

Combining color and depth for enhanced image segmentation and retargeting

Meir Johnathan Dahan · Nir Chen · Ariel Shamir · Daniel Cohen-Or

Published online: 29 December 2011
© Springer-Verlag 2011

Abstract As depth cameras become more popular, pixel depth information becomes easier to obtain. This information can clearly enhance many image processing applications. However, combining depth and color information is not straightforward as these two signals can have different noise characteristics, differences in resolution, and their boundaries do not generally agree. We present a technique that combines depth and color image information from real devices in synergy. In particular, we focus on combining them to improve image segmentation. We use color information to fill and clean depth and use depth to enhance color image segmentation. We demonstrate the utility of the combined segmentation for extracting layers and present a novel image retargeting algorithm for layered images.

Keywords Image segmentation · Retargeting · Depth maps

1 Introduction

In recent years we have witnessed a rapid advance in the development of cameras that capture depth. While this advance has been driven by video gaming, simultaneously acquiring depth with photometric imaging has an immense potential for a wide range of applications. In particular, having

commodity level cameras with depth sensing will considerably enhance the performance of image processing and digital photography applications. However, using data from such cameras may not be straightforward. First, the depth data spatial resolution does not always match the photometric resolution. Second, in many cases the depth image is incomplete, imperfect, and contains noise. Under these assumptions, we investigate the potential of combining the two modalities to enhance image segmentation, which is the basis of many image processing applications.

In image segmentation, boundaries between objects in the scene must be determined. In many cases, the photometric (color) image edges correspond to scene edges, i.e., to discontinuity in depth. However, there are many image edges created by illumination (e.g., shadows) or textures that do not signify object boundaries. Color segmentation usually finds coherent color regions in an image. Real objects in a scene such as people or animals can contain a variety of colors, but usually not a wide range of depth values. Hence, using depth information can assist the mapping of color-distinct regions to the same object and distinguishing between “false” image edges and real object boundaries.

Still, simply using the raw depth information is not sufficient. Our key idea is to utilize the information contained in one channel to enhance the other. Therefore, the RGB channel, being more accurate and having higher resolution, can, in turn, assist in enhancing the depth channel by reconstructing missing depth regions and enhancing the correctness of edges in the depth map.

Combining the two modalities together naively does not necessarily amount to unambiguous data from which the segmentation can be extracted. In this paper, we introduce a technique that combines the two modalities while increasing the agreement between them. We present an end-to-end depth-color segmentation that provides better results than

M.J. Dahan (✉) · D. Cohen-Or
Tel Aviv University, Tel Aviv, Israel
e-mail: meirjon@post.tau.ac.il

N. Chen · A. Shamir
Interdisciplinary Center Herzilya, Herzilya, Israel

N. Chen
e-mail: chen.nir@idc.ac.il

A. Shamir
e-mail: arik@idc.ac.il



Fig. 1 Segmentation and retargeting using depth: (from left to right) color image, depth image and our automatic segmentation using color and depth overlaid, color image retargeted by 20%, 30%, 40%, and 50%

a direct concatenation of the two (e.g., using mean shift or clustering). Our technique is initialized by using joint-bilateral upsampling of both depth and color and then applying graph cut segmentation based both modalities. We demonstrate the technique on depth data which is imperfect, i.e., some regions are missing, there are outliers, and noise is present. Furthermore, the depth spatial resolution is lower than the photometric one, and thus also the registration between the two is imperfect.

In addition, to demonstrate the use of the combined segmented image, we present a novel image retargeting algorithm. We show how the augmented segmentation allows a qualitatively better solution to the retargeting problem by exploiting the depth information to partially overlap the layers when size is reduced (see Fig. 1). We demonstrate our technique on both synthetic and real depth images.

2 Related work

Depth There are numerous ways for estimating depth in images, for instance, by using stereo, or multiple views in structure from motion [21]. Zhang et al. [29] introduced a method of recovering depth from a video sequence and show many applications; some of the images in this paper were acquired in this manner. Other works reconstruct depth using video sequences based on the motion between frames [8, 20]. Another possibility to acquire depth is simply to use depth cameras. Such cameras are becoming a commodity these days, and their use is widespread. Most examples in this paper were acquired using a depth camera. However, such depth sensors still have lower resolution than the RGB sensor, and their quality in general is inferior to photometric sensors.

Upsampling Some previous work attempted to deal with the lower spatial resolution of one modality versus the other by upsampling one to match the resolution of the other. Kopf et al. [10] introduced the joint bilateral filter for a pair of corresponding images. They upsample a low-resolution image while respecting the edges of the corresponding high-resolution one. This technique was then improved by using

iterations [16, 27]. Schuon et al. [22] combined several low-resolution depth images to produce a super-resolution image.

Except for low spatial resolution, depth imaging is also subject to noise, and, due to surface reflection properties, some regions of the surface are completely missing. This problem is often referred to as “depth shadow.” To address this issue, Becker et al. [3] use the color channels taken from a camera with a known location. They rely on the assumption that similar geometries possess similar colors and incorporate a patch-based inpainting technique. In our work we also complete missing color and depth values. We use techniques such as [12] for layer completion and reconstruct the depth with a multistep joint bilateral depth upsampling filter [16].

