, a reduction of the width is assumed.

Optical flow is used to determine if content moves in to or out of a frame. Actively moving foreground objects are detected by the entropy of the column's optical flow. This categorization is important, because the actual cropping is later done by placing non-critical regions either inside or outside of the target video borders. The shape of important objects is preserved by

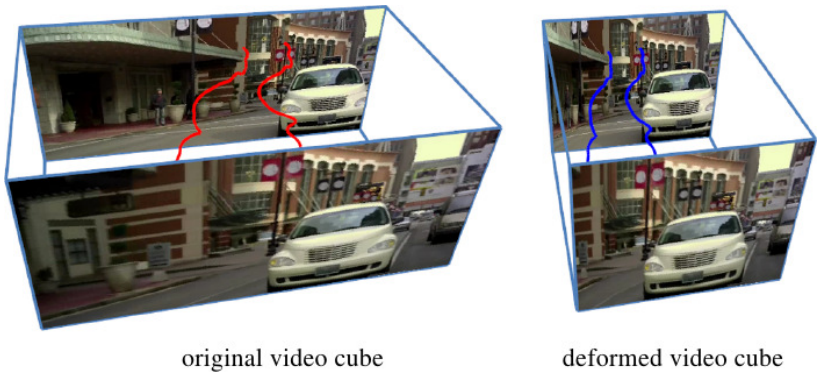


Figure 2.16: *The red lines show motion pathlines in a 3D video cube (from Y.-S. Wang et al. 2011 , shortened). In order to achieve temporal coherence, it is important that the deformation of these lines is consistent. The result of this operation is shown by the blue lines in the deformed video cube.*

adding an importance measure to the mesh in each frame. This measure is generated as a combination of gradient magnitudes, a saliency map [36] and face detection [89]. The goal is to keep the deformation of the quads with important content during the warping to a minimum. Additionally, two energy terms from Wolf *et al.* [99] are used to prevent strong bending of the grid lines in the mesh (one for the vertical direction, one for the horizontal).

For the preservation of temporal coherence, an energy term is added which represents the motion information. This information is gained by using the optical flow to determine the evolution of every quad. Lastly, another constraint reduces the virtual camera movement which may occur during resizing. The minimization of the objective function is a least-squares problem which is solved by iterative minimization.

"Scalable and Coherent Video Resizing with Per-Frame Optimization"

Many previously mentioned algorithms retarget videos via global optimization. However, this may lead to high memory requirements and long runtimes due to processing all the frames of a shot at once. In order to make video retargeting scalable, Y.-S. Wang *et al.* [94] present an approach which seperates it into a spatial and a temporal component that can be processed sequentially. First, each frame is resized seperately. Then, the motion pathlines of pixels in the optical flow are optimized. Finally, the retargeting is repeated with the new information. To improve their results, the authors also add cropping in the retargeting process, comparable to their previous crop-and-warp approach [95].

The algorithm starts by resizing each frame of a shot individually using the scale-and-stretch image retargeting technique [96]. Other operators can also be used as long as they are

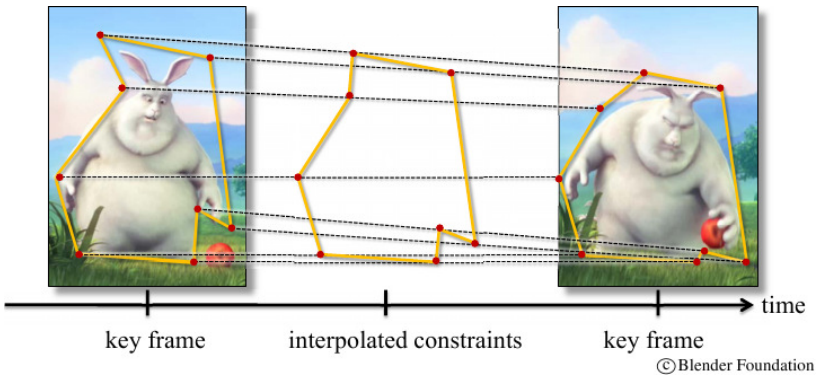


Figure 2.17: *Example of key frame editing (from Krähenbühl et al. 2009). The object is marked in the key frames and interpolated for the frames in between.*

able to track the correspondences of the pixels between the originial and the adapted images. This tracking is important as the pathlines of the pixel motion in the optical flow are optimized in the next step in the way that the deformation of these lines is consistent (see Figure 2.16). A regular grid with a size of 20×20 pixels is used with the vertices of the grid in the first frame as seeding points for the pathlines. Lastly, the resizing of each frame is repeated with the position of the nodes of the pathlines as an additional constraint.

As mentioned, the authors add cropping to the resizing steps in a comparable way to their crop-and-warp approach [95]. As they do not want a global optimization, they imitate the technique for the use on a per-frame basis. Two vertical lines contain the critical regions in each frame that should be preserved. The frames are then reduced in size via warping until the largest critical region of all the resized frames just fits into the target size. In the next step, the frames are adjusted so that all critical regions are positioned in the target video cube, and finally cropped to the target size.

"A System for Retargeting of Streaming Video"

An advantage of the warping technique from Krähenbühl *et al.* [45] is the ability to retarget a streaming video with low resolution in real-time. Also, they combine automatically detected image features with high-level features annotated by the producer. A distinctive technical feature is the computation of a per-pixel warp instead of the deformation of a coarser grid mesh. The output frames are additionally rendered by a hardware accelerated per-pixel EWA (elliptical weighted average) splatting [108] to minimize aliasing artifacts.

Visually important features are automatically detected in the first step of the algorithm. This is done by combining image gradients, saliency, motion and scene changes, with the ability

to work in real-time in mind. For this reason, the authors use a GPU implementation of a bottom-up strategy by Guo *et al.* [26] to generate an importance map. Temporal saliency is detected by estimating the optical flow between two consecutive frames.

Edges are identified by a standard Sobel operator. The bending of theses edges is avoided by a smoothness constraint following Wolf *et al.* [99] while an additional constraint prevents edge blurring or vanishing of detail. With the goal of retargeting real-time video streams, temporal coherence is difficult as an in-depth treatment requires knowledge of the full video cube. To solve this problem, a scene cut detector is applied first which detects discontinuities in the video. This is combined with a temporal filtering of the per-frame importance maps with a lookahead of a few frames.

As a complement to the automatically detected low-level features, interactive key frame editing is added to identify high-level features. This includes direct editing of the feature maps to add the position of objects or straight lines. The shape of these marked objects is interpolated linearly in frames between two key frames (see Figure 2.17). Generally, the same scale factor is used to change the scale of all features to retain the global spatial relations and the overall scene composition.

An iterative multi-grid solver running on the GPU computes the energy optimization for the per-pixel warp. Afterwards, the output frames are rendered using the EWA splatting technique [108] to avoid aliasing artifacts. This technique combines a reconstruction filter to continuously approximate the discrete input signal and an additional lowpass filter to bandlimit the maximum allowable frequencies set by the output resolution.

"Content-aware Video Retargeting Using Object-preserving Warping"

In contrast to the technique [45] that works on a pixel level, Lin *et al.* [50] focus on the preservation of whole objects in a video. Therefore, an object-based importance map is generated that identifies attention objects in the 3D space-time cube. These objects are then preserved while non-important regions are warped similar to linear scaling.

The algorithm uses a uniform grid mesh for the warping. It starts with a spatio-temporal segmentation [25] in order to find volumetric objects in the 3D space-time cube. It should be noted that an object in this case is a segmented connected region over all frames and not a defined object like a person or a chair. After the segmentation, a saliency value for each pixel in each frame is calculated. The values of the pixels within a segmented object are then averaged. All grids belonging to the same object are assigned the same importance value, representing the importance value of the object (see Figure 2.18).

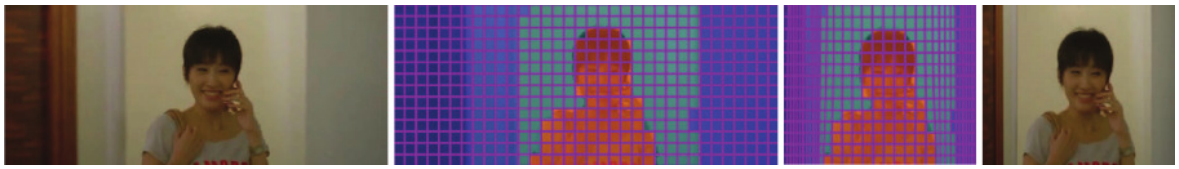


Figure 2.18: *Example of object-based warping (from Lin et al. 2013). Each grid belonging to the same object is assigned a similar importance value. That way, important objects like the person in the frame can be preserved as rigid as possible.*

In the optimization step, the algorithm aims to keep important volumetric objects as unchanged as possible while warping the non-important regions in a way similar to linear scaling. The optimization itself is an iterative process which is initialized with frame-based warps.

Zhang *et al.* [107] propose shrinkability maps to create a multi-size video that can be retargeted in real-time. The shrinkability maps are precomputed and determine the size of each pixel in different target sizes via a random walk model. This model is used to find a path in a weighted graph based on given probabilities by randomly moving to neighbouring nodes at each step. Because the graph does not need to be optimized globally, this technique is very efficient. To determine the importance of the pixels, the importance map from Wolf *et al.* [99] is used. In the processing step, a scaling function uses the cumulative shrinkability maps to generate the final pixel positions and sizes in the target video. After that, the output pixels are computed by a texture mapping process [98].

In the approach by Kim *et al.* [42], the frames of a video are divided into vertical strips which combine multiple columns with a similar texture. These strips are also connected with the corresponding strips from the previous and following frames, dividing the video cube into several 3D volumes. Each strip is adaptively scaled according to its gradient-based importance measure. To minimize artifacts generated by scaling, distortions are formulated in the frequency domain via the Fourier transform, and are minimized in the retargeting process as a constrained optimization problem. In order to preserve the temporal coherence in video retargeting, object motion is taken into account, and the entire shot is presented as a 3D video cube. Cutting planes are used to describe the strip borders in each frame over the time axis.

Consumer video retargeting is presented by Shi *et al.* [79], where sports videos and advertisements are examined. They begin with a shot classification followed by the extraction of visual concepts. These concepts include play-field, in-field, etc. for sports videos and brand,

Figure 2.19: *(a) Original image with three important objects, the two boys and the ball, (b) and (c) Retargeted images where the important objects were pasted back into the image (from Setlur et al. 2005).*

product image, etc. for advertisement videos. Based on this classification, different visual importance measurements are used to optimize a 3D grid representing the video cube to fit the frames consistently to the target size.

Yen *et al.* [103] propose to retarget videos by a mosaic-guided scaling. First, importance maps are used to create an individual scaling map for each frame. Then, a panoramic mosaic is constructed by aligning all the frames of a shot. A global scaling map is generated by utilizing the information of the individual scaling maps on the mosaic. The final scaling maps for each frame are based on the global map for temporal coherence and then iteratively computed under predefined spatial coherence constraints.

2.2.4 Miscellaneous

This section contains the techniques that do not fit into any of the three categories above. They are techniques that take a different approach to retargeting by exploiting characteristic properties of the underlying images. The important objects in the image are re-arranged in a way that remains mostly unnoticed by the viewer.

"Automatic Image Retargeting"

The idea of the algorithm proposed by Setlur *et al.* [77] is to cut out the Regions of Interest (ROI) and paste them back into a resized and repaired background (see Figure 2.19).

First, the mean-shift algorithm segments the given image into regions [14]. Then, the spatial distribution of color/intensity is used to combine adjacent regions. The importance map consists of saliency [36] and ROIs identified by face detection [74]. If the identified ROIs fit

Figure 2.20: *Left: Original image. Right: Width reduced by half through shift-maps (from Pritch et al. 2009, shortened).*

into the target size, the image is simply cropped. Otherwise, the ROIs are removed from the image and the gaps are filled by inpainting [33].

The repaired background is then scaled to the target size, and the removed objects are pasted back into the image. If necessary, the objects are scaled down in inverse proportion to their importance.

Similar to the approach of Setlur *et al.* [77] for images, Cheng *et al.* [12] extract important objects from a video scene to paste them back into a rescaled and repaired background. The objects are identified by a combination of visual attention features consisting of intensity, color and motion. After the objects are removed, the background is repaired with inpainting [11] and then rescaled to fit the target size. Finally, the extracted objects are reintegrated into the video.

"Shift-Map Image Editing"

The algorithm introduced by Pritch *et al.* [64] optimizes shift-maps to rearrange or remove the pixels in the source image to fit into the target size (see Figure 2.20). A shift-map describes the change in position of the pixels from the input image to the result. Pixels may also be removed entirely. The shift-map is calculated by an optimization via graph cuts.

The algorithm starts by assigning importance values to each pixel. Like the techniques before, the resulting importance map is used to express if a pixel should be kept or discarded. Based on this map, an optimization function is defined which consists of constraints and a smoothness term.

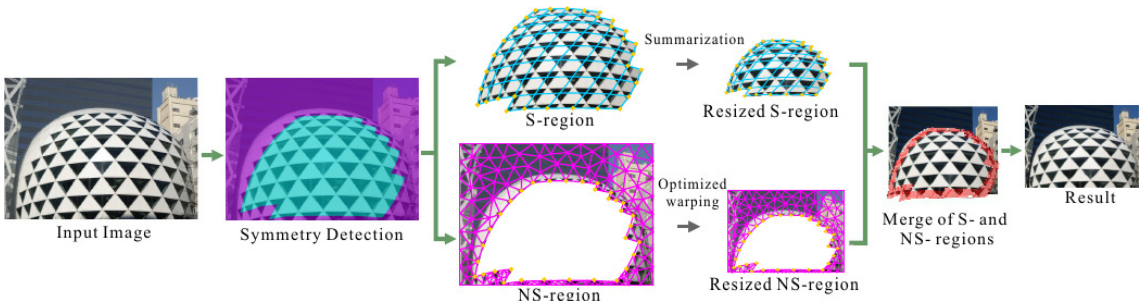


Figure 2.21: *Overview of the symmetry-summarization algorithm (from Wu et al. 2010). The image is analyzed to find symmetry (S-region) and non-symmetry (NS-region) regions. S-regions are resized by the proposed summarization technique while a warping algorithm is used for the other regions. For the final result, the regions are merged back together.*

The constraints include the importance values generated by the saliency map. They can also be used to protect pixels from removal or declare pixels for removal by manual annotations. This is especially important for object removal or image rearrangement. As discontinuities might be created by the removal of pixels, a smoothness term is used to prevent them during the optimization. This term includes color and gradient information in order to create smooth transitions where pixels have been removed.

For the optimization step, the optimal shift-map is described as a graph and solved by the graph- cut algorithm. In order to speed up the computation, the optimization is first calculated on a much lower resolution. The results are then interpolated and used as an initial guess for the computation on the actual resolution.

Resizing by symmetry-summarization

Symmetry-summarization is a novel idea presented by Wu *et al.* [100]. The algorithm divides images into *symmetry* (S) and *non-symmetry* (NS) regions. S-regions are analyzed in order to remove parts of the symmetrical structure – a process which is called *summarization* in the paper. Other regions are resized using a warping operator. In the end, the regions are combined to form the target image (see Figure 2.21).

In the first step, the image is analyzed, and symmetric patterns are detected by using the *maximally stable extremal region* (MSER) [59]. As one can see in Figure 2.21, the searched patterns are regions in the image with content that repeats itself and not symmetry in the mathematical definition, like axial symmetry. This technique finds locations with similar content which can be clustered to form an S-region. The number of these regions found depends on the image content. Other regions are declared as NS-regions.

In S-regions, the symmetric pattern is first described by a 2D lattice. Based on this lattice, a mesh for the region is constructed. Each cell in the mesh represents a recurring part of the pattern, e.g., the windows in a building. For resizing, cells from the middle of a S-region are removed or inserted. This operation is done by using the graphcut algorithm [46]. The changed image region is matched with the rest of the image by linear scaling. Each cell of the S-region is relighted to maintain a continuous lighting distribution between the cells of the mesh.

NS-regions are resized by a warping operator that ensures shape similarity and preserves straight lines. In the last step, the S- and NS-regions are merged back together. An overlapping area of the regions is generated, and a seamless cut path is computed with graphcut [46] in order to hide discontinuity artifacts.

2.2.5 Discussion of Media Retargeting

Retargeting is a popular research topic, and a large number of interesting techniques have been proposed. They all have in common that each method is well-suited for some scenarios while it is not the optimal technique in others. The reason for this can be the quality of the results or the performance of the algorithm in different situations like retargeting videos as a whole or as a stream in real time.

In the case of quality, each method has inherent shortcomings as retargeting cannot be done without a loss of information [27]. For instance, if cropping is used, parts of the source image must be completely discarded. When warping is used instead, the information of the pixels with low importance is spread over the other pixels, but objects or parts of the background may get squeezed.

In order to overcome some of the individual limitations of the algorithms, many authors use a combination of multiple operators (e.g., Rubinstein et. al. [70]). One example for a clear limitation is the disability to enlarge an image or video with cropping. A limitation specific to Seam Carving is the observation that in most cases Seam Carving produces artifacts when it is used to reduce the width to below one half of the original size (depends on the image content).

However, there are still limitations that nearly all techniques – multi-operator or not – share. Objects including structured textures or straight lines are difficult to preserve [96, 24, 50]. They can often be found in the background on buildings or as markings on a street. This is why some papers specialize on their preservation in particular (e.g., the symmetry summarization by Wu et al. [100]). Another typical problem for retargeting are very complex scenes with a large amount of objects and few homogeneous areas [95, 103].

In our opinion, an image or video should be scaled uniformly first, as the retargeting operators seem to be more fit to adjust the aspect ratio than to resize a frame. A good example is a widescreen movie with a high resolution that should be adapted to a smartphone display by using Seam Carving. The algorithm would have to remove a lot of seams to reach the target resolution, most probably introducing heavy artifacts. The main focus for the preservation should lie on faces and whole persons if there are any, as they usually draw the most attention of the viewer. However, not all faces are equally important, for instance a face of a basketball player compared to a face of the crowd behind him (as mentioned in [37]). In videos, motion is really important, and an object that moves on the screen is usually interesting to a viewer. The benefit of such objects is that they are able to mask distortions in homogeneous areas or regions that are not so visually dominant. This principle is used by the warping techniques that do not discard information like other operators.

From a technical perspective, image retargeting and video retargeting should be distinguished [95]. There are different core considerations in video retargeting, namely motion and temporal dynamics, which prevent most image algorithms to be simply expanded with just temporal constraints. The authors of the SeamCrop approach [40, 39] came to similar conclusions, where the inner workings of the algorithm had to be changed in order for it to work properly on videos. Interestingly, the other way around seems to work quite well: The video algorithm by Krähenbühl et al. [45] was one of the most preferred ones in the comparative image retargeting study by Rubinstein et al. [69].

In terms of performance of the algorithms, a trade-off must be made between the quality of the results and the computation time that is needed [87]. For instance, an optimization over an entire video sequence results in a highly complex problem with at least three dimensions (width, height and time). One way to reduce complexity is to use a heuristic and accept the fact that the result is not globally optimal. An example for this is the discontinuous Seam Carving approach by Grundmann et al. [24]. Another possibility is to look for ways of parallelizing individual steps of an algorithm and to use the GPU for faster processing (e.g., Chiang et al. [13]).

CHAPTER 3

Seam Carving

In this Chapter, the seam carving algorithm is enhanced for different use cases, and the first three retargeting questions posed in Chapter 1.2 are answered. First, errors occuring in images with straight lines or regular patterns are solved. Then, we extend seam carving for a fast retargeting of videos. Lastly, a novel version of the technique is described which is developed for the retargeting of stereoscopic videos. To our knowledge, there is no other algorithm capable of retargeting this kind of videos. The work described in this Chapter was published in [41, 44, 28].

3.1 Seam Carving for Images

There are regular patterns or straight lines in images that may cause errors if processed with traditional seam carving. During the retargeting, these lines may become curved or disconnected. Figure 3.1 shows an example of an image with straight lines which has been adapted through classical seam carving. The curved lines on the windmill vane are clearly visible. These distortions may occur if a seam crosses a non-vertical or non-horizontal line (see Figure 3.2).

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 several seams are crossing a straight line in adjacent intersection points and this is repeated, a distortion becomes visible when the pixels are moved left (see Figure 3.2 c)).

Figure 3.1: *Left side: original image. Right side: image adapted to 75% of the original width using seam carving (forward energy used). Noticeable curved lines are marked with red rectangles.*

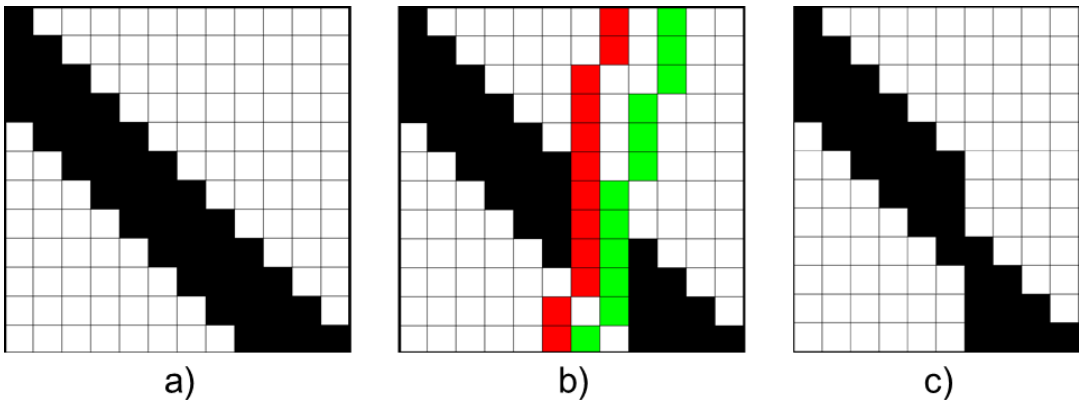


Figure 3.2: *a) A straight line. b) Two seams crossing the line. c) An obvious distortion occurs by removing adjacent seams.*

3.1.1 Workflow of our 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. By using a different energy map, we rather change some optimal seams so that the visible effect of such errors is reduced, and the viewer does not notice the distortion. It should be guaranteed that 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 not so obvious to the viewer.

