EUROGRAPHICS 2010 / T. Akenine-Möller and M. Zwicker
(Guest Editors)

Volume 29 (2010), Number 2

Two-Colored Pixels

Darko Pavić and Leif Kobbelt

RWTH Aachen University, Germany

Abstract

In this paper we show how to use two-colored pixels as a generic tool for image processing. We apply two-colored pixels as a basic operator as well as a supporting data structure for several image processing applications. Traditionally, images are represented by a regular grid of square pixels with one constant color each. In the two-colored pixel representation, we reduce the image resolution and replace blocks of $N \times N$ pixels by one square that is split by a (feature) line into two regions with constant colors. We show how the conversion of standard mono-colored pixel images into two-colored pixel images can be computed efficiently by applying a hierarchical algorithm along with a CUDA-based implementation. Two-colored pixels overcome some of the limitations that classical pixel representations have, and their feature lines provide minimal geometric information about the underlying image region that can be effectively exploited for a number of applications. We show how to use two-colored pixels as an interactive brush tool, achieving realtime performance for image abstraction and non-photorealistic filtering. *Additionally, we propose a realtime solution for* image retargeting, *defined as a linear minimization problem on a regular or even adaptive two-colored pixel image. The concept of two-colored pixels can be easily extended to a video volume, and we demonstrate this for the example of* video retargeting.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms

1. Introduction

Traditionally, digital images are represented by a regular array of mono-colored pixels (MCPs), i.e. small squares with one constant color each. While this representation has the decided advantage that the global regularity enables a very efficient processing, it also has some obvious disadvantages. The piecewise constant approximation of the continuous color distribution has only linear approximation order and the sampling nature of the representation leads to alias effects in regions with sharp contrast. The number of MCPs (per row or column) not only restricts the *size* of the detail that can be represented but also the *locations* where this detail can be displayed.

In order to address these problems we can generalize the MCP representation. We still split an image into an array $\mathbf{T}_{i,j}$ of square pixels, but a pixel does not have a single color. Instead, each pixel stores a *feature line* $\mathbf{L}_{i,j}$ and two colors $\mathbf{C}^{-}_{i,j}$ and $\mathbf{C}^{+}_{i,j}$ for the two pixel segments on the negative and positive side of the feature line respectively (see Fig. 1, left). This is actually equivalent to the concept of *wedgelets* introduced by Donoho [Don99]. In this paper we prefer to use the term *two-colored pixel* (TCP), since we are not aiming at image compression but rather at exploring TCPs for various image processing operations. Fig. 1 shows a comparison between the MCP and the TCP image representation. Obviously, TCPs much better capture the color discontinuities in the image due to the fact that the feature lines are placed on the high-contrast boundaries. Notice that the resolutions of the shown images are chosen such that the MCP image has three times more pixels than the shown TCP image, since TCPs require three times more memory.

The superiority of the TCPs compared to the MCPs with respect to the capturing capabilities of the features in the images have inspired us to explore the concept of the TCPs in more detail. In this paper we show how to exploit this concept for different image processing tasks. In particular the main contributions are the following:

- We introduce a hierarchical algorithm for fast computation of TCPs from MCP images, which builds the basis for all TCP-based applications. By using a CUDA imple-

Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

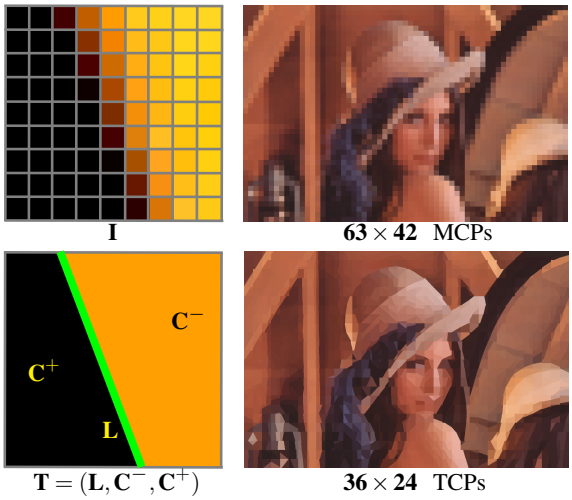


Figure 1: *Left: Example of a TCP* $\mathbf{T}$ *created from a* 9×9 *MCP array* $\mathbf{I}$ *and described by the two colors* $\mathbf{C}^-$ *and* $\mathbf{C}^+$ *and the feature line* $\mathbf{L}$ *(green). Right: Although the MCP image consists of three times more pixels than the TCP image, the latter one still much better captures the detail.*

mentation an additional speed-up of about one order of magnitude is achieved.

- We have developed an interactive brush tool by using the TCP concept as an edge-aware image processing operator. Besides the pure TCP generation in realtime, we also show a number of possible operations which can be derived.
- We show how to exploit the TCPs in order to define image retargeting as solution of a linear minimization problem. By using the contrast of the TCP colors and the minimal geometric information represented by the corresponding feature lines, our method is able to protect important (high-contrast) regions as well as line features from being distorted during the retargeting process.
- We show how to extend the TCP concept to a *two-colored voxel* (TCV) concept in a video volume, where feature lines become *feature planes* by least-squares optimization. This novel concept is demonstrated for the example of video retargeting.

2. Related Work

The underlying concept of two-colored pixels has been explored in different contexts before. In the area of signal processing wedgelets [Don99] or platelets [WN03] were effectively used for image approximation, compression of piecewise polynomial images [SDDV05] or in the context of depth video compression [MMS*09]. Recently, the TCP concept was used in the context of image mosaics [PvCK09] in order to distinguish between feature and non-feature image regions.

Other works use representations similar in spirit to TCPs: the standard pixel representation is enriched with feature information to improve pixel-based algorithms. Bala et al. [BWG03] use what they call *edge-and-point* representation in order to capture depth discontinuities for high-quality rendering. This idea was extended for feature-based textures [RBW04] and later for encoding feature distance functions [PZ08]. Tumblin and Choudhury [TC04] introduced *bixels*, a novel image representation by storing an additional sample feature point in each pixel, which inherits the idea of Dual Contouring used for surface extraction [JLSW02]. Concurrently Pradeep [Sen04] presented an idea similar to bixels, called silhouette maps for texture magnification.

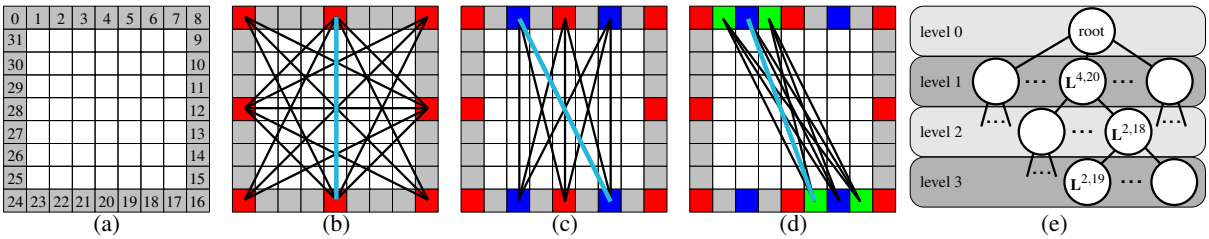
In this paper we adapt the TCP concept and extend previous works by showing how to efficiently compute TCPs on the GPU making them applicable in the context of realtime applications like interactive edge-aware image processing or realtime image retargeting. We also extend the TCP concept to a video volume for the example of video retargeting.