Segmentation Segmentation is one of the fundamental problems in image processing [9]. As we discussed in the introduction, the performance of segmentation can be greatly improved with the assistance of depth images. Recently, Crabb et al. [6] employed depth imaging to extract foreground objects in real time. Their segmentation requires a user-defined threshold depth to separate the foreground from the background. The color image is used only to support a small fraction (1–2%) of the pixels which are not solved by the depth threshold. Another interesting use of depth is to assist background subtraction in video. In [13] several channels of the depth camera are used and combined with the color channel.

Image retargeting To demonstrate the usefulness of our method, we show how it can be utilized for image retargeting. Techniques such as seam-carving [1, 19], shiftmaps [15], and non-homogeneous warping methods [11, 24, 26] all rely on the ability to automatically detect important regions in an image and preserve them under the resizing operator. If the image is condensed and does not contain smooth or textured areas, these techniques can cause serious artifacts. Using depth layers can assist in such cases by overlaying parts of the image while resizing. In [7] repetition is used to segment, layer, and rearrange objects in an image closer to our approach, but no depth input is used. For video, Zhang et al. [28] present a video editing system

for re-filming visually appealing effects based on the extraction of depth layers from the video. Recently Mansfield et al. [14] exploited the seam carving technique to cause pre-segmented image objects to occlude each other. However, due to the nature of seam carving, distortions are still introduced to the carved background image. We show how the addition of depth information can assist specifically in such cases, by allowing different parts of the image to overlap in the resized image based on their depth. We compare our results to various other techniques using the recent RetargetMe benchmark [17].

3 Color and depth imaging

Depth cameras provide an image where each pixel contains a measurement of the distance from the camera to the object visible in that pixel. There are several types of depth devices such as time-of-flight cameras and projected coded images. Most depth cameras use IR light for depth acquisition since IR is not visible to the human eye and does not interfere with the color sensor. The device we use is based on projected IR Light Coding image and provide a depth resolution of 640×480 . It is important to note that contemporary depth sensors are still rather noisy and there are significant regions in the image where depth is not accurate or even missing. Moreover, since the depth and color images are created and captured with different sensors, the two images are not perfectly aligned and may have different resolutions. Specifically, photometric images usually have much higher resolution than depth image (see Fig. 1). In our work, we exploit this to enhance the depth resolution.

As a first step, the capturing devices must be calibrated respectively. We seek the set of camera parameters (location, rotation, view plane) that best explain the perspective deformation observed from the devices. First, the user marks several corresponding pairs of points (see Fig. 2). Denote these points as $B_{\text{depth}} = \{p_1, p_2, \dots, p_m\}$, $B_{\text{RGB}} = \{q_1, q_2, \dots, q_m\}$ in the color and depth image, respectively. Next, we project every user marked pixel in B_{depth} to its 3D coordinates using the depth measurement. Denote as A the set of the 3D locations of the points. Then, we retrieve the perspective projection parameters θ by minimizing the L_2 distance between the RGB points and the perspective transformation T_θ of the corresponding projected depth points (described below). Finally, we apply the transformation T_θ to all the 3D points to generate a new depth image suiting the view-point of the color image. A more detailed description is provided in the next paragraph.

We use perspective projection transformation as specified in [5]. Let a be a 3D point ($a \in A$) and $\theta = \{c_{x,y,z}, \theta_{x,y,z}, e_{x,y,z}\}$ the camera parameters. Transformation $b = T_\theta(a)$ is



Fig. 2 Camera calibration: The perspective projection parameters are obtained by minimizing the distance between RGB points and the perspective transformation T_θ of the projected depth points. In the figure, the depth gradients image is superimposed on the color image before (top) and after (bottom) registration. The yellow marked points (q_i) are matched to the green (p_i) by the user. After registration procedure the new position of the yellow points is marked in red

defined as:

$$\begin{aligned}
 d &= Rot_{\theta_x} Rot_{\theta_y} Rot_{\theta_z} [a - c], \\
 b_x &= (d_x - e_x)(e_z/d_z), \\
 b_y &= (d_y - e_y)(e_z/d_z),
 \end{aligned}
 \tag{1}$$

where, $a_{x,y,z}$ is the point in 3D space that is to be projected, $c_{x,y,z}$ is the location of the camera, $\theta_{x,y,z}$ is the rotation of the camera, $e_{x,y,z}$ is the viewer's position relative to the display surface, and $b_{x,y}$ is the 2D projection of a . We search for the depth camera parameters that minimizes $\min_{c,\theta,e} \{\sum (T_{c,\theta,e}(a_i) - q_i)^2\}$, where q_i is the i th element in B_{RGB} , and a_i is the corresponding projected 3D location of p_i (in the depth image). Once the camera parameters are obtained, we project all depth pixels to 3D space and apply the perspective transformation on all the points.

