University of Mannheim/Computer Science

Technical Report TR-2013-003

Analysis of Disparity Maps for Detecting Saliency in Stereoscopic Video

Stephan Kopf, Benjamin Guthier
Philipp Schaber, Torben Dittrich, Wolfgang Effelsberg
Department of Computer Science IV
University of Mannheim, Mannheim, Germany
{kopf, schaber, guthier, effelsberg}@informatik.uni-mannheim.de
dittrich@pi4.informatik.uni-mannheim.de

ABSTRACT

We present a system for automatically detecting salient image regions in stereoscopic videos. This report extends our previous system [4] and provides additional details about its implementation. Our proposed algorithm considers information based on three dimensions: salient colors in individual frames, salient information derived from camera and object motion, and depth saliency. These three components are dynamically combined into one final saliency map based on the reliability of the individual saliency detectors. Such a combination allows using more efficient algorithms even if the quality of one detector degrades. For example, we use a computationally efficient stereo correspondence algorithm that might cause noisy disparity maps for certain scenarios. In this case, however, a more reliable saliency detection algorithm such as the image saliency is preferred. To evaluate the quality of the saliency detection, we created modified versions of stereoscopic videos with the non-salient regions blurred. Having users rate the quality of these videos, the results show that most users do not detect the blurred regions and that the automatic saliency detection is very reliable.

Keywords

Video saliency, depth saliency, visual attention, stereoscopic videos

1. INTRODUCTION

Automatically detecting salient regions in images is the basis of many different applications in image processing. For instance, saliency detection is used for image compression to encode salient regions in high quality and to increase the compression rate for non-salient regions. Another application is image retargeting, which automatically adapts the resolution of an image to the target resolution of a display [10, 21, 11, 15, 22]. Errors caused by cropping, scaling, or warping are significantly reduced when saliency information is taken into account. A third example is embedding digital watermarks in images. A watermark should be nearly *invisible* to a human observer but should be *robust* at the same time. Being robust means to be extractable even when subject to image distortions or compression artifacts [29]. Embedding watermark information in non-salient regions allows to increase the signal strength of the watermark without increasing the effect on the human visual system. Yet another example is to automatically produce short video summaries by selecting important shots and scenes from a video [19, 18]. Detecting salient image regions, salient shots, salient scenes, and audio saliency are fundamental requirements for video summarization.

There has been all kinds of research for detecting salient regions in still images and videos. As multiview videos are becoming more and more popular, depth has been added as a new factor in saliency. A special case of multiview videos are stereoscopic videos which provide two different views of a scene to add a depth impression[1]. The basis of stereoscopic perception is seeing with two eyes that are slightly horizontally shifted. This way, both eyes see the same object or scene from different angles. By this means, the images formed by each eye divert slightly concerning the location of objects within a scene. Those differences between object points are called *disparities*. Based on the disparities between both views, the human brain is capable of decoding depth information of a scene. Although the effect of depth is the main motivation for watching multiview or stereoscopic videos, only very few research results have been published yet that focus on the saliency of the depth information.

The number of movie theaters that support 3D content has increased significantly in the last few years. Also, all major enterprises that produce displays or TVs offer at least one screen that supports stereoscopic content. This trend continues now with smartphones and mobile phones. Because cinematic movies [14] are typically captured in wide screen resolution, that is, aspect ratio between $17:9$ and $22:9$ in contrast to the HD ratio of $16:9$, methods for automatically retargeting the resolution of stereo video become necessary to make the content available for other display devices.

In this paper, we analyze the impact of depth information in videos on the human visual system. We detect salient regions based on the colors of the content, the motion in a scene, and the depth differences of objects in a scene. Our main focus is to see how much influence the depth impression

[1]Note that in the case of cinematic movies, the commonly used term *3D movies* actually refers to stereoscopic video.

really has as a factor for the detection of salient regions in stereo videos.

This paper is structured as follows: Section 2 covers different approaches for the detection of salient regions in media. A number of methods to detect salient regions in still images as well as in videos and stereo videos are presented. Section 3 gives an overview of our system while Section 4 presents the implementation of the saliency detection algorithm in stereo videos. The results of the implementation and a discussion on how test users evaluated the accuracy of the implemented method are given in Section 5. Section 6 concludes the paper.

2. RELATED WORK

Saliency detection methods can be classified depending on the amount of visual information they consider. Image-based approaches consider the color and contrast of pixels in an image (2D content). Videos also include temporal information about object and camera motion. Stereo videos additionally provide depth information about the objects in a scene. In the following, we present selected approaches to saliency detection in images and video. Up to date, only one other method for saliency detection in stereoscopic video has been published. It is discussed in detail at the end of this section.

The goal of *image saliency detection* is to determine regions which are particularly noticeable to human observers. Image saliency detectors are usually mathematically or biologically inspired [1]. Several methods have been proposed that try to simulate the characteristics of the human visual system. Most approaches make the assumption that an object is salient if it significantly differs from other objects in its local neighborhood [7]. It is common to distinguish between top-down and bottom-up approaches [26]: *Top-down* approaches identify important image regions like faces, superimposed text [16], people [17], or objects [28], whereas *bottom-up* techniques analyze pixel values and group highly salient pixels into regions.

Ma and Zhang [24] proposed a bottom-up contrast-based saliency detection algorithm. Their general assumption is that a high contrast between pixels is a good indicator for image regions that are relevant to an observer. The *contrast value* C_i of pixel i is defined as $C_i = \sum_{q \in \Theta} d(p_i, q)$. p_i is the color of the current pixel, and $q \in \Theta$ is a pixel in the local neighborhood Θ of i. The Gaussian distance d is used to measure the color difference of two pixels in the LUV space. A fuzzy growing technique is then applied to the contrast map to identify regions with high saliency.

In previous work, we improved the method proposed by Ma and Zhang [20]. Instead of using fuzzy growing we implemented a color quantization technique based on the Linde-Buzo-Gray (generalized Lloyd) algorithm [23]. The values of C_i are mapped to a fixed number of new values which define the *codebook*. The values in the codebook are selected such that the average distortion caused by the mapping is minimized. To achieve this, the codebook must fulfill two constraints: A contrast value must be mapped to the nearest value in the codebook, and each codeword must be the centroid of the contrast values mapped to it.

Another approach was presented by Achanta et al. [1]. It focuses on objects with a high contrast to the image background. In a first step, the images are normalized by removing very high and very low contrast differences. A very high contrast typically indicates noise which distorts saliency results. Low differences are removed such that all pixels in a large region get similar saliency values. The Euclidean distance in the $L^*a^*b^*$ color space of a pixel to the average color of the normalized image is used to estimate the saliency value of a pixel. The advantages of the proposed approach are its computational efficiency and its ability to create saliency images with a correct localization of object borders.

A further novel approach was proposed by Goferman et al. [6]. It detects salient regions in the background as well as important foreground objects. The detection of local and global saliency is determined by the number of appearances of a pixel. This is done by centering a window of varying size around a pixel. Based on the prominence of this window, the saliency of the centered pixel is set. Next, a multi-scale saliency enhancement step is performed. The last step is to highlight those pixels that build the centers of gravity of salient regions. Therefore, the most salient regions are extracted based on a threshold. Then, the saliency value of each pixel outside a salient region is redefined on the basis of its distance to the closest pixel of the salient region. Consequently, parts of neighboring background pixels of an object are also considered to be salient.

Cheng and Zhang presented another bottom-up approach that analyses the global color contrast in images [3]. In order to compute the importance of a color, the algorithm computes its difference to all other colors in $L^*a^*b^*$ color space weighted by their frequencies. We selected this image saliency detector for our system due to the high quality of the results and its computational efficiency. The details of this approach are discussed in Section 4.1.

Compared to still images, video sequences contain *motion* which is another aspect of human visual attention. Motion is either caused by moving objects or by camera movements. In both cases, the regions of interest might change over time as objects enter or leave a scene.

The optical flow can be used to detect motion in image sequences. It describes a vector field that indicates the direction in which each pixel moved. Wixson and Hansen [32] developed an algorithm to detect salient motion based on flows. Using a surveillance scenario, the assumption is that important objects move in a constant direction. To measure the saliency of a pixel, a set of optical flow fields is considered and for each pixel the distance it has moved and its direction is taken as a measure for saliency.

Zhu et al. [36] propose an adaptive saliency algorithm for videos using motion vectors. The approach analyses motion vectors derived from block matching and optical flow. The salient region is defined by motion features in case of high motion; otherwise color and orientation is used. The approach is efficient to compute, because only a part of the saliency features are considered in each iteration.

Xia et al. [33] use motion history maps to create saliency maps. The model computes the spatial saliency as well as the temporal saliency. For motion analysis, each frame is divided into blocks and a motion vector field is computed to detect moving blocks. Next, region growing is applied to detect the interior of moving objects. To create the final temporal saliency map, the motion map of the current frame is combined with the motion history map. The resulting

saliency map for a frame is a combination of the spatial and temporal saliency map. An advantage of this approach is its ability to keep a moving object highlighted as salient even after it stopped moving.

Only few 3D visual attention models have been published that consider depth as an estimator for salient regions. An early approach was formulated by Maki et al. [25] which considers motion and depth information. First, preattentive cues are detected. These are stereo disparity, image flow, and motion. In a second stage, the cues are combined into one saliency map. For this, two masks are computed based on pursuit and saccade modes. In the pursuit mode, the target object is followed and masked, whereas the goal of the saccade mode is to switch the focus to another more relevant object. To determine the final target mask, depth is used as an indicator to tell which of the masks should be applied.

Riche et al. [27] present a similar saliency detection approach that combines motion and depth information. The proposed model consists of a horizontal, a vertical, and a depth motion feature extractor. The detected features are discretized into different speeds and directions. After extracting the features, a low-pass spatio-temporal filter is applied to summarize the feature maps of all speeds and directions. Although this bottom-up approach analyzes 3D object motion, it does not consider image saliency features like color or contrast. This makes the obtained saliency information inapplicable to many practical purposes.

An approach that combines depth maps with a region-based saliency map was developed by Zhang et al. [34]. The saliency value of a pixel region is computed from its color contrast to all other regions in the image. Supposing that a depth map for the input image is available, the saliency value of a region is computed as the combination of its depth value and its color-based saliency. Their work only considers depth information of images and does not take temporal information into account.

The only method that considers all three saliency factors (color contrast, motion, and depth) is the visual attention model proposed by Zhang et al. [35]. The saliency map of a frame is computed as a combination of three different saliency maps. First, salient regions for still images are detected by the bottom-up spatial attention model developed by Itti et al.[9]. This approach combines color contrast, intensity, and orientation. The second part is a temporal attention model. To detect motion of objects within a video sequence, block-based optical flow is used [35]. The consecutive frames build a frame group where vertical and horizontal motion is estimated and combined for each frame. The third component is the depth attention model. A graph cut algorithm [12] is used to create a disparity map, which is converted into the perceived depth Z with $Z = B * f/d_c$, where B is the baseline between both cameras, f is the focal length of the cameras, and d_c is the physical disparity between corresponding points. The physical disparity is calculated by converting pixel disparities into centimeters. With the knowledge of the perceived depth Z, pixels belonging to near objects get higher saliency values compared to those that do not. A disadvantage of this approach is that the camera parameters must be known. This is unrealistic in most scenarios.