Edge-aware image processing is a topic often addressed in the past [DS02,CPD07,OBBT07,FFLS08,KL08,KLC09]. A well-known operation is edge-preserving filtering. The most famous edge-preserving filter is probably the bilateral filter (BLF) [TM98]. BLF is a nonlinear filter, but when approximated by linear filters in high-dimensional space [PD06] and by using the bilateral grid data structure [CPD07] it can be applied in realtime. Adams et al. [AGDL09] proposed a general acceleration method for non-linear filters. Farbman et al. [FFLS08] discuss the limitations of the BLF and show that their weighted least squares (WLS) framework creates superior results when compared with BLF. Our TCP operator introduces a novel approach to edge-aware image processing and edge-preserving filtering is just one possible application (cf. Fig. 4 for comparison of our method with WLS filtering [FFLS08]).

A number of approaches for content-aware image and video retargeting have been presented in the past. *Seam carving* approaches [AS07, RSA08] remove non-significant pixels in images or videos. These approaches perform very well as long as the seams do not pass through feature regions, which is not avoidable in general. Removing pixels from the feature regions results in visible aliasing artifacts. Warping based approaches are able to create better results in general. We differentiate between *per-pixel warping* and *mesh warping* approaches.

Wolf et al. [WGCO07] introduced per-pixel warping for retargeting where for each pixel position in the source image the target position in the output is computed by solving a least-squares optimization problem. Zhang et al. [ZHM08] improved the performance of this method by using shrinkability maps and a multi-grid solver. Very recently Krähenbühl et al. [KLHGar] embedded the per-pixel warping idea along with a number of interactive tools in their video retargeting system.

Our TCP-based image retargeting falls into the group of mesh warping approaches [GSCO06, WTSL08, KFG09,



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

Figure 2: *Hierarchical approach. After enumerating all boundary pixels of the 9×9 image (a) we set up a hierarchical data structure for all valid lines between them. First, we choose the red boundary samples and all valid feature lines between them are building the first level of the hierarchy (b). The children of the line $\mathbf{L}^{4,20}$ on level 2 of the hierarchy are shown in (c), where additional blue boundary samples were chosen. Analogously we show the children of the line $\mathbf{L}^{2,18}$ on level 3 (additionally used samples in green) (d). The tree structure is visualized in (e). In this example, the hierarchical procedure has to check $16+8+8=32$ candidate lines while the brute force approach would have to check 352 possible feature lines.*

WFS*09]. The idea here is to use a regular quad mesh structure which is overlaid on the image. The retargeting result is then defined as a solution of an energy minimization problem. Karni et al. [KFG09] use a simple gradient measure and the iterative "local-global" optimization [SA07]. Wang et al. [WTSL08] use gradient and saliency maps [IKN98] and propose a nonlinear optimization method which was very recently extended to videos [WFS*09]. Our retargeting method uses the TCP grid as the helping mesh data structure and we define the retargeting problem as a simple linear minimization problem, which is faster than previous approaches. Additionally, we apply image retargeting on adaptive meshes, and, by extending the TCP concept to video, we are able to apply the same warping technique in the video volume for video retargeting.

3. TCP Computation

The core of the TCP concept is the *feature line*, which divides an image tile into two mono-colored segments. We will first introduce the general concept for computing feature lines and thus creating TCPs and then present our hierarchical approach for the efficient computation. Finally we will show how to exploit the CUDA platform for a parallel high-performance implementation of the algorithm.

3.1. General TCP Concept

For a given MCP image $\mathbf{I}$ let $\mathbf{L}(x,y) = a \cdot x + b \cdot y + c$ be the implicit representation of a line $\mathbf{L}$ in image coordinates. We define a TCP of $\mathbf{I}$ to be the 3-tuple $\mathbf{T} = (\mathbf{L}, \mathbf{C}^-, \mathbf{C}^+)$, where $\mathbf{C}^-$ and $\mathbf{C}^+$ are the average colors on the two sides of the line $\mathbf{L}$, computed as follows:

$$\mathbf{C}^- = \frac{1}{n^-} \cdot \sum_{L(x,y)<0} \mathbf{I}_{x,y} \;, \quad n^- = \sum_{L(x,y)<0} 1 \;, \qquad (1)$$

$$\mathbf{C}^+ = \frac{1}{n^+} \cdot \sum_{L(x,y)\geq 0} \mathbf{I}_{x,y} \;, \quad n^+ = \sum_{L(x,y)\geq 0} 1 \;. \qquad (2)$$

Here the values $\mathbf{I}_{x,y}, \mathbf{C}^-, \mathbf{C}^+ \in [0,1]^3$ are three-dimensional RGB-color vectors and n^- and n^+ define the number of pixels on the negative and on the positive side of the line $\mathbf{L}$ respectively. The feature line is chosen such that the following approximation error (energy) $E(\mathbf{I},\mathbf{L})$ is minimized [PvCK09]:

$$E(\mathbf{I},\mathbf{L}) = \sum_{L(x,y)<0} \left\| \mathbf{I}_{x,y} - \mathbf{C}^- \right\| + \sum_{L(x,y)\geq 0} \left\| \mathbf{I}_{x,y} - \mathbf{C}^+ \right\| \;. \qquad (3)$$

For $\mathbf{T}$ we define the *contrast value* $K(\mathbf{T})$ to be the maximal difference between color values in the RGB-channels of the corresponding average colors. For $\mathbf{C}^- = (r^- \;\; g^- \;\; b^-)^T$ and $\mathbf{C}^+ = (r^+ \;\; g^+ \;\; b^+)^T$ we obtain:

$$K(\mathbf{T}) = \max\{\|r^- - r^+\|, \|g^- - g^+\|, \|b^- - b^+\|\} \;. \qquad (4)$$

The contrast value defines the *importance* of the feature line or in other words: the higher the contrast, the sharper is the color discontinuity of the underlying image signal.

3.2. Hierarchical Approach

Let a square image $\mathbf{I}$ of $N \times N$ MCPs be given that should be converted into a single TCP. The task is then to find the optimal feature line $\mathbf{L}$ such that the approximation error (3) is minimized. If we restrict the precision of the line $\mathbf{L}$ to pixel precision then we can encode each line by the two boundary pixels where it enters and leaves the image $\mathbf{I}$.

More precisely, let $b_0, \ldots, b_{4N-5}$ be the boundary pixels of $\mathbf{I}$ enumerated in clockwise order. Then the line $\mathbf{L}^{i,j}$ connects the two boundary pixels b_i and b_j and thus splits the TCP into two regions. If we do not make any a priori assumptions about the color distribution within $\mathbf{I}$, then we have to check $O(N^2)$ candidate lines $\mathbf{L}^{i,j}$ in order to find the line that minimizes the equation (3). Since this leads to an infeasible computation complexity, we propose a hierarchical heuristic to find a good feature line. Even if we cannot guarantee that the optimal solution will be found, in practice we will obtain a solution $\hat{\mathbf{L}}$ with $E(\mathbf{I},\hat{\mathbf{L}})$ close to the global optimum.