This calibration process is done once, offline. Since the two cameras are close to one another (about 3 cm apart), the search domain is relatively small around the depth camera. It requires less than a dozen pairs of points to find these parameters. However a better estimation is reached as more pairs are given. Even after calibration is done, there are still discrepancies between the two images. In the following we describe how we further align the images, enhance the depth image based on the color image (Sect. 4), and then enhance the segmentation of the color image based on the enhanced depth image (Sect. 5).

4 Depth enhancement using color

There are four issues that we address in enhancing the depth image using color. First, due to various environmental reasons, specular reflections, or simply the device range, there are regions of missing data in the depth image. Second, the accuracy of the pixels values in the depth image is low, and the noise level is high. This is true mostly along depth edges and object boundaries, which is exactly where such information is most valuable. Third, despite the calibration, the depth and color images are still not aligned well enough. They are acquired by two close, but not similar, sensors and may also have differences in their internal camera properties (e.g., focal length). This misalignment leads to small projection differences, even, again, these small errors are more noticeable especially along edges. Lastly, usually the depth

image has lower resolution than the color image, and we would like to up-sample it in a consistent manner.

To account for the above four issues, we apply the following technique (see Fig. 3). First, we down-sample the high-resolution image to the same resolution as the depth image and then down-sample both images to a lower resolution. Next, we use joint bilateral up-sampling [16] in two consecutive steps. The first step is aimed at filling in the missing depth regions based on the color information, and the second step aligns the two images while up-sampling the depth to the higher resolution. Together this fills in the missing regions, aligns the depth edges with color image edges, removes inaccuracies caused by disparity estimation, and increases the accuracy and consistency at depth edges and in areas with high frequencies.

Let $I(p)$ be the RGB color value at pixel location p , and $D(p)$ the corresponding depth at p . To fill in a missing depth region, we down-scale the depth image and color image by 2^{-4} fraction. In the regions of missing depth data, we calculate the average depth value of the region boundary and assign it to all pixels inside the region. Next, we use joint bilateral up-sampling to bring the resolution of the depth image back to its original resolution while using the color image data to assist in filling the correct depth values. The up-sample filter uses samples from both the high-resolution and low-resolution images in the range term of the bilateral filter. The up-sampling is done in multiple stages, refining the resolution by a factor of 2×2 at each stage. Each up-sampling step is carried according the to following equation:

$$D_p^h = \sum_{q \in \Omega} w_{p,q} D_q^l / \sum_{q \in \Omega} w_{p,q}, \tag{2}$$

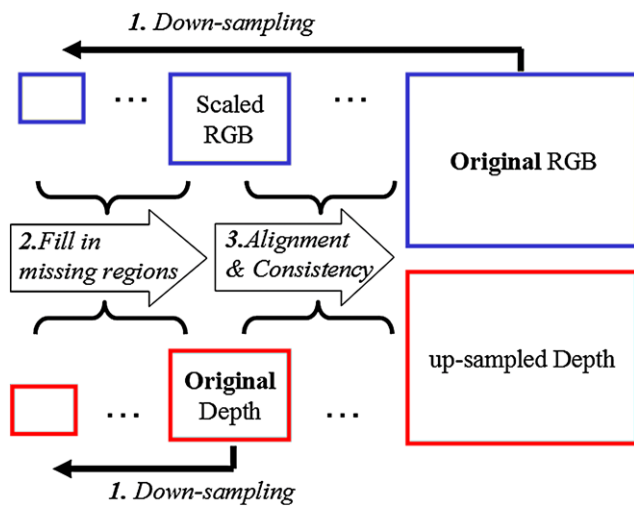


Fig. 3 Down-sampling the color and depth images and up-sampling the depth image using multistep joint bilateral up-sampling

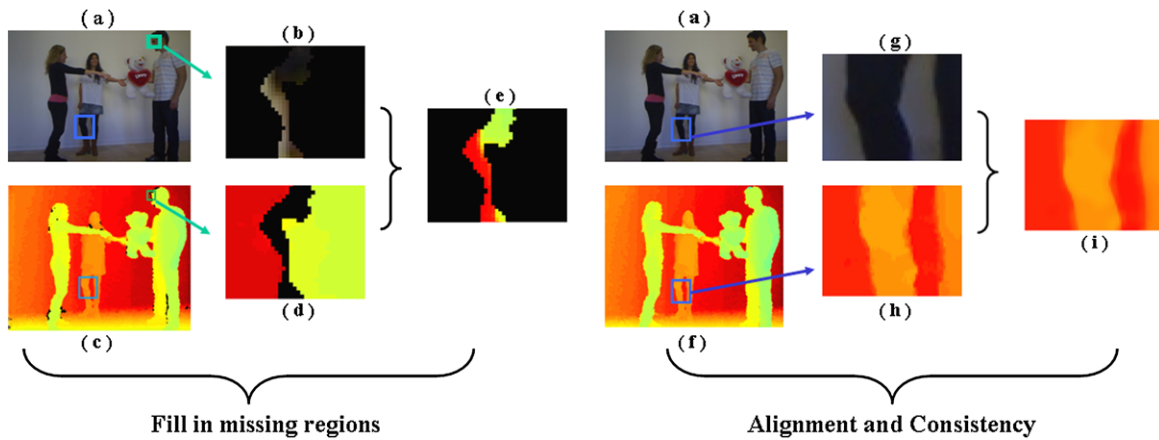


Fig. 4 Depth enhancement using multijoint bilateral upsampling. (a) Color image. (b) A close-up view. Note the existence of an edge. (c) Initial depth image with missing depth and rough edges. (d) A close-up view. (e) The missing depth is correctly filled by re-

specting the color edge. (f) Depth image after filling in of the missing depth. (g) A close-up view. (h) A close-up view. (i) The close up of (h) after the final depth alignment