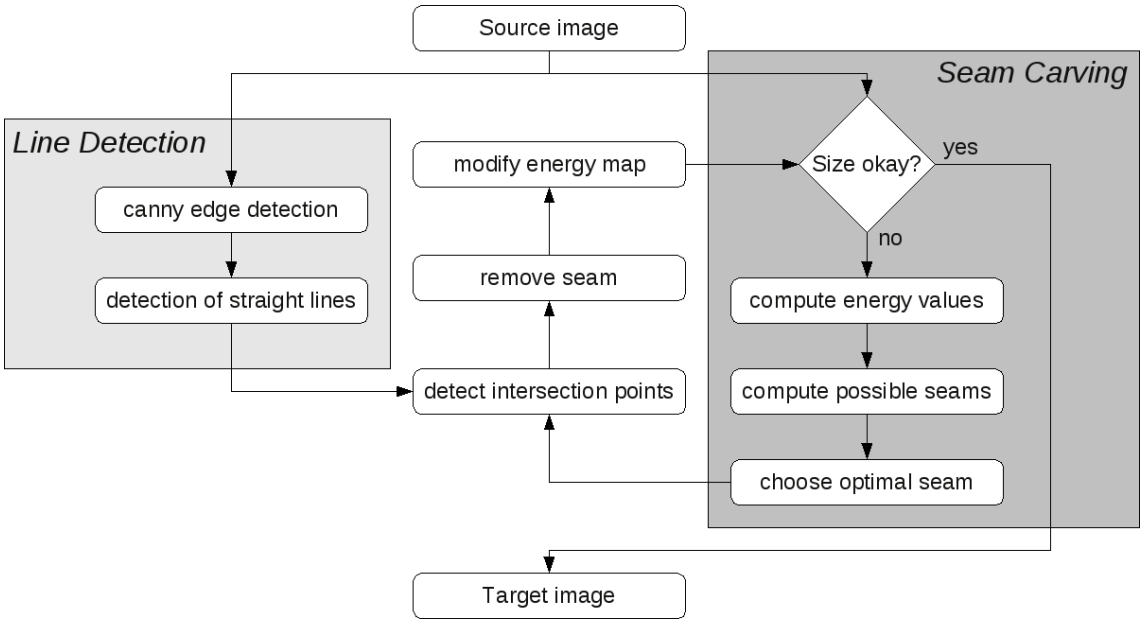


Figure 3.3: *Overview of the workflow of the algorithm*

We apply an automatic line detection algorithm to identify straight lines in images. Each time a seam crosses a line, the energy in a certain radius of the intersection point is increased in order to reduce the probability that other seams cross the line in adjacent pixel positions.

Our algorithm is based on the regular seam carving algorithm, and therefore most of the basic steps are similar. In addition, we use the information about the most relevant straight lines. We apply the *Canny edge detector* [7] to identify significant edges. Afterwards, the edge pixels are transformed into *Hough space* [18] to locate pixels on straight lines. Edge pixels that are located on straight lines will be called *line pixels* in the following.

The only enhancement required for seam carving is to change 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. Figure 3.3 gives an overview of the different steps of the algorithm. The details of the line detection algorithm and the improvements of seam carving are presented in the following sections.

3.1.2 Detection of Straight Lines

Edges in the image are detected based on the *Canny edge detector*. As the parameters of Canny, we use a Gaussian mask of size 3 for noise reduction, and $T_{up} = 100$ and $T_{low} = 20$

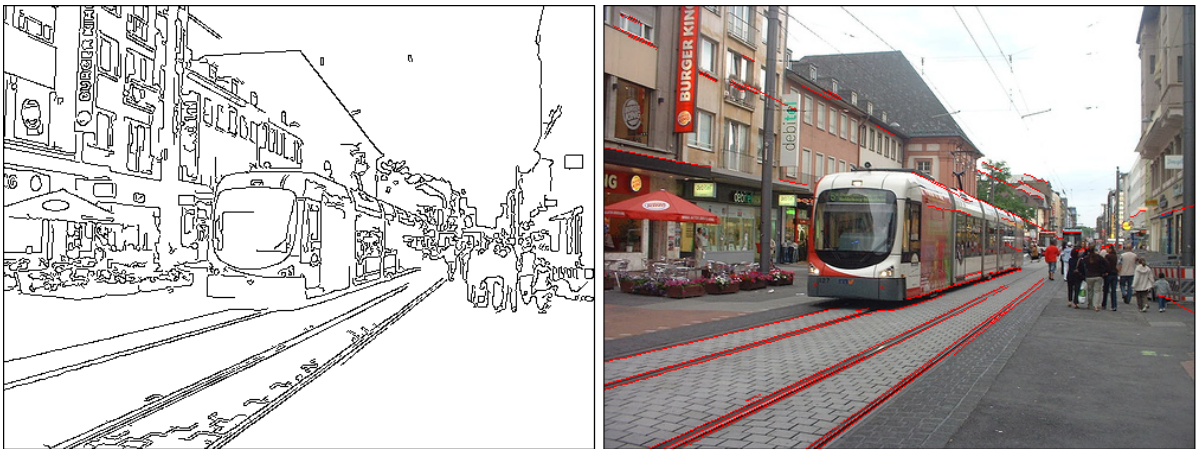


Figure 3.4: *Edge image and detected lines (red)*

as upper and lower thresholds for the hysteresis. These values, as well as the ones used as parameters in the follwing sections, have yielded the best results in our experiments.

Edge pixels are transformed into *Hough space* I_H next. Each point in Hough space corresponds to a straight line in the edge image. A threshold $T_{hough} = 0.6 \cdot \max\{I_H\}$ is derived from the maximum value in Hough space. Only the most significant straight lines are selected by considering Hough pixels that exceed this threshold.

For each line candidate, the number of edge pixels located on this line is counted. An edge pixel is considered as line pixel if the distance between edge pixel and line is below a threshold $T_{dist} = 0.5$ pixels, and if the line segment has a length of at least $T_{length} = 10$ pixels. Small gaps between valid line segments are filled up ($T_{gap} = 30$). Because the precision 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. Figure 3.4 shows an example of an edge image and the straight lines that are detected automatically.

3.1.3 Modification of the Energy Map

Our seam carving algorithm starts by computing an energy value for each pixel based on the gradient. We also use *forward energy* as proposed by Rubinstein [68]. An additional energy map that adds an offset value to each gradient is initialized with zero. All seams are calculated based on 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. By calculating the next optimal seam, the values of the offset energy map are added to the gra-

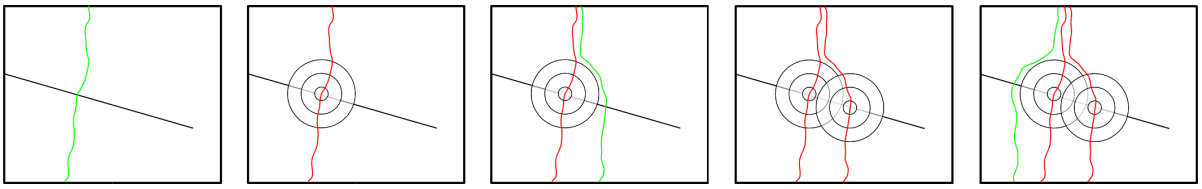


Figure 3.5: *Visualization of the basic idea of the modification of the energy map: the optimal seam (green) of one iteration may cross a straight line (black). After the removal of the seam (the removed seam is marked in red), the energy in the local neighborhood of the intersection point is increased. This decreases the probability that an optimal seam crosses these areas in the next iterations.*

dient values. This guarantees that seams avoid intersection points in the following iterations. Figure 3.5 visualizes the modification of the energy map.

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 (these parameters were estimated empirically). After the modification of the offset energy map, the pixels of the optimal seam are removed from the image and the offset energy map. The algorithm stops after a sufficient number of seams have been removed to reach the target image size.

If both dimensions of an image are to be changed, it is necessary to build the optimal order of vertical and horizontal seams. Therefore, the minimum energy of vertical and horizontal seams is compared, and the lower one is chosen. This is repeated, and an optimal order is built.

3.1.4 Evaluation

In the following, we evaluate our enhanced seam carving algorithm and compare the quality of the adapted images to the quality of regular seam carving based on forward energy. In case of landscape images where no straight lines are detected, traditional seam carving and our enhanced approach lead to identical results. However, when the image contains objects with straight borders, the quality of the adapted image is significantly better with our algorithm. Figure 3.6 compares the original seam carving with our enhanced approach. Most critical for seam carving are diagonal lines. The red circles mark distorted lines in the adapted images of regular seam carving.

The first two image rows contain images which show diagonal straight lines reaching from one side of the image to the other. This kind of images is problematic for seam carving because it can not avoid to cross these lines. The bridge in the first row becomes curved because seam

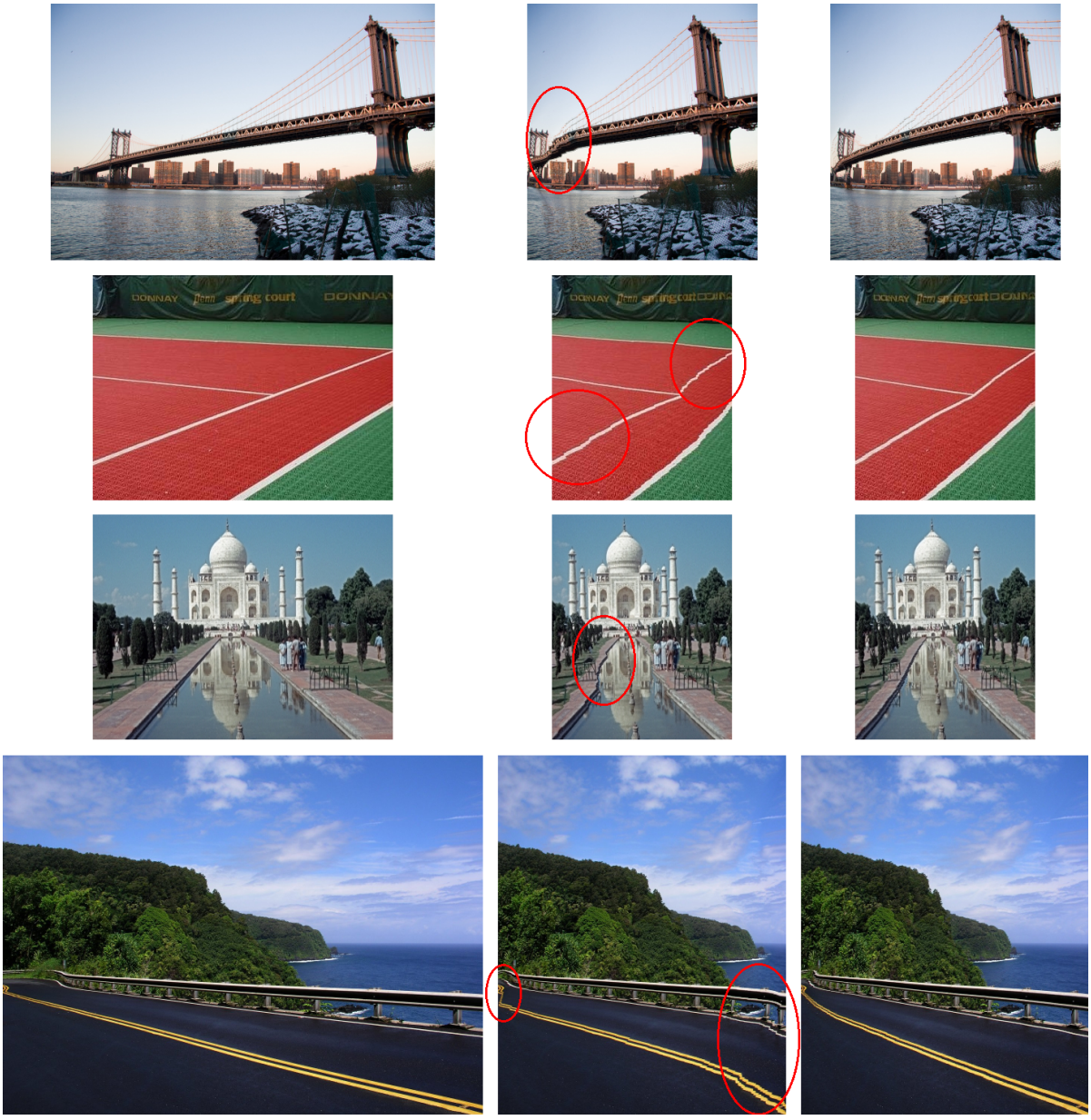


Figure 3.6: *Left: Original. Center: Seam carving with forward energy. Right: Our result. All the images have been reduced to 60% of their original width. The red circles mark distorted lines in the seam carving results.*

Figure 3.7: *Limitations of the algorithm*

carving accumulates the seams on the left side of the image due to large areas of water and sky. The algorithm does not take lines into account and removes these regions. The result of seam carving in the second row shows blurred and bent lines in most parts of the image. Our method distributes the seams evenly across the lines and reduces these errors. Although straight lines also become slightly curved in some cases, the visual distortion caused by the enhanced seam carving is less obvious.

Many objects like trees or people are depicted in the image of the third row, which makes it difficult to remove seam pixels and maintain these objects at the same time. Although the most relevant straight lines do not reach over the entire image, they are located in an area with less relevant content and thus become blurred.

The image in the last row is similar to the first two images based on straight lines spanning the image but depicts a landscape with no structured background. The adapted image based on classical seam carving has blurred and broken lines on the left and the right side, while the other regions of the image show no distortions. Again, our method overcomes these distortions and achieves a higher visual quality.

An example of the **limitations** of our approach is shown in Figure 3.7. The pedestrian underpass contains many straight lines which cover all regions of the image. The algorithm can only preserve straight lines if there is sufficient space to move the seams. In this example, the lines are too close to each other and the angles between the lines differ, so it is not possible to preserve one line without distorting another. If a large number of straight lines or straight structures are contained in an image, the algorithm may not be able to prevent them from bending or getting distorted.

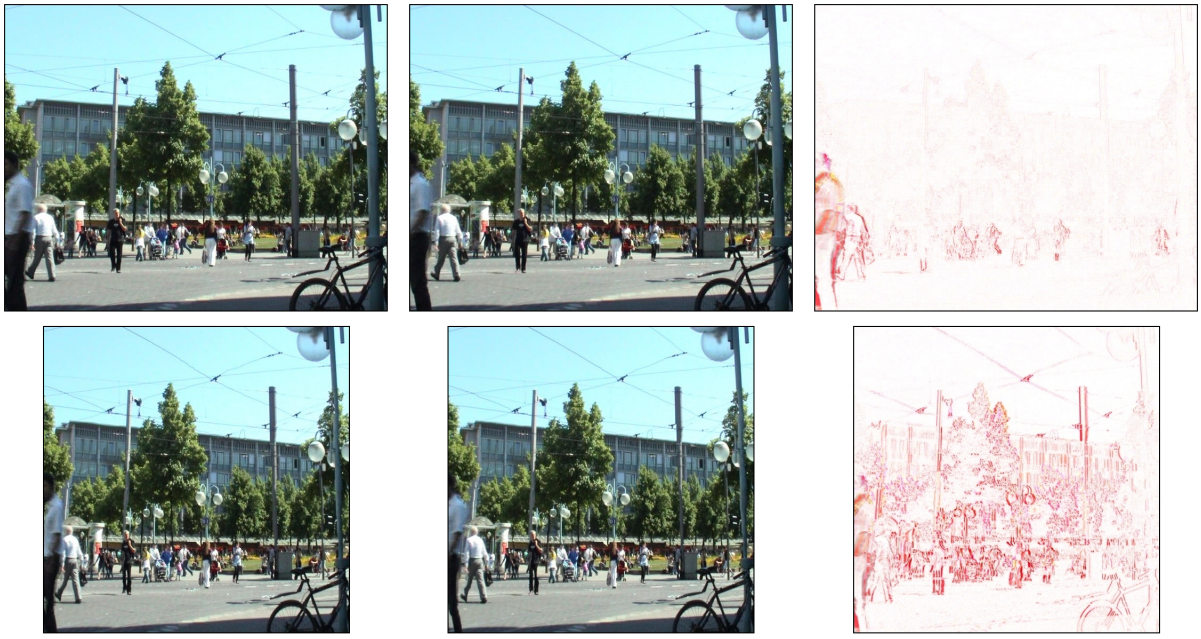


Figure 3.8: *Top: Two original video frames and difference image. Bottom: Seam carving applied on individual frames and difference image. The image difference of the adapted video frames is significantly higher and causes a shaky video.*

3.2 Efficient Seam Carving for Videos

After we have enhanced the seam carving algorithm for images, we want to extend the technique for the use in videos. As Rubinstein et al. presented in [68], the quality of adapted videos is very low if seam carving is used on each frame separately because this leads to visible artifacts, and the video becomes blurred and shaky. The errors 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 these changes. Even with a static camera, small differences in pixel values lead to different seams.

The absolute difference of pixel values in the top right image of Figure 3.8 visualizes minor differences between two adjacent frames of the original video. The differences increase significantly after seam carving is applied on individual frames (Figure 3.8, lower right). In this example, background objects like buildings or pillars are constantly changing their proportions and positions. These observations underline the importance of processing the sequence as a whole and not each frame separately.

Our *FSCAV*[1] algorithm operates on the level of shots (continuous camera recordings). Even if we adapt full-length videos, the algorithm considers all shots separately. We put the focus on the following requirements for the development of the FSCAV algorithm:

1. An optimal seam should be *robust*, that is to say the removal of this seam should not cause a blurred or shaky video.

2. To limit the computational effort and memory consumption, 1D seams should be calculated in images instead of 2D seam manifolds in 3D space-time volumes.

We assume in the first part of this section that we want to adapt a video without object motion, and changes are only caused by camera motion. The first of the above mentioned requirements is valid if a pixel of 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. The idea of FSCAV is to identify robust seams by estimating and compensating the camera motion between all frames.

The analysis of the camera motion also makes it possible to maintain the second requirement: instead of searching 2D seam manifolds in a 3D cube, we analyze and compensate the camera motion between consecutive frames. This enables the aggregation of pixel values, respectively 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 the implementation of the FSCAV algorithm in the next sections and consider the following challenges:

1. How should the algorithm avoid that an optimal seam removes too many pixels from moving foreground objects?

2. How can we handle robust seams that are 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 reducing the visual quality too much?

[1] Fast Seam Carving for the Adaptation of Videos

3.2.1 Camera Motion Compensation

Image registration techniques can be used to track the image content in consecutive frames. We use the projective camera model [19] which uses eight parameters to describe the motion of the camera in consecutive frames:

$$x' = \frac{a_{11}x + a_{12}y + t_x}{p_x x + p_y y + 1}, \quad y' = \frac{a_{21}x + a_{22}y + t_y}{p_x x + p_y y + 1}. \tag{3.1}$$

Six parameters (a_{ij}, t_x, t_y) specify affine motion and two parameters (p_x, p_y) a change of the perspective. To identify the parameters of the model, point correspondences between two frames have to be identified first. A large number of techniques have been proposed to identify characteristic feature points in images like the Moravec detector [60], corner detectors like Harris [32] or SUSAN [80], or detectors that are invariant to image transformations like SIFT [55]. Frame rates of at least 25 fps are typical in videos that were produced for cinema or television, and we do not require invariant features to track the correspondences because the camera motion between consecutive frames is relatively small. We have selected the Harris detector with sub-pixel refinement for our purpose due to its better repeatability and accuracy [19].

In a second step, correspondences between features have to be identified. The features of two frames are considered only if the spatial distance between two feature points is below a predefined threshold T_S. We set T_S to 30 in our experiments. The visual distance between two arbitrary feature points is defined as the sum of absolute differences in a small local window of 8×8 pixels. A greedy-based approach selects corresponding feature points by choosing the two features with the smallest visual distance. The algorithm terminates if the distance exceeds a threshold ($T_D = 4000$).

We use a robust algorithm for motion estimation due to the wrong assignment of some feature points (outliers). One of the most popular techniques is the RANSAC algorithm [21]. A subset of four corresponding features is randomly drawn, and the parameters of the camera model are calculated based on these features. We classify the number of inliers (features that fit to the model) and outliers and keep the parameters with the largest number of inliers. By repeating this step (we use 300 iterations), the probability that the features chosen describe the camera motion correctly is very high.

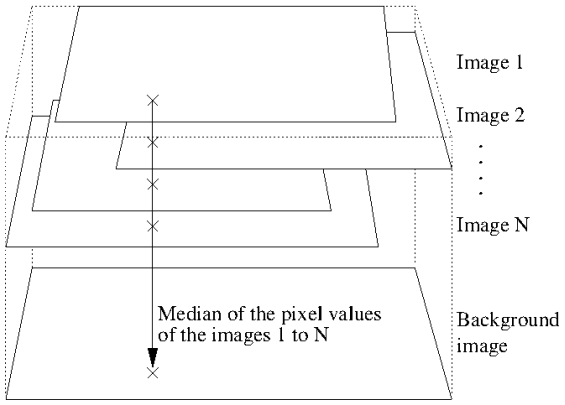


Figure 3.9: *The background image is constructed by applying the median filter on the aligned frames.*

3.2.2 Aggregation of Frames

In the next step an *energy map* based on all motion compensated frames is calculated. The idea is to aggregate the frames into one (possibly much larger) *background image* [20]. The center frame of a shot is selected as a reference frame, and all other frames are aligned to this frame. 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 at one position. Figure 3.9 visualizes the construction of the background image. The background in videos is preserved well if an optimal seam is detected in the background image and transformed back into all frames of the shot.

On the other hand, this approach does not consider foreground objects at all. To reduce deformations in moving objects, we identify foreground objects by comparing each aligned frame with the background image. A pixel is characterized as an object pixel if the absolute difference of aligned pixels exceeds a threshold ($T_A = 30$). 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. Figure 3.10 shows an example of a background image and an image with foreground objects.

3.2.3 Identification of Robust Seams

To identify *robust seams*, we apply the seam carving algorithm to the energy map of the background image and get 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 may cause visible artifacts in the adapted video.

Figure 3.10: *Top: Three sample frames of a video sequence with horizontal pan. Bottom: Background image based on the median (left) and foreground objects (right).*

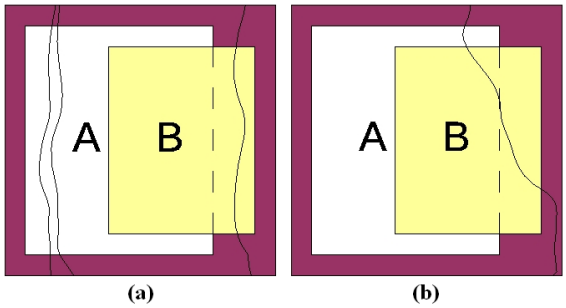


Figure 3.11: *Example of missing seams: The seams are located in frame A or B, but not in both frames (a). The vertical seam does not pass through frame A or B from top to bottom (b).*

The characteristics *completeness* and *visibility* of seams are introduced in the following, to validate whether a mapped seam is suitable or not. A seam is *visible* if it is included in all frames of the shot. For example, in case of a camera pan, the seams at the borders are not visible in most cases. A seam is *complete* if a pixel is assigned in each row (vertical seam) or in each column (horizontal seam). This corresponds with the first constraint of the definition of a seam. We classify a seam as *robust seam* if it is *visible* and *complete* in all frames. In the following, we analyze how to process seams that are not robust:

- *Missing seams:* in case of camera motion like a pan or tilt, it may happen that the image content covered by a seam is not included in all frames (these seams are called *missing seams*). Figure 3.11 (left) shows an example of missing seams, where two seams are only mapped to frame A and another one to frame B. In Figure 3.11 (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. Missing seams are ignored to avoid shaky videos.

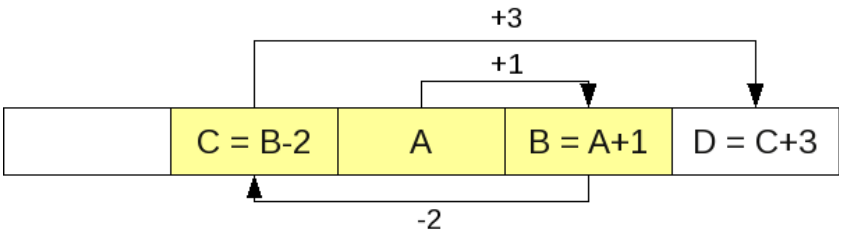


Figure 3.12: *Example of the search for unoccupied pixels. If the position of a robust seam pixel is already occupied (yellow pixels) by another seam, a new and unused position for this pixel has to be searched. This is done by alternately looking at the positions to the right and to the left until a unoccupied position is found.*

- *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 background image. These gaps appear periodically and cause frayed edges in the adapted frames. The gaps are filled by interpolating adjacent seam pixels.