The basic assumption of our method is that significant color discontinuities can be detected at rather coarse image resolutions and that in the vicinity of the optimum, the energy (3) is convex. The second statement can be justified by

the observation that, in non-degenerate configurations, if we shift the optimal feature line slightly, the energy (3) increases monotonically. The closest local minimum can be expected to have a safe distance to the global minimum. Otherwise the color discontinuity would not have been significant.

Our strategy is to first compute a candidate feature line on a coarse level and then to iteratively refine the location of that line. Here, the coarseness of the levels is controlled by only considering a subset of the boundary pixels b_i. Coarsening the pixel resolution of the tile $\mathbf{I}$ by some low-pass filter is not necessary since the energy (3) is based on the computation of an average color anyway.

Let $b_{i\cdot 2^k}$ be the coarsest level boundary pixels where $k+1$ defines the number of hierarchy levels. In the first step we compute the energy (3) for each possible feature line $\mathbf{L}^{i\cdot 2^k, j\cdot 2^k}$ and choose the one with minimal energy. On the next hierarchy level, we select the boundary pixels $b_{i\cdot 2^k}$ and $b_{j\cdot 2^k}$ as well as their neighbors $b_{i\cdot 2^k \pm 2^{k-1}}$ and $b_{j\cdot 2^k \pm 2^{k-1}}$. These six boundary pixels define a set of 9 candidate lines, among which we again look for the one with minimal energy. This recursion is continued until we reach the finest level. See Fig. 2 for an illustration of the hierarchical refinement. If we start with a coarsest level that has only 8 active boundary pixels then we have to check 16 candidate lines in the first step and then 8 on each hierarchy level (since one has already been computed on the previous level).

3.3. CUDA-based Implementation

Since the evaluation of (3) is essentially accumulation of pixel colors, it can be implemented on the GPU in a very efficient manner. In order to avoid the recursive refinement in the hierarchical approach, we build a pre-computed data structure that unfolds the recursion into a tree traversal. Notice that this tree has to be computed only once for a given resolution $N \times N$ and not for each TCP conversion.

The needed accumulation operation can be implemented by using CUDA *parallel reduction* kernels [Har07, NBGS08]. With our hierarchical data structure the execution of these reduction kernels can be done in parallel for a number of TCPs (depending on the available GPU-memory on the CUDA device) per level of the hierarchy.

4. Interactive Brush Tool

We have implemented an interactive edge-aware image processing tool with a simple brush metaphor where the TCP concept is used as a basic operator. The user can select a shape (square or circular) and size of the brush as well as the resolution $N \times N$ of the TCP operator and then apply it to the input image. Moving such a brush across the input MCP image results in generating a number of TCPs inside the brush region (please see the accompanying video). The TCP computation is always based on the original MCP input image. Regarding the distribution of the TCPs inside the brush, the user can choose between the *regular* or the *spraying* brush. For the regular brush we apply regular subdivision of the brush region in $N \times N$ TCPs. For the spraying brush we randomly distribute the TCPs across the brush region (see Fig. 3).

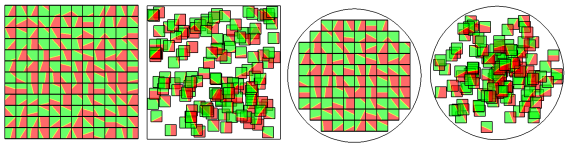


Figure 3: *Brush tool examples. From left to right: square regular, square spraying, circular regular and circular spraying (radial TCP distribution) brushes.*

Next we will explain the different modes for applying our TCP operator and then a number of different image operations we have implemented based on these modes.

4.1. TCP Operator Filter Modes

With our efficient procedure to convert an $N \times N$ MCP sub-image $\mathbf{I}$ into a TCP $\mathbf{T} = (\mathbf{L}, \mathbf{C}^+, \mathbf{C}^-)$ (Section 3), we can now define powerful filter operations on images. In order to be able to predict the result of an operation, we first have to better analyse the elements and properties of TCPs.

- The feature line $\mathbf{L}$ provides a linear approximation to the strongest color discontinuity in the sub-image $\mathbf{I}$. Hence it can be used for edge enhancement, e.g. by rendering it as a line of a certain thickness.
- The two colors $\mathbf{C}^+$ and $\mathbf{C}^-$ represent average, i.e. low pass filtered, colors and hence provide noise reduced color information. However, since the two regions are separated by the strongest local color discontinuity, this low pass filter does not blur over the discontinuity. This effect can be exploited for edge-preserving filtering.
- The local contrast value $K(\mathbf{T})$ can be understood as a measure of the strength of the discontinuity. This information can be used to implement non-linear filters that adapt the filter characteristics depending on the contrast.

The effect of an operation is controlled by the way how the information from all TCPs overlapping a pixel is combined to determine the pixel color of the output image. Here we can distinguish several different modes:

- We can compute a (weighted) average of the overlapping TCPs, or we can select just the most dominant one, e.g., based on the contrast value $K(\mathbf{T})$.
- We can either use the entire square TCP or we can restrict its support to a circular disc to avoid anisotropic effects, or we can restrict its support to a strip along the feature line to enhance discontinuities.

4.2. Edge-aware Operations using TCP Operator

Let $\mathbf{I}$ be the $N \times N$ input MCP image corresponding to a specific TCP $\mathbf{T} = (\mathbf{L}, \mathbf{C}^-, \mathbf{C}^+)$ and let $\mathbf{J}$ be the corresponding $N \times N$ output array. The $\mathbf{J}_{x,y}$ are 4D-vectors

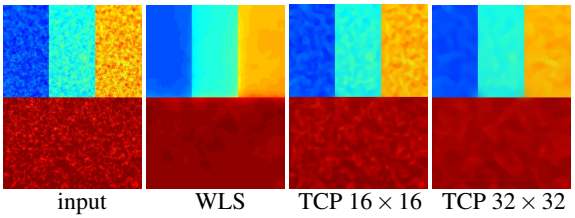


Figure 4: *Comparison of the WLS filtering [FFLS08] and our TCP-filtering. Intuitively, the used TCP size determines the minimal feature size to be preserved during the filtering operation. The input synthetic image as well as the WLS solution are taken from [FFLS08].*

$(J^r_{x,y}\ J^g_{x,y}\ J^b_{x,y}\ J^a_{x,y})^T$, where the *rgb*-channels store the final colors of the specific operation and the content in the a-channel depends on the used filter.

Depending on the combination of the modes described in Section 4.1 we distinguish between different filters and different rendering domains. We apply either a *maximum filter* or an *averaging filter*, depending on whether the most dominant value is chosen or the average is computed. For rendering we use either the *full* or the *line rendering* domain, depending on whether the full TCP area (square or circular) is regarded or only a strip along the feature line.