The final step of the stereoscopic attention model by Zhang is a dynamic fusion of all saliency values. The different values of the spatial and temporal models are weighted dynamically on the basis of motion contrast. This means that in sequences with low motion contrast the weight for the static image increases. Pixel values of the depth map are weighted by a static factor. However, in their paper the authors did not present a user evaluation that measures the perceived quality of the saliency maps.

In comparison to the previous research in this area, the distinctive features of our proposed system are:

1. We built a system that computes saliency maps for stereoscopic video based on a combination of image saliency, motions saliency, *and* depth saliency. Almost all previous system focus on only one or two saliency features at a time.

2. Our proposed method is suitable for arbitrary stereo videos. We do not assume that information about the capturing process is available (e.g., camera parameters like focal length or distance between the two cameras).

3. The computational effort of the algorithms is considered in our approach. Efficiency is preferred over precision of the disparity maps in order to reduce the overall computation time to process a video.

4. Many automatically computed disparity maps are very noisy. We propose techniques to reduce such noise by including the reliability of the image, motion, and depth saliency maps into the calculation of suitable weights.

5. To evaluate the performance of the saliency detector, we generated test videos with strong distortions in regions that were marked as non-salient by various saliency measurements. A user evaluation compares the perceived visual quality of the different algorithms.

3. SYSTEM OVERVIEW

Figure 1 gives an overview of the system. In a first step, the algorithm decodes a stereo video and splits each frame into the left and right views. We require rectified views in order to create valid disparity maps. It can be assumed that all input videos are shot with calibrated cameras and that the images are already rectified.

Now, the actual process of saliency detection starts. The spatial and temporal salient regions are detected from only the left view, since they are usually applied to 2D videos. However, both views need to be considered for the creation of depth saliency. Each of the three detectors for salient regions creates an individual saliency map. The disparity map, created by a stereo correspondence algorithm, is used as an indicator for pop-out regions in a frame.

In the next step, all three saliency maps are fused to a single map by weighting each map differently. The result is a combined saliency map that contains the final salient regions. This information is now available to other applications to enhance the processing of stereo videos. Typical examples for such applications are video compression, video retargeting, or video summarization. To create test videos for our user evaluation, a sample application blurs non-salient regions in both views and encodes them into a new stereo video file.

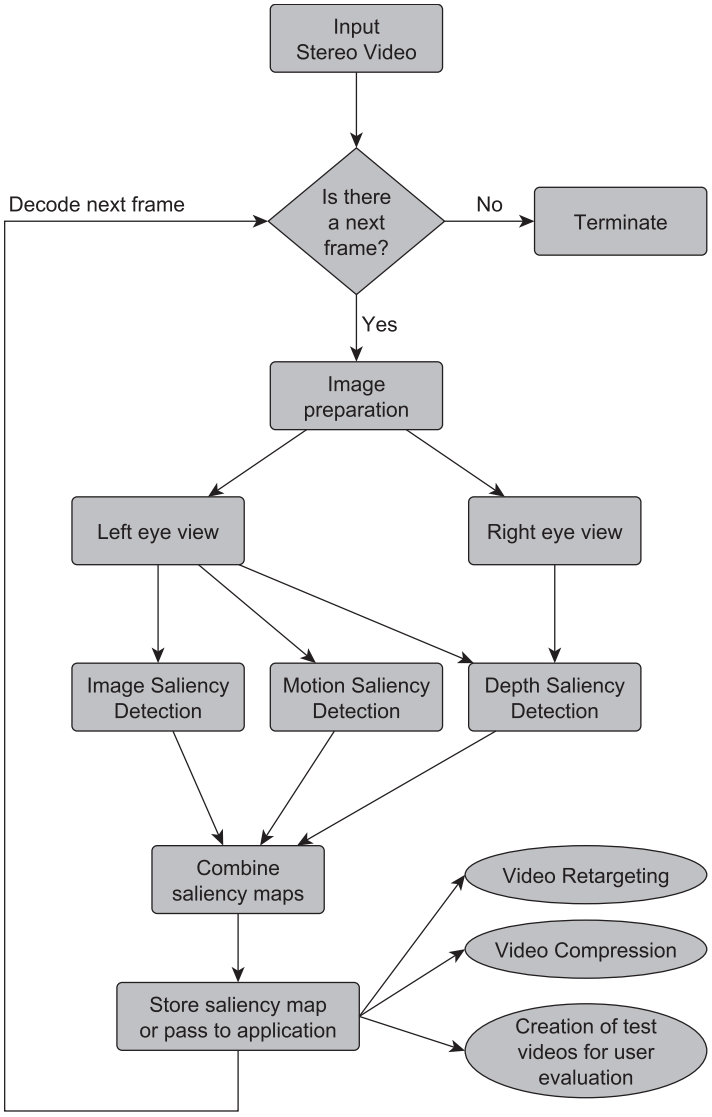


Figure 1: System overview

4. SALIENCY DETECTION

4.1 Saliency in Still Images

For the computation of image saliency we use a histogram-based algorithm proposed by Chen et al. [3]. This algorithm utilizes the fact that the human visual system is sensitive to colors which occur seldom within an image. The algorithm works in three steps: *quantization*, *color difference estimation*, and *smoothing*.

Quantization is done to improve the run-time of the algorithm. Instead of using the full color space, each color channel (RGB) is reduced to only 12 values. This reduces the maximum number of different colors to $12^3 = 1728$. Furthermore, there are colors that occur too infrequently to be considered salient by the human eye. Hence, they are replaced by colors that lie closest to them in the histogram. The remaining colors – typically less than 100 different color values – constitute a new color space for the analyzed image.

In order to compute the importance of these colors, the algorithm uses the difference between them. This is done by first converting the RGB-space into the $L^*a^*b^*$ color space, where the L*-axis represents the brightness, the a*-axis the green to red part of a color, and the b*-axis the blue and yellow part. A major advantage of $L^*a^*b^*$ is the fact that it contains all colors in a *perceptually uniform* way. This means that a fixed absolute change of a color value has the same visual importance (based on human perception) for all colors.

After this conversion, the image saliency value of a color value c_l can be defined as:

$$S(c_l) = \sum_{j=1}^{n} f_j * D(c_l, c_j),$$

where f_j is the frequency of pixel occurrences with color value c_j, n is the number of different color values occurring in the complete image (after quantization), and $D(c_l, c_j)$

Figure 2: Left view of stereo videos and corresponding image saliency maps. Pixels with high saliency values are printed in dark colors.

computes the absolute color distance between two colors c_l and c_j in the $L^*a^*b^*$ color space.

The final step of this algorithm is color space smoothing. This step is performed to reduce noise and thereby refine the saliency value for each color. The current saliency value of a color is replaced by a weighted average saliency value of similar colors.

The image saliency value $I_S(p)$ of pixel p is defined as

$$I_S(p) = S(c_l) \tag{1}$$

on condition that the color of pixel p is mapped to c_l during the quantization step.

Figure 2 shows two video frames and the corresponding image saliency maps. The red car is correctly detected as salient region in the first video. In the second video, the red posts and the bus are correctly detected but the sky is marked as salient, too.

4.2 Motion Saliency

The algorithm to estimate motion saliency analyzes the differences between consecutive frames. First, the moving average between two frames I_{i-1} and I_i is computed. The image acc_i serves as an accumulator and stores the weighted sum of input frames in a sequence. It is computed for all pixels (x, y) as

$$acc_i(x, y) = (1 - \alpha) \cdot acc_{i-1}(x, y) + \alpha \cdot I_i(x, y), \tag{2}$$

where I_i specifies the current frame and α is an indicator of how fast the accumulator forgets about previous frames.

Next, the absolute difference between the current frame I_i and the current moving average acc_i is calculated. In order to get blobs of moving objects, multiple erosion and dilation steps are performed. Dilation increases any bright regions in an image whereas erosion shrinks them. Because we do not need information about exact boundaries of moving objects and to make sure that the full shape of moving objects is covered, the number of iterations for dilation is larger than the one for erosion.

Therefore, the motion saliency value $M_S(p)$ of pixel p is defined as

$$M_S(p) = (|I_i - acc_i| \ominus_n B) \oplus_m B. \tag{3}$$

$\ominus_n$ and $\oplus_m$ denote n respectively m iteration steps of erosion and dilation using the structuring element B.

Figure 3 shows the motion saliency maps of two stereo videos. The first video visualizes two moving objects (cars) which are clearly visible in the saliency map. Although the red car is partially occluded, its pixels are estimated very well due to the dilation step. The second saliency ignores the colorful background (in contrast to image saliency) and shows the movement of the person.

4.3 Depth Saliency based on Semi-Global Block Matching

We make the assumption that objects close to the camera are more relevant in terms of saliency that other objects. Our goal is to identify these objects (pop-out regions) by comparing both views of a stereo video. The depth of an object pixel is approximated by its horizontal shift (disparity). For the computation of disparity maps, semi-global block matching is used. It is a slightly modified version of the semi-global matching algorithm developed by Hirschmüller [8].

We assume that the stereo video is already rectified. Therefore, the disparity d of a pixel p describes the horizontal shift of a pixel between both views of a frame in a stereo video.

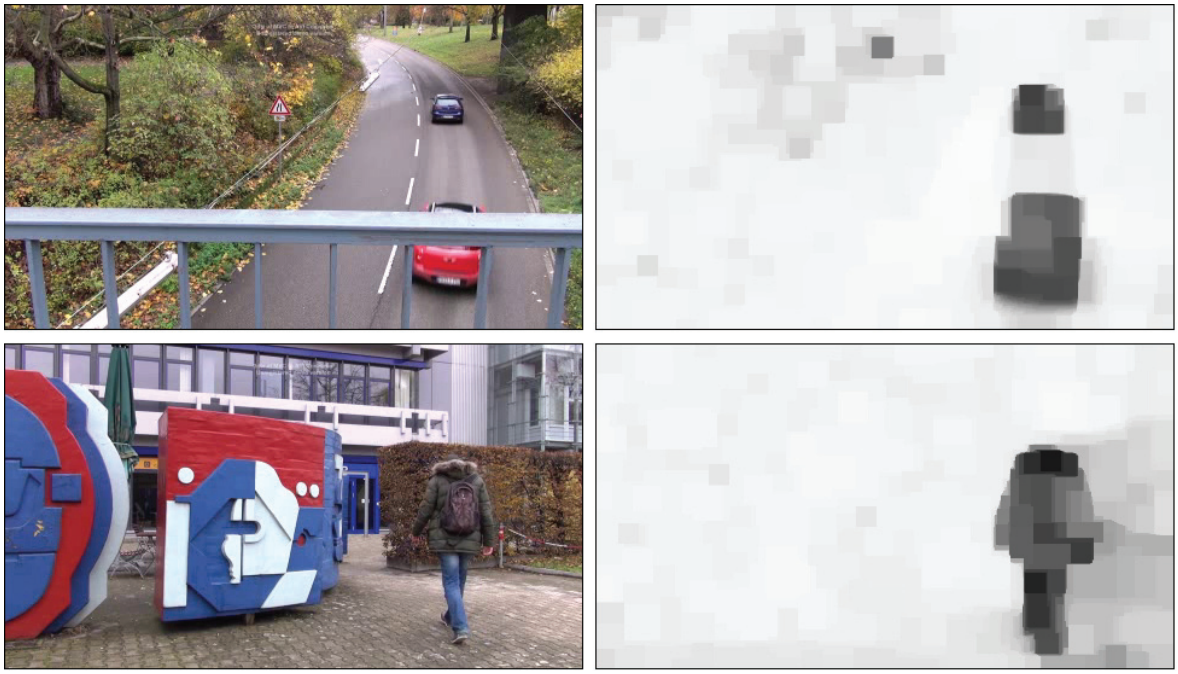


Figure 3: Left view of stereo videos and corresponding motion saliency maps.