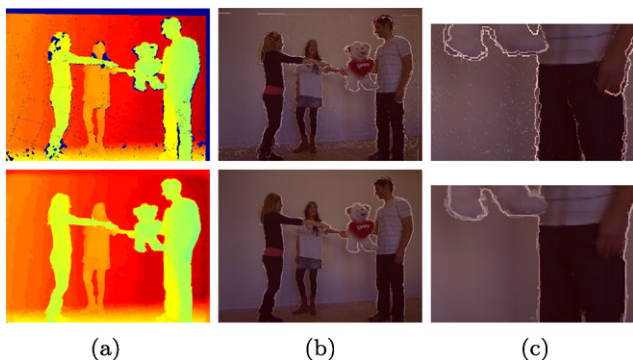


Fig. 5 (a) Calibrated depth image (*top*) and the enhanced depth image (*bottom*). (b) The depth gradients before (*top*) and after (*bottom*) superimposed on the color image. (c) A close up to the superimposed image

where $w_{p,q} = r(I_p^h - I_q^l)$, r is a Gaussian range filter kernel, h stands for high-resolution, l is the low resolution, and Ω is a neighborhood of position p .

Since the up-sampling works jointly on both color and depth data, it changes the values inside the smoothed filled regions to be consistent with their depth boundary values, while preserving existing *color* edges within the region (Fig. 4(b, e)). However, the up-sampling also changes (smooths) the values in other regions of the depth image, possibly to superfluous values. Hence, we only use the up-sampled pixel values inside the missing regions and maintain the original depth in all other regions.

The process described above results in a depth image with the original resolution and all missing depth interpolated. However, it is still noisy and inconsistent with the original color image (Fig. 4(h)). Therefore, we run the multi-step joint bilateral up-sampling algorithm once more on the new depth image, but this time only for one or two iterations until we reach the original color size. This aligns the depth with the color image information all over the scene (Fig. 4(i)). The final result (Fig. 5) is a registered depth map which gives a good cue for the depth of pixels in the scene with no missing data and where color and depth edges are aligned.

5 Segmentation using depth and color

Now that we have two aligned images, a color image I and a depth image D , we present an automatic segmentation method using both the color and depth information. Note that joint segmentation is similar to extracting consistent layers from the image where each layer (e.g., segment) is accordant in terms of both color and depth. It is common for real scene object, such as a person, to contain several different colors, but a rather small range of depths. Using depth information allows one to detect real object boundaries instead of just coherent color regions. We therefore initialize

our segmentation with layers based only on depth. Next, instead of searching for a single color value for each layer, we use a probabilistic model (Gaussian Mixture Model) for all colors in the region. Using this model, we then create the final segmentation by minimizing a common energy function,

$$E(X) = \sum_p E_d(x_p) + \lambda \sum_{p,q} E_s(x_p, x_q), \tag{3}$$

where p and q are two adjacent pixels, and $X : I \rightarrow L$ is the assignment of a label to each pixel. This function consists of two terms: The data term E_d which assigns each pixel to a specific label (segment) according to its color and depth and the smoothness term E_s that encourages similar neighboring pixels to belong to the same segment, i.e., have the same label.

In practice, we induce the energy over a graph constructed from the image pixels and solve it using graph-cut [2]. The segmentation gives an assignment of L labels where each label $l \in L$ defines a unique segment. Since we use both depth and color information in the energy, these segments represent real objects in the scene. “False” color image edges have less effect on the segmentation since the depth information reduces their relative weight. On the other hand, since color information is used as well, layers with similar depths can be separated into objects based on color.

Initialization We initialize the process using mean-shift segmentation [4] on the depth image alone. This step defines $|L|$ layers in the depth. These layers are built from objects, parts of objects, and some objects joined together. In our experiments, we used between five to seven depth regions. Each region is considered to be the seed for a label l in the segmentation. Hence, $|L|$ is the number of labels used in the graph-cut procedure.

For each of the $|L|$ segments, a Gaussian Mixture Model, similar to [18], is fit. We use two Gaussians for each modality: two for the color image $pc^l(\cdot)$ and two for the depth image $pd^l(\cdot)$. These models are used to define the distance between elements in the definition of the energy function of the graph-cut segmentation (see below).