- *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. Figure 3.12 visualizes our approach to detect the next unoccupied pixel position. Because pixel A is already occupied, the adjacent pixel B and C 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.

- *Fast camera motion:* some camera operations like long camera pans are a great challenge for this algorithm. No robust seams are detected 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 (this depends on the desired size 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 dissolve 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.

3.2.4 Quality Measurements for Video Adaptation

In the following, we present a heuristic to estimate the degradation of the quality of an adapted video. We analyze how much the seam to be removed next reduces the image quality. Although shaky videos were not observed, other noticable image errors may occur by deformations of foreground objects. The seam carving algorithm stops if the heuristic indicates that the quality drops below a certain level. In this case, no more seams are removed from the shot; another adaptation technique is chosen to adapt the video to its final size (we use scaling).

In a first step, the parameters of the camera model are analyzed to guarantee a correct background image. Errors occur in case of large foreground objects or an insufficient number of characteristic features in the image background. An error is detected if a parameter is not within a characteristic interval or if a parameter considerably changes between adjacent frames. The binary variable $C_{i,i+1}$ defines whether the parameters of the camera model $(t_x, t_y, a_{i,j}, p_x, p_y)$ between frames i and $i+1$ are correct or not:

$$C_{i,i+1} = \begin{cases} 1 \text{ if } & |\frac{t_x}{W}|, |\frac{t_y}{H}| \leq 0.05 \ \wedge \\ & \text{rotation angle} \leq 5 \text{ degrees} \ \wedge \\ & \text{scaling factor} \leq 4 \text{ percent} \ \wedge \\ & |p_x|, |p_y| \leq 10^{-5} \\ 0 & \text{else.} \end{cases} \tag{3.2}$$

The thresholds were estimated empirically, the parameters W and H define the width and height of a frame. We switch to the alternative retargeting technique if the parameters of at least one pair of adjacent frames are invalid.

In a second step, we analyze the cost of the next seam to be deleted. We assume that pixels from relevant objects are deleted, and errors become obvious in case of seams with high cost values. No more seams are deleted if the cost $E(s_i^*)$ of the optimal seam in iteration i exceed a threshold T_S. After the removal of the first i seams ($E(s_i^*) \leq T_S < E(s_{i+1}^*)$), the image is scaled to the final resolution. The threshold T_S is defined based on the costs of the seams in the original image:

$$T_S = \alpha \cdot E(s_1^*) + (1 - \alpha) \cdot E(s_1^{MAX}) \tag{3.3}$$

$E(s_1^*)$ specifies the cost of the optimal seam in iteration 1, $E(s_1^{MAX})$ the maximum cost of a seam in this iteration. The costs are weighted by a parameter $\alpha = 0.3$.

3.2.5 Evaluation

Fast adaptation techniques

In a first step, we compare the quality of the adapted videos based on scaling, cropping and *FSCAV*. To this end, 45 video sequences (shots) have been selected from television, the Internet or recorded with a HD camcorder. We consider individual shots because seam carving operates on the level of single shots. The resolution of the sequences varies between PAL resolution (720×576 pixel, 25 fps) and HD resolution (1920×1080 pixel, 25 fps).

Due to the fact that *FSCAV* is especially sensitive to object motion, the video sequences have been grouped into five categories: *static* (no camera motion, average number of object pixels is less than 1 percent of all pixels), *camera motion only*, *small object motion* (1-10 percent object pixels), *high object motion* (>10 percent), and *large objects* (occupy at least 50 percent of the height or width of an image). Camera motion is usually visible in all videos except static sequences. The following Table 3.1 gives an overview of the sequences in each category.

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

Table 3.1: *Categorization of the sequences that were used during the evaluation.*

One video of each category was selected for the evaluation. Ten students between 21-24 years old evaluated the test series by watching the original video sequence first and the three adapted versions in the following. 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 in the case of PAL resolution). In the case of cropping, we had to set the borders manually to get acceptable results. The quality of cropping is completely unacceptable without this manual setting. The subjects filled out a questionnaire and answered the following questions for the test series: How well are details preserved? What 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 1 (excellent) to 5 (insufficient), and additional comments were collected. This scale corresponds to the german school grading system. Another task was to sort the adapted sequences of each test series by visual quality.

Cropping always leads to a loss of relevant content in the adapted video. It achieved the worst results in the evaluation (4, *sufficient*). Ranking the quality of *scaling* and *FSCAV* is not so easy: although the average quality of *FSCAV* is better than scaling (between 3, good and 2, very good) the evaluations of the different sequences differ a lot. *FSCAV* is significantly better when object and camera motion are relatively low (*static*, *camera motion only* and *small object motion* sequences) and ranges from 1 (excellent) to 2 (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 if the camera or the objects move in parallel to the seams (e.g., vertical motion does not degrade the image quality so much when the width of a video is reduced). On the other hand, the visual quality drops significantly if objects move orthogonally to the direction of a seam. The best case occurs when the seams are uniformly distributed, causing an effect similar to scaling (see Figure 3.13 (d)). If many seams are located in a small area, a moving object is significantly distorted in this region. This is very annoying, it reduces the average visual quality to 4 (sufficient). Most problematic are sequences with large objects where the object covers a major part of a frame. The quality is insufficient in these cases, and the test persons ranked the sequence even below the cropped videos.

Figure 3.13 shows sample video frames from each category and their adapted versions. In case of the static sequence (a), *FSCAV* preserves the building and trees much better compared to the scaled frame. In (b), the visual quality of the adapted videos are very similar. The quality of sequence (c) based on *FSCAV* is rated significantly higher, because parts of both persons are missing in the cropped video, and the faces are heavily deformed in case of scaling. Many fast moving objects (cars) cross the image in sequence (d). The differences of scaling and seam carving are low if we compare the cars in the foreground. Major differences can only be recognized in the building. The quality of the scaled video is much higher in sequence (e). The tram crosses the full image and is heavily deformed by the removal of seams. Another general problem of seam carving is also recognizable at the overhead contact lines of the tram: Straight lines in the original image become curved. This is very disturbing compared to scaling.

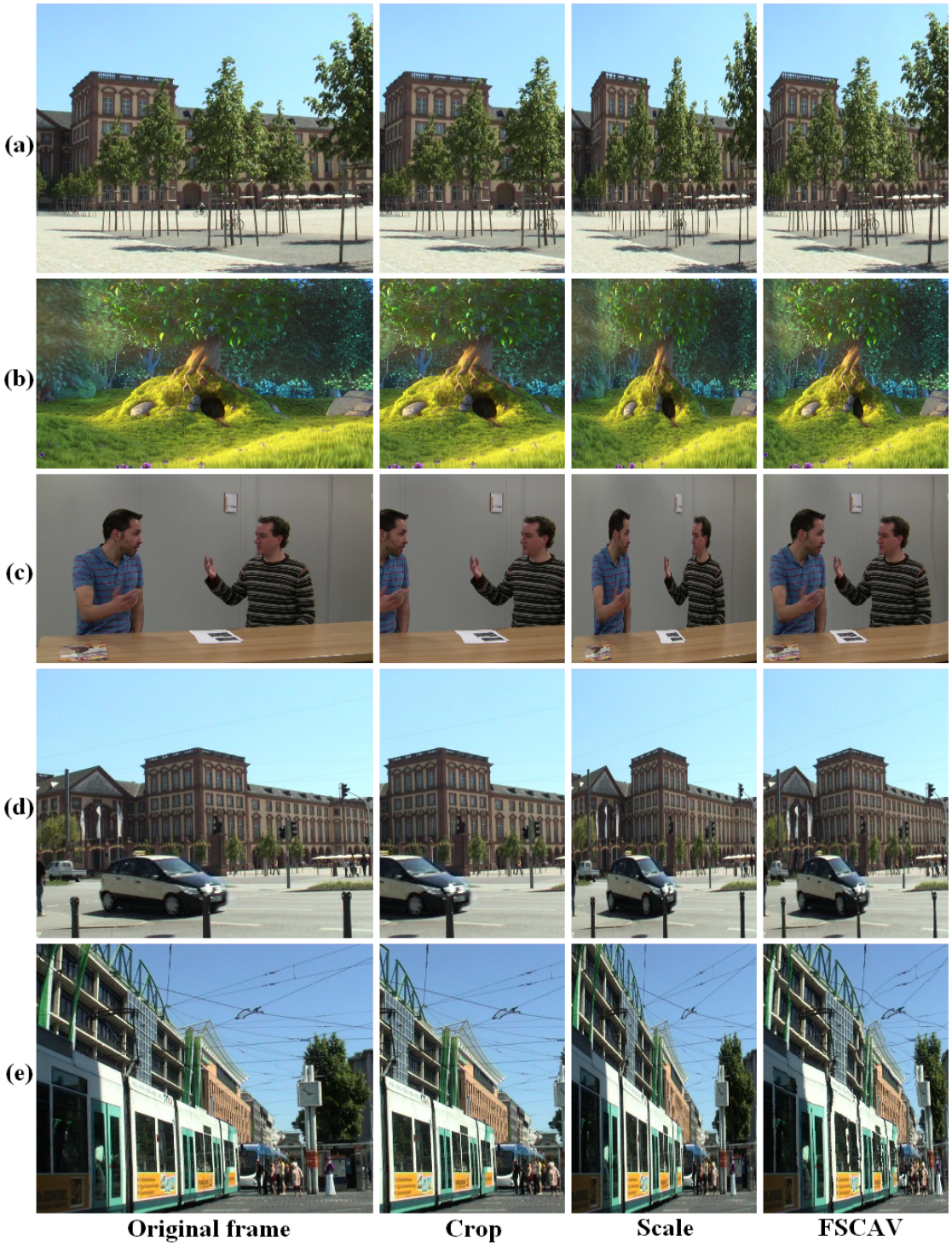


Figure 3.13: *Top to bottom: Sample sequences from the five categories: (a) static, (b) camera motion only, (c) small object motion, (d) high object motion, and (e) large objects. The width of each video is reduced by 45 percent.*

Video adaptation based on seam carving

The second part of the evaluation compares two seam-carving-based adaptation techniques (*graph cut* [68] and *FSCAV*). The visual differences of adapted videos based on these approaches are very low. In most cases, it is not possible to recognize differences if only one frame is observed (this is also true for all video sequences of Figure 3.13). In videos without object motion (*static* or *camera motion*) the quality of *FSCAV* adapted videos is slightly better. The reason is that the seams calculated by graph cut change from frame to frame and introduce a small amount of shakiness, which is especially obvious in background objects.

The graph cut technique generates videos of higher quality if small objects move in parallel to the direction of the seams (seams usually avoid the foreground objects). In case of *fast moving objects* or *very large objects*, the visual quality of the adapted videos based on both seam carving techniques is much lower. Due to the fact that 2D seam manifolds are connected in the temporal direction, the position of a seam pixel may only change one pixel position in adjacent frames. Even slow-moving objects that move orthogonally to the seams will cross them and cause visual errors that are comparable to *FSCAV*.

Another aspect is the *computational effort* and *working memory* required by both algorithms. *FSCAV* is separated into an *analysis* and an *adaptation* phase. As the 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 very efficient and can be handled nearly in real-time on a standard personal computer if the decoding and re-encoding of the adapted video stream is not considered.

The memory requirements of *FSCAV* are defined by the maximum number of frames which are loaded into memory at one time. Sequences in PAL resolution are processed entirely in memory. If sequences in HD resolution are processed, repeated decoding of a video sequence might be necessary to limit the total amount of memory. In this case, the memory requirements are less than 200 MB, but runtime 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 Kolmogorov [6]. The following Table 3.2 lists the measurements of the runtime and the memory requirements

on a standard personal computer (Athlon 64 Dual Core, 2.4 GHz, 2 GB RAM). In case of the PAL and HD sequences, it was not possible to run 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 requirements and computational effort of the *FSCAV* algorithm are much lower.

	Low res. 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 - Analysis - Adaptation	 14 s 1 s	 8 min 11 s	 51 min 83 s
Graph Cuts	17 min 290 MB	N/A (44 GB)	N/A (292 GB)
Graph Cuts (hierarchical)	N/A N/A	49 min 530 MB	123 min 820 MB

Table 3.2: *Performance of FSCAV compared with other retargeting methods (Cropping, Scaling and Seam Carving based on Graph Cuts).*

A hierarchical approximation was proposed by Rubinstein et al. [68] to reduce the memory requirements. We implemented a hierarchical graph cut algorithm which reduces the resolution of a video to the low-resolution video defined above. In case of HD videos, the spatial and temporal resolutions 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 two iterations. The major disadvantage of using a hierarchical approximation is the fact that a seam may be trapped in a local minimum; detection of the optimal cut is no longer guaranteed.

Limitations

Calculation of the camera parameters fails in some shots, especially in the cases of 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. The quality of scaled videos is much better if 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 increase the perceptible errors significantly like straight lines which become curved. Another disadvantage in comparison to scaling or cropping is the computational effort. Despite these limitations, the visual quality of adapted videos based on *FSCAV* is significantly higher in most cases.

3.3 Seam Carving for Stereoscopic Videos

The popularity of stereoscopic videos is rapidly increasing, mainly due to the use in mainstream cinema movies. Also, more and more devices are able to produce or play these kinds of videos. Therefore, we want to be able to adapt their sizes with seam carving as well. In this Section, we introduce a seam carving algorithm for the retargeting of stereoscopic videos. To our knowledge, there is currently no other technique capable of this task.

In comparison with FSCAV, we changed the inner workings of the video adaptation. Instead of aligning all the frames of a shot and calculate the optimal seams on it, our new algorithm works on a frame-by-frame basis as this yields better results in sequences with large pans. Remember that FSCAV can not deal well with long-distance panning operations because they imply that there is screen content that is not shared by all frames. Additionally, stereoscopic videos introduce new challenges that have to be taken into consideration, like the need to synchronize the left and the right view of a frame. For instance, the seams found in the left and right background images would need to be synchronized between the views and also when transferred back to the individual frames through the camera model. This would leave a lot of room for inaccuracies in the computation step.

The input to our algorithm is a video sequence consisting of left frames $I_t^L(x, y)$ and right frames $I_t^R(x, y)$. Since most of the processing is done on one frame at a time, the frame index t is dropped in the following unless needed for clarification. Each frame of the input sequence is of size $w \times h$. We retarget the video by removing vertical seams to reduce the width of each frame. In this work, a bar over a mathematical symbol is used to denote that it is a result after removing one or more seams. The output of our algorithm is a video sequence of left and right frames $\bar{I}_t^L(x, y)$ and $\bar{I}_t^R(x, y)$ with reduced width. Their size is now $\bar{w} \times h$. This is done by removing one seam after another. Our description is thus limited to removing one seam at a time. In order to reduce the image width from w to $\bar{w}$, this process is iterated $w - \bar{w}$ times. Each pair of left and right frames of the video sequence is processed individually. The only

exception is that seams are carried over from the previous frame in order to achieve temporal consistency. This is described in detail later.

A frame is retargeted in a number of steps. The process starts by computing a disparity map between I^L and I^R to establish pixel correspondence among the views. A disparity map is a mapping between the pixels of the left view and the right view. For each pixel position (x, y) in the left view, the disparity value $D(x, y)$ states by how many pixels it is shifted to the left in the right view. As such, the disparity map establishes a correspondence between left and right pixels:

$$I^L(x, y) \approx I^R(x - D(x, y), y) \tag{3.4}$$

In our implementation, disparity values range between 0 and 16 (the unit is pixel width). Higher values mean that the pixel is closer to the camera; far away objects have roughly the same position in both images. The disparity for far-away pixels is thus close to zero. We use semi-global block matching to compute the disparity map [34]. A disparity map needs to be computed only once for the frame pair. All other steps of the retargeting are repeated for each of the seams.

An energy function is computed for the current frame that incorporates knowledge from both views at once. The energy value of a pixel represents its importance in the image; low energy pixels are removed first. A pixel's energy value depends on a large number of factors, including local contrast, depth and its location with respect to seams in the previous frames. The energy values are accumulated row by row to calculate an accumulated energy map. Based on this map, seams of pixels with low energy are detected and removed from the two views. In the last step, the seam is also removed from the disparity map, and disparity values are updated. The entire process is then repeated until the target width is reached.

Our approach is focused on finding and removing vertical seams in a stereo pair. Unless explicitly noted otherwise, we are always referring to seams in the *current* frame. A vertical seam consists of exactly one x coordinate for each row in an image. It is a function of y. Removing a seam means deleting the seam pixel in each row and shifting all pixels to the right of the seam left by one. This reduces the width of the image by one, as illustrated in Figure 3.14. More formally, the i–th detected vertical seam $S_i(y), i = 1, \ldots, w - \bar{w}$ in a frame is a function mapping each row index y to an x coordinate between 0 and $w - i$. The removal of seam S_i reduces the width of the frame from $w - i + 1$ to $w - i$. We distinguish between

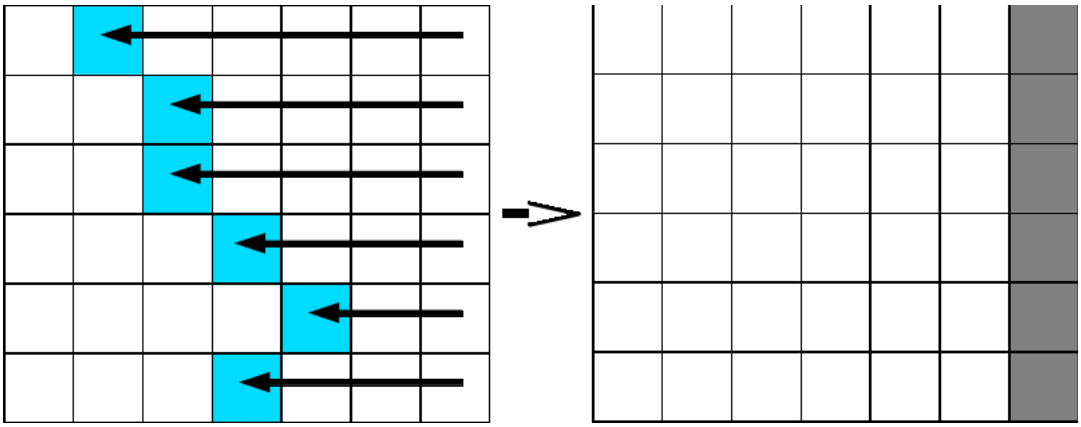


Figure 3.14: *The blue squares are pixels that belong to a detected vertical seam. Removing the seam pixels from the image shifts the entire remainder of the row left by one pixel. This reduces the width of the image by one (see grey pixels).*

seams in the left and the right view by using the superscripts L and R. The pair of seams is connected by the disparity map:

$$S_i^R(y) = S_i^L(y) - D(S_i^L(y), y) \tag{3.5}$$

3.3.1 Energy Function

The energy value of a pixel denotes its importance in the image. It is determined from a large number of factors which are outlined in this section. Some components of a pixel's energy not only depend on the pixel itself, but also on seam pixels in the row above. Because of this dependency, it is not efficient to precompute and store energy values. They are instead represented as an energy function which is evaluated as needed. In our approach, the energy function is composed of appearance energy E_{app}, disparity energy E_{3D}, and temporal energy E_{temp}. Appearance energy measures edges in the intensity image that are introduced when removing a pixel. Disparity energy takes into account the removal of seams in the disparity map similar to appearance energy, as well as the depth of a pixel. Note that disparity and depth are not the same, as disparity describes the the change in position of a pixel between the left and the right view, while depth measures the distance of a pixel to the camera. Temporal energy helps to achieve temporal consistency by giving a higher energy to pixels that are far away from the seams of the previous frame. These three components are summed up to a total energy E:

$$E(x, y, \hat{x}) = \alpha_1 E_{app}(x, y, \hat{x}) + \alpha_2 E_{3D}(x, y, \hat{x}) + \alpha_3 E_{temp}(x, y) \tag{3.6}$$

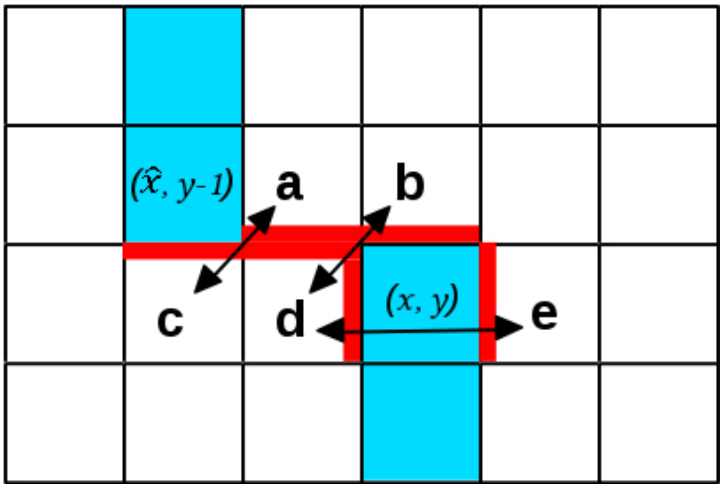


Figure 3.15: *The blue squares are pixels belonging to a seam. After removing it, the pixels labeled a through e change their neighbors. The affected sides of the pixels are marked in red. In this example, the forward energy is $|d-e|+|a-c|+|b-d|$.*

Total energy is a function in three variables: x and y coordinate of the pixel and the horizontal location $\hat{x}$ of the seam pixel in the row above. This is explained in more detail later. Throughout this section, the hat over a symbol is used when referring to values in the previous row or previous frame. The α are weights for the three different types of energy. In our implementation, pixel intensity and disparity values are normalized to [0..1] when used in the energy function. We use $\alpha_1 = 5$, $\alpha_2 = 0.5$, and $\alpha_3 = 0.1$.

3.3.2 Appearance Energy

Appearance energy describes the effect of introducing new edges into the frames by removing seams and bringing pixels together that were originally seperated by a seam. In the literature, this is also known as *forward energy* [68]. The appearance energy $E_{app}(x, y, \hat{x})$ at a pixel position (x, y) depends not only on the pixel position itself, but also on the horizontal position $\hat{x}$ of a potential seam pixel in the row above $(\hat{x}, y-1)$. This is illustrated in Figure 3.15. Depending on which pixel in the row above ends up being part of the same seam, a different set of pixels become adjacent, introducing different new edges. In Figure 3.15, the pixels labeled a through e change their neighbor after removing the seam.

Seam pixels do not need to be diagonally connected. In the case of stereo frames, there are situations where the seam may need to become discontinuous. The pixels of the seams in the left and the right view are connected by the disparity map, as shown in Equation 3.5. If a seam

crosses the border of an object that is closer or further away, the disparity value changes from one seam pixel to the next. One of the seams thus inevitably becomes discontinuous. As a consequence, E_{app} must be defined in a way that allows to compute it for an arbitrary distance between x and $\bar{x}$.