Maximum filter: We use the contrast value $K(\mathbf{T})$ as a priority value for the maximum filter. The a-channel of $\mathbf{J}$ is used for storing the maximal contrast value and the other three channels are used for storing the RGB output color. For each $\mathbf{J}_{x,y}$ we check if the contrast value of the currently explored TCP $\mathbf{T}$ is higher than the stored one. If $K(\mathbf{T}) > J^a_{x,y}$ then:

$$\mathbf{J}_{x,y} = \begin{cases} (r^-\ g^-\ b^-\ K)^T, & \mathbf{L}(x,y) < 0 \\ (r^+\ g^+\ b^+\ K)^T, & \mathbf{L}(x,y) \geq 0 \end{cases} \tag{5}$$

Averaging filter: This filter does averaging over all contributing TCP colors instead of choosing only one among them. Here the a-channel of $\mathbf{J}$ is used for storing the number of accumulated colors and the *rgb*-channels sum up the corresponding color contribution, i.e., we get:

$$\mathbf{J}_{x,y}\ +\!= \begin{cases} (r^-\ g^-\ b^-\ 1)^T, & \mathbf{L}(x,y) < 0 \\ (r^+\ g^+\ b^+\ 1)^T, & \mathbf{L}(x,y) \geq 0 \end{cases} \tag{6}$$

Line rendering: Instead of updating each value $\mathbf{J}_{x,y}$ (full rendering domain) we can choose to update only values lying inside a stripe defined by the thickness value l (given in % of the TCP size N), i.e., $\mathbf{J}_{x,y}$ is updated only if $|\mathbf{L}(x,y)| \leq 0.5 \cdot l \cdot N$.

After applying one of the above operations we generate the final output RGB-color of the pixel (x,y) on the screen as $(J^r_{x,y}\, J^g_{x,y}\, J^b_{x,y})^T$ for the maximum filter or as $(\frac{J^r_{x,y}}{J^a_{x,y}}\ \frac{J^g_{x,y}}{J^a_{x,y}}\ \frac{J^b_{x,y}}{J^a_{x,y}})^T$ for the averaging filter. Additionally we use two other implementations of the above filters: For computing grayscale images we replace the colors $\mathbf{C}^-$ and $\mathbf{C}^+$ with the color vector $(1-K\ 1-K\ 1-K)^T$ in the above computations (5) and (6).

4.3. Discussion and Results

We have tested our algorithm on an Intel 3GHz CPU with 4GB RAM and a NVIDIA GeForce GTX280 graphics card. In Table 1 we show a comparison for TCP computation between the naive and our hierarchical approach implemented on the CPU as well as with CUDA on the GPU. The hierarchical CPU method is already very fast, but the CUDA implementation is especially adept at handling large brush sizes, with a speed improvement of one order of magnitude.

Using 16 × 16 TCP resolution			
brush size	naive CPU	hier. CPU	hier. CUDA
2 · 16	13.3 fps	422 fps	820 fps
4 · 16	3.3 fps	106 fps	488 fps
8 · 16	0.8 fps	26 fps	202 fps
16 · 16	0.2 fps	6.6 fps	66 fps
32 · 16	0.05 fps	1.7 fps	18 fps

Table 1: *Comparison of different TCP implementations.*

In Fig. 4 we compare our TCP-based averaging filter with the WLS approach of Farbman et al. [FFLS08] on a synthetic image. Although both approaches preserve the feature lines very well there are some differences. First, WLS tends to create piecewise constant regions, whereas with our method piecewise linear regions are created due to the nature of the TCP averaging process. Second, WLS creates gradual segments in the vicinity of the feature lines, which results in a bleeding effect. Our TCP-based filtering does not produce such artifacts, but it introduces edges not present in the input, which best approximate the underlying, noisy area in the image. We can conclude that the decision which filtering method to use is strongly application dependent.

In Fig. 5 we show several examples for different effects which can be achieved with our interactive brush tool. Applying our averaging filter results in edge-preserving filtering (Fig. 5 (b)). Line rendering with the averaging filter can be used for edge-enhancement (Fig. 5 (i)) or for creating grey-valued drawings which expose the structural information in the images (if wished on different scales by using different TCP sizes) (Fig. 5 (c)). By using the maximum filter we create nice painterly image abstractions (Fig. 5 (e), (f)). Finally, by combining the averaging and the maximum filter we can achieve the "structure highlighting" effect (see Fig. 5 (l), (j)).

When using line rendering it may happen that some pixels of the output do not receive any color contribution from any TCP (see Fig. 5 (e)). Such pixels belong to low contrast regions. We can fill in smooth color information into those "holes" by using a simple color diffusion technique known from image inpainting [BSCB00] (see Fig. 5 (f)).

The presented variety of effects shows that the TCP operator can be applied very effectively in the context of edge-aware image processing. In the next section we will show how to exploit the TCP concept in order to provide a novel,

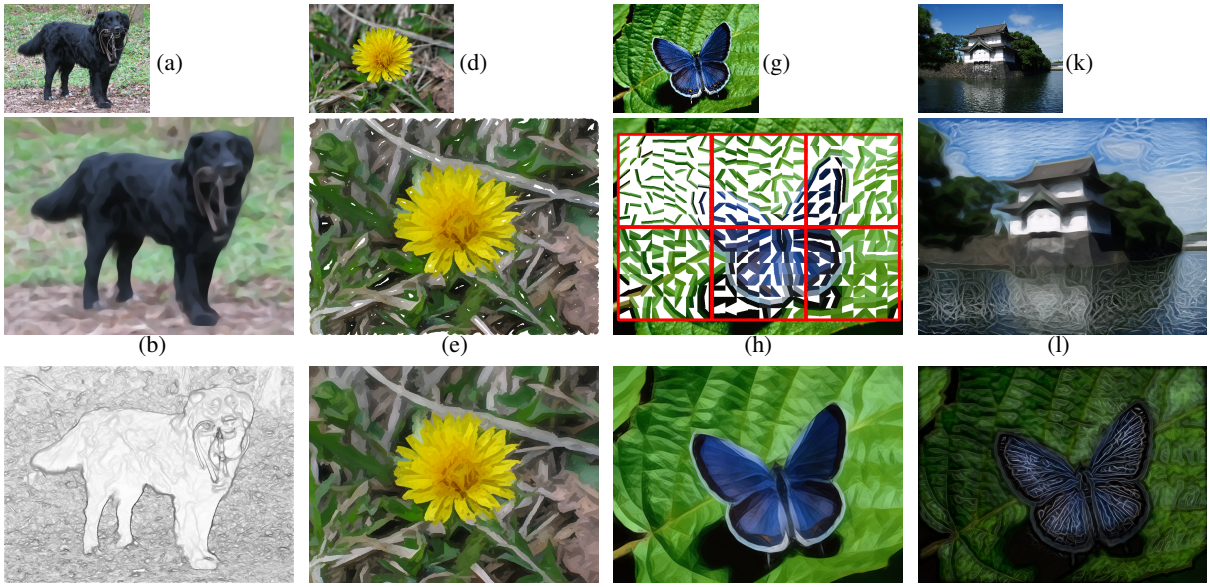


Figure 5: *Examples created with our interactive TCP brush tool (unless otherwise noted 32×32 TCPs are used): Input images (a), (d), (g), (k). Averaging filter (b). Averaging filter with grey-valued line rendering and different TCP sizes ($2^k \times 2^k, k = 2, ..., 5$) for capturing edge features on different scales (c). Line rendering with maximum filter before (e) and after image inpainting (f). In (h): visualization of brushes with different line thickness values $l \in \{25 + i \cdot 15\% | i = 0, ..., 5\}$) used. For edge enhancement in (i) we use, e.g., $l = 25\%$ and $l = 100\%$ and our averaging filter. The images in (j) and (l) show two different examples where we combine grey-valued line rendering and averaging filter in order to create "structure highlighting" in the final output.*

fast solution to the image retargeting problem and then extend the concept for the example of video retargeting.