The aim of the algorithm is to compute the optimal disparity d for each pixel p. The cost $C_1(p,d)$ should be minimized and is defined as the absolute luminance difference

$$C_1(p,d) = |I_p - I_q| \tag{4}$$

of the pixels p and q of the left and the right frame. Considering the position of a pixel $p = (p_x, p_y)^T$, the position of pixel $q = (p_x - d, p_y)^T$ depends on the value of the disparity and describes a horizontal shift.

A direct computation of the cost $C_1(p,d)$ typically leads to very noisy results. Therefore, two additional constraints are added to smooth the resulting disparity map. Small changes of disparity values should be avoided. If two disparity values D_p and D_q slightly differ between pixels p and q (q is in the neighborhood of p), then a constant penalty P_1 is added. This defines the additional cost C_2 for a pixel p and its disparity d:

$$C_2(p,d) = \sum_{q \in N_p} P_1 * T[|D_p - D_q| = 1] \tag{5}$$

The binary function $T[v]$ is zero or one depending on the value of parameter v which checks whether the difference value is small (equals 1). In case of large disparity changes (which typically occur at object boundaries), a larger penalty value P_2 is added as cost term C_3:

$$C_3(p,d) = \sum_{q \in N_p} P_2 * T[|D_p - D_q| > 1] \tag{6}$$

To compute gradual changes of the depth, $P_1 \leq P_2$ should be valid. The overall cost C_{sum} of a pixel with disparity d is now defined as:

$$C_{sum}(p,d) = \sum_p (C_1(p,d) + C_2(p,d) + C_3(p,d)) \tag{7}$$

To find an optimal disparity map, Equation 7 is to be minimized.

The disparity map is defined by the disparity values D_p of all pixels p. The depth saliency value $D_S(p)$ of pixel p highlights pop-out regions which correspond to large disparity values. $D_S(p)$ is therefore defined as:

$$D_S(p) = \arg\min_d C_{sum}(p,d). \tag{8}$$

The computation of the global minimum of C_{sum} is NP-complete [8]. Therefore, a heuristic is used to compute a local minimum. This is done by analyzing different paths that move to the current pixel p. The reduction of a two-dimensional into a one-dimensional problem allows the computation of the optimal disparity values for the path pixels. The penalty values P_1 and P_2 are still used to avoid noisy results. The original algorithm is further modified as follows:

- Instead of searching matches for single pixels, blocks are matched (we use the sum of absolute differences with a fixed window size of 3).
- A Birchfield-Tomasi sub-pixel metric [2] is used to replace mutual information as proposed in the original implementation [8].
- We do not use a fixed value for the maximal disparity value, but set the maximum depending on the width of the input image.

Table 1: Weights for the edges of the graph cut algorithm

edge	weight	for
(s, a)	$D_{occ}(a)$	$a \in A^0$
(a, t)	$D_{occ}(a)$	$a \in A^\alpha$
(a, t)	$D(a) + D_{smooth}(a)$	$a \in A^0$
(s, a)	$D(a)$	$a \in A^\alpha$
$(a1, a2), (a2, a1)$	$V_{a1,a2}$	$\{a1, a2\} \in N$
$(a1, a2)$	∞	$p \in P, a1 \in A^0, a2 \in A^\alpha$, $a1, a2 \in N_p(\widetilde{f})$
$(a2, a1)$	C_p	$p \in P, a1 \in A^0, a2 \in A^\alpha$, $a1, a2 \in N_p(\widetilde{f})$

4.4 Depth Saliency based on Graph Cuts

A second approach to estimate the disparity uses the graph cut algorithm which was developed by Kolmogorov and Zabih[13] and can be divided into the following steps:

1. Preprocessing of the input image
2. Define energy function
3. Create graph of stereo image
4. Use graph cuts to minimize the engery function

In the following, A will be a set of unordered pairs of pixels which are likely to correspond. In the case of rectified images and pixels p with coordinates (p_x, p_y), A is defined as $A = \{\langle p, q\rangle \mid p_y = q_y \; and \; 0 \leq q_x - p_x < k\}$ where k is the maximum disparity between to pixels. An assignment $a = \langle p, q\rangle \in A$ is given a value f_a which is set to 1 if the assignment consist of corresponding pixels (*active assignment*) and otherwise 0.

First, the algorithm needs to do some image preparation before it can start. For this, the input image is segmented into regions with possible similar disparities. For instance, this can be done with color segmentation. In this case, it is assumed that the disparities in areas with similar colors do not differ too much. Then, each segment of the image is used as a configuration f which is called *unique* if each pixel of it is involved in at most one active assignment. In unique configurations, occluded pixels are in no active assignments.

Next, an energy function needs to be created which in this case is defined as:

$$E(f) = E_{data}(f) + E_{occ}(f) + E_{smooth}(f) \qquad (9)$$

The data term $E_{data}(f)$ is a result from the differences in intensity between pixels. The second term in the function is the occlusion term E_{occ} which imposes a penalty if a pixel is occluded. The smooth term E_{smooth} imposes a penalty if one assignment is in the configuration f, whereas a neighboring assignment with the same disparity value is not. Hence, if neighboring pixels have the same disparity this penalty is zero.

Now that an energy function is given, the following steps are the creation of a graph and the minimization of the function with graph cuts. Those steps will be explained in theory and practice with the example shown in Figure 4. In this example, the left and right view consist of four pixels each. The solid lines represent active assignments that were found in the preparation of the image. The dashed lines indicate the assignments that are considered to be better matches.

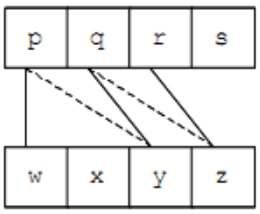


Figure 4: Example for the graph cut algorithm

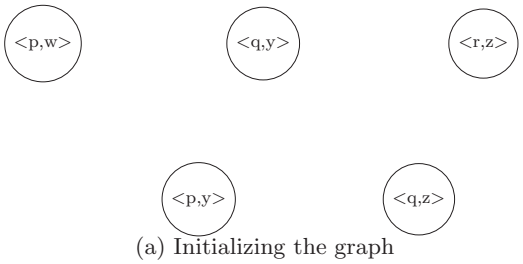
Given a configuration f and a disparity α, the algorithm starts with a unique configuration f^0 with:
$A^0 = \left\{a \in A(f^0) \mid d(a) \neq \alpha\right\}$, where $d(a)$ is the disparity value of the assignment a, and an α-expansion set of inactive assignments $A^\alpha = \{a \in A \mid d(a) = \alpha\}$. In the example, the left and right image consists of four pixels each. The solid lines define the active assignments, so A^0 would consist of the assignments $\{\langle p, w\rangle, \langle q, y\rangle, \langle r, z\rangle\}$, whereas $A^\alpha = \{\langle p, y\rangle, \langle q, z\rangle\}$. The goal is to find a new configuration with active assignments within one α expansion as a subset of $\widetilde{A} = A^0 \cup A^\alpha$.

Now a weighted graph $G = (V, E)$ can be created where each assignment $a \in A^0 \cup A^\alpha$ is a vertex, as shown in Figure 5(a). Next, terminals s and t are added to the graph. These are needed to compute the flow in the graph. Afterwards, edges are added to the graph. Edges (s, a) are set from the source to each vertex in the graph and from each vertex to the sink edges $(a, t)$are added. Further edges are inserted for neighboring assignments in the same set. Hence, in Figure 5(c) the assignments $\langle q, y\rangle$ and $\langle r, z\rangle$ as well as $\langle p, y\rangle$ and $\langle q, z\rangle$ are connected. The assignments $\langle p, w\rangle$ and $\langle q, y\rangle$ are not connected because w and y are no direct neighbors.

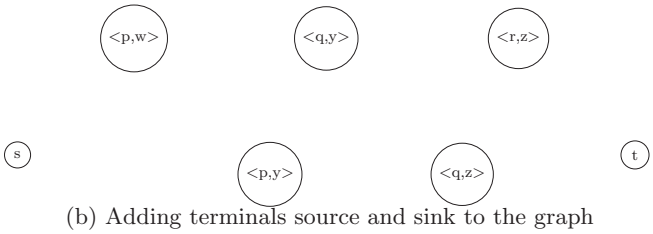
Furthermore, assignments from different sets are connected if they contain the same pixels. This needs to be done, because the final result should be a unique configuration, so pixels cannot occur in more than one assignment. In the example in Figure 5 the assignments $\langle p, w\rangle$ and $\langle p, y\rangle$ both contain pixel p, thus, they need to get connected.

The last step of the graph creation is setting the weights for each edge. The different weights for edges can be seen in Table 1 and are taken from [13].

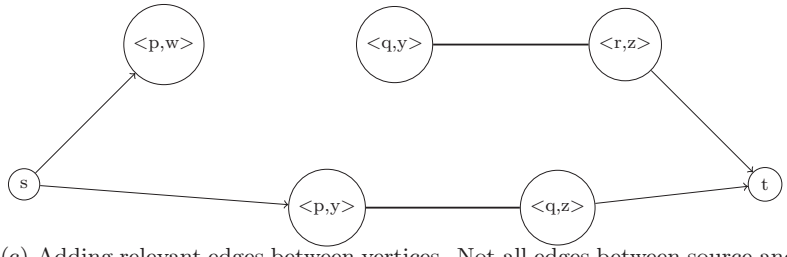
The occlusion cost D_{occ} is defined as $D_{occ}(\langle p, q\rangle) = D_{occ}(p) + D_{occ}(q)$, where $D_{occ}(p) = 0$ if p has more than one entering edge and C_p, which is a penalty if the pixel is occluded, if there is only one. The weight $D(a)$ comes from the data term which considers the differences in intensities I of pixels. It is defined as $D(a) = (I(p) - I(q))^2$ for an assignment $a = (\langle p, q\rangle)$. $V_{a1,a2}$ is a penalty based on the smoothness



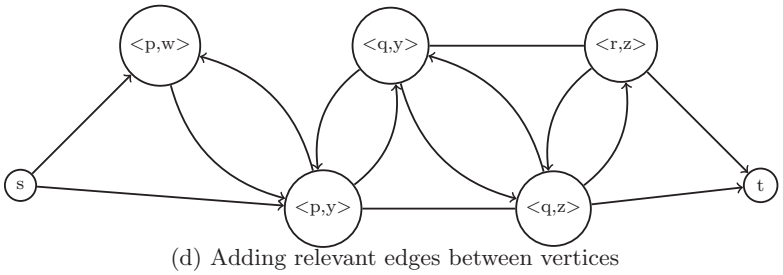
(a) Initializing the graph



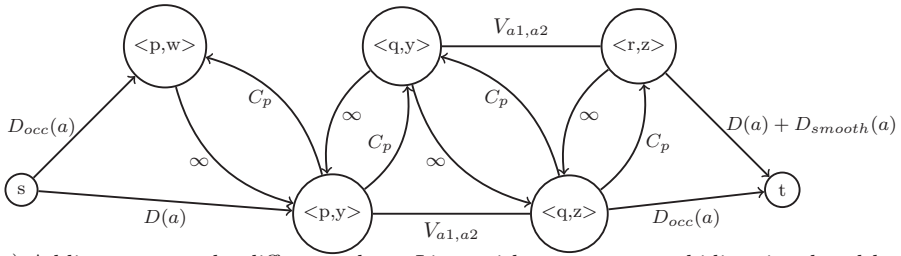
(b) Adding terminals source and sink to the graph



(c) Adding relevant edges between vertices. Not all edges between source and sink to each assignment are drawn for visualization



(d) Adding relevant edges between vertices



(e) Adding costs to the different edges. Lines without arrows are bidirectional and have the same cost in both directions.