The appearance energy can be computed based on two parts:

$$E_{app}(x, y, \hat{x}) = E_{hor}(x, y) + E_{ver}(x, y, \hat{x}) \tag{3.7}$$

There are horizontal (E_{hor}) and vertical energies (E_{ver}). When a pixel at (x, y) is removed, its left and right neighbors become adjacent, introducing a new edge. This is measured by the horizontal energy which is simply the difference between the intensities of the left and the right neighbor:

$$E_{hor}(x, y) = |I(x-1, y) - I(x+1, y)| \tag{3.8}$$

If $x \neq \hat{x}$, removing a seam causes a shift between rows $y-1$ and y over the length of $|x-\hat{x}|$. In Figure 3.15, pixels ac and bd become adjacent, and new edges are introduced between them. This is measured by the vertical energy:

$$E_{ver}(x, y, \hat{x}) = \begin{cases} \sum_{k=\hat{x}+1}^{x} |I(k, y-1) - I(k-1, y)| & \text{if } \hat{x} < x \\ \sum_{k=x+1}^{\hat{x}} |I(k-1, y-1) - I(k, y)| & \text{if } \hat{x} > x \end{cases} \tag{3.9}$$

$E_{app}(x, y, \hat{x})$ is computed once for the pixels in the left frame and once for the right frame. The horizontal pixel positions x and $\hat{x}$ are mapped into the right frame by subtracting the disparity. Like this, the appearance energy is calculated for the left and the right view simultaneously. The final value for $E_{app}(x, y, \hat{x})$ is then obtained by adding the energy values of the two corresponding pixels.

3.3.3 Disparity Energy

Detected seams are not only removed from the left and right views, but also from the disparity map. Similar to intensity images, removing seams in the disparity map also introduces undesirable edges. Furthermore, the disparity map gives clues about the importance of pixels. We make the assumption that objects that are closer to the viewer are more relevant and should be removed less likely. These criteria are incorporated into the disparity energy E_{3D}. E_{3D} is

composed of forward energy in the disparity map E_{disp}, the distance of a pixel from the camera E_{dist}, and the confidence of the disparity estimation E_{conf}:

$$E_{3D}(x, y, \hat{x}) = E_{disp}(x, y, \hat{x}) + \alpha_4 E_{dist}(x, y) + \alpha_5 E_{conf}(x, y) \tag{3.10}$$

This definition of disparity energy is similar to the one in [3]. Disparity is normalized to values between 0 and 1, and based on our experiments, we chose the weights to be $\alpha_4 = 0.1$ and $\alpha_5 = 1$.

$E_{disp}(x, y, \hat{x})$ is defined in the same way as E_{app} above, except that it is computed over the disparity map instead of the intensity image. Objects that are closer to the camera have a higher disparity. The energy from object distance E_{dist} is thus simply defined as the normalized disparity D:

$$E_{dist}(x, y) = D(x, y) \tag{3.11}$$

The estimation of the disparity map may be noisy and contain errors. In order to cope with noisy measurements, we include E_{conf} into the disparity energy, which represents the confidence in the disparity measurement at a pixel. For a good disparity value, the two sides of Equation 3.4 only differ by a small amount. If the difference is large for two pixels (x, y) and $(x - D(x, y), y)$ in the left and right views, respectively, it is likely that $D(x, y)$ is erroneous. The confidence in the disparity estimation is thus defined as the difference between the left and the right pixel:

$$E_{conf}(x, y) = |I^L(x, y) - I^R(x - D(x, y), y)| \tag{3.12}$$

3.3.4 Temporal Energy

When applying seam carving frame by frame to a video, the seams take a different path in every frame. This introduces artificial motion into the frame which is perceived as a disturbing flicker artifact. To avoid flicker, it is necessary to make sure that seams do not differ from the seams in the previous frame too much. This is done by adding temporal energy to the energy function, as was shown in [24]. During the detection of the i-th seam in the *current* frame, the temporal energy E_{temp} for a pixel measures by how much the result differs if this pixel is removed instead of removing the i-th seam of the *previous* frame again.

More formally, when computing the i-th seam $S_i^L(y)$ in the left frame at time t, the i-th seam in the left frame at time $t-1$ is taken into account. This seam in the previous frame is denoted by $\hat{S}_i^L(y)$. If the exact same seam $\hat{S}_i^L(y)$ was used again as the i-th seam of the current left frame I_t^L, the resulting frame after removing the seam would be $\hat{I}_t^L$. Row y of frames I_t^L and $\hat{I}_t^L$ are shown on the right side of Figure 3.16. Frame $\hat{I}_t^L$ would have perfect temporal consistency, because the same pixels as in the previous frame were removed. For each pixel position (x, y) in the left frame, the temporal energy $E_{temp}^L(x, y)$ is thus computed as the difference between frame I_t^L as if it were carved by a seam going through pixel (x, y), and the perfectly consistent frame $\hat{I}_t^L$. Removing a seam pixel at position (x, y) in frame I_t^L means that all pixels to the right of x are shifted left by one. Hence, E_{temp}^L is defined as:

$$\begin{aligned} E_{temp}^L(x,y) = & \sum_{k=0}^{x-1} |I_t^L(k,y) - \hat{I}_t^L(k,y)| \\ + & \sum_{k=x+1}^{w-i+1} |I_t^L(k,y) - \hat{I}_t^L(k-1,y)| \end{aligned} \tag{3.13}$$

In Figure 3.16, $\hat{S}_i^L(y)$ is greater than x, so all pixels up to $x-1$ are identical in the two images I_t^L and $\hat{I}_t^L$. This means that the first sum is zero. The second sum of differences is shown as diagonal arrows in the Figure. It only has $\hat{S}_i^L(y) - x$ nonzero terms.

Analogous to the left frame, the i-th seam $\hat{S}_i^R(y)$ of the previous right frame is used on the current right frame I_t^R to produce a right frame $\hat{I}_t^R$ with perfect temporal consistency. E_{temp}^R is then computed in the same way for the right view by mapping x into the right frame by subtracting the disparity $D(x, y)$. Total temporal energy E_{temp} is then obtained by adding the values of both views:

$$E_{temp}(x,y) = E_{temp}^L(x,y) + E_{temp}^R(x - D(x,y), y) \tag{3.14}$$

3.3.5 Finding and Removing Seams

After fully defining the energy function, it can be used to detect and remove seams with low energy in the video frames.This is done in the following steps. The energy function is accumulated row by row and stored as an accumulated energy map. This map is used to find a pair of seams with minimal energy, which are then removed from the left and right frame. Lastly, the left seam is also removed from the disparity map, and the disparity values are updated to accurately represent the differences of the positions of the remaining pixels between the views.

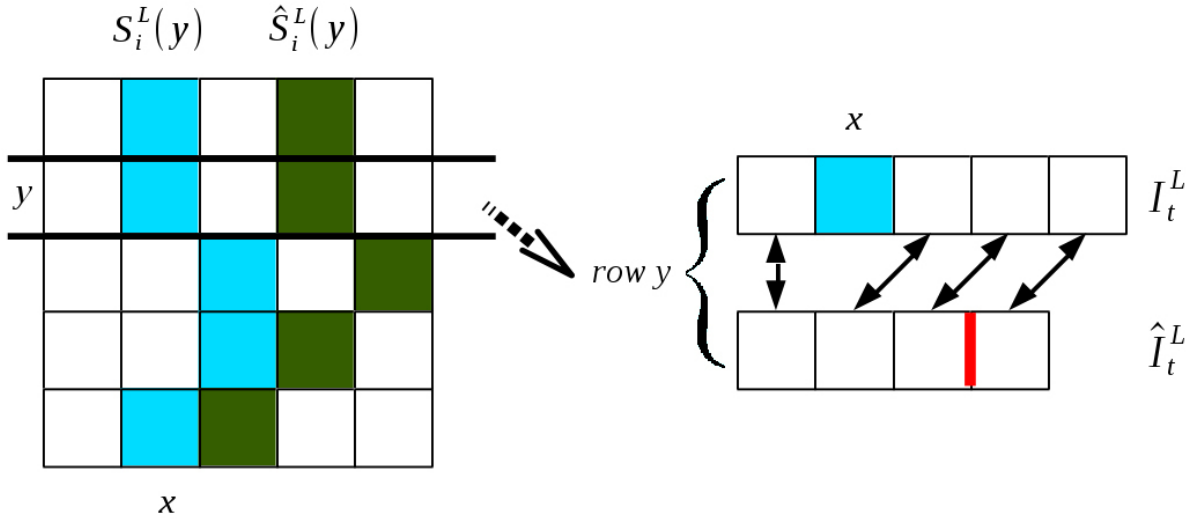


Figure 3.16: *The blue seam is a potential seam in the current frame. The green one is the unchanged seam $\hat{S}_i^L(y)$ from the previous frame. For pixel (x, y), temporal energy is computed as a sum of differences between the current frame I_t^L and the frame $\hat{I}_t^L$, which is the result of removing seam $\hat{S}_i^L(y)$ from I_t^L. The red line marks the pixel that was removed. Pairs of pixels for which the difference is calculated are marked with an arrow. The leftmost and rightmost pair of pixels have zero difference.*

Note that only one seam pair is detected and removed at a time, so the seam index i can be omitted.

In order to compute a pair of seams $S^L(y)$ and $S^R(y)$, the energy function is accumulated over each row of the frame, starting from the top. The result is an accumulated energy map $M(x, y)$. $M(x, 0)$ simply consists of those types of energy that do not depend on pixels in the row above (all but E_{ver} and E_{disp}). For each pixel position (x, y), all potential predecessor pixels $(\hat{x}, y - 1)$ in the row above are considered. For each potential predecessor location $\hat{x}$, the accumulated energy $M(\hat{x}, y - 1)$ of the predecessor is added to the energy $E(x, y, \hat{x})$ of the current pixel. The $\hat{x}$ for which this sum becomes minimal is chosen as the predecessor of pixel (x, y):

$$M(x, y) = \min_{\hat{x}} M(\hat{x}, y - 1) + E(x, y, \hat{x}) \tag{3.15}$$

$\hat{x}$ is stored for each pixel position (x, y).

The last row of the accumulated energy map $M(x, h-1)$ then contains the accumulated energy of a left seam ending in location $(x, h-1)$. The minimum of the entire last row marks the endpoint of a left seam with the lowest energy:

$$S^L(h-1) = \arg\min_x M(x, h-1) \tag{3.16}$$

$(S^L(h-1), h-1)$ is thus the last pixel of the seam. For this location, a predecessor $\hat{x}$ was stored during energy accumulation. Consequently, $(\hat{x}, h-2)$ is the second to last seam pixel. By following the stored predecessors in this fashion, the seam $S^L(y)$ is defined for each row from bottom to top.

Note that M was computed using information from both views simultaneously. This means that the detected seam has minimum energy with respect to the left *and* the right view. The left seam S^L can now simply be mapped to the right frame by using Equation 3.5.

The i-th detected vertical seams S_i^L and S_i^R for the left and the right view are now removed from their respective frames. Since this is done in the same way for both views, the superscripts are dropped here. To remove seam $S_i(y)$ from frame $I(x, y)$, each row y is processed individually. All pixels to the right of seam position $(S_i(y), y)$ are shifted left by one pixel:

$$I(x, y) := I(x+1, y) \qquad \text{for} \qquad x = S_i(y), \ldots, w-i-1 \tag{3.17}$$

Doing this for each row y reduces the width of I from $w-i+1$ to $w-i$.

For reasons of efficiency, the disparity map is not recomputed after the removal of each seam. Instead, the seam is also removed from the disparity map and the disparity values around the removed seam are updated [3]. For the description of how the disparity map is updated, we use the following notation: x^L is the horizontal position of a pixel in the left frame before seam removal. x^R is this pixel's horizontal position in the right frame. The mapping is done by subtracting the disparity from x^L:

$$x^R = x^L - D(x^L, y) \tag{3.18}$$

After removing the pair of seams, the pixel's new horizontal coordinate is $\bar{x}^L$ in the left frame and $\bar{x}^R$ in the right frame. In accordance with Equation 3.17, this coordinate is calculated as:

$$\hat{x}^L = \begin{cases} x^L & \text{if } x^L < S^L(y) \\ \text{undef.} & \text{if } x^L = S^L(y) \\ x^L - 1 & \text{if } x^L > S^L(y) \end{cases} \tag{3.19}$$

$\bar{x}^R$ is defined analogously. For each pixel position $(\bar{x}^L, y)$, the new disparity value is calculated as the horizontal distance of the corresponding left and right pixels after seam removal:

$$D(\bar{x}^L, y) = \bar{x}^L - \bar{x}^R \tag{3.20}$$

3.3.6 Evaluation

We evaluated the achieved quality of our algorithm by resizing five challenging stereoscopic videos. As there is currently no other method for content-aware resizing of stereo videos, we compare our new technique to our implementation of [3]. It employs appearance and disparity energy and avoids removing occluded or occluding pixels. However, the energy function in [3] has no temporal component as it is a still image approach. In the following, we refer to our own approach as SV for "stereo video" and abbreviate the other method by SFW for "stereo frame-wise".

The evaluation was a no-reference comparison where the test subjects only got to see the retargeted results, but not the original sequence. This is comparable to the real-world situation where users only see the resized videos on their devices. As test sequences, five stereo videos depicting indoor and outdoor scenes with moving objects were used. We refer to them as: "dialog", "office", "street", "table" and "walking". Example frames of the resized sequences are shown in Figure 3.17[2]. The full videos with a side-by-side frame format can be found online[3]. The original size of the videos was 480 x 270. They were resized to a size of 384 x 270, which is a reduction in width by 20%.

The evaluation was conducted on a desktop computer with an NVIDIA GeForce GTX 560 graphics card, NVIDIA GeForce 3D Vision shutter glasses, and a Samsung Sync Master 2233 display operating with a refresh rate of 120 Hz. For each video sequence, the results of the two algorithms were shown in random order. The participants were first asked which of the two videos they preferred. Then the subjects assigned scores to the two sequences in four

[2]The stereo vision frames for viewing with anaglyph (red/cyan) glasses were created with `http://instantsolve.net/anaglyph/`

[3]`http://ls.wim.uni-mannheim.de/de/pi4/research/projects/retargeting/`

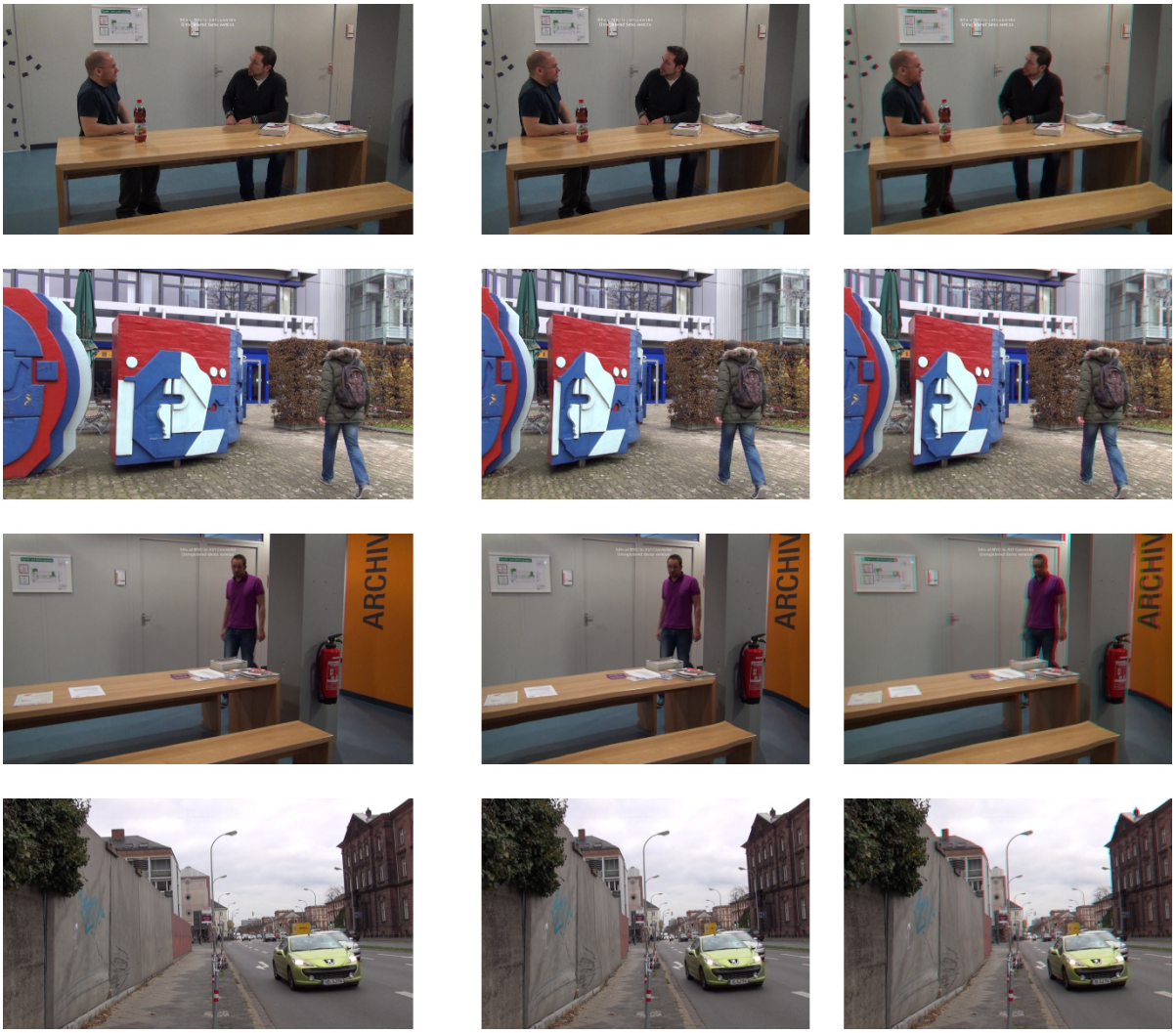


Figure 3.17: *Example frames from the test sequences "dialog", "walking", "table", and "street" that were used in our evaluation. The width of the videos was reduced by 20%. Left: left view of the original frame. Middle: left view of the resulting frame. Right: stereo vision of the resulting frame for viewing with anaglyph (red/cyan) glasses.*

	Deformation		Cut-off objects		Flicker		3D effect		Preferred by	
	SFW	SV	SFW	SV	SFW	SV	SFW	SV	SFW	SV
"dialog"	2.18	1.76	1.59	1.35	3.00	1.18	1.41	1.12	0	17
"office"	2.06	1.59	1.29	1.94	2.88	1.65	1.18	1.06	3	14
"street"	2.65	2.47	1.65	1.76	2.76	1.76	1.88	1.76	3	14
"table"	2.06	1.88	1.00	1.00	2.82	1.12	1.24	1.18	0	17
"walking"	1.71	1.41	1.18	1.53	2.88	1.35	1.24	1.12	1	16
Average	2.13	1.82	1.34	1.52	2.87	1.41	1.39	1.25	1.4	15.6

Table 3.3: *Detailed overview of the scores given in the user evaluation.*

categories: deformation, cut-off objects, flicker, and distortion of the 3D effect. One of the following three grades could be given to each video in each category:

1. not noticeable

2. noticeable, but not disturbing

3. noticeable and disturbing.

A total of 17 participants took part in the evaluation, three of which were knowledgeable in the field of video processing. Only fully executed surveys were used.

Analysis and Discussion

The evaluation showed that results of our stereo video approach were significantly preferred over the frame-wise approach without a temporal component. When asked which of the two compared videos had a higher overall quality, the subjects chose the video produced by our method 92% of the time. The scores in the four categories which were given by the participants are shown in Table 3.3. It can be seen that the viewers' preference is mainly influenced by the improved temporal stability of our approach, which leads to considerably less flicker. The scores in the other three categories were largely the same for both approaches, as was expected.

Deformations were noticed in both approaches equally but were classified as not disturbing. The least distortions were spotted in the "walking" sequence while the most were found in the "street" sequence. This is because "street" contains a lot of structured background and fast moving objects which move over a large portion of the screen. As mentioned above, this is not a beneficial scenario for seam-carving-based algorithms in general. The "walking" sequence shows art in the form of abstract patterns in which deformations cannot be detected easily.

Our algorithm performed slightly worse in the category of cut-off objects. Both scores are in the range that indicates that this artifact remained mostly unnoticed. When they were detected in a video, they were not disturbing to the viewer. Because SFW works on a per frame basis, it is more flexible in avoiding collisions of seams with moving objects. This led to a slightly better score than SV.

Flicker is an artifact which nearly all participants found to be very disturbing in the videos that were resized using the SFW approach. It received the worst possible score in almost all of the ratings in this category. This is because the frames are processed individually without taking temporal information into account. The seams can thus vary freely between the frames which creates a disturbing flicker effect. In our approach, seams are kept more stable between the frames, which resulted in a better score. Flicker was not noticed in the SV sequences most of the time.

The 3D impression of the sequences achieved high scores in both approaches. The subjects did not notice an impairment of the 3D effect on the average. This category achieved the highest overall score.

Limitations

The approach described in this Section produces visible distortions in some of the shots. As the seams are temporally connected, they may cross objects that are large or fast moving. In such situations, seam carving may not be the resizing technique of choice.

We also found it difficult to obtain good disparity maps in our approach. The requirements for the computation of a disparity map contradict the requirements of seam carving. While seam carving works best in large untextured areas where there is little energy, pixel correspondences for disparity maps are best computed over highly textured regions. Erroneous disparity values have negative effects on many aspects of the energy function, which makes seam carving of stereoscopic media difficult in general.

CHAPTER 4

SeamCrop

Seam carving has some limitations that can not be solved without combining it with other retargeting operators. For instance, if there are a lot of objects in an image or if a frame has to be reduced in size significantly, it is most likely that interesting objects collide with seams, resulting in visible artifacts. Therefore, we looked at other retargeting operators that are able to overcome these limitations. In a comparative study of image retargeting algorithms with users by Rubinstein *et al.* [69], manual cropping outperformed the state-of-the-art automatic techniques in a large evaluation. This leads them to the assumption that the automatic search of a cropping window is still a viable research topic.

SeamCrop combines *cropping* and *seam carving* to benefit from the advantages of both techniques. Cropping can adapt images or frames without introducing any artifacts besides possible cuts of objects at the borders. This is enhanced by carefully used seam carving that is able to remove pixels from within the image, thus getting more important content into the cropping frame. We do not use warping in our approach as it may introduce squeezing artifacts to faces and objects.

The remaining research questions presented in Chapter 1.2 are solved in this Chapter. We first present SeamCrop for the retargeting of images. This is further enhanced by a new visual importance measure that is able to determine whether a face in an image is important or not. Then the algorithm is extended for the adaptation of videos. Lastly, the already efficient SeamCrop for videos is parallelized and implemented on a GPU for a significant performance gain. The algorithms presented in this Chapter were published in [40, 37, 39, 38].

4.1 SeamCrop for Images

Our SeamCrop algorithm is designed with the preference of content loss over the insertion of deformations [69]. It combines the two image retargeting operators *seam carving* and *cropping*. Several hybrid techniques use *warping* instead of cropping or as an additional third operator. We have chosen not to include it in the algorithm because it may cause squeezing artifacts when the aspect ratio is changed. Scaling is used, however, to uniformly scale down very large images before they are adapted to a different aspect ratio.

4.1.1 Algorithm

In the following, we assume that the width of an image is reduced. The reduction of the height can be done analogously. First, a saliency map of the source image is calculated. Based on this map, vertical seams are taken out of the image until the target size is reached or more than α percent of the energy has been removed. If the target size has not been reached, the algorithm switches to cropping. An optimal cropping window with the smallest possible size is searched in the image until the target size is reached or more than β percent of the energy has been cropped. As the switching point between seam carving and cropping is crucial for the quality of the result, we use thresholds that dynamically adapt to the image content. The algorithm is repeated until the target image size is reached. Figure 4.1 gives an overview of the workflow.