5. TCP-based Image Retargeting

For content-aware image retargeting, we allow for arbitrary retargeting of an input image. In the "regions of interest" the aspect ratios should be preserved as well as possible, which means that these regions are only allowed to scale (nearly) uniformly. The distortion in the rest of the image can be arbitrary, but it should be equally distributed. We adapt the idea used in some previous works [GSCO06, WTSL08, KFG09] and overlay a mesh on the image. Hence, image retargeting becomes mesh deformation.

We first compute a regular TCP representation of the input image which defines the quad mesh $M = (V, E, T)$, where $V = \{\mathbf{v}_i \in \mathbb{R}^2 | i = 0, ..., n-1\}$ is the set of grid vertices, $E \subseteq \{(i, j) | \mathbf{v}_i, \mathbf{v}_j \in V\}$ is the set of grid edges and $T = \{\mathbf{T}_k\}$ is the set of TCP 3-tuples which correspond to the quad mesh faces. In order to deform the mesh we have to move the vertices $\mathbf{v}_i$ to some new positions $\mathbf{v}'_i$. This mapping will be defined as a solution of a linear minimization problem, derived from the feature-aware structure of the TCPs.

5.1. Deformation Energy

Our approach is inspired by the method of Wang et al. [WTSL08] who define the distortion energy per quad. Instead we define the distortion energy per edge, which simplifies the following computations. We measure the distortion of an edge by the deviation from being uniformly scaled after the retargeting. For an edge $e = (i, j) \in E$ the rotational deformation energy is:

$$D(e) = \|(v'_i - v'_j) - s_e \cdot (v_i - v_j)\|^2 \tag{7}$$

The optimal scaling factors s_e can be computed with the following simple partial derivative computation:

$$\frac{\partial D(e)}{\partial s_e} = -2 \cdot (v_i - v_j)^T \left[(v'_i - v'_j) - s_e \cdot (v_i - v_j)\right] \tag{8}$$

$$\frac{\partial D(e)}{\partial s_e} \stackrel{!}{=} 0 \;\Rightarrow\; s_e = \frac{(v_i - v_j)^T \cdot (v'_i - v'_j)}{\|(v_i - v_j)\|^2} \tag{9}$$

Now, in contrast to Wang et al. [WTSL08], we insert the scaling factor (9) into (7) and get the final formulation for the deformation energy of the given edge e:

$$D(e) = \|(v'_i - v'_j) - \frac{(v_i - v_j)^T \cdot (v'_i - v'_j)}{\|(v_i - v_j)\|^2} \cdot (v_i - v_j)\|^2 \tag{10}$$

which is quadratic in the unknown target locations $\mathbf{v}'_i$. However, this energy can be minimized by solving a sparse linear system. The total deformation energy for all grid edges is simply the sum over all edge deformation energies, namely:

$$D_{grid} = \sum_{e \in E} D(e) \tag{11}$$

By only using the energy in equation (11) the mesh would always scale uniformly. Previous retargeting approaches de-

fine an importance function, which is used for weighting of the individual energy terms. The higher the importance is, the higher the weight, and thus less deformation will be allowed in areas of high importance. Often gradient magnitude is used as an importance measure [AS07, RSA08, KFG09]. Wang et al. [WTSL08] use a more sophisticated measure which combines gradient and saliency maps. In our method the importance is defined by the TCPs in two different ways. Similar to previous works we also use a scalar weighting factor, which is in our case the contrast K of the TCP. Additionally, in our formulation we also exploit the directional information stored in the feature lines of the TCPs.

In order to embed the TCP information into the problem formulation the idea is to add feature lines of the TCPs as additional (virtual) edges in our mesh. The vertices of the feature lines do not introduce new unknowns in our problem, but instead they additionally constrain the grid vertices.

Let the feature line $\mathbf{L}_k$ of the TCP $\mathbf{T}_k$ be defined by the vertices $\mathbf{w}_0^k$ and $\mathbf{w}_1^k$. These vertices are lying per construction on two different grid edges $e_0^k = (i_0^k, j_0^k) \in E$ and $e_1^k = (i_1^k, j_1^k) \in E$. Hence, the feature line vertices can be expressed through linear combinations of grid vertices:

$$\mathbf{w}_m^k = \alpha_m^k \cdot \mathbf{v}_{i_m^k} + (1-\alpha_m^k) \cdot \mathbf{v}_{j_m^k}, \;\; \alpha_m^k \in [0,1], \;\; m \in \{0,1\} \quad (12)$$

We can define the deformation energy of the feature line $D(L_k)$ analogously to how we have defined the deformation energy for the grid edges in equation (10). The total energy of all feature lines is defined by the following equation:

$$D_{feature} = \sum_{\mathbf{T}_k \in T} K(\mathbf{T}_k) \cdot D(L_k) \quad (13)$$

The deformation energy per feature line is weighted with the contrast of the corresponding TCP, which is our scalar importance measure.

5.2. Relaxation Energy

In order to give the user more control over the retargeting process we also introduce a relaxation energy term. Intuitively the relaxation should push the mesh towards the linearly scaled solution. Let $\bar{\mathbf{v}}_i$ be the vertex positions of the uniformly scaled mesh. For the edge $e = (i, j) \in E$ we define the deviation of the edge scaling factor s_e from the uniform scaling $\|\bar{\mathbf{v}}_i - \bar{\mathbf{v}}_j\| / \|\mathbf{v}_i - \mathbf{v}_j\|$ to be the relaxation energy per edge. Therefore we define the total relaxation energy for grid edges and feature lines to be:

$$D_{relax,grid} = \sum_{e=(i,j)\in E} \left\| s_e - \frac{\|\bar{\mathbf{v}}_i - \bar{\mathbf{v}}_j\|}{\|\mathbf{v}_i - \mathbf{v}_j\|} \right\|^2 \quad (14)$$

$$D_{relax,feature} = \sum_{\mathbf{T}_k \in T} \left\| s_{\mathbf{L}_k} - \frac{\|\bar{\mathbf{w}}_k^0 - \bar{\mathbf{w}}_k^1\|}{\|\mathbf{w}_k^0 - \mathbf{w}_k^1\|} \right\|^2 \quad (15)$$