Figure 5: Graph construction of the graph cut algorithm

constraint. The smoothness cost is set to

$$D_{smooth}(a1) = \sum_{\{a1,a2\}\in N, a2\notin\widetilde{A}} V_{a1,a2}. \quad (10)$$

How the costs are assigned is shown in Figure 5(e). For instance, the edge from $\langle q, y\rangle$ to $\langle q, z\rangle$ has costs ∞ because both contain pixel q which cannot be active in two assignments.

The final step of the algorithm is to find the best matches between pixels. This is done by a cut $C = V^s, V^t$ which divides the vertices into two sets such that $s \in V^s$ and $t \in V^t$. Moreover, each vertex in A can be either in V^s or in V^t. Now, the goal is to find the minimum cut which is the cut with the smallest cost. This can be solved by computing the maximum flow between source and sink. For instance, a well known algorithm to compute the maximum flow is given by Ford and Fulkerson [5].

4.5 Fusion of the Saliency Maps

After the three saliency maps are created, they are fused into a single map. The final saliency value of a pixel is computed by

$$S(p) = w_i * I_S(p) + w_m * M_S(p) + w_d * D_S(p) \quad (11)$$

where $S(p)$ is the final saliency value of a pixel p. The values of each map are considered as $I_S(p)$ for the saliency value of still images (see Equation 1), $M_S(p)$ for the one computed for motion (Equ. 3), and $D_S(p)$ which is the disparity value of a pixel (Equ. 8). Moreover, each value is weighted differently by the weights $w_i, w_m, w_d \in [0, 1]$ which need to hold $w_i + w_m + w_d = 1$.

In our implementation, the weight w_m is set to 0.3. This parameter was empirically determined and is approximately the average of w_i and w_d. Motion is therefore always considered but does not outweigh the other weights.

The other weights can be chosen dynamically based on different factors with the constraint that $w_i + w_d = 0.7$. To estimate these weights, the maximum distance between colors and the maximum depth value is considered. These measures are used because the saliency detection algorithm in still images usually produces incorrect results if the maximum distance of colors is low. Then, the algorithm detects either too much of the image as salient or hardly any regions.

Another indicator are strong pop out regions in an image, that have pixel disparity values above a certain threshold. Hence, $w_d > w_i$ should be set if either the maximum distance between colors is low or the average disparity is high. Vice versa, if the maximum color distance is large enough, the $I_S(p)$ is usually a better indicator for salient regions. This is mainly based on the fact that the automatically computed disparity maps usually show a lot of errors and do not represent ground truth data.

The pseudo code in Algorithm 1 illustrates the details and parameters for the computation of the weights. The first case checks whether the color distance or the number of pop-out pixels are above or below a certain threshold. If both values are very high, it can be assumed that both saliency maps are giving important information about the overall saliency. In contrast, if both values are below their thresholds, it is assumed that the saliency maps are not very reliable. In both cases, the weight for the image saliency map is set a little higher because the reliability of the image saliency detector is a little higher in these cases.

Algorithm 1 Computing weights

```
// colDist: color distance
// pD: percentage of pixels with disparity above a thresh.
if colDist ≥ maxVal && pD ≥ 0.05 ||
colDist < maxVal && pD < 0.05 then
    w_i ← 0.4
    w_d ← 0.3
else
    if colDist ≥ maxVal && pD < 0.05 then
        w_i ← 0.5
        w_d ← 0.2
    end if
    if colDist < maxVal then
        if pD > 0.05 && pD < 0.3 then
            w_d ← 0.62 * sqrt(pD) + 0.16
            w_i ← 0.7 − w_d
        end if
        if pD ≥ 0.3 then
            w_d ← 0.5
            w_i ← 0.2
        end if
    end if
end if
```

Next, if the distance between colors is over its threshold but there is hardly any depth in a scene, then, the weight for the image saliency is set much larger than the one for the disparities. The last case is the one in which the maximum color distance is below its threshold and there are more than 5% of pixels with high values in the disparity map. If this is the case, the weight for pixels in the disparity map is increased by the percentage of pixels with large disparity values. If more than 30% of the complete image corresponds to pop out regions then the pixels in the disparity map will influence the final map by 50%.

The last step of the fusion is checking whether $S(p)$ is below a certain threshold to remove small areas, which usually are non-salient regions.

4.6 Application: Blur Non-Salient Regions

In order to analyze the quality of the detected salient regions, a sample application blurs out non-salient regions in each frame (see Figure 17). As input, this application needs the combined saliency map and both views. First, multiple erode and dilate steps are done in order to remove small artifacts in the saliency map (we use 10 iterations of each operator). Such a high value is used because usually there are small salient regions around a main salient region. If the number of iterations is set high enough, these regions are merged to one.

As a next step, the algorithm finds the outer contours to detect salient regions in the saliency map. The identification of outer contours is sufficient, because if one contour lies within another, it will be detected as salient anyway. The algorithm iterates through the contours and builds bounding rectangles around them. As occluded pixels are usually at the border of objects, the width and height of the rectangle is increased by a small value to make sure that these pixels are not blurred. However, it has not been found that occluded pixels influenced the results negatively.

As the saliency maps for the still image and for the motion

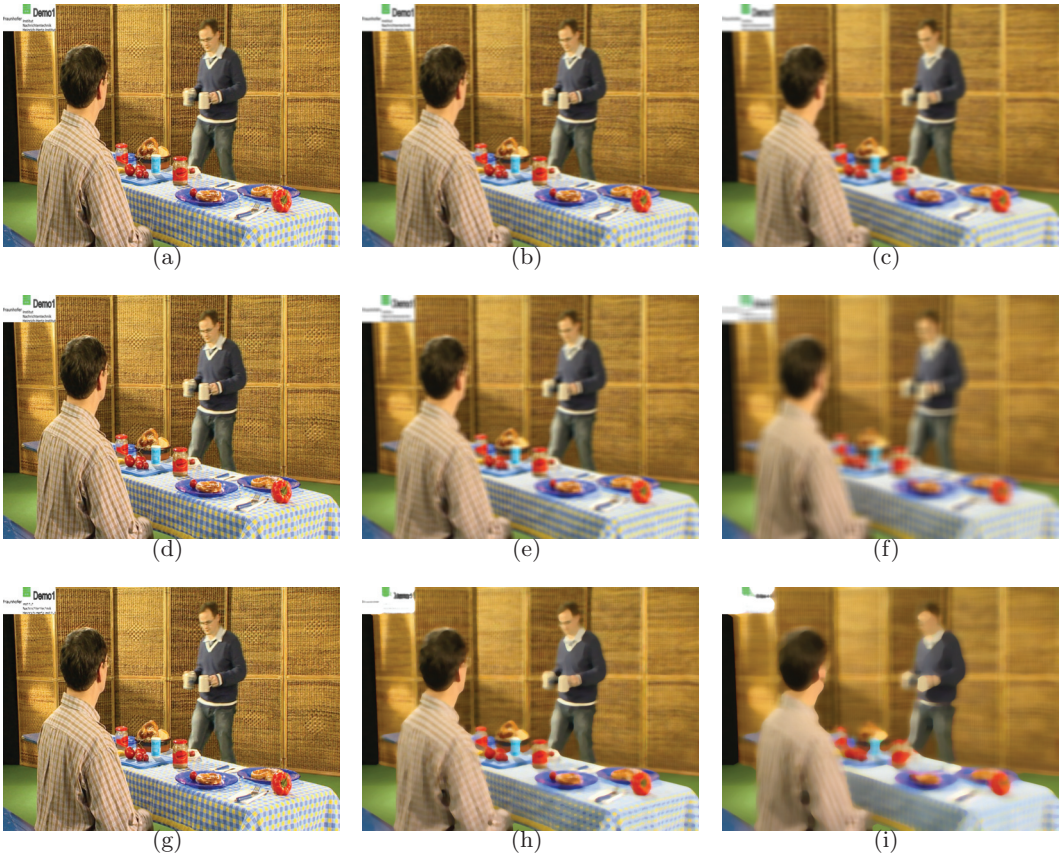


Figure 6: Blurring effects with different algorithms

are based on the left view, the bounding rectangle is only defined for the left view. Therefore, the salient region in the left view is matched against the right view to find the corresponding region. Due to the erosion and dilation in the beginning, the salient regions are large enough to find confident matches.

The last step before the encoding of the stereo video is to smooth all non-salient regions outside the bounding rectangles in both views. We use *Gaussian blur* with standard deviation σ_x and σ_y defined as:

$$\sigma_x = (\frac{n_x}{2} - 1) * 0.3 + 0.8,$$
$$\sigma_y = (\frac{n_y}{2} - 1) * 0.3 + 0.8. \quad (12)$$

n_x and n_y specify width and height of the windows size that is used for smoothing. The smallest window size that can be used to smooth an image is 3×3, for which the blurring effect can hardly be seen in the sample videos. When the window size increases the blurring is more noticeable.

The visual impact on smoothing an image is highly dependent on the number of different colors, which is a good indicator for the level of details. This is why we use an adaptive window size. The number of different colors defines the parameter of the window size which is kept between 3×3 and 35×35. Experimental results have shown that the number of colors in the test video sequences usually varies between 30 and 300. Thus, the window size of the blurring algorithm is set to:

$$V_{odd} = \begin{cases} 0 & \text{if } (\lfloor \frac{numColors}{10} \rfloor \ mod \ 2) = 0, \\ 1 & \text{otherwise.} \end{cases} \quad (13)$$

$$n_x = n_y = n_{max} - \lfloor \frac{numColors}{10} \rfloor - V_{odd} \quad (14)$$

The modulo operation in Equation 13 ensures that the window size is odd, which is a requirement for the smoothing operation. n_{max} defines the odd maximum window size that is used for blurring (we use $n_{max} = 33$). If the number of colors is larger than 300, the corresponding images are blurred with the minimum window size of 3×3. Images with a number of colors smaller than 30 can be seen as images with only few textures, thus, the window size is set to 35×35.