Energy function

The energy function we use is a contrast-based saliency map. It is computed on a copy of the original image that was converted into the LUV color space. This color space has the advantage of being perceptually uniform. The perceived distance between two colors can thus simply be calculated using the Euclidean distance between the LUV coordinates. For each image pixel, the color distance to all neighbor pixels in a rectangular window is computed. It is weighted by a Gaussian function to create a value between 0 and 1. The saliency value for the considered pixel is then obtained by averaging the color distance over the entire window. Areas with little color variation thus have lower energy than those with high contrasts.

Marking seams

For the removal of seams, we mark the paths of the seams in the image and give the included pixels a very high energy value to make them unusable for further seams. Because of this, it may happen that a seam different than the energy-optimal one is chosen, since the seam may

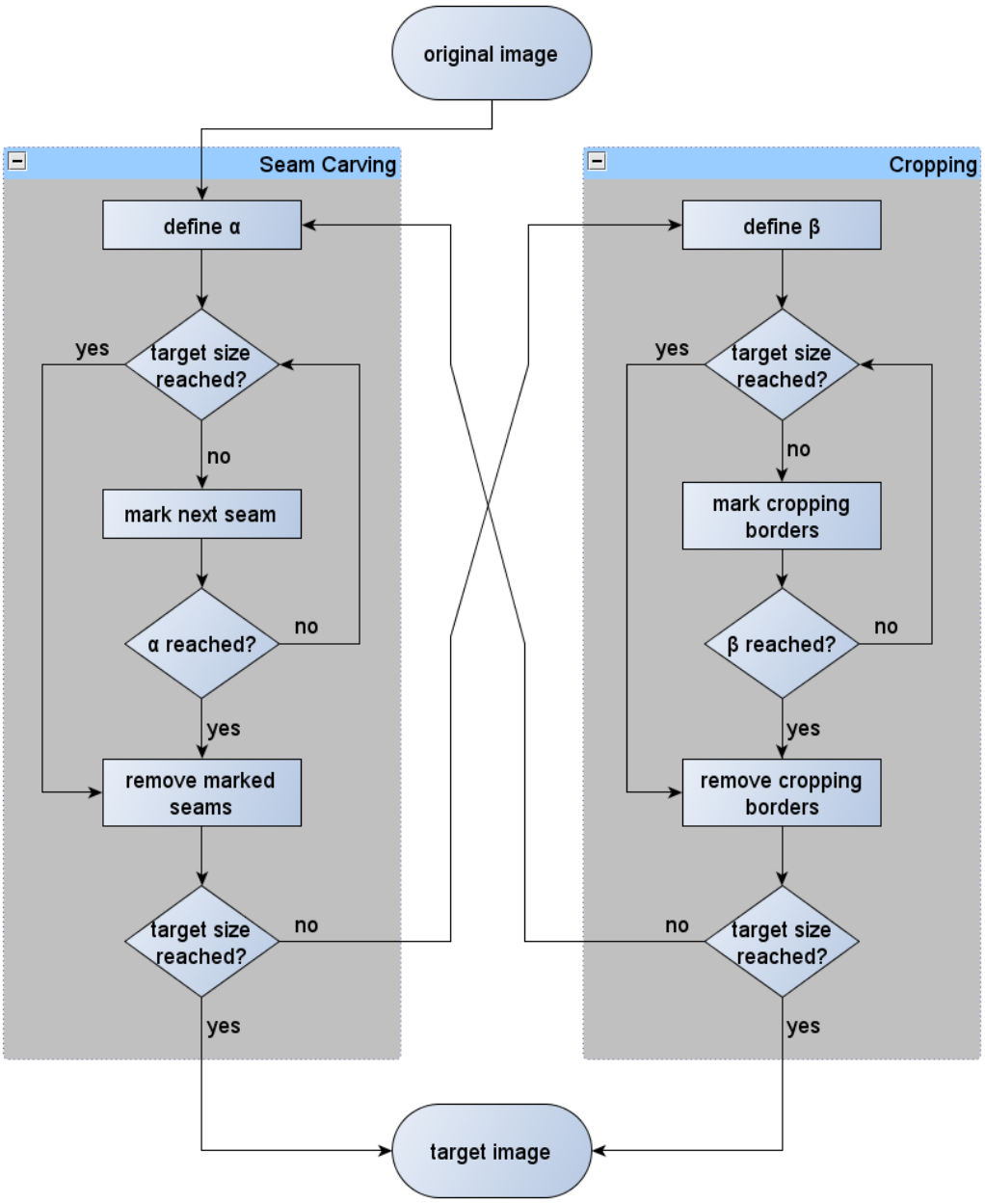


Figure 4.1: *The operator switches between seam carving and cropping based on a dynamic threshold. This is repeated until the target size is reached.*

not be able to cross a previously marked path. In practice, this is a major advantage because the seams are spread more evenly over the image. Our intention is to remove seams very carefully to prevent visible artifacts. Only if the summed up energy of the marked seams hits the threshold and fulfills its requirement or if the target size is reached, the seams are actually removed.

Dynamic threshold

The dynamic thresholding is composed of two parts: First, seams may only be removed up to a certain percentage α of the total energy of an image. The second part is the requirement that the seams discarding about α percent of the total energy must remove at least the same percentage of image pixels. This is based on the idea that an operator is only effective when it removes a higher percentage of pixels than it removes energy, i.e., relevant content. Only if this requirement is fulfilled, the marked seams are actually removed. If necessary, the algorithm then switches to the cropping operator.

Positioning of the cropping window

As the width of the image has been modified by seam carving, a new saliency map has to be computed first. This saliency map is projected to a one-dimensional energy array by summing up the saliency values for an entire column. After that, we use a brute-force 1D search to find the optimal window position with the smallest possible size that discards less than β percent of the energy of the image. β should be chosen higher than α as the intention is to remove seams only carefully, and cropping should not produce artifacts like bend or broken objects.

Cropping starts with the width of the cropping window being one column smaller than the width of the image. The total energy of all possible positions (initially two) is calculated and the position with the highest energy inside the window is picked. Then, if less than β percent of the energy is discarded, the step is repeated again with the window size reduced by one. This is continued until the removed energy exceeds β percent. The last size and position with discarded energy below the threshold is chosen as the optimal cropping window. Like the marked seams, the cropping window has to fulfill a requirement in order to be executed. The algorithm stops when β percent of the total *energy* of the image would be discarded. At the same time, the crop has to remove at least β percent of the *width*. If the requirement is not met, the crop is skipped. After that, the algorithm switches the operator again if necessary.

It may happen that both operators are skipped repeatedly due to the thresholds. In this case, the requirements are relaxed to allow the removal of lesser width than energy. In practice, this is very unlikely: if high energy regions at the borders prevented a crop, the energy in the center would be lower, allowing the removal of seams there. When there is high energy in the middle, a crop is possible.

4.1.2 Evaluation

We performed an evaluation in order to compare our algorithm to current state-of-the-art image retargeting techniques. The comparative study [69] and the images from the benchmark data set[1] provided by Rubinstein *et al.* were used as the basis for our comparison; they were indeed exceptionally helpful for our study.. Our results of the full benchmark are available on the web[2]. A manually chosen crop, *multi-operator retargeting* [70] and *streaming video* [45] were the best techniques in nearly all test cases of the study. Therefore, the authors suggest that it is sufficient to demonstrate that a new algorithm outperforms these three.

Our evaluation was a no-reference comparison where the original image was not shown to the subjects. This simulates the real-world situation in which a user only gets to see the retargeted result. Nine image sets consisting of four resized images each were evaluated by the participants. Each set consisted of retargeted results of the same source image calculated by the three methods mentioned above and by our new SeamCrop algorithm. The nine image sets were randomly chosen by our evaluation software out of the images provided in the benchmark data set. We did not include a reduction in height in our evaluation. This led to 71 possible image sets out of 80.

In an image set, the participants could rank the results by giving 1 to 4 points to each resized image, with 1 being the best and 4 the worst. Additionally, they were asked a number of questions.

A total of 16 subjects (14 male, 2 female) took part in the evaluation. One half were students, the other half were colleagues from our department. A total of 144 image sets was evaluated. In each set, the images are reduced to either 75% or 50% of their original width, depending on the values used in the benchmark data set. As parameters, we used $\alpha = 1\%$ as the seam carving threshold, and $\beta = 15\%$ for cropping.

Analysis and Discussion

Based on the evaluation results, the mean rank and the standard deviation σ for each method have been calculated. They are presented in Table 4.1.

Similar to the comparative study [69], the manually chosen crop was clearly the preferred technique by the participants. The gap between the other techniques is much smaller with our approach being ranked second. As the visual attention analysis of an image is one of

[1] `http://people.csail.mit.edu/mrub/retargetme`
[2] `http://ls.wim.uni-mannheim.de/de/pi4/research/projekte/retargeting/`

Original Manual Automatic

Figure 4.2: *A comparison of manual and automatic cropping. The automatic crop was done with brute force on the same energy map that is used by our SeamCrop algorithm.*

	Manual Cropping	Multi-Operator	Streaming Video	Seam Crop
Mean rank	1.72	2.45	2.53	2.21
σ	1.02	1.05	1.04	1.11

Table 4.1: *Mean rank and standard deviation σ of the four evaluated techniques.*

the most challenging parts of image retargeting, the advantage of a manually picked cropping window over automatic detection is obvious: The human editor can identify all important objects and pick the perfect position by also taking the composition of an image into account (see Figure 4.2). Additionally, a crop does not cause any artifacts in the retargeting process. This may be the reasons why manual cropping ranks first in the no-reference comparison. It can be assumed that the quality of the other three techniques would benefit from a manually created saliency map.

After the evaluation, the participants were asked if there were things in the resized images that bothered them. Many subjects noticed squeezing artifacts in several sets. While all of them found the artifacts disturbing when persons were displayed, some found them acceptable for images depicting only buildings or nature without any important objects in the foreground. The participants were also asked if they generally preferred the loss of content or squeezing. All of the subjects stated that they prefer loss of content, although some specifically added the restriction that prominent foreground objects should not be truncated. This supports our decision not to include warping in the algorithm due to this effect. An example of an image set from the evaluation is shown in Figure 4.3.

Manual crop

Multi-operator

Streaming video

SeamCrop

Figure 4.3: *Example image set from the evaluation.*

	Seam Carving	Cropping	σ
To 75% width	20.23%	79.77%	9.62%
To 50% width	24.96%	75.12%	15.70%

Table 4.2: *Mean ratio and standard deviation σ of the operators used in the benchmark.*

A comparison between the single operators and our approach demonstrates that it is effective to combine these two (see Figure 4.4). Table 4.2 gives an overview of the average seam carving to cropping ratio that occurred in the benchmark data set images.

When images are reduced to 75% of their width, about 20% of the pixels are removed with seam carving and 80% with cropping. In case of a 50% target width, about 25% of the reduction is done with seam carving and 75% with cropping. As cropping also cuts image content already manipulated by seam carving, this indicates that the seam carving operator is used only carefully, like we intended it.

Figure 4.4: *A comparison of the individual operators and SeamCrop. As mentioned before, the automatic crop was done with brute force on the same energy map that is used by our SeamCrop algortihm.*

4.2 SeamCrop for Images with Improved Face Detection

As mentioned in Section 1.1, the quality of the results of a retargeting algorithm significantly depends on the accurancy of the importance map. Especially faces draw a lot of attention from the viewer. But not all faces in an image are equally important, for instance, the face of a basketball player in the foreground is more important than the face of a person in the audience behind him. These unimportant faces are often blurry and out of focus. Nonetheless, modern face detectors find all faces, regardless of focus. We have developed a new technique that is able to distinguish between these types of faces in order to improve the results of image retargeting algorithms.

For the detection of faces in images, we use the Viola and Jones framework [88]. Face regions are used as the input for the algorithm which is presented in the following. In a first step, each face is classified as in-focus or as out-of-focus. Next, in-focus faces are automatically segmented from the background with *GrabCut* [67] in order to generate face masks. Finally, these masks are encoded as binary maps which can then be used in image retargeting algorithms.

4.2.1 Face Detection with Multiple Cascades

It is crucial that the face detection algorithm has a high detection rate while at the same time keeping the number of false positives low. For example, whenever a non-face region is classi-

fied as a face, this region would be included in the face masks, and the other areas of the image would be deformed by a resizing operator using this information.

To achieve higher detection rates, the algorithm uses multiple cascades. A cascade is a set of filters used on the image in order to determine whether a region is a face or not. In order to be recognized as face, the region has to pass through the whole cascade. Multiple cascades generate multiple detections of the same face. However, cascading will also increase the number of false detections. In order to identify only one pair of coordinates for each face and eliminate probable false detections, the detected face regions are clustered. A threshold parameter δ is calculated, derived from the width w and height h of two rectangles:

$$\delta = s * [min(w_1, w_2) + min(h_1, h_2)] * 0.5 \tag{4.1}$$

If the absolute difference between the rectangles' upper corners is both smaller than or equal to δ, the rectangles are labeled as belonging to the same face. In case of $s = 0$, each rectangle belongs to a separate cluster, whereas $s = \infty$ aggregates all rectangles into one cluster. Considering that four cascades are used in our algorithm, a value of $s = 0.2$ provides good results. If less than three rectangles have been assigned to a cluster, it is not considered as a face, and the detections belonging to the cluster are removed. The four cascades we use are provided by the OpenCV software development kit[3].

4.2.2 Focus Detection

A focus detection algorithm is used to exclude all blurry faces. First, a face region is classified as an elliptical object (or blob). Each face includes characteristic edges due to nose, mouth, eyes and eyebrows, among others. Strong edges are visible when a face is in focus. A face is classified as in focus when at least one of its edges is classified as a strong edge. Figure 4.5 visualizes the absence of strong edges in blurry faces.

An obvious approach to the edge detection problem is to analyze luminance variations in an image. We use the gradient magnitude [5] to identify edge pixels. A threshold is applied to remove weak edge pixels with a lower edge value. The threshold is set to 12.5% of the maximum value of the gradient magnitude. Based on the maximum gradient magnitude in the whole image, different threshold values were tested: $1/16$ (6.25%), $1/8$ (12.5%) and $1/4$ (25%) were examined in different images. A value of 12.5% empirically kept most of the in-focus edges and excluded most of the out-of-focus edges, and was thus chosen for our algorithm.

[3] opencv.willowgarage.com/

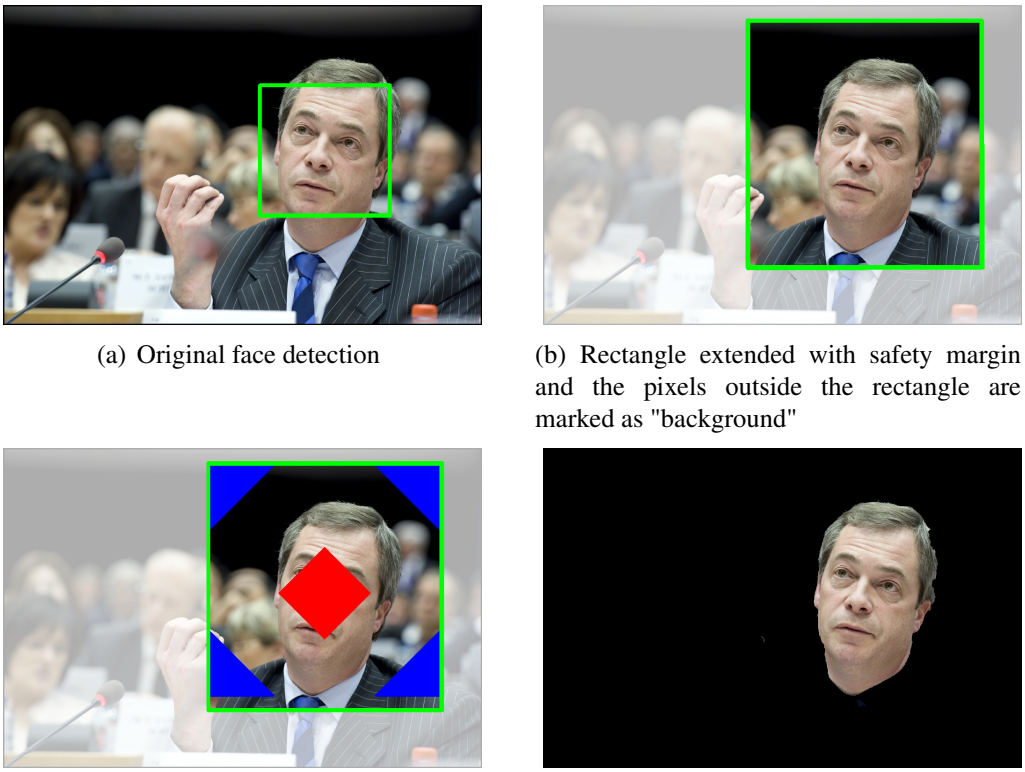
Figure 4.5: *Distinguishing blurred faces (red) from in-focus faces (green) based on strong edges (blue).*

Border pixels between face and background may cause additional strong edge pixels. Therefore, only a small area in the center of the face corresponding to 25% of the face rectangle's area is considered. A face is classified as in focus if at least one strong edge pixel is found in this area.

4.2.3 Creating Face Masks with GrabCut

In order to use the information of our improved face detection in an image retargeting algorithm, we want to segment the in-focus faces from the background and encode them as binary maps. For the segmentation, the GrabCut algorithm is used [67]. To segment objects, the original GrabCut algorithm requires some user interaction. The idea of the algorithm is to use Gaussian Mixture Models to specify the color distribution of background and foreground pixels and to use this information for the segmentation. Initial labels (foreground, background, probably foreground, or probably background) are provided by a user [71]. In our case, to avoid manual classification, we have implemented an enhancement to the algorithm where information provided by the face detection module is used.

A face region is defined by the center and the width and height of a rectangular region (see Figure 4.6(a)). The position or size of such a rectangle might not describe the exact face region as the alignment or size of the detected face might be inaccurate. Therefore, we define a safety margin and add an additional 40% to each border of the rectangle. Figure 4.6(b) shows the



(a) Original face detection

(b) Rectangle extended with safety margin and the pixels outside the rectangle are marked as "background"

(c) Foreground (red) and probably background (blue) labels

(d) Segmented face

Figure 4.6: *Creation of a face mask from the initial detection.*

enlarged face region. In order to obtain a more accurate segmentation of the face, labels are assigned to pixels inside and in the neighborhood of the rectangle. All pixels outside of this region are labeled as background and will not be part of the final segmentation. The labels 'foreground', 'probably foreground', and 'probably background' are assigned to the pixels of the enlarged face region depending on the distance of each pixel from the center of the region. Figure 4.6(c) visualizes the result of the labeling step.

GrabCut is applied in the last step to identify face pixels (see Figure 4.6(d)). To reduce the effect of discontinuities at the borders of a segmented face a morphological dilation operation with a square 3×3 structuring element is applied to the image. This reduces minor inaccuracies and also adds a safety margin so that no face pixels are missed in the mask.

4.2.4 Evaluation

We also performed a user evaluation of our improved face detection algorithm. It consisted of two parts. In the first part, we discuss the results and analyze the reliability of our enhanced

face saliency technique. In the second part, we show its application in the context of image retargeting by using it together with the previously introduced SeamCrop algorithm (4.1).

Focus Detection

We conducted an evaluation in order to analyze the reliability of our new face detection algorithm. 35 images with a total of 42 in-focus and 46 out-of-focus faces were used in this study. Each image contained at least one face that is in focus and one that is out of focus. As the ground truth, the faces in all images were manually marked and classified as in focus or out of focus. If the decision was not clear, at least three people classified such a face, and the majority of the classifications was used. We want to point out here that the automatic face detection does not recognize all out-of-focus faces that can be found manually, as visualized in Figure 4.7.

We analyzed the percentage of missed and correctly identified faces in a first step. By using multiple cascades, the number of false detections was reduced by 43% while increasing the number of missed faces by 5% compared to using only one cascade. In a second step, all correct face regions were analyzed further. 90% of the detected faces were assigned correctly as in focus or out of focus.

Although the algorithm performs a Gaussian smoothing operation before the focus detection is applied, it is still not completely immune to noise. Areas affected by noise can still have high intensity variations which results in high values of the gradient magnitude. This was the main reason for the 10% false assignments in the focus detection.

Figure 4.7 exemplarily shows an image where two face regions are detected. For visualization, in-focus faces are marked with a green rectangle and out-of-focus faces are marked with a red rectangle. In the image, the woman in the foreground is clearly the center of attention. If the unsharp face of the woman in the background was also considered in a retargeting algorithm, this would preserve both faces but at the same time would cause severe distortions in other regions of the image. In case of several in-focus faces that are spread over the image, the result of image retargeting with our new algorithm is comparable to normal face detection.

Application in Image Retargeting

As mentioned before, the results of the face detection are encoded as binary maps that are used as additional visual importance information in the image retargeting process. We chose to use SeamCrop in our application scenario and set the threshold values to $\alpha = 1\%$ and $\beta = 15\%$,

Figure 4.7: *An example of our focus detection algorithm for faces. One face that is in focus (green) and another out-of-focus face (red) are detected.*

like in the evaluation of the algorithm in Section 4.1.2. All images are reduced to 50% of their original width.

Figure 4.8 shows some results of the retargeting algorithm. In the top row (a), the shoulder and face of the man in front becomes severely distorted if the detected face in the background is treated like a normal face. With our technique, that face is classified as out of focus and therefore is allowed to be removed. In the second row (b), the face of the woman in front of the couple is unsharp. If this face region is considered in the retargeting with high priority, the target image is suboptimal. In the basketball image (c), many faces are found in the background. These faces distract the algorithm from the player in the foreground. The images with the two men in the last row (d) include detected faces but also a false detection. This incorrect region would be classified as out of focus. Focus detection also helps in this case; it would move the center of attention to the right. As our focus detection algorithm is also able to identify false positives in the background, it further improves the precision of faces that are in focus.

The shown images prove that our new face detection technique is able to significantly enhance the results of the retargeting algorithm. As this is an evaluation about the effectiveness of the distinguished face detection and we have already done a comparision of normal SeamCrop with other state-of-the-art image retargeting techniques in the previous Section (4.1), we have chosen not to do another one.