Finally, the total relaxation energy is simply the sum:

$$D_{relax} = D_{relax,grid} + D_{relax,feature} \quad (16)$$

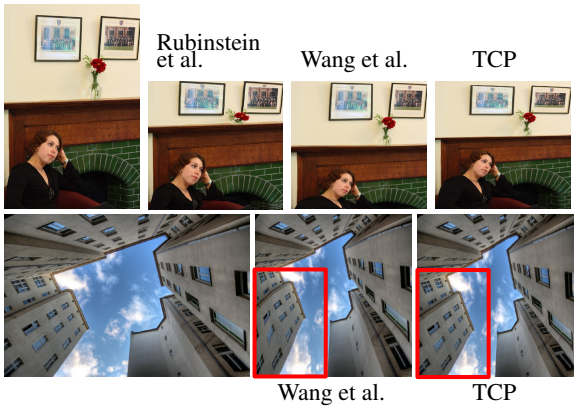


Figure 6: *Comparison of our image retargeting solution with previous works (images taken from [WTSL08]). In the top row seam carving of Rubinstein et al. [RSA08] destroys the vase and the approach of Wang et al. [WTSL08] removes the girl from the focus of the image. In our case we preserve the girl in the focus as well as the vase. In the bottom row we show that our approach is able to preserve line features much better than the approach of Wang et al. due to the exploited feature line information in the TCP representation.*

5.3. Linear Minimization

In order to find the optimally deformed mesh we solve the minimization problem for the following quadratic, deformation energy functional:

$$D = D_{grid} + \lambda_{feature} \cdot D_{feature} + \lambda_{relax} \cdot D_{relax} \quad (17)$$

We usually set $\lambda_{feature} = 10$ and $\lambda_{relax} = 1$. The solution of the minimization problem is found by solving a sparse linear system $A \cdot X = b$ in the least squares sense, where the vector of unknowns $X = (\ldots v'_{i,x} \ldots v'_{i,y} \ldots)^T$ consists of the x- and y-components of the target vertex positions $\mathbf{v}'_i = (v'_{i,x} \;\; v'_{i,y})^T$, and each row of the matrix A corresponds to a partial derivative of the function D.

Let $w \times h$ be the input image resolution and $w' \times h'$ the target image resolution after the retargeting. We want to preserve the rectangular form of the image during the retargeting step. Therefore we solve our linear system subject to the following boundary constraints:

$$v'_{i,x} = \begin{cases} 0, & if \;\; v_{i,x} = 0 \\ w', & if \;\; v_{i,x} = w \end{cases}, \quad v'_{i,y} = \begin{cases} 0, & if \;\; v_{i,y} = 0 \\ h', & if \;\; v_{i,y} = h \end{cases} \quad (18)$$

For solving the system we use the constrained solver proposed recently by Bommes et al. [BZK09]. In contrast to the method of Wang et al. [WTSL08] the unknowns in x- and y-direction are coupled in our case due to equation (10). This means that we have to solve a system with twice as many unknowns, but: 1) our system is only linear and 2) we have to factorize the matrix only once and for the rest of the retargeting procedure we only have to adjust the constraints (18) and the right-hand side of the system. This makes our method applicable for interactive deformation of very fine

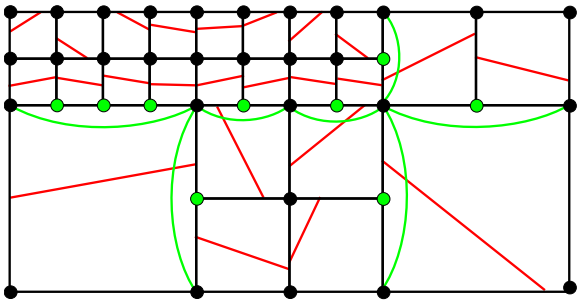


Figure 7: *Adaptive TCP grid for image retargeting. Black vertices represent the unknowns. Green vertices are additional constraints to our system, represented as linear combinations of the end vertices of the corresponding long edges (depicted as green arcs for better illustration).*

meshes (please see the accompanying video), whereas Wang et al. [WTSL08] have to use rather coarse meshes for interactive deformation.

In Fig. 6 we show a comparison of our method and approaches of Rubinstein et al. [RSA08] and Wang et al. [WTSL08]. In the "girl and vase" example we see that our approach nicely distributes the deformation so that the important objects like the vase are preserved, but also the focus on the girl is not lost. The "building" example shows that our approach is better in preserving line features which is one of the main advantages of our TCP based image retargeting.

Decreasing the TCP size results in decreasing feature awareness of our method. In the limit case where only one pixel is used for computing the corresponding TCP imagine that we have one arbitrary feature line and two equal average colors. In this case we get uniform scaling of the image. Also for this reason we introduce an adaptive retargeting method in the following section.

5.4. Adaptivity

Using a regular TCP grid as discussed so far makes the method easy to describe and to implement. But with an adaptive TCP grid we could use more samples (grid vertices) in regions of higher importance and thus control deformations in these regions on a finer scale allowing for stronger distortions in the more sparsely sampled, homogeneous regions. For this reason we propose an adaptive approach for image retargeting.

We start with a regular TCP grid; then by using a given error threshold θ we create a quadtree hierarchy by recursive 1-4 subdivision of each TCP for which the approximation error (3) lies above θ. The subdivision is continued until we reach the finest TCP level.

Once we have the adaptive TCP representation we can easily create an adaptive mesh $M_a = (V_a, E_a, T_a)$ from it. Each TCP $\mathbf{T} \in T_a$ is spanned by four vertices in V_a (we write $\mathbf{v}_i \in \mathbf{T}$ for such a vertex). Note that M_a is not a quad mesh

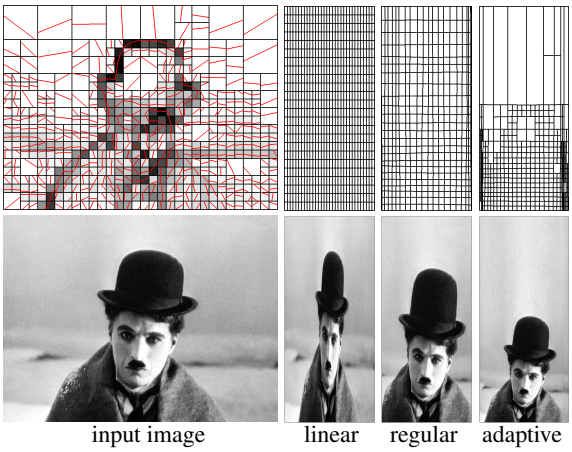


Figure 8: *Comparison of the regular and adaptive retargeting approach for extreme retargeting to less than a third of the width of the input image. In the top left image we visualize the used adaptive TCP grid, where the grey shading represents the contrast values.*

in general (see Fig. 7). Still, all its faces are by construction quad-shaped polygons and we want to force them to stay quad-shaped also after the deformation. For this reason we first construct the set E_{long} of so-called *long edges* (green arcs in Fig. 7). These are edges which connect two vertices of a TCP, but are not topological edges of the mesh. Formally:

$$E_{long} = \{e = (i,j) \mid \exists \mathbf{T} \in T_a : v_i, v_j \in \mathbf{T},\ e \notin E_a\} \quad (19)$$