Graph cut We use the regular construction of the graph G : every pixel in the image is a node v in the graph, and we use 4-connectivity for the edges e which are called *n-links*. Every node v is also connected with edges, which are called *t-links*, to the terminal nodes t_l . Below we give the description of the weights assignment for the *n-link* and the *t-link*. After the calculation of a minimum cut on the graph, we are left with a labeling assignment for pixels, which leads directly to a minimization of the energy function defined above. The two types of energy terms fit the two types of edges in the graph: the smoothness term as the *n-link* and the data-term as the *t-link*.

Fig. 6 (a) Color image. (b) Depth image. (c) Automatic segmentation using depth only. Note that the *woman on the right* cannot be separated from the grass. (d) Automatic segmentation using color only. No layers created. (e) Automatic segmentation using color and depth. As can be seen, the layers in the image are segmented. (f) Mean shift using XY and depth. (g) Mean shift using XY and RGB. (h) Mean shift using XY, RGB, and depth. (i) *k*-means using XY and depth. (j) *k*-means using XY and RGB. (k) *k*-means using XY, RGB, and depth

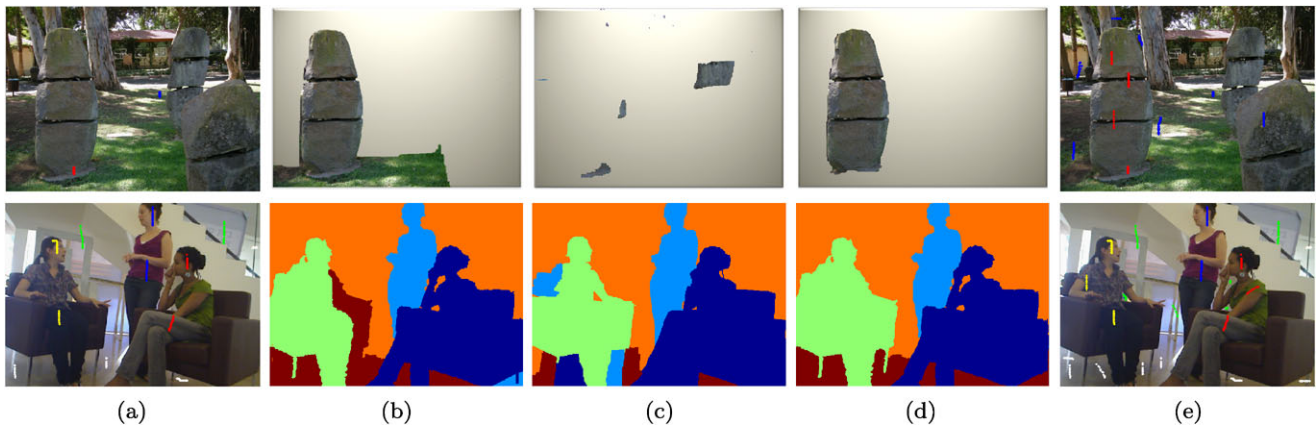
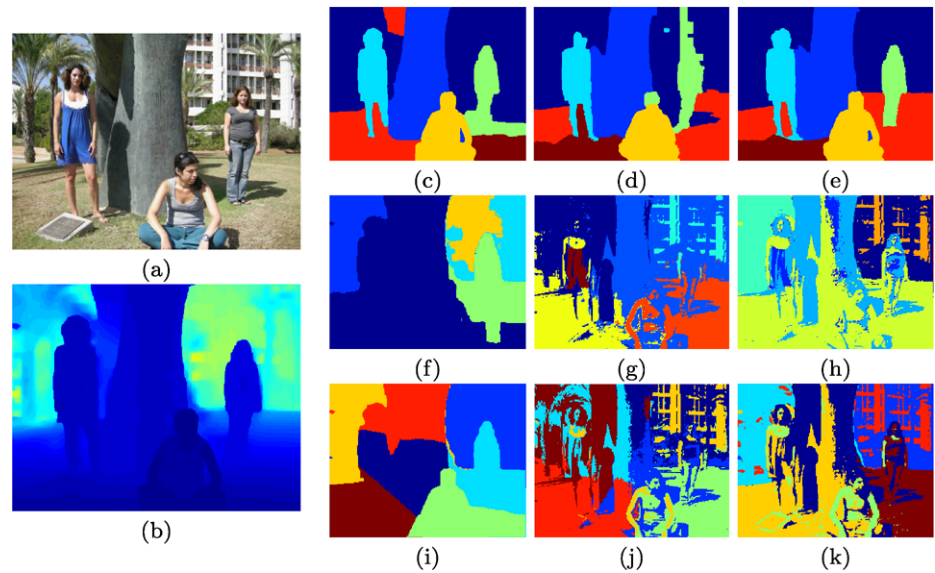


Fig. 7 The effectiveness of scribbles when color and depth are considered. Color or depth information alone cannot segment the scene using such a small number of scribbles. (a) *Top*: 2 scribbles; *bottom*: 11 scribbles. (b) Segmentation using depth only. (c) Segmentation using color only. (d) Segmentation using depth and color. Using only color

segmentation, 11 (e, *top* includes 2 of (a) + 9 more) or 21 (e, *bottom* includes 11 of (a) + 10 more) scribbles are required to match the segmentation with 2 scribbles (a, *top*) or 11 scribbles (a, *bottom*) using both color and depth