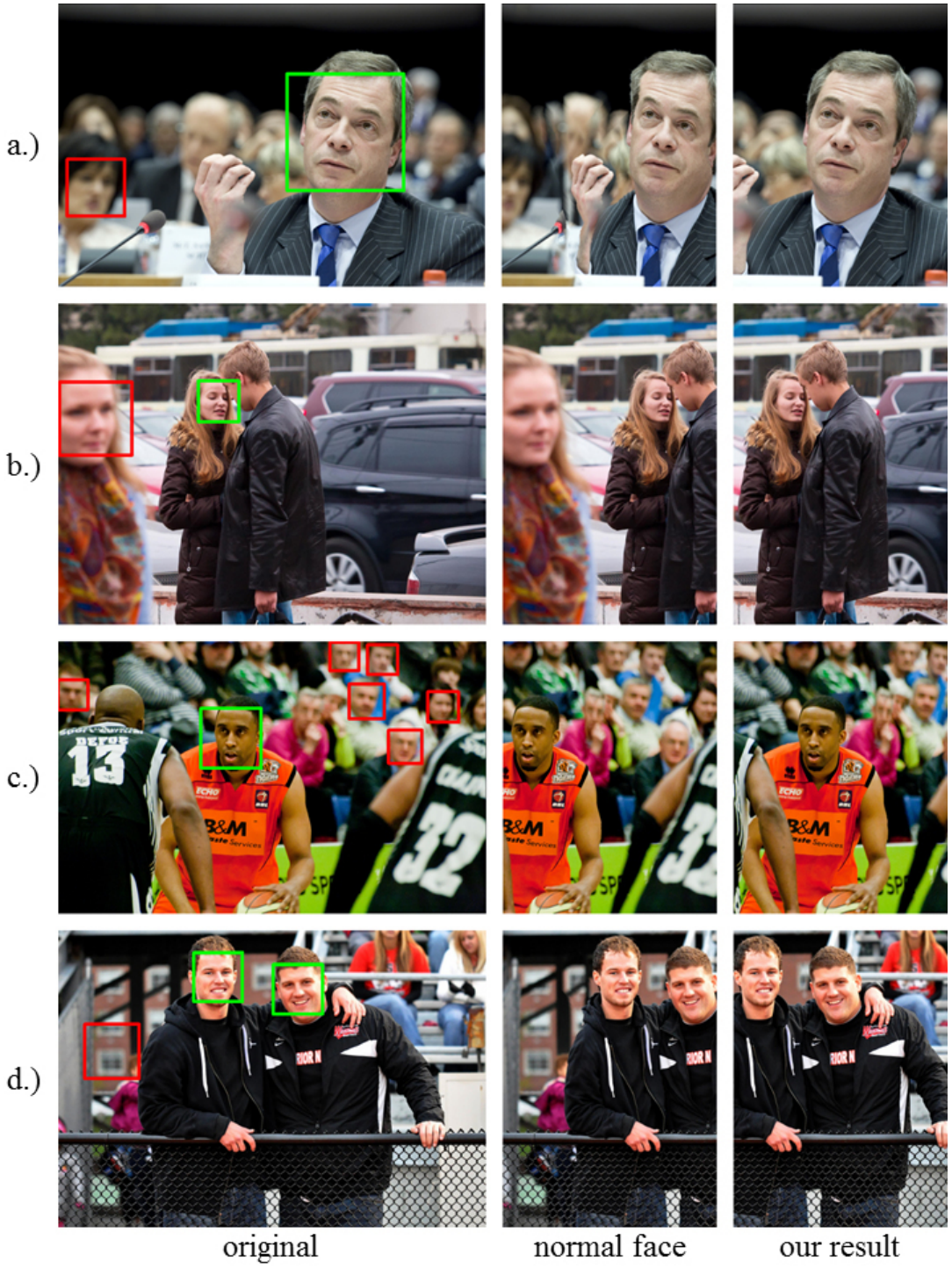


Figure 4.8: *Results of our image retargeting technique: The width of the original image is reduced by* 50%. *The results of the retargeting operator are compared when using all faces (center) and only faces that are in focus (right).*

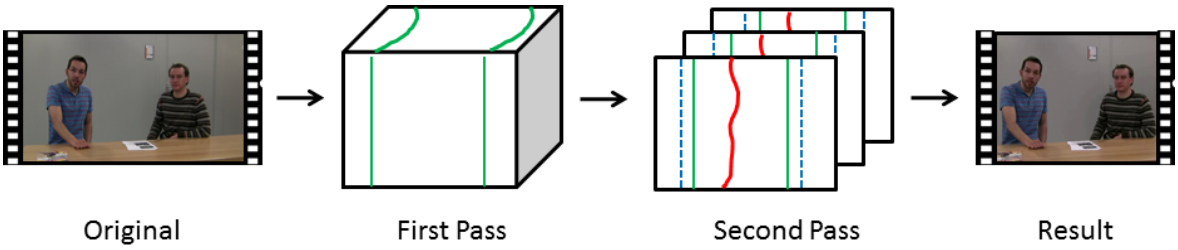


Figure 4.9: *Overview of the basic workflow of SeamCrop for videos. In pass one, the optimal cropping window is searched via dynamic programming (green lines) in a global optimization over all frames. Seams are searched in the second pass (red lines) inside the extended borders of the cropping window (blue dotted lines) frame-by-frame.*

4.3 SeamCrop for Videos

In this Section, we extend the SeamCrop algorithm for the retargeting of videos. Due to the additional temporal component of videos, the iterative approach of SeamCrop for images is no longer viable and has to be changed. The intention of SeamCrop is to have fast computation times while providing improved or at least comparable results to other state-of-the-art algorithms. Many of these algorithms use an optimization over the entire video sequence which is very time consuming. Instead, we only use the entire sequence for computing a cropping window while the seam carving is done on each frame separately (see Figure 4.9). This makes the optimizations that have to be computed less complex and consequently reduces the processing time.

The algorithm is done in two passes: the first pass calculates the positions of a cropping window with the target size over the course of the video, the second extends the cropping window and then removes seams to reach the target size again. As the movement of the cropping window depends on the content shown on the screen, a virtual pan may be introduced.

In the following, the reduction of the frame width is used to illustrate our algorithm. The reduction of the height is achieved in a similar manner.

4.3.1 Finding a Cropping Window

The video sequence to be resized consists of a sequence of T frames F^t. Unless required for clarity, we omit the time index t. Each frame has the original width m and height n and is resized to a target width of $m' < m$. At any given point in time, a cropping window can thus take on $m-m'+1$ possible horizontal positions. Over the duration of the entire video, the path of a cropping window is a path through a two-dimensional space of size $(m-m'+1) \times T$. This

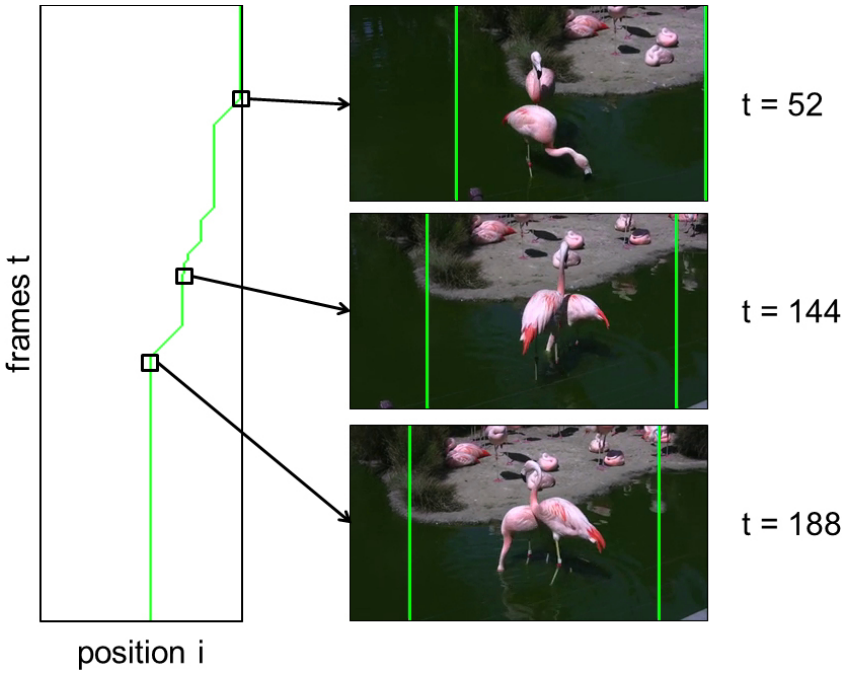


Figure 4.10: *Path of the cropping window over time. Each point in the path stands for the position of the cropping window in the corresponding frame.*

path is calculated via dynamic programming, similar to the computation of a seam in the seam carving algorithm (see Chapter 3.1) [2]. Figure 4.10 shows the path of the cropping window over time.

We begin by computing an energy map E for each frame in the video. The energy value of each pixel in E is representing the importance of the corresponding pixel in the frame. The energy map is composed of the rate of temporal change of a pixel (i.e., its motion) E_M and the measure E_G depicting the L_1 length of its gradient normalized to $[0..1]$. We combine them in the following way using weights that worked well in our experiments:

$$E = \frac{3}{4}E_M + \frac{1}{4}E_G. \tag{4.2}$$

The temporal change of a pixel is estimated by the difference of its values between the preceding and the following frame. As pixel values may differ slightly between frames due to lighting or small camera movements instead of object motion, the values in E_M are thresholded to either 0 or 1 using a threshold T_M. The threshold is computed for each frame individually.

First, the maximum difference value for each column i of E_M is determined as

$$e_i = \max_{j=1,..,n} E_M(i,j). \tag{4.3}$$

The threshold T_M is then set to 25% of the average of these maxima:

$$T_M = 0.25 \sum_{i=1}^{m} \frac{e_i}{m}. \tag{4.4}$$

Since thresholded values of 1 mainly appear on the edges of moving objects, a Gaussian smoothing filter is used to distribute the non-zero energy values over the total space of the object. Empirical tests have indicated that this simple measure works sufficiently well for detecting the temporal change of pixels in a video.

The values of the columns of the total energy map E computed above are now summed up. The result is a list of column costs

$$c_i = \sum_{j=1}^{n} E(i,j) \tag{4.5}$$

for each column $i = 1,..,m$. These costs are then used to determine the total energy W_i contained within each possible cropping window for position $i = 1,..,(m - m' + 1)$

$$W_i = \sum_{k=0}^{m'-1} c_{i+k}. \tag{4.6}$$

Calculating this total energy for every frame in the video yields a 2D array W_i^t where each value represents the total energy of one cropping window position i in one frame t (see Figure 4.11).

On this 2D array, dynamic programming is used with similar restrictions as in the seam carving algorithm in order to find the path with the maximum energy [2]. The energy of a path is the sum of the energy values of all path positions. It is determined by traversing the W_i^t space along the time axis and calculating the cumulative maximum energy $\widetilde{W}_i^t$ for each position as

$$\widetilde{W}_i^t = W_i^t + \max(\widetilde{W}_{i-1}^{t-1}, \widetilde{W}_i^{t-1}, \widetilde{W}_{i+1}^{t-1}). \tag{4.7}$$

The maximum value of $\widetilde{W}_i^t$ in the last row ($t = T$) indicates the total cost of the path with the highest energy. By backtracking from this maximum, we find the optimal path for a cropping

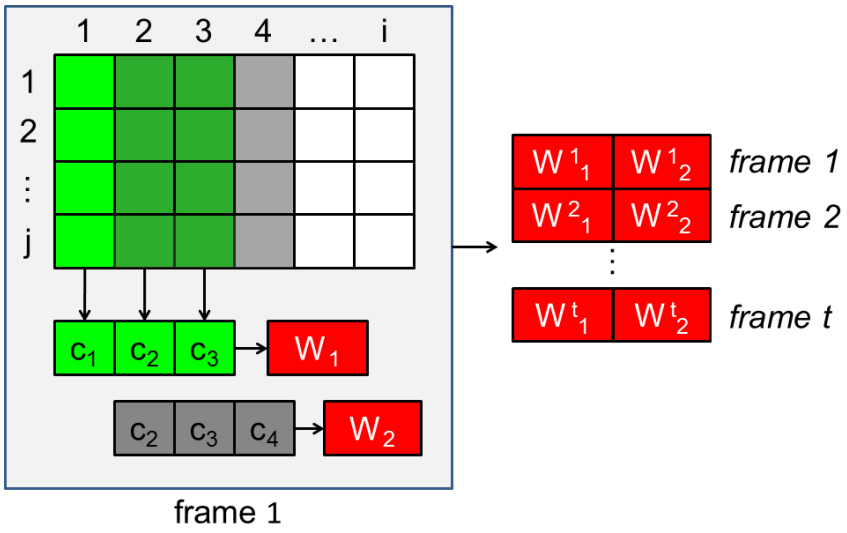


Figure 4.11: *The energy values in each column are summed up first. Next, this is also done with the summed up values included in a cropping window position (e.g., the green values are summed up to W_1). Lastly, the values for the positions of all frames are combined to a 2D array.*

window. As the paths found by the algorithm are connected, the resulting cropping window path is temporally coherent but it may contain jitter. In order to lessen the effect of jitter of the window, the computed positions are smoothed with a Gaussian filter.

4.3.2 Finding Seams

In order to include more possibly important content in the cropping window without introducing artifacts, the borders of the cropping window are slightly extended, and then seam carving is used frame by frame to reach the target size again. To ensure temporal coherence, the energy map E^t of each frame is modified by the seams found in the previous frame. We use a simplification of the temporal coherence costs introduced by Grundmann *et al.* [24]. They base their measure on the new edges that are introduced by moving the seam to a different pixel position. We, on the other hand, use costs increasing linearly from the position of the previous seam because it is sufficient for our technique, and the computation time is much lower.

As the cropping window already has the target size, it has to be extended before seams can be removed. It is extended equally on both sides (see Figure 4.12). The amount of enlargement is a parameter that controls the tradeoff between seam carving and cropping. We chose to enlarge the window by 20% in our experiments. If the cropping window moves towards a border of the frame and the added space would lie outside of the border, the space on the other

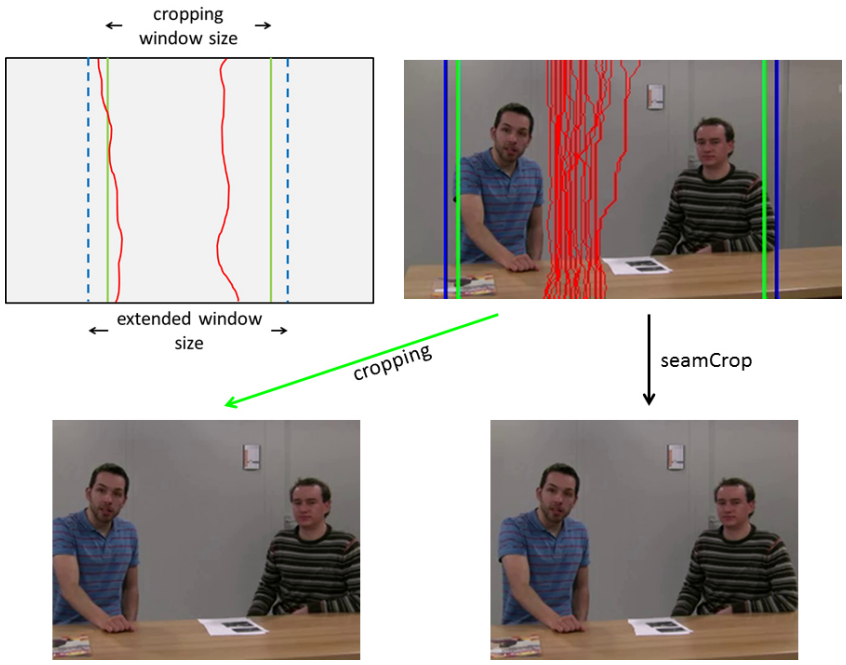


Figure 4.12: *Top row: The estimated cropping window (green lines) is extended (blue dotted lines) and then reduced to the target size again by removing a small number of seams (red). Bottom row: results of the frames being reduced by the cropping operator alone (left) and by SeamCrop (right).*

side is extended by an equal amount. Similarly, if the window leaves the border, its space is again equally distributed.

The k-th vertical seam in frame t is defined by the list of pixels it includes. It includes exactly one pixel per row of the frame. The seam can thus be fully described by a horizontal pixel position $s_k^t(j)$ for each row $j = 1, .., n$. We omit the row index j when referring to the entire seam. In contrast to the computation of the cropping window, which is optimized over all frames, the seams are calculated for each frame separately. To prevent temporal discontinuities and jitter that may occur by a frame-wise search for seams (see Chapter 3.2), temporal coherence costs are used to modify E before computing a seam.

We assume that a number of seams s_k^{t-1} have been calculated in the previous frame. Now, the same number of seams must be calculated for the current frame. We want to make it more likely for a new seam to be close to the seam with the same index in the previous frame. We thus add temporal coherence costs C_k^t to the energy map E before calculating seam s_k^t. The costs C_k^t are zero at the location of the corresponding seam s_k^{t-1} in the previous frame. They then increase linearly with increasing horizontal distance up to an upper bound β beyond a

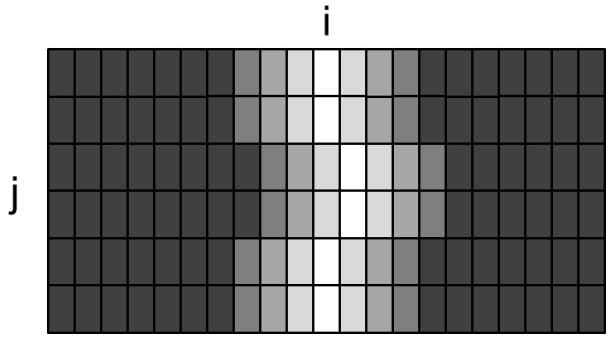


Figure 4.13: *Visualization of temporal coherence costs. The positions of the pixels of the corresponding seam s_k^{t-1} from the preceding frame are white, and the energy increases the darker the positions get.*

pixel distance of α (see Figure 4.13). The temporal coherence costs are thus defined as

$$C_k^t(i,j) = \begin{cases} \frac{\beta}{\alpha} \left| i - s_k^{t-1}(j) \right| & for \left| i - s_k^{t-1}(j) \right| < \alpha \\ \\ \beta & otherwise. \end{cases} \tag{4.8}$$

$(E + C_k^t)$ is then used as the energy function for calculating seam s_k^t. The parameter β adjusts how strongly the position of the seam of the previous frame is imposed on the new energy map. We chose an upper bound of $\beta = 0.3$ for a normalized energy map. The choice of the parameter α is dependent on the image size. We set it to 3% of the image width in our experiments.

Seams are calculated in a manner similar to finding the optimal path of the cropping window, as described earlier. To calculate the k-th vertical seam in frame t, the modified energy map $(E + C_k^t)$ is traversed from top to bottom (in the direction of index j). The *cumulative minimum energy* for each pixel position is calculated by adding the energy of the current pixel to the minimum of the cumulative energy values of the three adjacent pixels above (similar to Equation 4.7).

The positions of all seams s_k^t are stored and are all removed in one step after an additional condition is checked. If a position of a previous seam $s_k^{t-1}(j)$ lies outside the extended window, all costs in row j are set to the upper bound: $C_k^t(i,j) = \beta, \forall i$. When more than a certain percentage of the seam lies outside the extended window, no temporal costs are added to E so that a new seam can be found. We found that setting this value to 20% gives good results. With this restriction, the seams may first be deferred a few frames so that more points of the seam lie at the border before they disappear, which is visually less disturbing in the result.

4.3.3 Evaluation

We conducted an evaluation in order to compare our algorithm to the similar technique of *multi-operator retargeting* [70]. The evaluation was a no-reference comparison, like in previous Sections (3.3.6 and 4.1.2). As test sequences, twelve videos belonging to different categories like animations, movies or sports were used. These videos are available on the Web[4].

The evaluation was conducted online on a web site for scientific surveys[5] and consisted of twelve comparisons and five questions at the end. In each comparison, a video was shown which presented the retargeted results side by side. The side on which our results appeared was chosen randomly for each video. For each comparison the participants were asked which result they prefer or if they could not find any visual differences. The last page was meant to get feedback on the decisions the subjects made while rating the videos. It was asked if their decisions were influenced by squeezing artifacts, visible deformations in general, cut off objects or abnormal camera motion. Lastly, there was an optional open question about the reasons for their rating.

A total of 19 participants took part in the evaluation. 13 were students, 6 were colleagues from our department. A total of 228 video comparisons were thus evaluated, and only fully executed surveys were utilized.

Analysis and Discussion

The survey shows that none of the techniques is superior in all of the test sequences, each has its strong and weak points (see Figure 4.14). SeamCrop is preferred where persons or objects that appear large on the screen are squeezed, e.g., in a video where a train with visible passengers crosses the screen or a sequence with football players. Multi-operator retargeting is favored in sequences with many moving objects or with structured backgrounds, e.g., a small person walking between buildings with a structured facade.

This coincides with the answers the participants gave at the end of the survey. The most disturbing artifacts are deformations of relevant objects, followed by cropped objects. Also, the squeezing of persons is explicitly stated in the optional open question about the reasons of their voting. Some participants additionally noticed unstable backgrounds and did not like camera pans from one side to another and then vice versa in one scene. Figure 4.15 shows some examples of the videos used in the survey. The original frame and a symmetrically

[4] `http://ls.wim.uni-mannheim.de/de/pi4/research/projekte/retargeting/`
[5] `http://www.soscisurvey.de`

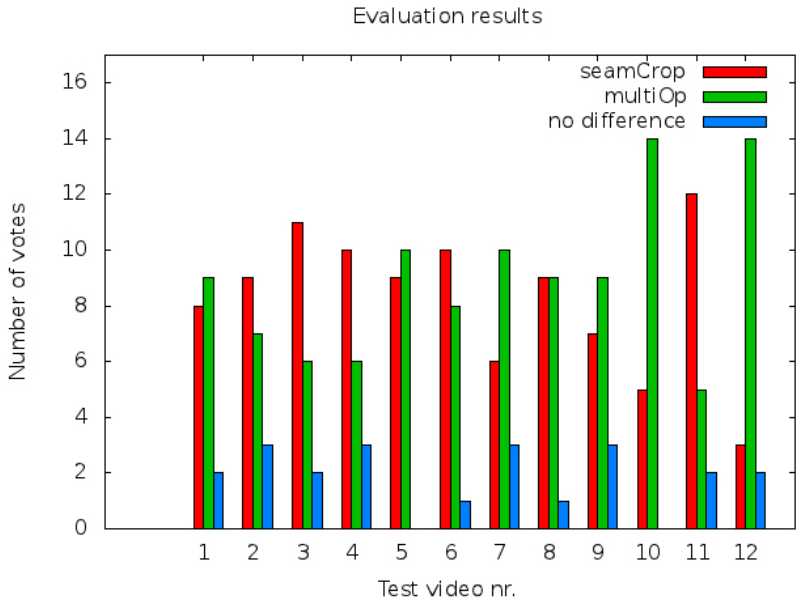


Figure 4.14: *Number of votes for each method in all test sequences of the evaluation.*

scaled down version are added for better comparison. It can be clearly seen that some objects like persons or cars get squeezed by the scaling operators of the multi-operator algorithm.

Many current state-of-the-art video resizing techniques solve optimization problems on the entire video cube, which takes a lot of processing time. For example, the resizing of a video sequence (400×300 pixels, 400 frames) to 50% of the original width takes about 10 to 20 minutes with seam carving based on graph cuts [68]. Similarly, multi-operator retargeting has average optimization times of 10 minutes for one key frame with resolutions between 600×400 and 400×300 pixels. In contrast, our new approach is very fast as it only uses information of the entire video to calculate the optimal cropping path as a 2D problem, and it searches for seams in each frame individually. The performance values of the following Table 4.3 were measured on a Intel Core 2 Quad desktop with 2.4 GHz and 4 GB memory. Please note that our algorithm runs on one (sequential) processor[6].

	400×300 400 Frames	720×432 261 Frames	1920×1080 72 Frames
Processing time	1 min 6 sec	3 min 21 sec	18 min 27 sec

Table 4.3: *Processing time of SeamCrop for a single test video in three different screen resolutions. All sequences were retargeted to 50% of the original width.*

[6]We will discuss a GPU-based parallel version in the following section.