A vertex $\tilde{\mathbf{v}} \in V_a$ which lies in the interior of a long edge $e_l = (i_l, j_l) \in E_{long}$ can be represented by a linear combination of the vertices $\mathbf{v}_{i_l}, \mathbf{v}_{j_l} \in V_a$ with a $\beta \in [0,1]$ (green vertices in Fig. 7). This linear combination is an additional constraint to our system:

$$\tilde{\mathbf{v}} = \beta \cdot \mathbf{v}_{i_l} + (1-\beta) \cdot \mathbf{v}_{j_l} \quad (20)$$

Finally, the grid energy (equations (11) and (14)) consists of energy terms of all *valid* edges. An edge e is said to be valid if it is either a long edge ($e \in E_{long}$), or if it is an edge of the adaptive mesh ($e \in E_a$) that is incident to equally sized TCPs. In other words, the mesh edges which lie on the long edges are called *invalid*. For the feature energy (equations (13) and (15)) the end vertices of the feature lines are defined as linear combinations of the end vertices of the incident valid edges.

Additionally, we want to treat all edges in the same way, i.e. similar rotational distortions should have similar impact on the energy no matter which length the edges have. Therefore we rescale the energy terms of the "longer" edges. If $N_{min} \times N_{min}$ is the minimal TCP size used, then the energy terms corresponding to edges having the length N are scaled by N_{min}^2/N^2. Notice that by the definition of valid edges above, this scaling factor is unambiguous.

In Fig. 8 a comparison of our regular and our adaptive image retargeting approach is shown. In the adaptive case more

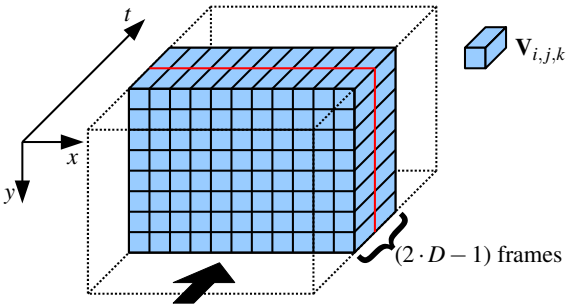


Figure 9: *Video retargeting process by moving a block of two volume slices with TCVs $\mathbf{V}_{i,j,k}$ of depth D along the time axis. The TCP grids on the three keyframes of this block define the unknowns of a linear system to be solved. The solution of the inner keyframe (red) is used for retargeting.*

deformation is pushed into the homogenous regions leading to a more uniform resizing of the person in the image.

6. Two-Colored Voxels

In this section we will extend the TCP concept to a video volume and apply it for the example of video retargeting. We will describe only the regular setting, but the extension to an adaptive one is straight-forward.

Let $F = \{f_0, ..., f_{t-1}\}$ be the input image sequence consisting of t frames. We split this video volume in volumetric elements $\mathbf{V}_{i,j,k}$, having the spatial size $N \times N$ and the temporal depth D. In this notation (i, j) are the spatial TCP coordinates of $\mathbf{V}_{i,j,k}$ and k is the coordinate of the first frame covered by $\mathbf{V}_{i,j,k}$. Analogously to the TCP concept we define each $\mathbf{V}_{i,j,k}$ to be a 3-tuple $(\mathbf{N}_{i,j,k}, \mathbf{C}^-_{i,j,k}, \mathbf{C}^+_{i,j,k})$, where $N_{i,j,k}$ is the *feature plane* which splits $\mathbf{V}_{i,j,k}$ in two parts and $\mathbf{C}^-_{i,j,k}$ and $\mathbf{C}^+_{i,j,k}$ define the average colors on the negative and on the positive side of the feature plane, respectively. We call $\mathbf{V}_{i,j,k}$ *two-colored voxels* (TCVs).

Like a feature line of a TCP, the feature plane of a TCV should provide a linear approximation to the strongest color discontinuity inside the volume covered by the TCV. Exhaustive computation of such a plane by using similar hierarchical structure as done for feature lines in Section 3.2 would be too expensive. Hence, an alternative method is to approximate the best feature plane by defining a finite number of directions and offsets to be tested. Since we already have a very efficient method for computing TCPs, we choose a different approach.

Each frame f_m is split into TCPs $\mathbf{T}_{i,j,m}$. The TCV $\mathbf{V}_{i,j,k}$ covers all TCPs in the set $\{\mathbf{T}_{i,j,m} | m = k, ..., k + D - 1\}$. We compute the feature plane $\mathbf{N}_{i,j,k}$ as the least-squares plane of the feature lines from the set $\{\mathbf{L}_{i,j,m} | m = k, ..., k + D - 1\}$.

6.1. TCV-based Video Retargeting

Applying TCP-based image retargeting to each input frame f_m results in strong flickering because time coherence is completely ignored. In order to get smooth transitions between the consecutive frames in the retargeted video, we exploit our TCV concept.

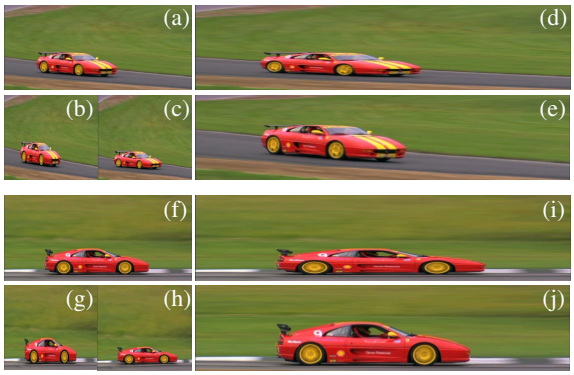


Figure 10: *Video Retargeting example visualized for two input frames (a) and (f). Linear scalings (b), (d), (g), (i) compared with our solutions (c), (e), (h), (j), respectively.*

We compute a regular TCV representation $\mathbf{V}_{i,j,k}$ for F by using a fixed depth D, which divides F in *volume slices*. Each two consecutive volume slices share a *keyframe* in the sequence. The computed TCVs implicitly define TCP grids and corresponding quad meshes on the keyframes. All *keyframe meshes* together define the set of unknowns in our video retargeting problem. Solving this problem on the whole volume is too expensive. Therefore we propose to use a window that is two volume slices wide, and for each position of this window only the solution of the inner keyframe mesh is used later for retargeting (see Fig. 9). The retargeting solutions for other frames are computed by linear interpolation between the keyframes. Notice that the keyframes f_0 and f_{t-1} are special cases since they never become inner keyframes of a volume slice, thus they are computed together with keyframes f_{D-1} and f_{t-D}, respectively.

In order to set up the linear system, we follow the method for image retargeting and define the deformation energy for the grid edges in the same manner as for the image case. Including the deformation energy of a feature plane $N_{i,j,k}$ is done by computing the feature lines for each TCP included in the corresponding TCV as lines on the plane $N_{i,j,k}$. Notice that in this setting feature planes are parametrized by the 8 corners of the TCV (in the image case 4 corners of the corresponding TCP were used (see Section 5.1)). In Fig. 10 we show two frames from a car image sequence retargeted with our method (please see also the accompanying video for a better impression of the quality of our results).