Data term The *t*-links, from each t_l to each node v , are assigned with weights according to the data term calculated by both color and depth channels. Formally, we calculate:

$$E_d(x_p) = \max(x_p \cdot L_p^{cl}, \alpha \cdot x_p \cdot L_p^{dl}), \quad (4)$$

where α is the weight between the color and depth energy. $pc^l(\cdot)$ and $pd^l(\cdot)$ are the GMM models for each l , for color and depth, respectively, $L_p^{cl} = -\ln(pc^l(I_p))$, $L_p^{dl} = -\ln(pd^l(D_p))$, and I_p and D_p are the color and depth image values at pixel p , respectively.

We merge the color and depth channels by using the maximum of their distance to the respective model. This way fitting each pixel gives lower energy for a good match. The actual weights on the *t*-links are calculated using the negative data term.

Smoothness term The smoothness term is also constructed using both the color and depth information. The energy weights on the *n*-links are calculated from the similarity between the two connected nodes. This similarity is calculated on the color and on the depth separately, and then the final value assigned to the graph is again the higher value between them, giving stronger weight to the larger similarity. Formally, when p and q have the same label, then $E_s(x_p, x_q) = 0$, and for pixels p, q that have different labels, we define:

$$E_s(x_p, x_q) = (\max(\mu \|I_p - I_q\|, \beta \cdot \varphi \|D_p - D_q\|))^{-1}, \quad (5)$$

where β is the weight between the color and depth energy, and $\mu = (\|I_p - I_q\|)^{-1}$ and $\varphi = (\|D_p - D_q\|)^{-1}$ are the

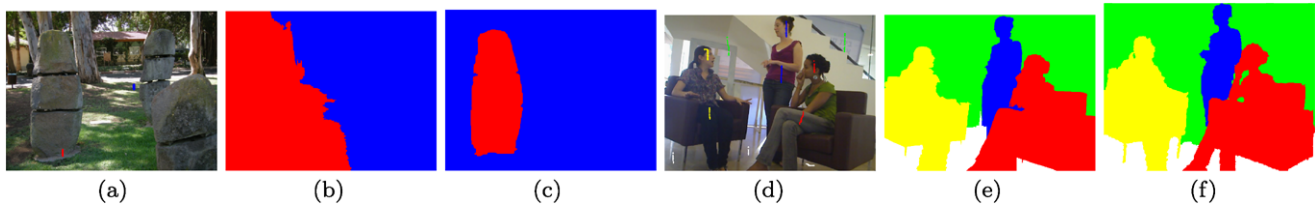


Fig. 8 Manual segmentation comparison: (a, d) The original with scribbles. (b, e) GrowCut segmentation result [23]. (c, f) Our combined depth and color segmentation



Fig. 9 Automatic segmentation using color and depth can also outperform manual segmentation using color alone. Here we compare our automatic segmentation results (a and d) with GrowCut segmentation result [23] (b and e) using the scribbles shown in (c and f)

expectation operators over the whole color and depth map image, respectively.

After running a minimum cut on the graph, each pixel is assigned a label defining its segment. To create more coherent layers in the segmentation, we use superpixels defined on the color image using mean-shift segmentation and apply a voting scheme. The label of each superpixel is determined as the majority of its pixel assignments. Figure 6 demonstrates the strength of using both color and depth information for segmentation rather than using only one of them. Running on a Pentium 4, 3 GHz with 2 GB memory, on average it takes around one minute for the joint bilateral upsampling and 5–10 minutes for the graph-cut depending on the number of layers.

The segmentation can be also applied interactively. In this case, instead of the voting scheme, the user marks scribbles to initialize the segmentation. As can be seen in Fig. 7, the use of both color and depth can help not only for a better and precise capturing of the object, but also for a more convenient scribble processing. When using the scribbles in Fig. 7(a) on the segmentation with both color and depth, the result, Fig. 7(d), describes the objects in the scene naturally. While the segmentation using only depth (Fig. 7(b)) or only color (Fig. 7(c)) is not sufficient to capture the objects in a natural way. We have also compared our method to the state-of-the-art algorithm such as GrowCut [23]. The combination of depth and color can enhance manual segmentation as demonstrated in Fig. 8. We also illustrate that combining depth and color with no manual intervention can outperform manual segmentation in Fig. 9.

6 Retargeting using depth

Image retargeting deals mostly with changing the aspect ratio of images. This process involves inserting or removing parts from the image either directly (e.g., seam carving or shift-map) or indirectly (i.e., warping the image to a different domain). The main difficulty in this process is how to choose the right image parts so as to create a realistic result with little artifacts. All image retargeting techniques have difficulty especially when images contain many details, many foreground objects, and, most specifically, people. In this section we show how using the additional information provided by image layers can assist to create better retargeting results even in such situations.