Figure 6 shows the differences between selected smoothing methods. The presented image was smoothed with a window size of 3×3, 15×15, and 31×31 from left to right. The Figures 6 (a) to (c) were blurred with Gaussian blur, (d) to (e) with a normalized box filter, and the last three images with a median filter. Using the smallest window size

does not create noticeable differences between all methods. But increasing the window size, the effects of the methods become more obvious. As one can see, the box and median filter destroy the image structure the most. Especially little details are very blurry and with the median filter the people can hardly be recognized. Even in images with a low texture level, these blurring methods create obvious errors in the image. That is why the Gaussian blur is used in this implementation. As the images show, even with a large window size, colors and details are still perceptible.

5. EVALUATION

The evaluation is split in two parts. First, we present the tested video sequences, visualize results of the saliency computation, and discuss typical errors. The second part then presents the user evaluation.

5.1 Experimental Results

Table 2: Properties of the tested video sequences

Video	Parameters of the video		
	Image size (one view)	Frame rate	Length
HHI Tridelity Demo1	960×1080	$25fps$	$9s$
HHI Tridelity Demo4	960×1080	$25fps$	$27s$
HHI Tridelity Demo5	960×1080	$25fps$	$27s$
Ice Age	816×1920	$25fps$	$23s$
Sammy's Adventure 2	640×720	$29fps$	$60s$

Table 2 lists the video sequences that are used to evaluate the results of our implementation. The first three sequences listed in Table 2 are are taken from the multiview video sequences provided by Fraunhofer Heinrich-Hertz-Institute (HHI)[2]. Further, we use two animated video sequences from the movies Ice Age and Sammy's Adventure 2.

Figure 7 shows single frame's saliency maps for still images, motion, and depth information. For both videos, the left view frame of the video (a), the three separate saliency maps (b)-(d), and the combined saliency map (e) are visualized.

Image Saliency

In most cases, the saliency maps of the histogram-based approach (b) are accurate enough to be used as an indicator for salient regions. Overall, these saliency maps are very detailed, with some exceptions in which the algorithm either returns hardly any salient regions or identifies far too many pixels as salient. In these cases, it might be better to use the other maps to determine salient regions. Thus, we analyzed for which kinds of pictures the algorithm returns incorrect saliency maps.

The following conditions on the source frames are indicators of incorrectly calculated image saliency maps:

- There are only few colors and the distance between them is low.
- The distance between colors of salient objects and the background colors is low in general.
- Parts of salient objects have the same color as background colors.

The best results are usually achieved if the color distance is much larger than the number of colors and if the salient objects do not contain colors that occur the most.

The resulting image saliency map from the first video, shown on the left side in Figure 7 (b), detected most of the salient regions. Intuitively, the two persons and the table seem to attract the human visual attention the most; only the head of the person in the background is not highlighted as salient. This is caused by the fact that its color is similar to one of the wall's colors in the background. However, this error is later compensated in the combined saliency map.

The saliency map of the second video, on the right in Figure 7 (b), detected all salient regions with only some small errors in the background. In this example, the person, the flowers, and the lamp seem to be most important. Highlighting parts of the background at the right image border are again an error that will be compensated by the other algorithms when the maps are combined.

Motion Saliency

Figure 7 (c) shows motion saliency maps. Naturally, motion can only be detected in non-static scenes. The results are especially reliable when there is almost no camera movement, which means that the background does not change.

In the first video, the sitting person moves only a little, which can be seen by the fact that the pixel values in these regions are not as high as the ones for the person in the background. The second person is walking through the room which emphasizes the person very well in the motion saliency map.

The second video illustrates the movement of the person behind the flowers very well, whereas the flowers are static and are thus not detected in the motion saliency map.

Depth Saliency

Especially the block matching algorithm and the semi global block matching algorithm are based on many different parameters which can improve or degrade the disparity maps. The *number of disparities* and the *SAD window size* have the most influence on the results. In order to find a possibility to automatically determine these values based on characteristics of the input image, different parameters have been tested.

The images that were used for testing the parameters are taken from the Middlebury Stereo Vision Page[3] [31, 30]. First, block matching is analyzed and the SAD window size was set to a fixed value of 9, and only the horizontal search range, defined by the number of disparities, has been edited. Because of the horizontal search, the width of an image plays an important role in the decision for setting the parameter. The resulting disparity maps of the Aloe data set with a size of 1282×1110 pixels is shown in Figure 8. The first image is the original left view. Figure 8(b) is the disparity map that was created with a search range of 64 pixels, whereas the saliency map in Figure 8 (c) was computed with a number of disparities set to 160. The third disparity map was computed by using 320 pixels as the search range for matches. The last image shown in Figure 8 is the ground truth map, which can be seen as the perfect result that needs to be

[2] `http://sp.cs.tut.fi/mobile3dtv/stereo-video/`

[3] http://vision.middlebury.edu/stereo/

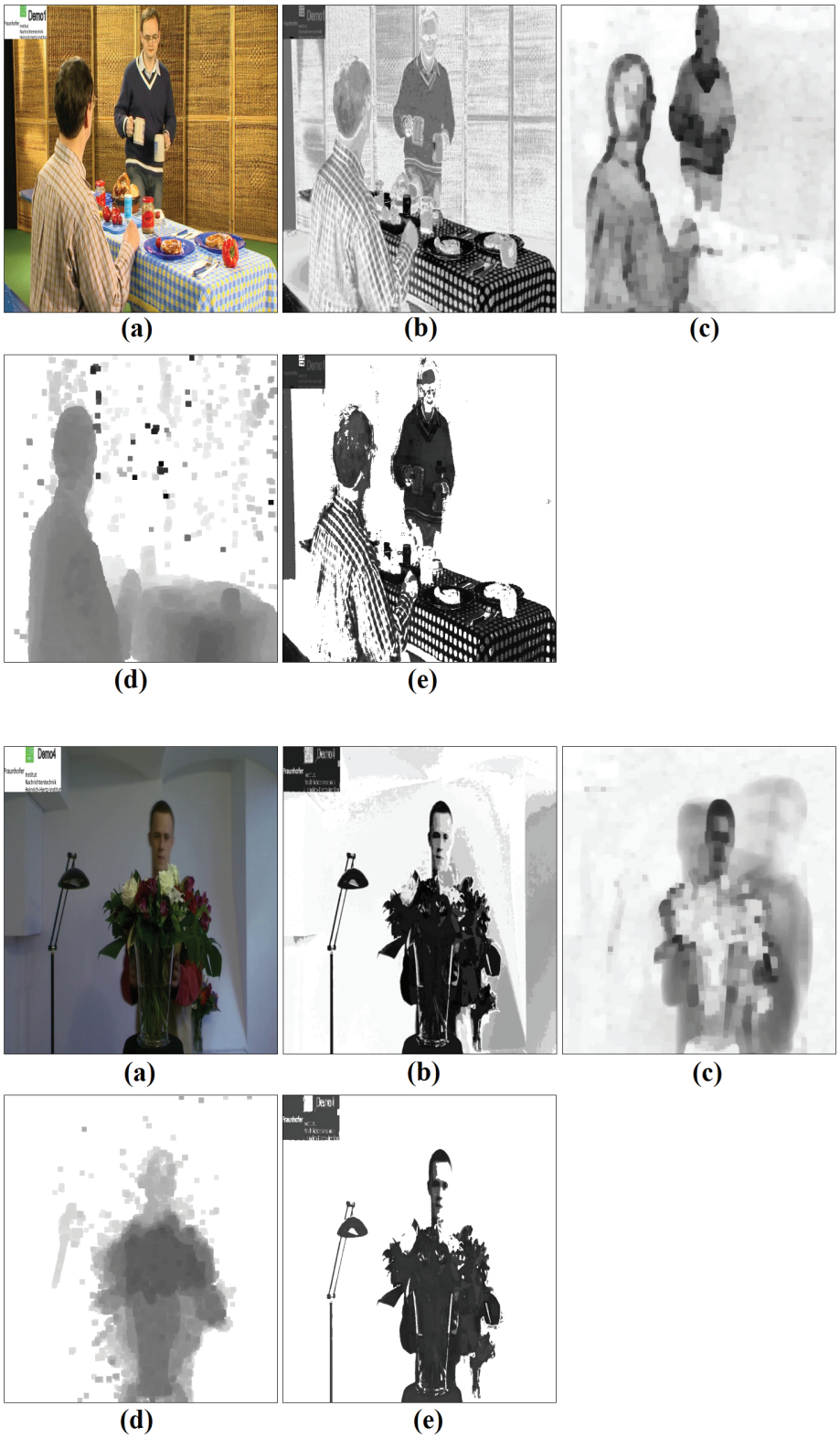
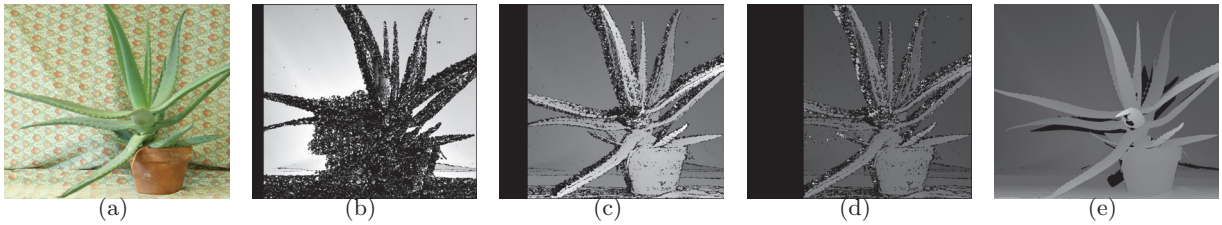


Figure 7: Comparison of saliency maps: Left frame of stereo video (a), image saliency (b), motion saliency (c), depth saliency (d), and combined saliency map (e). Pixels with high saliency values are printed in dark colors.



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

Figure 8: Increasing the number of disparities to achieve better disparity maps. (e) is the ground truth data to compare the disparity maps.

approximated.

The results show that the disparity map in Figure 8 (c) is the best approximation. If the search range is to small, a lot of false matches will be found, which the algorithm will interpret as disparities even if these regions are in the background. Choosing the number of disparities too large will result in quite good disparity maps, but the larger it gets the more of the picture is cut. As a consequence, these regions cannot be detected as salient regions anymore.

Due to the fact that block matching searches horizontally, it can be assumed that the number of disparities varies according to the width of an image. To find a correlation between image width and search range, the block matching algorithm was tested on the Aloe data set with half the size of the one tested before. It can already be estimated that using 160 as the number of disparity (as in the full-sized image) would cut $\approx 25\%$ of the image, which would be a drawback for the saliency detection.

The resulting disparity maps for the smaller Aloe set are shown in Figure 9, in which (a) is the disparity map with a search range of 64. For (b) the search range was set to 80, and in (c) the number of disparities was set to 160. It clearly shows that with 80 pixels, as the search range, the ground truth data can be approximated very well.

The same tests were done with several other images in order to detect a correspondence between the image width and the number of disparities. Based on these sample images, a suitable approximation can be formulated as follows:

$$numDisparities = \begin{cases} \frac{w}{8}, & \text{if } \frac{w}{8} \ mod\ 16 = 0, \\ \lfloor \frac{w}{8} \rfloor + (16 - (\frac{w}{8} \ mod\ 16)), & \text{otherwise.} \end{cases} \quad (15)$$

with w being the width of an image.