Discussion and Future Work The computation of feature lines from feature planes within volume slices guarantees smooth transitions between keyframes, but the smoothness property cannot be guaranteed across the keyframes. Nevertheless, the results we have achieved with our method are very satisfying and by using additional temporal constraints we could achieve smoothness property for the whole retar-

geted sequence. This is one of the main issues we would like to address in the future work.

7. Conclusion

In this paper we have shown how to efficiently compute a TCP representation of an image using our hierarchical algorithm on the GPU. This allowed us to use the TCP concept as a basic operator in the context of an interactive brush tool. We have implemented a number of operations like painterly image abstraction or edge-preserving filtering and have demonstrated the effectiveness of the TCP operator in the context of edge-aware image processing. Further, we have exploited the TCP grid as a main data structure for mesh warping based image retargeting. In this context we have also introduced image retargeting on adaptive grids which provides more control over the retargeting process. Finally, we have extended the TCP concept to a video volume creating the two-colored voxel representation and demonstrated its application to video retargeting.

References

[AGDL09] ADAMS A., GELFAND N., DOLSON J., LEVOY M.: Gaussian kd-trees for fast high-dimensional filtering. *ACM Trans. Graph. 28*, 3 (2009), 1–12.

[AS07] AVIDAN S., SHAMIR A.: Seam carving for content-aware image resizing. In *Proc. of ACM SIGGRAPH* (2007).

[BSCB00] BERTALMIO M., SAPIRO G., CASELLES V., BALLESTER C.: Image inpainting. In *Proc. of ACM SIGGRAPH* (2000).

[BWG03] BALA K., WALTER B., GREENBERG D. P.: Combining edges and points for interactive high-quality rendering. *ACM Trans. Graph. 22*, 3 (2003), 631–640.

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. In *Proc. of ACM SIGGRAPH* (2009).

[CPD07] CHEN J., PARIS S., DURAND F.: Real-time edge-aware image processing with the bilateral grid. In *Proc. of ACM SIGGRAPH* (2007).

[Don99] DONOHO D. L.: Wedgelets: nearly minimax estimation of edges. *Annals of Statistics 27*, 3 (1999), 859–897.

[DS02] DECARLO D., SANTELLA A.: Stylization and abstraction of photographs. In *Proc. SIGGRAPH* (2002), pp. 769–776.

[FFLS08] FARBMAN Z., FATTAL R., LISCHINSKI D., SZELISKI R.: Edge-preserving decompositions for multi-scale tone and detail manipulation. In *Proc. of ACM SIGGRAPH* (2008).

[GSCO06] GAL R., SORKINE O., COHEN-OR D.: Feature-aware texturing. In *Proc. of EGSR* (2006).

[Har07] HARRIS M.: Optimizing CUDA. Online Tutorial: Super Computing 2007 on http://www.gpgpu.org/static/sc2007/, 2007.

[IKN98] ITTI L., KOCH C., NIEBUR E.: A model of saliency-based visual attention for rapid scene analysis. *PAMI '98* (1998).

[JLSW02] JU T., LOSASSO F., SCHAEFER S., WARREN J.: Dual contouring of hermite data. In *Proc. of ACM SIGGRAPH* (2002).

[KFG09] KARNI Z., FREEDMAN D., GOTSMAN C.: Energy-based image deformation. *Computer Graphics Forum* (2009).

[KL08] KANG H., LEE S.: Shape-simplifying image abstraction. *Computer Graphics Forum 27*, 7 (2008), 1773–1780.

[KLC09] KANG H., LEE S., CHUI C. K.: Flow-based image abstraction. *IEEE Transactions on Visualization and Computer Graphics 15*, 1 (2009), 62–76.

[KLHGar] KRÄHENBÜHL P., LANG M., HORNUNG A., GROSS M.: A system for retargeting of streaming video. In *Proc. of ACM SIGGRAPH ASIA* (to appear).

[MMS*09] MERKLE P., MORVAN Y., SMOLIC A., FARIN D., MÜLLER K., DE WITH P. H. N., WIEGAND T.: The effects of multiview depth video compression on multiview rendering. *Image Commun. 24*, 1-2 (2009), 73–88.

[NBGS08] NICKOLLS J., BUCK I., GARLAND M., SKADRON K.: Scalable parallel programming with cuda. In *ACM SIGGRAPH classes* (2008).

[OBBT07] ORZAN A., BOUSSEAU A., BARLA P., THOLLOT J.: Structure-preserving manipulation of photographs. In *Proc. of ACM NPAR* (2007).

[PD06] PARIS S., DURAND F.: A fast approximation of the bilateral filter using a signal processing approach. In *Proc. of ECCV* (2006).

[PvCK09] PAVIC D., V. CEUMERN U., KOBBELT L.: GIzMOs: Genuine image mosaics with adaptive tiling. *Computer Graphics Forum 28*, 8 (2009), 2244–2254.

[PZ08] PARILOV E., ZORIN D.: Real-time rendering of textures with feature curves. *ACM Trans. Graph. 27*, 1 (2008), 1–15.

[RBW04] RAMANARAYANAN G., BALA K., WALTER B.: Feature-based textures. In *Rendering Techniques* (2004).

[RSA08] RUBINSTEIN M., SHAMIR A., AVIDAN S.: Improved seam carving for video retargeting. In *Proc. of ACM SIGGRAPH* (2008).

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Proc. of Eurographics Symposium on Geometry Processing* (2007).

[SDDV05] SHUKLA R., DRAGOTTI P. L., DO M. N., VETTERLI M.: Rate-Distortion Optimized Tree-Structured Compression Algorithms for Piecewise Polynomial Images. *IEEE Transactions on Image Processing 14*, 3 (2005), 343–359.

[Sen04] SEN P.: Silhouette maps for improved texture magnification. In *Proc. HWWS '04* (2004), pp. 65–73.

[TC04] TUMBLIN J., CHOUDHURY P.: Bixels: Picture samples with sharp embedded boundaries. In *Rendering Techniques* (2004).

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *Proc. of ICCV* (1998).

[WFS*09] WANG Y.-S., FU H., SORKINE O., LEE T.-Y., SEIDEL H.-P.: Motion-aware temporal coherence for video resizing. *Proc. of ACM SIGGRAPH ASIA 28*, 5 (2009).

[WGCO07] WOLF L., GUTTMANN M., COHEN-OR D.: Non-homogeneous content-driven video-retargeting. In *Proc. of ICCV* (2007).

[WN03] WILLETT R., NOWAK R.: Platelets: a multiscale approach for recovering edges and surfaces in photon-limited medical imaging. *Medical Imaging, IEEE Transactions on* (2003).

[WTSL08] WANG Y.-S., TAI C.-L., SORKINE O., LEE T.-Y.: Optimized scale-and-stretch for image resizing. In *Proc. of SIGGRAPH Asia* (2008).

[ZHM08] ZHANG Y.-F., HU S.-M., MARTIN R. R.: Shrinkability maps for content-aware video resizing. In *Proc. of Pacific Graphics* (2008).