The key point in a layered image is that it provides an ordering on the image parts. This ordering can be utilized to assist image retargeting simply by concealing farthest parts of the image behind closer ones in a realistic manner and without distorting these parts. For instance, the basic operation for reducing the size of an image would be to reduce each layer separately (i.e., shift it toward the center of the image) and then reconstruct the image from the layers. Indeed, by reducing each layer, some farther pixels will be concealed by closer ones in the resized image (Fig. 10). However, because of shifting, there will be pixels in the reconstructed image with no value at all. These pixels need to be filled with correct colors from the correct layer.

Assume that after segmentation we have k layers $\{L_1, L_2, \dots, L_k\}$. Each layer is assigned an ordinal depth value $\tau(L_i)$. This value is usually found by sorting the mean depth value of each layer L_i . That is, for a layer i closer than layer j , it holds that $\tau(L_i) < \tau(L_j)$. For simplicity, we will consider $\tau(L_i) = i$. One exception to this rule is the ground

layer. A layer is considered a ground layer if it touches the bottom of the image and the differences between its minimum depth d_{\min} and maximum depth d_{\max} is larger than a given threshold. The ground layer is always considered at the back of any layer whose depth is smaller than d_{\max} .

Assume that we need to reduce the width of the image from W to W' (height changes are done similarly). This means that we need to reduce the width by a factor of $r = \frac{W'}{W}$. Let $c_i = (c_{ix}, c_{iy})$ denote the center of mass of layer L_i . Our first step is to shift each layer by $\frac{r}{2}(c_{ix} - \frac{W}{2}) + \frac{W'}{2}$. Hence, all layers move uniformly toward the center of the image. After the shift, pixels that are outside the width W' are cropped. Inside the new image, several layers may overlap, meaning that some pixels may contain more than one layer. The color value of these pixels is defined as the color of the closest layer. On the other hand, because of the shifts, some pixels will not be covered by any layer (Fig. 11(b)).

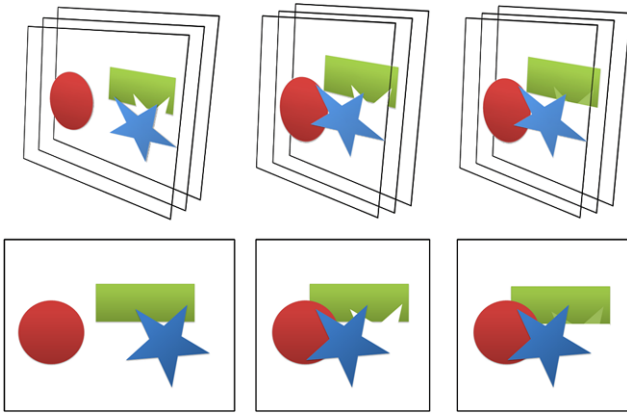


Fig. 10 Retargeting using layers: each layer is rearranged in the new size. This causes holes and missing areas which are inpainted by associating them to the most likely layer

We call these pixels null pixels as they do not have any color associated to them.

Our first attempt at reducing the number of null pixels is to perturb the layers position by up to $\pm\Delta$ where $\Delta = 5$ pixels. We use a greedy algorithm (see Algorithm 1) that checks all shifts of each layer individually from back to front, each time checking if the number of null pixels are reduced by the shift. If they are reduced, we keep the new layer position and continue to the next layer, looping back to the last layer after we reach the front one. We continue in this manner until there is no more change (Fig. 11(c)).

To fill in the missing regions that still remain, one could use simple inpainting. First, this does not always give satisfactory results as the pixels must be filled from the correct layer (see Fig. 12). Second, there are cases where layers have regions that should be filled even if they are not null

Algorithm 1 Layer jittering

```

1: function  $J = \text{JITTERLAYERS}(I = (L_1, L_2, \dots, L_n))$ 
2:    $\text{count} \leftarrow \|\text{null}(I)\|$   $\triangleright$  the number of null pixels
3:   repeat
4:      $\text{prev\_count} \leftarrow \text{count}$ 
5:     for  $i = n \rightarrow 1$  do
6:       for  $j = -\Delta : \Delta$  do
7:          $I' = (L_1 \dots L_i + j \dots L_n)$ 
8:          $N_j \leftarrow \|\text{null}(I')\|$ 
9:       end for
10:       $J_i = \text{argmin}_j \{N_j\}$ 
11:       $L_i \leftarrow L_i + J_i$ 
12:    end for
13:     $\text{count} \leftarrow \|\text{null}(I)\|$ 
14:  until  $\text{prev\_count} = \text{count}$ 
15:  return  $J = (J_1, J_2, \dots, J_n)$ 
16: end function