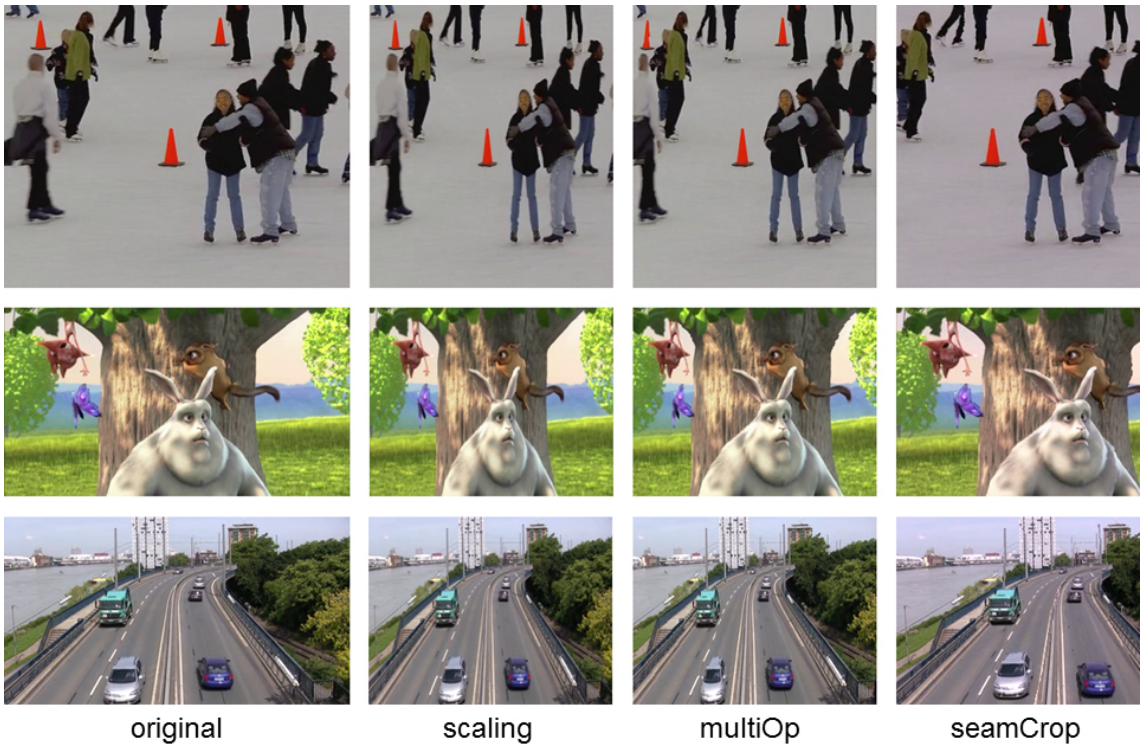


Figure 4.15: *Example frames from the evaluation. The original frame and an asymmetrically scaled version are added for better comparison of the results.*

Limitations

SeamCrop produces clearly visible distortions in some of the shots. Like all resizing techniques, the algorithm depends on the accuracy of the visual importance function. If the cropping window is positioned in the wrong spot, relevant persons or objects might get cut. Also, even though only a small percentage of the resizing is done with seams, fast moving objects may cross their path before they can jump to a new position. Additionally, artifacts may be introduced if the important content takes up the entire screen.

4.4 SeamCrop for Videos using the GPU

The SeamCrop version presented in the last section is already very efficient, using a 2D optimization for the search of the optimal cropping window and doing seam carving on a frame-by-frame basis. However, it is still not fast enough for a popular retargeting use case, the adaptation of streaming video in real-time. Therefore, to further enhance the processing speed, we want to make an implementation of our algorithm on the GPU using *CUDA* [7], a parallel

[7] `https://developer.nvidia.com/what-cuda`

programming language by NVIDIA specifically designed for this task. Some changes have to be made to the algorithm in order to parallelize more tasks and optimize the performance. These changes and some insights of our CUDA implementation are presented in detail in the following. If not stated otherwise, we assume that all steps to be executed on the GPU in order to save copy operations between CPU and GPU.

4.4.1 System Overview

The basic workflow of the algorithm remains the same: two passes are done on the video, the first to find a cropping window over the entire sequence and the second to extend the cropping window and search for seams to be carved out frame by frame.

In both passes, frames are treated in multiple CPU-threads. These threads are responsible for loading the frames to the CPU memory, copying them from CPU to GPU and removing them from the GPU when they are not needed anymore. Also, they organize the workflow on the GPU as they start kernels for the frames and watch the synchronization between the frames, for example that a frame has to wait in the seam carving step until enough seams are found in the previous frame because of the temporal coherence costs. By using multiple CPU-threads and assigning a different CUDA stream to each thread, multiple kernels can be executed on the GPU in parallel. This ensures that the GPU utilization is high throughout the runtime of the algorithm.

On the GPU, frames are usually divided into regions that are computed independently and in parallel, the so-called *blocks*. Each block consists of several threads that have a shared memory. As CUDA has no mechanism for synchronizing blocks, we implemented one ourselves. This is important because most of the steps of the algorithm require that the steps before are finished. For instance, when finding the optimal seam paths the result of each row depends on the previous row. The block synchronization function uses CUDA's thread synchronization to ensure that all threads of a given block enter and leave the block synchronization function at the same time. The first thread of each block is chosen as a representative of the block. This representative uses atomic functions to increase a variable and perform a busy wait on it. Once all blocks have reached this point the representatives will end their busy wait, and normal operation resumes. As an alternative, kernel launches could have been used for synchronization; but doing so experimentally decreased the performance of the program.

4.4.2 Importance Function

As the pixels within a frame are independent of each other during the importance calculation, this step of the algorithm is highly parallelizable. There are several possibilities to determine the importance values, like a saliency map or a histogram of gradients that can be used. We chose to stay with the simple gradient function from [39], as it provides sufficient information for visually good results.

Also, we keep the simple motion saliency function. As this function is calculated dependent on the previous and the succeeding frame, the thread of the frame has to wait until these other two frames are also loaded into memory. The resulting motion saliency map is smoothed with a Gaussian function. We implemented a horizontal smoothing kernel and transposed it for the vertical direction. This way, only one implementation is necessary, and there are no conflicts with the memory banks of the GPU that would occur in an additional vertical kernel.

After the importance map is calculated, the frame is dropped from memory, and the thread can be assigned to another frame. A major difference is the drop of the importance information in the first pass after the necessary data for the calculation of the cropping window has been gained. This is due to the possibility of parallelizing this step efficently so that it is fast to recalculate the importance function. This allows to process videos with higher resolutions, as the algorithm needs less memory compared to an implementation that keeps all frames and importance maps in memory.

4.4.3 Cropping Window

In the calculation of the cropping window path, there are some dependencies that have to be considered in the parallelization. For instance, the columns have to be summed up before the costs of each cropping window position can be calculated. The columns themselves are independent, all the sums can thus be calculated at the same time.

In the next step, the cropping path is searched via dynamic programming in a 2D array similar to seam carving [68]. This leads to a dependency between the rows, as each position in a row depends on the values of its potential predecessors in the line above (see Figure 4.16). Therefore, rows are done one after another while the positions in a row can be calculated inependently. When the costs of the path have been summed up, the optimal cropping path can be found by backtracking from the cheapest value in the last row, which is not very complex. Everything except the calculated cropping path is then dropped from the memory before the

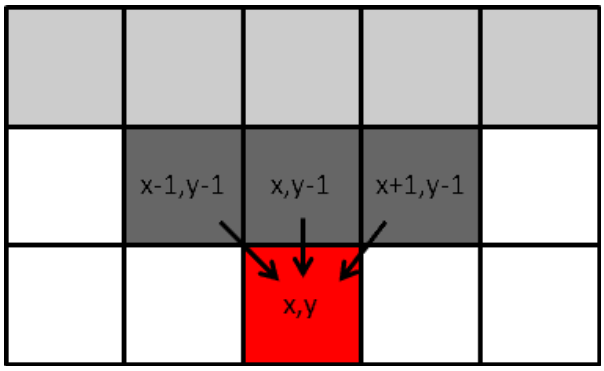


Figure 4.16: *Example for the pixels depending on each other during the search for a seam. The cost of the current pixel (red) is the addition of its importance value with the cost of the cheapest predecessor in the line above (dark grey). Only the three adjacent pixels are taken into account. These themselves depend on the values of the line above (light grey).*

second pass starts. The smoothing of the path is done on the CPU as it is not a complex operation. It is the only operation in our implementation that is not computed on the GPU.

4.4.4 Seam Carving

Seam Carving is done in the second pass of the algorithm. As mentioned above, the seams are now computed frame-by-frame instead of an optimization over all frames like in the cropping step. Each seam depends on the seams that have been previously calculated in the current frame t, as well as the corresponding seam from the frame $t-1$. This means that only one seam is searched in each frame at any time, although the frames are done in parallel. Because the algorithm uses information of the seams from the previous frame $t-1$ via temporal coherence costs, each thread waits until the thread of frame $t-1$ has found enough seams. For instance, if seam i is to be calculated, the thread t waits until thread $t-1$ has found at least seam $i+1$ before it starts with it. As a lookup table, an array is used to keep track of the number of seams that have been found in all frames. If a thread has to wait because not enough seams are found in frame $t-1$, the time is given to other threads.

Like mentioned before, the rows are computed one after another in the dynamic programming step where the cheapest seam is searched (see Figure 4.16). For the calculation, each column is assigned its own thread so that each pixel in a row can be computed independently. Because the rows depend on each other, each one has to wait until the previous one is finished. After this computation, the cheapest value in the last row has to be found because it marks the

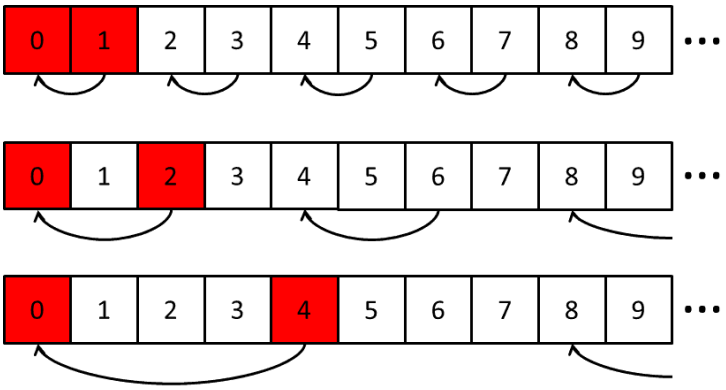


Figure 4.17: *Search for the minimum value in the last row of a frame. Top row of the Figure: each second thread checks if its neighbor has a lower value, and copies it and its position in this case (i.e., 0 compares to 1 (red), 2 to 3,..). Middle row: the same is done with each fourth thread and the thread two IDs before (i.e., 0 compares to 2 (red), 4 to 6, ...). This is repeated until the first thread has the optimal value of the block and its position.*

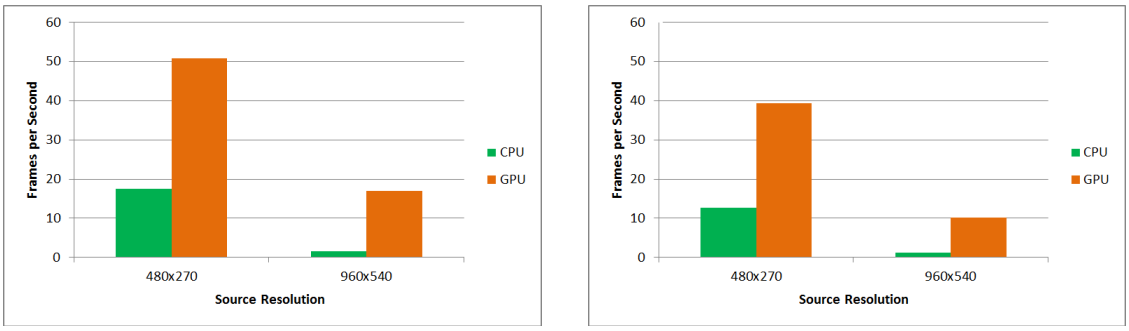
end of the optimal seam. This is done differently than in the sequential approach, where the minimum is found by just going through all of them from left to right.

Instead, all threads of a block are used to find the minimum. First, each thread copies its importance value and its position into the shared memory. Then, every other thread checks if its neighbor has a better value and copies it and its position (see Figure 4.17). After that, every fourth thread checks if the thread two IDs after it has a better value. This goes on until the optimal value and its position have been copied all the way to the first thread of each block. Lastly, each block writes its best value and position into the global memory, and the first thread of the first block determines the global minimum and more importantly the position of the global minimum.

In order to save computation time, the importance map is only calculated once and then modified by each found seam in a way that the following seams will not pick the same pixels. This is done by setting the visual importance values of the used pixels to really high values.

4.4.5 Evaluation

We conducted a detailed performance test in order to evaluate the efficiency of the GPU implementation. In the performance test, we compared GPU against CPU, measured how long pass one and two of the algorithm took, and tested different parameters. As we did not fundamentally change any of the core principles of the SeamCrop algorithm [39], the quality of visual results remains the same.



(a) Retarget Factor 25%

(b) Retarget Factor 50%

Figure 4.18: *Performance comparison between the CPU and the GPU version of the algorithm.*

All the tests were performed on a PC with the following specifications: Intel i7-3770 processor with four cores at 3.4 GHz, 16 GB DDR3 RAM and a NVIDIA GeForce GTX 650 TI with 1024 MB memory and 768 CUDA cores. In the implementation, eight CPU threads are used as the PC has four cores and is able to support two threads per core. The factor for the extended borders is set to 20% (see Section 4.3.2), and only the width is reduced in all tests while other parameters vary. Also, except for GPU test 4, all sequences have 183 frames for a better comparision of the results.

We first did a performance comparison between the already efficient CPU implementation and our new GPU version of the algorithm (see Figure 4.18). The tests where done on a 480×270 and a 960×540 sequence with the *retarget factors* (RF) 25% and 50%. We took those factors from a comparative study on image retargeting by Rubinstein et al. [69], where they are regarded as a considerable resizing. In the 480×270 sequence, the GPU version is about 3 times as fast as the CPU one. The factor gets higher for the 960×540 sequence as the parallel processes come more into effect. There, the parallel implementation is 10.5 times faster for a retarget factor of 25% and 8 times faster for 50%.

In addition to the comparison with the CPU version, we also did detailed performance tests of the GPU implementation. There are four tests with varying parameters that are presented in the following.

In our first GPU test case, we measured how long the algorithm took to reduce the size of a video by 25% and by 50% (see Figure 4.19).

For the reduction, six sequences are used in four different resolutions (1920×1080, 1440×810, 960×540 and 480×270). For the 480×270 sequence, the gap when reducing the RF from 25% to 50% is small. This is caused by the computation overhead of steps that have to be done regardless of the target size. The gap between the RFs gets even smaller with

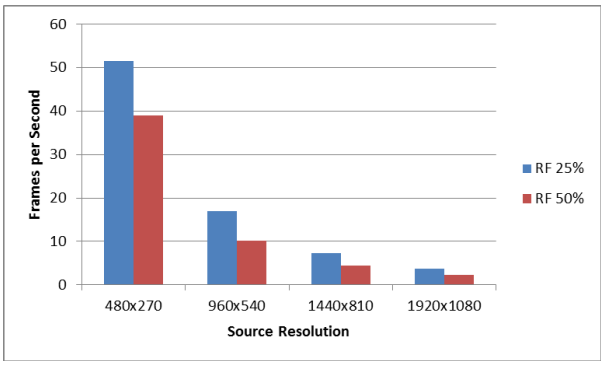


Figure 4.19: *GPU test 1: Frames per second on four different resolutions and two retargeting factors (RF). For the Figure, the results from the six used sequences are averaged.*

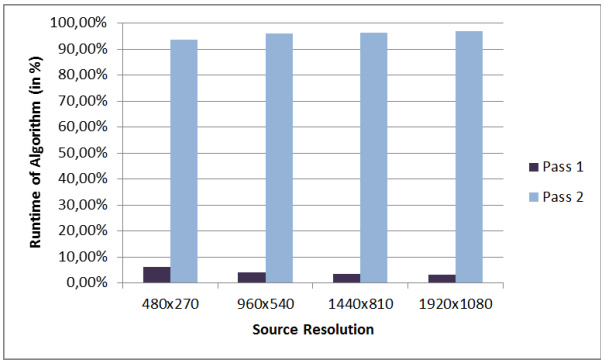


Figure 4.20: *Percentage that the two passes take in relation to the pure processing time (without loading and saving the frames) on the GPU. For this figure, the results for both retarget factors are averaged as the percentages are nearly identical.*

increasing resolutions as the capacity of the GPU is fully used, and the framerate drops in general. When analyzing the percentage load of each pass, there are clear differences (see Figure 4.20). The first pass is less complex and has more potential to be parallelized than the second pass. Especially the seam carving in pass two with its need to synchronize row after row takes time. Nonetheless, the GPU version is considerably faster than the CPU version.

GPU test case 2 uses a 960×540 sequence and varies the RF width from 10% to 50% (see Figure 4.21). The decrease in fps over the increasing RF results from the increased number of possible cropping window positions per frame, as well as a larger number of seams that have to be computed. It is not a linear progression because the use of the GPU becomes more efficient with higher RFs as there are more computations that can be done in parallel.

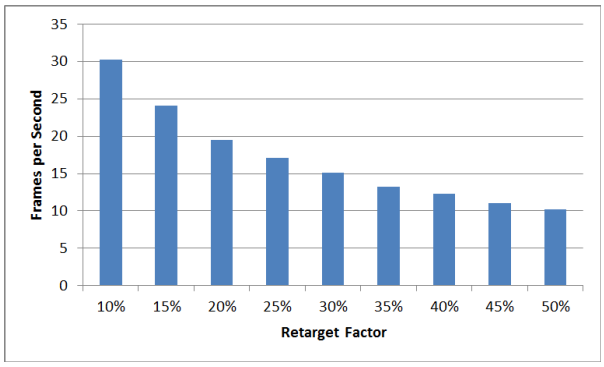


Figure 4.21: *GPU test 2: Frames per second of a* 960×540 *sequence with a varying retarget factor.*

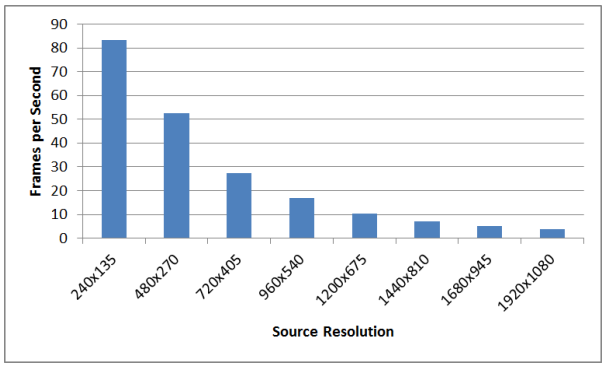


Figure 4.22: *GPU test 3: Frames per second with increasing frame resolutions and a retarget factor of* 25%.

In GPU test case 3, we retarget a sequence by 25% but vary the resolution of the source video (240×135, 480×270, 720×405, 960×540, 1200×675, 1440×810, 1680×945, and 1920×1080). Like in the previous test, the progression is also non-linear (see Figure 4.22). This has to do with two factors: First, not all steps in the Figure have the same amount of pixel increase. For instance, 480×270 has four times the pixels than 240×135, but 720×405 has only 2.25 times the pixels than 480×270. Second, the GPU is not working to full capacity in the beginning with the low resolutions and is later more efficient in parallelizing at the high resolutions. Assuming 25 frames per second and always keeping a shot in the buffer, our algorithm is able to achieve real-time retargeting up to a resolution of 720×405 pixels.

Lastly, in our fourth GPU test case, a 960×540 sequence is shrunk by 25% with a varying number of frames (183, 244, 305, 366, 427 and 488). As expected, the fps stays at the same amount (in this case seventeen) no matter how many frames the sequence has (see Figure 4.23).

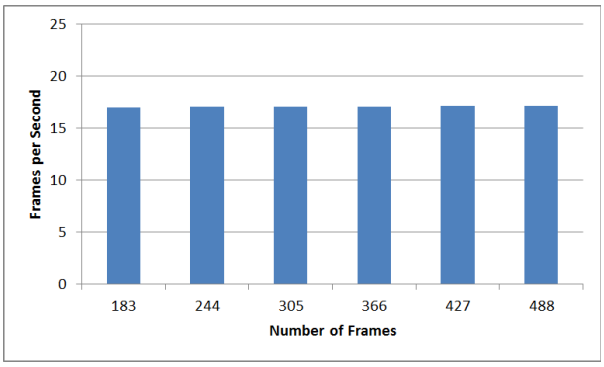


Figure 4.23: *GPU test 4: The number of frames of the video sequence (*960×540*, retarget factor* 25%*) does not affect the frames per second.*

This is because the processing time of a frame does not depend on the number of frames that follow .

CHAPTER 5

Conclusions and Future Work

In this chapter, the dissertation is concluded, and an outlook on future work is given.

5.1 Conclusions

With the increasing popularity of smartphones and similar mobile devices, the demand for media to consume on the go rises. As most images and videos today are captured with HD or even higher resolutions, there is a need to adapt them in a content-aware fashion before they can be watched comfortably on screens with small sizes and varying aspect ratios. This process is called *retargeting*. Most distortions during this process are caused by a change of the aspect ratio. Thus, retargeting mainly focuses on adapting the aspect ratio of a video while the rest can be scaled uniformly. An example would be viewing a widescreen movie created in $16:9$ on a device with a $4:3$ display. The movie can be scaled down to fit the height without loosing important details but the borders have to be cropped in order to fit the width.

Although the retargeting of media data is currently a very popular research area, there are still unsolved questions and limitations. Back when this dissertation project was started, there were even more research questions unanswered. This dissertation contributes to the media retargeting communtity by answering several of these questions. In the following, the contributions of this dissertation are described in detail:

Seam carving generally achieves a high adaptation quality for images depicting scenes with homogenous backgrounds like landscapes, and the distortions caused by the removal of

seams are very low compared to other methods like cropping and scaling. However, if there are straight lines or structured backgrounds in an image, the intersection points between the seams and the lines create visually disturbing artifacts. In order to prevent these distortions, we introduced an enhanced seam carving for images that puts the focus on the preservation of straight lines. When a seam crosses a straight line, adjacent importance values are increased in order to prevent the following seams from crossing the line nearby. The distribution of the seams preserves straight lines much better, and less distortion is introduced in the adapted image. Compared to the original seam carving, our method achieves significantly better results when used on images with prominent straight lines or structures.

The seam carving approach for *videos* presented by the authors of the original paper [68] is computationally very complex. They use the graph cut algorithm for a global optimization over all frames which takes a lot of processing time and is not applicable to HD 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 reasonable amounts of memory. To overcome this problem, we presented a new algorithm called FSCAV that also takes all frames into account but tackles the retargeting problem with less complexity. FSCAV uses image registration to create a so-called background image. On this image, seams are searched and then tracked back to the individual frames. Through this technique, FSCAV has a fast processing time and does not need much memory for the computation. In contrast to the previoulsy mentioned graph-cut-based algorithm by the original authors, our technique is able to retarget videos with HD resolution in a reasonable time. User evaluations indicated the high quality of the adapted videos. The visual quality of adapted videos based on FSCAV and on graph cut is very similar. The image background of video sequences is more stable in case of FSCAV whereas small foreground objects or objects in slow motion are better preserved when graph cut is used.