The next step is to see if the SAD window size changes the results significantly with the number of disparities set dynamically. As seen before, a fixed SAD size could already be used, but there are some special cases in which the window size needs to be edited. This is usually the case in small images. During several tests, the window size did not have much influence on disparity maps of large images. However, as shown in figure 10, there are some obvious differences between the disparity maps of the respective image.

The size of the Tsukuba image in figure 10(a) is 384×288 pixels; and based on equation 15, the algorithm should set the number of disparities to 48. A search range of 64 would be set for the Cones image in Figure 10(e), which is 450×375 pixels of size.

Figure 10 (b) shows the disparity map with the fixed SAD window size of 9. Even though the objects can be detected quite well, the depth values of them are not high enough. Compared to the ground truth data in Figure 10 (d), for instance, the lamp in the picture should have much higher disparities than those that have been found. If the window size is increased, the disparity values are approximated much better, as Figure 10(c) and (g) shows, in which a window size of 19 was used.

Similarly, the disparity map for the Cones image can be improved with a SAD window size of 19. A drawback of larger values is that they usually result in blurry disparity maps, as the Cones example shows very well. However, the goal of the disparity map in this work is to find image regions with large disparities; this goal is still given with larger window sizes. Furthermore, larger window sizes are more robust to noise, thus, only the strong regions are detected.

To see if the window size is also dependent on the image size, it has been varied for larger images, too. Results can be found in Figure 11 with (b) using a SAD window size of 19 and (c) using 25. This actually shows that, even for the full sized Aloe image set, the window size does not have much effect on the disparity maps except for reducing noise. Hence, for this application it is sufficient to work with a fixed window size for any image size.

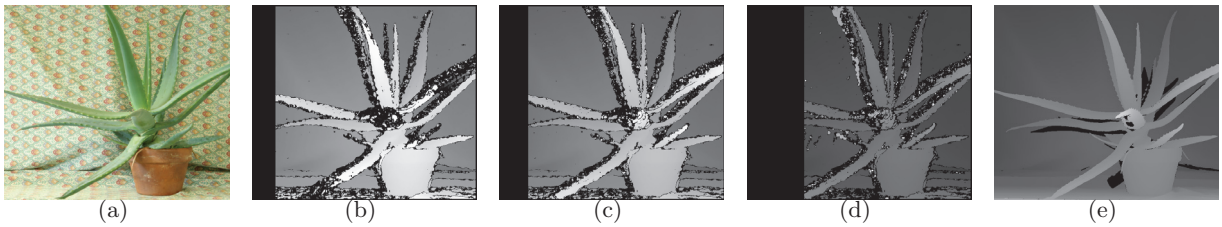
Compared to the block matching algorithm, semi global block matching usually presents much better results. This algorithm is also very dependent on the number of disparities and the SAD window size. In order to see if there are any differences between the choice of parameters for both algorithms, the same input images have been taken and a disparity map has been created.

The computation for the number of disparities has been taken from the block matching algorithm, which was appropriate for this algorithm, too. As Figure 12 shows, the window size needs to be reduced from 19 (12(b)) to 3 (12(c)) to improve the disparity map. This is the minimum size that can be chosen and it has given good results for most images independent of the width and height of an image.

Of all three algorithms that have been tested, the graph cut approach creates the best results. Parameters of this algorithm are the number of disparities and the number of iterations. For the number of disparities, the same computation as for the block matching algorithms is done. As shown in Figure 13, there are no noticeable differences between choosing 2 iterations (13(b)) and running the algorithm with 6 iterations (13(c)).

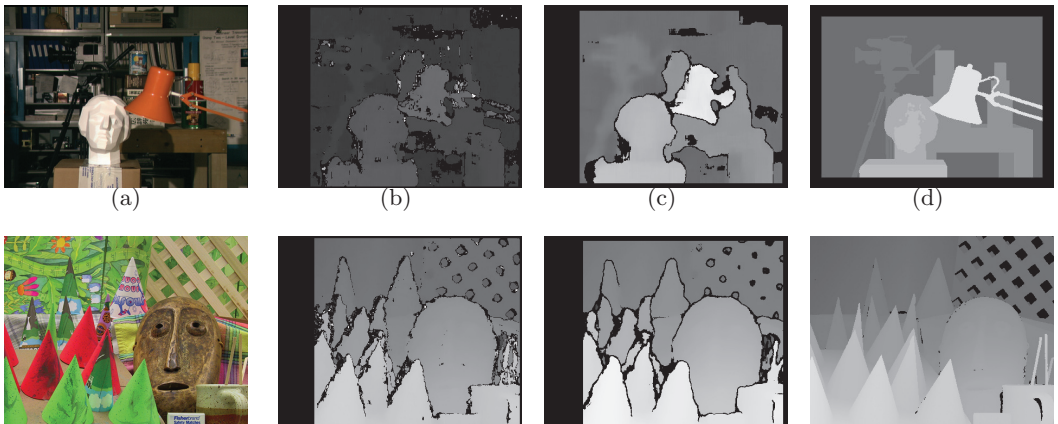
Hence, the number of iterations is set to 2, which is usually sufficient and reduces the runtime a little.

It can be seen that in most cases, all algorithms produce



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

Figure 9: Increasing the number of disparities to achieve better disparity maps with reduced image resolution. (e) is the ground truth data to compare the disparity maps.



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

(e) (f) (g) (h)

Figure 10: Influence of the SAD window size on the disparity map. (d) and (h) is the respective ground truth data

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

Figure 11: Increasing the window size for BM. (d) is the ground truth data to compare the disparity maps

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

Figure 12: Refining the SAD window size for the semi global block matching algorithm, (d) is the ground truth data to compare the disparity maps

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

Figure 13: Resulting disparity maps for the Tsukuba image set created with the graph cut algorithm. (d) is the ground truth data to compare the disparity maps

good results with the correct parameters. However, the data sets provided by Middlebury Stereo Vision have several advantages towards images taken from stereo videos. On the one hand, the colors and objects can be divided very good. This means that the first computation step of disparities can already create good results and the aggregation and final disparity computation can refine the disparity map without many errors. Furthermore, the disparities are very large, and with greater resolution of an image the disparities increase, too. This assumption cannot be taken for stereo videos. With frames of stereo videos, there are additional problems that are not present in the tested data sets. First, the disparities might not be constant during a sequence of images. Moreover, the maximum disparities also differs between different types of films. Especially in computer generated videos, the disparities are usually larger than in image sequences taken with a camera. This is mainly based on the camera setup and as long as the MVC video coding extensions are not used, the number of disparities can only be approximated. Nevertheless, if the computation of the number of disparities gets slightly edited, in most cases, the resulting disparity maps are satisfying and can be used as an approximation for pop out regions.

Hence, Equation 15 has been edited to fit the needs of this work to the following:

$$numDisparities = \begin{cases} \frac{w}{32}, & \text{if } \frac{w}{32} \ mod\ 16 = 0 \\ \lfloor \frac{w}{32} \rfloor + (16 - (\frac{w}{32} \ mod\ 16)), & \text{otherwise.} \end{cases} \quad (16)$$

Even though the number of disparities either is 16 or 32 for the stereo videos that were tested, the algorithms created much better results than with Equation 15. A comparison for one frame can be found in figure 14. The width of the shown image is 960 pixels, thus, the number of disparities would have been set to 128 based on Equation 15.

The resulting disparity map for this value with the semi global block matching algorithm is shown in Figure14(b) where depth is hardly detected. By decreasing the number of disparities, the quality of the disparity map increases. With Equation 16 the value is set to 32 which creates the best result.

In some cases, further decreasing the number of disparities would improve the detected pop out regions even more. However, the maximum disparities vary between different kinds of stereo videos a lot, hence, a further reduction could decrease the quality of disparity maps of videos with very large disparities. Thus, as long as this parameter is not known, Equation16 gives a reliable measure for the number of disparities in most stereo videos. Beyond that, choosing a small value for the number of disparities, reduces the runtime as shown below.

Both block matching algorithms usually have problems with images with large background areas because in these images mismatches are easy to happen. In some cases, even the graph cut algorithm produces noise in similar image areas. That is why in this implementation the resulting disparity maps of all stereo correspondence algorithms are post processed by an erosion and dilation step. The effects of this step are illustrated in Figure 15. The original disparity map that is shown was computed by the semi-global block matching algorithm.

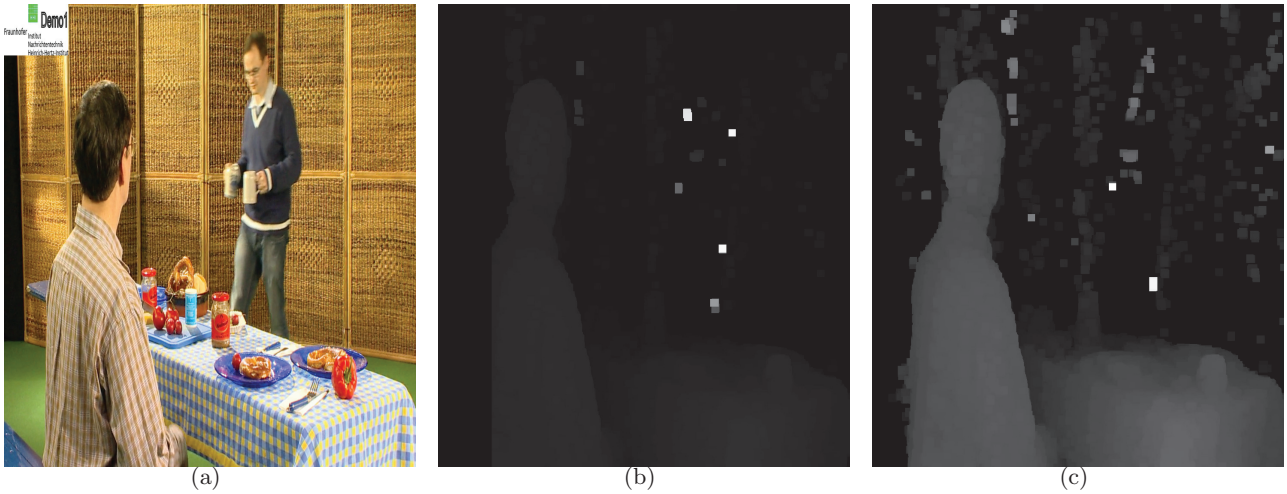
Even though some depth information gets lost, small artifacts caused by false matches are reduced. This does not have much effect on the saliency detection as such small areas can be seen as non-salient. One case that can be thought of as problematic, is an image with rain or small artifacts. Such images are very common in the latest 3D movies but as in these images the algorithm hardly produce usable disparity maps, this case can be neglected.

In Figure 7 (d), sample disparity maps computed by the semi-global block matching algorithm are shows. Typically, the block matching algorithm has problems with images containing large background areas, since in these images mismatches are easy to happen. That is why we post-process the resulting disparity maps by an erosion and dilation step. Due to this step, all major pop-out regions are detected and only small pixel areas in the background are incorrectly highlighted as salient.

Weighted Saliency

Most errors are further reduced by the combination of all saliency maps as the last row in Figure 7 illustrates. The combined map is calculated with the dynamic weights for the image saliency map and the disparity map. In case of the first video, the saliency maps were weighted with 0.5 for image saliency and 0.2 for the disparity map. These values are determined by the algorithm because the number of unique colors and the distance between them is very high. It can be further seen that the noise in the background of the image is reduced when combining the different maps.