The research area of retargeting is well explored for 2D images and videos. This is not the case for stereoscopic content. While there are algorithms for the automatic resizing of stereoscopic images [86, 3], to our knowledge there are no approaches for video yet that go beyond cropping or linear scaling. The difficulty of adapting this kind of content is to achieve consistency between the left and the right view in order to preserve the 3D effect, as well as temporal consistency between the frames to avoid flickering artifacts. In our novel approach, we used seam carving as a basis and took forward energy in the left and right view as well as the disparity map into account. Additionally, the algorithm calculates energy from depth and adds temporal consistency to the seams. Our evaluation showed that temporal consistency

is an important criterion when applying stereo seam carving to video. Its absence leads to flicker and strongly decreases the perceived video quality. Subjectively, the 3D effect was not impaired by seam carving. We believe that this effect may be too subtle to notice in a complex video scene.

As mentioned before, there are types of images where seam carving does not produce visually pleasing results. For instance, if there are numerous objects on the screen, it is difficult for the seams to avoid cutting through them. Other retargeting operators like cropping also have their problems with certain types of content. In order to overcome these limitations, we introduced a combination of seam carving and cropping called *SeamCrop*. The goal of the algorithm is to resize an image without manipulating the important objects in an image too badly. Therefore, SeamCrop switches between seam carving and cropping based on a dynamic threshold. The process is repeated iteratively until the target size is reached. Our results show that by using this method, more important content of an image can be included in the result than in normal cropping. The ”squeezing” of objects which might occur in approaches based on warping or scaling is also prevented.

The identification of relevant objects in an image, the visual attention analysis, is highly relevant in the context of image retargeting. Especially faces draw the attention of viewers. But the level of relevance may be different for different faces depending on the size, the location, or whether a face is in focus or not. For instance, the detection of unimportant faces in the background may lead to visual distortions of really important objects when used in an image retargeting algorithm. Therefore, we presented a novel technique for distinguishing between faces that are in focus and those out of focus. Faces are first detected with the use of multiple cascades. As faces with strong edges are assumed to be in focus, we use the gradient magnitude to classify the faces to be in or out of focus. GrabCut [67] is finally used to segment the faces and create face masks which can then be used as an additional input to the image retargeting algorithms. Our evaluation showed that the new algorithm is reliable, and most of the faces are correctly assigned as in focus or out of focus. Additionally, we selected SeamCrop as our application scenario and demonstrated that the quality of the results could be further enhanced with the improved face detection.

Operators in video retargeting share similar limitations as the ones used to resize images. Therefore, other authors also suggest the combination of multiple operators for this task. However, most of these algorithms use an optimization over the entire video sequence which is very time consuming. In order to be computationally more efficient, we introduced SeamCrop for the retargeting of videos. The main idea of the algorithm is to find an optimal path for the crop-

ping window that reduces the width or height and then use seam carving to get more useful content into it without creating visible artifacts. In contrast to other multi-operator retargeting approaches, for instance by Rubinstein et al. [70], we do not use non-homogeneous scaling in our approach as it may introduce squeezing artifacts to faces and objects. A user study was conducted that compares our approach to a similar video retargeting technique, showing that our new algorithm has faster computation times while providing a comparable quality of the results.

Video streams transmitted over the Internet are a popular use case for retargeting algorithms. Especially the resizing in real-time is thus a very challenging problem. In order to achieve this goal, the GPU has considerable potential for saving processing time. Currently, the best published algorithms achieve real-time only in sequences with low resolutions up to 480×320 pixels. To achieve real-time performance with higher resolutions, we presented an accelerated version of our SeamCrop approach for videos using a CUDA implementation. It uses parallel processes to significantly enhance the performance compared to the original implementation. The differences and the adjustements between the two versions were thoroughly discussed, and measurements of the efficiency are shown in a detailed performance test. In comparison to the already efficient CPU implementation, the computation time of our algorithm is 10.5 times faster (on a 960×540 video with a retarget factor of 25%). Also, our algorithm is able to retarget videos with a resolution up to 720×405 pixels in real-time (assuming 25 frames per second).

5.2 Future Work

The retargeting of images and videos is still a valid research area even though a lot of work has already been done in this field. Current mobile devices are getting displays with resolutions comparable to modern televisions, but the physical size of the displays does not change. This means that images and videos can theoretically be viewed in their natural resolution, but the details of the scenes will most likely become to small to be recognized by the viewer. Therefore, there remains a need for intelligent and fast retargeting algorithms in the future.

All current retargeting methods - multi-operator or not - have in common that there is no technique that is able to retarget all different kinds of images or videos with the same quality. Each method has its shortcomings in certain situations. A possible solution to this problem would be a framework that is built around the best methods for each category of media and then chooses dynamically which operator or combination of operators to use depending on

the content. This could either be done by using tools to analyse the sequence beforehand and choosing the best technique, or by using a method to automatically measure the quality of possible results produced by the operators. An automatic quality measure algorithm would also be helpful in other situations. Currently, new retargeting algorithms get evaluated in user studies to compare the results with state-of-the-art algorithms. Although there have been some promising approaches for automatic evaluations [69], there is still no accurate and and widely accepted method.

Another interesting idea would be to extend the face focus detection to derive depth information from 2D images or videos. As we mentioned before, the quality of the results of a retargeting algorithm depends a lot on the accuracy of the visual importance map. Like in the face detection, areas in images could be marked as not important because they are out of focus and therefore in the background. Also, this information could be used for a more accurate object detection and object segmentation.

With the increasing distribution of stereoscopic cameras and viewing devices, the retargeting of stereoscopic media has recently become very popular. As discussed in Chapter 3.3, a key requirement for stereo retargeting is the synchronization between the left and the right view, which is done by using a disparity map. In addition to the importance map, stereoscopic retargeting algorithms heavily depend on an accurate disparity map in order to get a pleasing result. There is still a lot of of potential to make the disparity maps more accurate and robust. This is especially tricky when used in seam carving as the disparity algorithm commonly favors an opposite image composition: seam carving works best on images or videos with unstructured and homogenouos areas where seams can be removed without going through objects or edges. The disparity algorithm wants structured scenes with a lot of contrast in them in order to find corresponding points in both views.

References

[1] G. Abdollahian, Z. Pizlo, and E. Delp. A study on the effect of camera motion on human visual attention. In *IEEE International Conference on Image Processing (ICIP)*, pages 693 –696, 12-15 2008.

[2] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. *ACM Transactions on Graphics, SIGGRAPH*, 26(3), 2007.

[3] T. Basha, Y. Moses, and S. Avidan. Geometrically consistent stereo seam carving. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1816–1823, 2011.

[4] M. Begum, F. Karray, G. K. I. Mann, and R. Gosine. A probabilistic model of overt visual attention for cognitive robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 40(5):1305–1318, 2010.

[5] A. Bovik. *Handbook of Image and Video Processing*. Elsevier Academic Press, 2. edition, 2005.

[6] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI)*, volume 26(9), pages 1124–1137, 2004.

[7] J. F. Canny. A computational approach to edge detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, volume 8(6), pages 679–698, 1986.

[8] A. Carlier, V. Charvillat, W. T. Ooi, R. Grigoras, and G. Morin. Crowdsourced automatic zoom and scroll for video retargeting. In *ACM Proceedings of the international conference on Multimedia (MM)*, pages 201–210, 2010.

[9] R.-M. Chao, H.-C. Wu, and Z.-C. Chen. Image segmentation by automatic histogram thresholding. In *International Conference on Interaction Sciences: Information Technology, Culture and Human (ICIS)*, pages 136–141, 2009.

[10] D.-Y. Chen, H.-R. Tyan, S.-W. Shih, and H.-Y. Liao. Dynamic visual saliency modeling for video semantics. In *International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP)*, pages 188 –191, 2008.

[11] W.-H. Cheng, C.-W. Hsieh, S.-K. Lin, C.-W. Wang, and J.-L. Wu. Robust algorithm for exemplar-based image inpainting. In *The International Conference on Computer Graphics, Imaging and Vision (CGIV)*, pages 64 – 69, 2005.

[12] W.-H. Cheng, C.-W. Wang, and J.-L. Wu. Video adaptation for small display based on content recomposition. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(1):43 –58, 2007.

[13] C.-K. Chiang, S.-F. Wang, Y.-L. Chen, and S.-H. Lai. Fast jnd-based video carving with gpu acceleration for real-time video retargeting. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(11):1588–1597, 2009.

[14] D. Comaniciu and P. Meer. Mean shift: A robust approach towards feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24:603–619, 2002.

[15] W. Dong, G. Bao, X. Zhang, and J.-C. Paul. Fast multi-operator image resizing and evaluation. *Journal of Computer Science and Technology*, 27(1):121–134, 2012.

[16] W. Dong and J.-C. Paul. Adaptive content aware image resizing. *Eurographics 2009*, 28(2), 2008.

[17] W. Dong, N. Zhou, J.-C. Paul, and X. Zhang. Optimized image resizing using seam carving and scaling. *ACM Transactions on Graphics, SIGGRAPH*, 28(5):1–10, 2009.

[18] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[19] D. Farin. *Automatic Video Segmentation Employing Object/Camera Modeling*. PhD thesis, Technische Universiteit Eindhoven, Einhoven, The Netherlands, 2005.

[20] D. Farin, T. Haenselmann, S. Kopf, G. Kühne, and W. Effelsberg. Segmentation and classification of moving video objects. In *Handbook of Video Databases: Design and Applications*, volume 8 of *Internet and Communications Series*, pages 561–591. CRC Press, 2003.

[21] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[22] S. Frintrop, E. Rome, and H. I. Christensen. Computational visual attention systems and their cognitive foundations: A survey. *ACM Transactions on Applied Perception (TAP)*, 7(1):1–39, 2010.

[23] R. Gal, O. Sorkine, and D. Cohen-Or. Feature-aware texturing. In *Proceedings of Eurographics Symposium on Rendering*, pages 297–303, 2006.

[24] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Discontinuous seam-carving for video retargeting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 569 –576, 2010.

[25] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2141–2148, 2010.

[26] C. Guo, Q. Ma, and L. Zhang. Spatio-temporal saliency detection using phase spectrum of quaternion fourier transform. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[27] Y. Guo, F. Liu, Z.-H. Zhou, and M. Gleicher. Image retargeting using mesh parameterization. *IEEE Transactions on Multimedia*, 11(5):856–867, 2009.

[28] B. Guthier, J. Kiess, S. Kopf, and W. Effelsberg. Seam carving for stereoscopic video. In *IEEE Workshop: 3D Image/Video Technologies and Applications (IVSMP)*, pages 1–4, 2013.

[29] D. Han, M. Sonka, J. E. Bayouth, and X. Wu. Optimal multiple-seams search for image resizing with smoothness and shape prior. *The Visual Computer*, 26(6-8):749–759, 2010.

[30] D. Han, X. Wu, and M. Sonka. Optimal multiple surfaces searching for video/image resizing - a graph-theoretic approach. In *IEEE Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1026 –1033. IEEE Press, 2009.

[31] J.-W. Han, K.-S. Choi, T.-S. Wang, S.-H. Cheon, and S.-J. Ko. Improved seam carving using a modified energy function based on wavelet decomposition. In *IEEE International Symposium on Consumer Electronics (ISCE)*, pages 38 –41, 2009.

[32] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, volume 15, pages 147–151, 1988.

[33] P. Harrison. A non-hierarchical procedure for re-synthesis of complex textures. In *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 190–197, 2001.

[34] H. Hirschmüller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVRP)*, pages 807–814, 2005.

[35] D.-S. Hwang and S.-Y. Chien. Content-aware image resizing using perceptual seam carving with human attention model. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 1029–1032, 2008.

[36] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 20(11):1254–1259, 1999.

[37] J. Kiess, R. Garcia, S. Kopf, and W. Effelsberg. Improved image retargeting by distinguishing between faces in focus and out of focus. In *IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 145–150, 2012.

[38] J. Kiess, D. Gritzner, B. Guthier, S. Kopf, and W. Effelsberg. GPU video retargeting with parallelized SeamCrop. In *ACM Multimedia Systems (MMSys)*, 2014 (To appear).

[39] J. Kiess, B. Guthier, S. Kopf, and W. Effelsberg. SeamCrop: Changing the size and aspect ratio of videos. In *ACM Workshop on Mobile Video (MoVid)*, pages 13–18, 2012.

[40] J. Kiess, B. Guthier, S. Kopf, and W. Effelsberg. SeamCrop for image retargeting. In *IS&T/SPIE Multimedia on Mobile Devices*, volume 8304, 2012.

[41] J. Kiess, S. Kopf, B. Guthier, and W. Effelsberg. Seam carving with improved edge preservation. In *IS&T/SPIE Multimedia on Mobile Devices*, volume 7542, pages 75420G–75430G, 2010.

[42] J.-S. Kim, J.-H. Kim, and C.-S. Kim. Adaptive image and video retargeting technique based on fourier analysis. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1730–1737, 2009.

[43] S. Kopf, B. Guthier, H. Lemelson, and W. Effelsberg. Adaptation of web pages and images for mobile applications. In *IS&T/SPIE Multimedia on Mobile Devices*, volume 7256, pages 72560C:01 – 72560C:12, 2009.

[44] S. Kopf, J. Kiess, H. Lemelson, and W. Effelsberg. FSCAV: Fast seam carving for size adaptation of videos. In *ACM International Conference on Multimedia (MM)*, pages 321–330, 2009.

[45] P. Krähenbühl, M. Lang, A. Hornung, and M. Gross. A system for retargeting of streaming video. *ACM Transactions on Graphics, SIGGRAPH*, 28(5):1–10, 2009.

[46] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH*, 22:277–286, 2003.

[47] Y. Li, Y.-F. Ma, and H.-J. Zhang. Salient region detection and tracking in video. In *IEEE International Conference on Multimedia and Expo (ICME)*, volume 2, pages II – 269–72 vol.2, 2003.

[48] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Transactions on Graphics, SIGGRAPH*, 23(3):303–308, 2004.

[49] Y. Li, Y. Tian, J. Yang, L.-Y. Duan, and W. Gao. Video retargeting with multi-scale trajectory optimization. In *ACM International Conference on Multimedia Information Retrieval (MIR)*, pages 45–54, New York, NY, USA, 2010. ACM.

[50] S. Lin, C. Lin, I. Yeh, S. Chang, C. Yeh, and T. Lee. Content-aware video retargeting using object-preserving warping. *IEEE Transactions on Visualization and Computer Graphics*, 19(10):1677–1686, 2013.

[51] F. Liu and M. Gleicher. Automatic image retargeting with fisheye-view warping. In *ACM Symposium on User Interface Software and Technology*, pages 153–162, 2003.

[52] F. Liu and M. Gleicher. Video retargeting: Automating pan and scan. *ACM international conference on Multimedia (MM)*, pages 241–250, 2006.

[53] H. Liu, S. Jiang, Q. Huang, C. Xu, and W. Gao. Region-based visual attention analysis with its application in image browsing on small displays. In *ACM International Conference on Multimedia (MM)*, pages 305–308, 2007.

[54] H. Liu, X. Xie, W.-Y. Ma, and H.-J. Zhang. Automatic browsing of large pictures on mobile devices. In *ACM international conference on Multimedia (MM)*, pages 148–155, 2003.

[55] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[56] Z. Lu, W. Lin, X. Yang, E. Ong, and S. Yao. Modeling visual attention's modulatory aftereffects on visual sensitivity and quality evaluation. *IEEE Transactions on Image Processing*, 14(11):1928 –1942, 2005.

[57] Y.-F. Ma and H.-J. Zhang. Contrast-based image attention analysis by using fuzzy growing. In *ACM international conference on Multimedia (MM)*, pages 374–381, 2003.

[58] B. S. Manjunath, P. Salembier, and T. Sikora. *Introduction to MPEG-7: Multimedia Content Description Interface*. John Wiley and Sons, Ltd., 2002.

[59] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference*, pages 384–393, 2002.

[60] H. Moravec. Visual mapping by a robot rover. In *International Joint Conference on Artificial Intelligence*, pages 599 – 601, 1979.

[61] H. Noh and B. Han. Seam carving with forward gradient difference maps. In *ACM International Conference on Multimedia (MM)*, pages 709–712, 2012.

[62] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*, chapter The Max-Flow, Min-Cut Theorem, page 120–128. Dover, 1998.

[63] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, 1992.

[64] Y. Pritch, E. Kav-Venaki, and S. Peleg. Shift-map image editing. In *IEEE International Conference on Computer Vision (ICCV)*, pages 151–158, 2009.

[65] T. Ren, Y. Liu, and G. Wu. Image retargeting based on global energy optimization. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 406–409, 2009.

[66] T. Ren, Y. Liu, and G. Wu. Image retargeting using multi-map constrained region warping. In *ACM international conference on Multimedia (MM)*, pages 853–856, 2009.

[67] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH 2004 Papers*, pages 309–314, 2004.

[68] M. Rubinstein, S. Avidan, and A. Shamir. Improved seam carving for video retargeting. *ACM Transactions on Graphics, SIGGRAPH*, 27(3), 2008.

[69] M. Rubinstein, D. Gutierrez, O. Sorkine, and A. Shamir. A comparative study of image retargeting. In *ACM Transactions on Graphics, SIGGRAPH ASIA*, pages 160:1–160:10, 2010.

[70] M. Rubinstein, A. Shamir, and S. Avidan. Multi-operator media retargeting. *ACM Transactions on Graphics, SIGGRAPH*, 28(3):1–11, 2009.

[71] M. A. Ruzon and C. Tomasi. Alpha estimation in natural images. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:18–25, 2000.

[72] H. Sakoe. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978.

[73] A. Santella, M. Agrawala, D. DeCarlo, D. Salesin, and M. Cohen. Gaze-based interaction for semi-automatic photo cropping. *ACM Conference on Human Factors in Computing Systems (CHI)*, pages 771–780, 2006.

[74] H. Schneiderman. Face detection demonstration. Technical report, Robotics Institute, Carnegie Mellon University, http://www.vasc.ri.cmu.edu/cgi-bin/demos/findface.cgi, 2010.

[75] H. Schneiderman and T. Kanade. A statistical model for 3D object detection applied to faces and cars. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[76] R. Seidel. Constrained delaunay triangulations and voronoi diagrams with obstacles. Technical Report 260, Institute for Information Processing, 1988.

[77] V. Setlur, S. Takagi, R. Raskar, M. Gleicher, and B. Gooch. Automatic image retargeting. In *ACM International Conference on Mobile and Ubiquitous Multimedia (MUM)*, pages 247–250, 2005.

[78] G. Shen and P. Caines. Hierarchically accelerated dynamic programming for finite-state machines. *IEEE Transactions on Automatic Control*, 47(2):271 –283, 2002.

[79] L. Shi, J. Wang, L. Duan, and H. Lu. Consumer video retargeting: context assisted spatial-temporal grid optimization. In *ACM international conference on Multimedia (MM)*, pages 301–310, 2009.

[80] S. M. Smith and J. M. Brady. Susan – new approach to low level image processing. *International Journal of Computer Vision (IJCV)*, 23(1):45 – 78, 1997.

[81] B. Suh, H. Ling, B. Bederson, and D. Jacobs. Automatic thumbnail cropping and its effectiveness. In *ACM Symposium on User Interface Software and Technology*, pages 95–104, 2003.

[82] M. Sujaritha and S. Annadurai. A three-level clustering algorithm for color texture segmentation. In *International Conference and Workshop on Emerging Trends in Technology (ICWET)*, pages 645–650, 2010.

[83] R. Szeliski. Image alignment and stitching: a tutorial. *ACM Foundations and Trends in Computer Graphics and Vision*, 2(1):1–104, 2006.

[84] J. Tang. A color image segmentation algorithm based on region growing. In *International Conference on Computer Engineering and Technology (ICCET)*, volume 6, pages 634–637, 2010.

[85] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *IEEE International Conference on Computer Vision (ICCV)*, page 839, 1998.

[86] K. Utsugi, T. Shibahara, T. Koike, K. Takahashi, and T. Naemura. Seam carving for stereo images. In *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pages 1–4, 2010.

[87] D. Vaquero, M. Turk, K. Pulli, M. Tico, and N. Gelfand. A survey of image retargeting techniques. In *IS&T/SPIE Applications of Digital Image Processing*, pages 779814–779828. SPIE, 2010.

[88] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 511–518, 2001.

[89] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[90] L. Wang, Y. Zhang, and J. Feng. On the euclidean distance of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27:1334–1339, 2005.

[91] S.-F. Wang and S.-H. Lai. Fast structure-preserving image retargeting. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1049–1052, 2009.

[92] S.-F. Wang and S.-H. Lai. Compressibility-aware media retargeting with structure preserving. *IEEE Transactions on Image Processing*, 20(3):855 –865, 2011.

[93] Y.-S. Wang, H. Fu, O. Sorkine, T.-Y. Lee, and H.-P. Seidel. Motion-aware temporal coherence for video resizing. *ACM Transactions on Graphics, SIGGRAPH ASIA*, 28(5), 2009.

[94] Y.-S. Wang, J.-H. Hsiao, O. Sorkine, and T.-Y. Lee. Scalable and coherent video resizing with per-frame optimization. *ACM Transactions on Graphics, SIGGRAPH*, 30(4), 2011.

[95] Y.-S. Wang, H.-C. Lin, O. Sorkine, and T.-Y. Lee. Motion-based video retargeting with optimized crop-and-warp. *ACM Transactions on Graphics, SIGGRAPH 2010*, 29(4):article no. 90, 2010.

[96] Y.-S. Wang, C.-L. Tai, O. Sorkine, and T.-Y. Lee. Optimized scale-and-stretch for image resizing. *ACM Transactions on Graphics, SIGGRAPH*, 27(5):1–8, 2008.

[97] J. V. D. Weijer, T. Gevers, and A. W. M. Smeulders. Robust photometric invariant features from the color tensor. *IEEE Transactions on Image Processing*, 15:2006, 2004.

[98] G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, 1990.

[99] L. Wolf, M. Guttmann, and D. Cohen-Or. Non-homogeneous content-driven video-retargeting. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1 –6, 2007.

[100] H. Wu, Y.-S. Wang, K.-C. Feng, T.-T. Wong, T.-Y. Lee, and P.-A. Heng. Resizing by symmetry-summarization. *ACM Transactions on Graphics, SIGGRAPH*, 29:159:1–159:10, 2010.

[101] P. Wu, B. S. Manjunanth, S. Newsam, and H. Shin. A texture descriptor for image retrieval and browsing. In *IEEE Workshop on Content-Based Access of Image and Video Libraries (CBAIVL)*, pages 3–7, 1999.

[102] B. Yan, K. Sun, and L. Liu. Matching-area-based seam carving for video retargeting. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(2):302–310, 2013.

[103] T.-C. Yen, C.-M. Tsai, and C.-W. Lin. Maintaining temporal coherence in video retargeting using mosaic-guided scaling. *IEEE Transactions on Image Processing*, 20(8):2339–2351, 2011.

[104] Z. Yu and H.-S. Wong. A rule based technique for extraction of visual attention regions based on real-time clustering. *IEEE Transactions on Multimedia*, 9(4):766 –784, 2007.

[105] R. Zabih, J. Miller, and K. Mai. A feature-based algorithm for detecting and classifying scene breaks. In *ACM International Conference on Multimedia (MM)*, pages 189–200, 1995.

[106] J. Zhang and C.-C. Kuo. Region-adaptive texture-aware image resizing. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 837–840, 2012.

[107] Y.-F. Zhang, S.-M. Hu, and R. R. Martin. Shrinkability maps for content-aware video resizing. *Computer Graphics Forum, Special issue of Pacific Graphics 2008*, 27(7):1797–1804, 2008.

[108] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002.