The weighted saliency map of the second video in Figure 7 (e) is created with the disparity map weighted by 0.3 and the image saliency map by 0.4. These values are calculated because there are fewer pop-out regions and the maximum



(a) (b) (c)

Figure 14: Changing the number of disparities for stereo videos

color distance is much lower.

In both cases, it can be seen that individual errors in the respective saliency maps are reduced by the combination of all maps. Sometimes, the interior parts of salient objects are not detected correctly. However, this usually does not pose a problem, as the outer contours of the salient objects are easily detected.

In the following, the effects of the weighting process are discussed. The previous results are all created with a static weight for the motion map of 0.3, whereas the image saliency map and the disparity map are weighted dynamically as described in the previous sections. Figure 16 shows some experimental results with differently weighted motion maps. In Figure 16 (a), the motion map was weighted with a factor of 0.2. It clearly shows that some salient regions are missing, e.g., the head of the second person. The image saliency map does not detect the head of the person as salient as its color is too similar to the background. Furthermore, the disparity map only gives information about pop-out regions. Hence, salient objects in the background are not detected. Consequently, weighting the motion map with a factor of 0.2 is not sufficient, as it does not have enough influence to correct the errors of either of the other saliency maps.

Using higher weights for the motion map does not change the final saliency map significantly. Nevertheless, a weight of 0.4 removes some static objects like the logo in the upper left corner, parts of the head of the first person, and some items on the table. This example shows that if the motion map is weighted too high, it might negatively influence the final result by removing static pixels which might be relevant.

Figure 16 (b) shows the resulting saliency map that was automatically computed by our algorithm. All intuitively salient regions are highlighted as salient and the previously seen errors do not occur.

Computational Effort

Table 3 lists the run-time to compute the image and motion saliency maps[4]. It shows that both algorithms perform their computations very fast, and as shown before, they create very good results. Even for frames of high resolution videos, the time consumption is still low enough, so the algorithms are appropriate for many applications.

Table 3: Run-time of the image and motion saliency detection

Frame size	Saliency Detection Algorithms	
	Image Saliency	Motion Saliency
320×480	$0.06s$	$0.02s$
960×1080	$0.25s$	$0.10s$

Table 4 shows the computation time to calculate disparity maps. To compare the computational effort, we also used the graph cut based disparity algorithm presented by Junhwan et al. [12]. The measurement shows that the block matching algorithm significantly outperforms the graph cut algorithm concerning computational effort.

Table 4: Run-time of the disparity detection

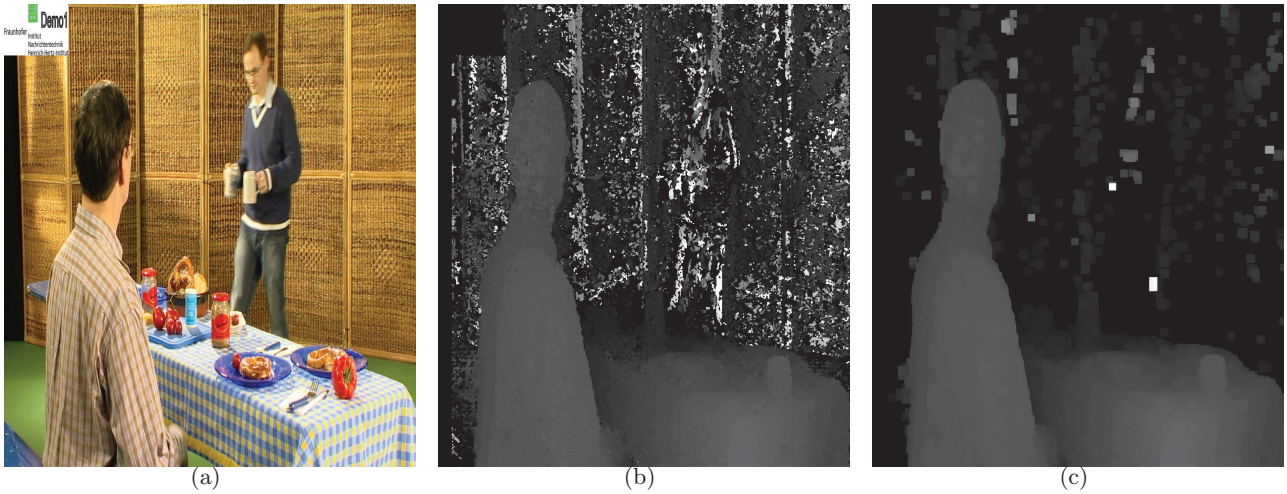
Frame size	Disparity Detection Algorithms	
	Semi Global block matching	Graph Cut
320×480	$0.15s$	$4.86s$
960×1080	$0.92s$	$55.30s$

The algorithm's computational effort mainly depends on the size of a frame. The run-time to completely decode, process (using semi global block matching), and encode one video frame is listed in Table 5.

5.2 Subjective Visual Perception

To measure the quality of the developed algorithm, video examples were shown to 12 test users aged between 21 and

[4]Intel Core 2 Duo 6400 processor with 2.13 GHz and 3 GB RAM.



(a) (b) (c)

Figure 15: Effects of erosion and dilation on disparity maps

Table 5: Run-time of the complete algorithm

Frame size	Run-time for each frame
640×480	$0.45s$
1920×1080	$12.0s$

57. Except for one person who suffered from a dyschromatopsia (red-green color blindness), nobody suffered from any defects in vision. The video sequences that were shown to the users are the Tridelity Demo1 and Demo4 sequences, as well as the Ice Age and Sammy trailers.

All videos were presented either on an Acer HS244HQbmii display with its respective glasses, played with PowerDVD, or on a Samsung SyncMaster 2233 display that uses the Nvidia 3D Vision Kit and the respective Nvidia player. Both displays are capable of 120Hz and use the shutter technology with an infra-red connection to the glasses. For each video, four different versions of the same sequences were shown to the test users, created using differently weighted saliency maps.

The first video presented was always the original video. Next, the test uses saw a version in which the image saliency map was weighted with 70%, the motion saliency map with 30%, and the disparity map was not considered as a factor for saliency. The third version did not consider the image saliency map at all, but weighted the disparity map with 70%. A video generated with the dynamic weights for the combined saliency map was presented as a last version. The users could see the video sequences more than once and were asked to rank the quality of the videos afterwards. All videos were shown without sound as it can actually direct the viewers' attention to specific objects.

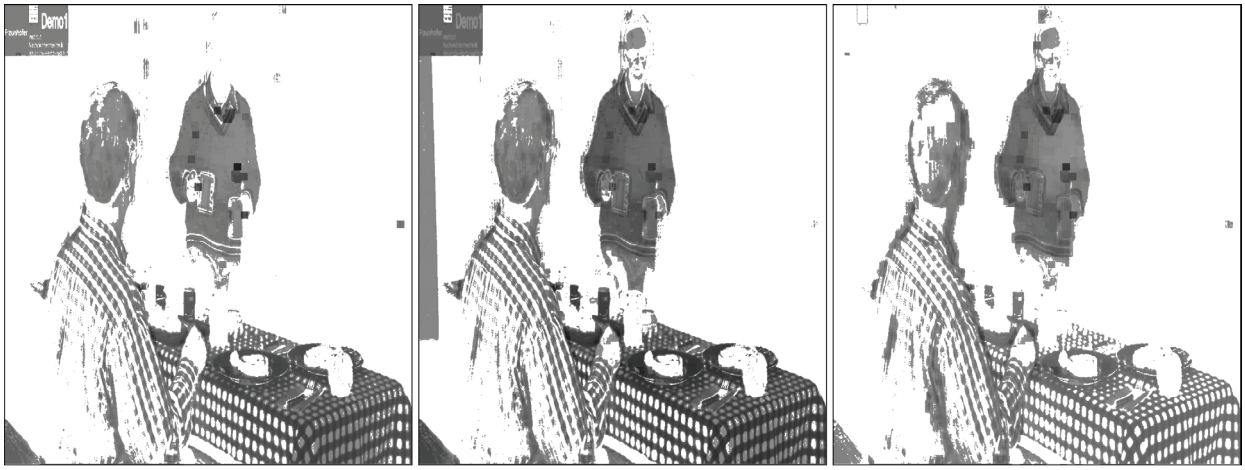
An example of a blurred video frame can be seen in Figure 17. A noticeable difference is the background region around the head of the walking person, which was detected as non-salient and was thus blurred. Also, the same applies to the top region above the head of the sitting person. At first, most users did not notice this modification; however, in this case it became obvious for most test users as soon as the person walks into the scene: At this point, the region is detected as salient by the algorithm, and is hence no longer blurred.

Table 6: Rating of the quality of the different video sequences between 1 (worst) and 5 (best)

Video	Average grades for the different saliency detectors			
	Orig.	Image	Depth	Dynamic
HHI Demo1	3.75	2.92	2.83	2.92
HHI Demo4	4.42	4.00	1.10	3.50
Ice Age	4.33	2.83	1.46	2.25
Sammy's Adv. 2	4.54	3.20	2.80	3.71

Table 6 lists the results of the evaluation. The videos were ranked on a Likert-scale of one to five, where five is the best. The first video (original) was always evaluated the best and the goal was to see how the different versions of the respective video were evaluated. As there are small artifacts in the video or some ghosting effects, the original videos were not rated as perfect by everyone. Our dynamic approach is not compared to Zhang's algorithm [35] because we do not know the camera parameters of the test videos.

The second column shows the results for the videos that were blurred based on the image saliency map and the motion map. Most users rated the blurred videos very well. Solely using the disparity and motion map as indicator for salient regions (third column) does not work very well in many cases; accordingly, this version was rated the lowest for every sample video. This either results from the disparity map sometimes not calculating the depth correctly, or from the fact that regions with strong depth values are often very small. Thus, large parts of the images were detected non-salient and were thus blurred, which was obvious for the users. The grades in the last column are the ones for the algorithm that estimates dynamic weights as discussed in Section 4. The grades of image saliency and the dynamic weights are comparable. The errors in the *HHI Demo4* video and the *Ice Age* video are slightly higher, whereas in the



(a) **(b)** **(c)**

Figure 16: Results of weighting the motion map differently. Chosen weights: $w_m = 0.2$ (a), $w_m = 0.3$ (b), and $w_m = 0.4$ (c).

Sammy's Adventure 2 video sequence, the proposed algorithm got rated best compared to the other edited videos. This video has the largest disparities, and hence, the disparity maps are very accurate in most cases. Choosing a better algorithm to compute disparity maps like Graph Cut improves the overall quality but also increases the computation time by a factor of ten.

We do not normalize the different saliency maps because the saliency value of a pixel should be derived from the content of a frame. E.g., depending on the amount of motion or the occurence of pop-out regions, the motion or depth saliency maps of a frame may only contain low values. Therefore, we decided not to normalize the average number of salient pixels for each detector.

Overall, the users usually did not detect the video sequences' blurred regions in the first iteration, but needed more than one iteration to take notice of them. Analyzing the ratings of the different versions, the ones ignoring the disparity map and the ones created with our algorithm were usually ranked best. However, it has been further found out that adding the disparity map as a measure for saliency can correct errors in the other saliency maps (based on spatial and temporal saliency detection).

Discussion

Even more interesting than the absolute ranking are the results collected from the user's feedback. It has been noticed that the following conditions influence the subjective visual perception of the test users:

1. Routine
2. Stereo video experience
3. Static background
4. Fast cuts
5. Genre
6. Previous knowledge of the video

The first indicator for the detection of modifications/errors in a video sequence is *routine*. Our test users had the possibility to watch the video sequence multiple times. Thus, with each iteration, they could focus on new parts within the scene. Most users actually needed more than one iteration to detect the differences between the videos. Moreover, the users knew what to search for after detecting the first errors. As a consequence, if the first video sequence had certain parts blurred, the user's attention automatically was on these regions in the next video.

The second parameter which influences the perception of stereo 3D videos is the experience a user has with the 3D effect. Since more and more 3D movies for cinemas are produced in recent years, the younger generation is actually used to watching films in 3D. Hence, they do not focus popping out objects as much as people who had no experience with 3D videos. Consequently, we could notice that people who watch 3D movies regularly detected blurred parts in the background much faster than these who have never seen a 3D video before.

The third and fourth condition are closely related: In video sequences with a static background and no changes in the scenery, the test users did have more time to disengage their focus of attention from the salient objects. Thus, they usually detect errors in the background in the first or second iteration already. In contrast, for videos with many short scenes that complete changed the scenery every few seconds, the test users did not have much time to scan the scene. As a result, they only perceived very few of the blurred regions.

The fifth condition that was found to influence the visual perception of the test users is the *genre*. In animated videos, the depth effect is usually much stronger than in live action films. Although filmed scenes can be post-processed to create a stronger depth impression, this can also appear unnatural for the human eye, and is thus most of the times not exaggerated. In animated videos, however, the viewer has no comparison to objects in the real world. Consequently, scenes do not look unnatural even if the disparities are very large.

As a result, the conditions 2 and 5 are also related because people with a lot 3D video experience are more focused to

Figure 17: Left: Original left view frame. Right: Non salient regions are blurred (also left view).

the depth impression in animated videos than in others.

The last condition indicating if a test users can detect blurred non-salient regions is *previous knowledge of the video*. In contrast to the first condition, this means a user already knows the salient objects in the scene. Thus, he or she detected the blurred regions much faster than test users who did not have any previous knowledge about the characters shown in the video. Intuitively, detecting specific characters as salient might be a good approach; however, we observed that only people who did *not* know these characters concentrated on them, and thus detected the background as blurred only in the second iteration.

All these conditions show that it is very hard to tell what is really salient in an image or a video sequence. One conclusion we can derive is that salient regions are highly dependent on the individual, its previous knowledge of presented content of the video, and the previous experience with the depth impression in 3D videos.

6. CONCLUSIONS

In this work, a new approach for saliency detection in stereo videos was presented, which works without the need to know any camera parameters or a depth video stream as additional information. The algorithm calculates three different saliency maps, which are finally merged.

First, a histogram-based approach detects salient regions in still images. Second, the motion within a video sequence is measured and taken as a second indicator for salient regions. As stereo videos add a 3D effect, a disparity map is calculated which shows the pop out regions in a frame, which are used as the third indicator for saliency. The resulting saliency maps is created by merging the individual maps into one. The motion map is weighted statically, whereas the image saliency map and the disparity map are weighted dynamically based on their characteristics and reliability. This way, it is ensured that the depth impression gets more influence on the final saliency map if the video contains large pop out regions.

Even though the disparity maps created with the semi-global block matching algorithm are sometimes very noisy and hardly usable, it is not necessary to use ground truth data or highly complex stereo correspondence algorithms: Our algorithm still creates suitable results, as combining all saliency maps with dynamic weighting can reliably compensate errors of individual saliency maps in most cases.

To verify our results, the calculated saliency maps have been used to create stereo videos in which the detected salient regions are shown with original quality, whereas the non-salient regions are blurred. Several users watched and evaluated these test videos, resulting that saliency maps calculated by our algorithm are actually capable of telling what is salient in a video frame.

Additionally, our tests have shown that in general the saliency implied by the depth information highly depends on the individual person. Especially people with little experience in watching 3D videos are very focused on pop out regions, whereas people who regularly watch stereo videos are used to the 3D impression and are less focused on these regions. One implication is to query the 3D experience of individual users and to derive user specific parameters for depth saliency.

7. REFERENCES

[1] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk. Frequency-tuned salient region detection. In *Computer*

Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 1597 –1604, june 2009.

[2] S. Birchfield and C. Tomasi. Depth discontinuities by pixel-to-pixel stereo. *Int. J. Comput. Vision*, 35(3):269–293, Dec. 1999.

[3] M.-M. Cheng, G.-X. Zhang, N. J. Mitra, X. Huang, and S.-M. Hu. Global contrast based salient region detection. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 409–416, Washington, DC, USA, 2011. IEEE Computer Society.

[4] T. Dittrich, S. Kopf, P. Schaber, B. Guthier, and W. Effelsberg. Saliency detection for stereoscopic video. In *Proceedings of ACM SIGMM conference on Multimedia systems (MMSYS)*, February 2013.

[5] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[6] S. Goferman, L. Zelnik-Manor, and A. Tal. Context-aware saliency detection. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2376 –2383, june 2010.

[7] S. He, J. Han, X. Hu, M. Xu, L. Guo, and T. Liu. A biologically inspired computational model for image saliency detection. In *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, pages 1465–1468, New York, NY, USA, 2011. ACM.

[8] H. Hirschmüller. Stereo processing by semiglobal matching and mutual information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(2):328–341, Feb. 2008.

[9] L. Itti and C. Koch. Computational modelling of visual attention. *Nature reviews. Neuroscience*, 2(3):194–203, Mar. 2001.

[10] J. Kiess, B. Guthier, S. Kopf, and W. Effelsberg. SeamCrop: Changing the size and aspect ratio of videos. In *Proceedings of the 4th Workshop on Mobile Video*, MoVid '12, pages 13–18, New York, NY, USA, 2012. ACM.

[11] J. Kiess, S. Kopf, B. Guthier, and W. Effelsberg. Seam carving with improved edge preservation. In *Proceedings of IS&T/SPIE Electronic Imaging (EI) on Multimedia on Mobile Devices*, volume 7542(1), pages 75420G:01 – 75420G:11, January 2010.

[12] J. Kim, V. Kolmogorov, and R. Zabih. Visual correspondence using energy minimization and mutual information. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, Washington, DC, USA, 2003. IEEE Computer Society.

[13] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions via graph cuts. In *In International Conference on Computer Vision*, pages 508–515, 2001.

[14] S. Kopf and W. Effelsberg. Mobile cinema: Canonical processes for video adaptation. In *Multimedia Systems*, volume 14(6), pages 369–375. Springer, December 2008.

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

[16] S. Kopf, T. Haenselmann, and W. Effelsberg. Robust character recognition in low-resolution images and videos. Technical Report TR-05-002, Department for Mathematics and Computer Science, University of Mannheim, Germany, 2005.

[17] S. Kopf, T. Haenselmann, and W. Effelsberg. Shape-based posture and gesture recognition in videos. In *Proceedings of IS&T/SPIE Electronic Imaging (EI) on Storage and Retrieval Methods and Applications for Multimedia*, volume 5682, pages 114–124, January 2005.

[18] S. Kopf, T. Haenselmann, D. Farin, and W. Effelsberg. Automatic generation of summaries for the Web. In *Proceedings of IS&T/SPIE Electronic Imaging (EI) on Storage and Retrieval Methods and Applications for Multimedia*, volume 5307, January 2004.

[19] S. Kopf, T. Haenselmann, D. Farin, and W. Effelsberg. Automatic generation of video summaries for historical films. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, volume 3, pages 27–30. IEEE Computer Society Press, June 2004.

[20] S. Kopf, T. Haenselmann, J. Kiess, B. Guthier, and W. Effelsberg. Algorithms for video retargeting. *Multimedia Tools and Applications (MTAP), Special Issue: Hot Research Topics in Multimedia*, 51:819–861, January 2011.

[21] S. Kopf, J. Kiess, H. Lemelson, and W. Effelsberg. FSCAV: Fast seam carving for size adaptation of videos. In *Proceedings of the 17th ACM international conference on Multimedia (MM)*, pages 321–330, October 2009.

[22] S. Kopf, F. Lampi, T. King, and W. Effelsberg. Automatic scaling and cropping of videos for devices with limited screen resolution. In *Proceedings of the 14th ACM international conference on Multimedia (MM)*, pages 957–958, October 2006.

[23] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84 – 95, jan 1980.

[24] Y.-F. Ma and H.-J. Zhang. Contrast-based image attention analysis by using fuzzy growing. In *Proceedings of the 11th ACM international conference on Multimedia*, pages 374–381. ACM Press, 2003.

[25] A. Maki, P. Nordlund, and J.-O. Eklundh. A computational model of depth-based attention. *Pattern Recognition, International Conference on*, 4:734, 1996.

[26] V. Navalpakkam and L. Itti. An integrated model of top-down and bottom-up attention for optimizing detection speed. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2049 – 2056, 2006.

[27] N. Riche, M. Mancas, B. Gosselin, and T. Dutoit. 3D saliency for abnormal motion selection: The role of the depth map. In *Proceedings of the 8th international conference on Computer vision systems*, ICVS'11, pages 143–152, Berlin, Heidelberg, 2011. Springer-Verlag.

[28] S. Richter, G. K"uhne, and O. Schuster. Contour-based classification of video objects. In *Proceedings of IS&T/SPIE conference on Storage and Retrieval for Media Databases*, volume 4315, pages

608–618, January 2001.

[29] P. Schaber, S. Kopf, F. Bauer, and W. Effelsberg. Robust digital watermarking in videos based on geometric transformations. In *Proceedings of the international conference on Multimedia (MM)*, pages 1219–1222, 2010.

[30] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, Apr. 2002.

[31] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *Proceedings of the 2003 IEEE computer society conference on Computer vision and pattern recognition*, CVPR'03, pages 195–202, Washington, DC, USA, 2003. IEEE Computer Society.

[32] L. Wixson and M. Hansen. Detecting salient motion by accumulating directionally-consistent flow. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 797–, Washington, DC, USA, 1999. IEEE Computer Society.

[33] Y. Xia, R. Hu, and Z. Wang. Salient map extraction based on motion history map. In *Image and Signal Processing (CISP), 2011 4th International Congress on*, volume 1, pages 427 –430, oct. 2011.

[34] H. Zhang, J. Lei, X. Fan, M. Wu, P. Zhang, and S. Bu. Depth combined saliency detection based on region contrast model. In *Computer Science Education (ICCSE), 2012 7th International Conference on*, pages 763 –766, july 2012.

[35] Y. Zhang, G. Jiang, M. Yu, and K. Chen. Stereoscopic visual attention model for 3D video. In S. Boll, Q. Tian, L. Zhang, Z. Zhang, and Y.-P. Chen, editors, *Advances in Multimedia Modeling*, volume 5916 of *Lecture Notes in Computer Science*, pages 314–324. Springer Berlin / Heidelberg, 2010.

[36] Y. Zhu, N. Jacobson, H. Pan, and T. Nguyen. Motion-decision based spatiotemporal saliency for video sequences. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 1333 –1336, may 2011.