```

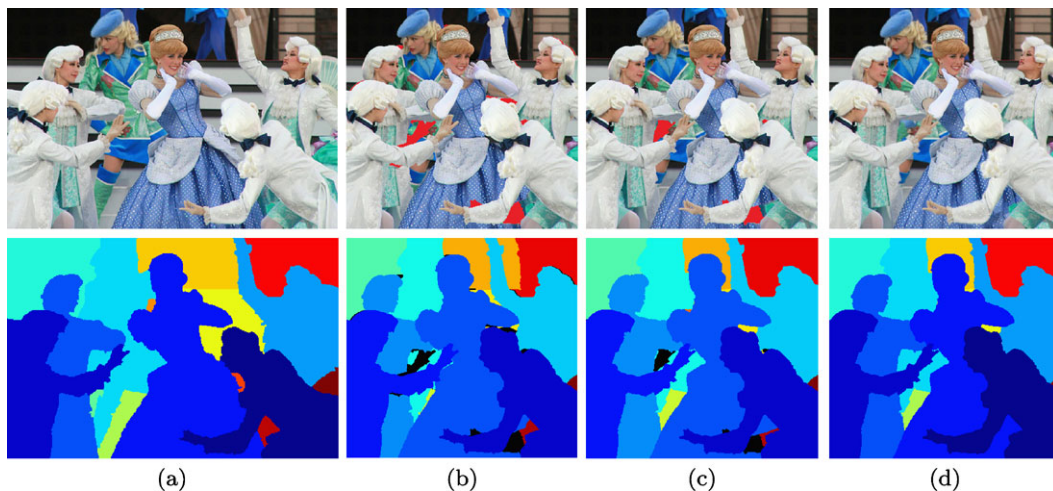


Fig. 11 (a) Original and layer image; (b) after shifting the layers; (c) after jittering; and (d) final result after inpainting

(see Fig. 14). For instance, some pixels may have a color obtained by a back layer while they are part of a front object that was concealed in the original image (see Fig. 13). Yet again, better results are obtained by utilizing the depth information. We use the layers from our segmentation algorithm to associate the most probable depth layer for each pixel and use the color information from that layer to inpaint that pixel. Note that this procedure is done not only for null pixels, but also for pixels that may contain a wrong color from a background layer.

In the original image, each layer L could be adjacent to several different layers. We define $B = B(L)$ as the boundary pixels of a layer L that connect to layers in front of L . B can be segmented into piecewise connect parts $B_i = (b_1, b_2, \dots, b_l)$ where each b_i is the pixel on the boundary that is adjacent to a single layer (but cannot be expanded). Next, we examine the shape of the underlined borders. Assume that pixels b_i are adjacent to layer L' that is in front of layer L . If layer L' penetrates into layer L (see Fig. 13), then after layer L' is moved due to retargeting, the pixels in the penetrated part must be filled by layer L and not by

any background layer that could be revealed behind it (see Fig. 14). To account for these situations, we calculate the convex hull for every boundary part b_i (see Algorithm 2). Later, in the retargeted image, if the pixels in the convex part are not covered by any layer in front of L , we fill them by inpainting using layer L itself. This is performed even if these pixels are not null pixels.

Lastly, if a region still remains null (unassigned) after filling all convex parts of all layers (meaning that it is not inside a convex part of any layer), then we assign it to the layer with largest boundary adjacent to the region. In practice, we check the designated location, only if it is concealed by frontal layer (Fig. 13(left)) at that relative location. Otherwise, it is discarded, and we move on to the next layer in boundary length. In cases where there are no suitable adjacent layer (see Fig. 12 between the legs of the girl on the right), we assign it to the closest layer behind all the layers

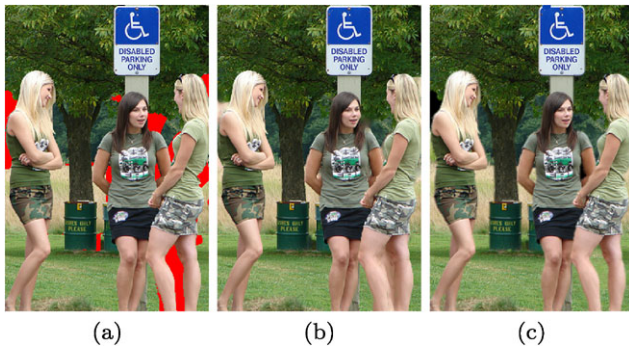


Fig. 12 (a) Retargeted image before image completion with missing parts (red). (b) Filling the missing parts using [25] completion method. (c) Filling these parts using our layer-coherent image completion. Note specifically the area between the legs of the girl on the right

Algorithm 2 Boundary analysis and convexification

```

1: function  $B = \text{BOUNDARYANALYSIS}(I =$ 
   ( $L_1, L_2, \dots, L_n$ )
2:   for all neighbors  $(p, q)$  in  $I$  do
3:     if  $p \in L_i, q \in L_j, (j > i)$  then
4:        $\text{Border}_i \leftarrow \text{Border}_i \cup p$ 
5:     end if
6:   end for
7:   for  $i = 1 \rightarrow n$  do
8:      $B_i = \emptyset$ 
9:     loop and convexify connected sets of pixels
10:    for  $j = 1 \rightarrow \|\text{conSet}(\text{Border}_i)\|$  do
11:       $b_j = \text{getConSet}(\text{Border}_i, j)$ 
12:       $B_i \leftarrow B_i \cup \text{Convex}(b_j)$ 
13:    end for
14:  end for
15:  return  $B = (B_1, B_2, \dots, B_n)$ 
16: end function

```

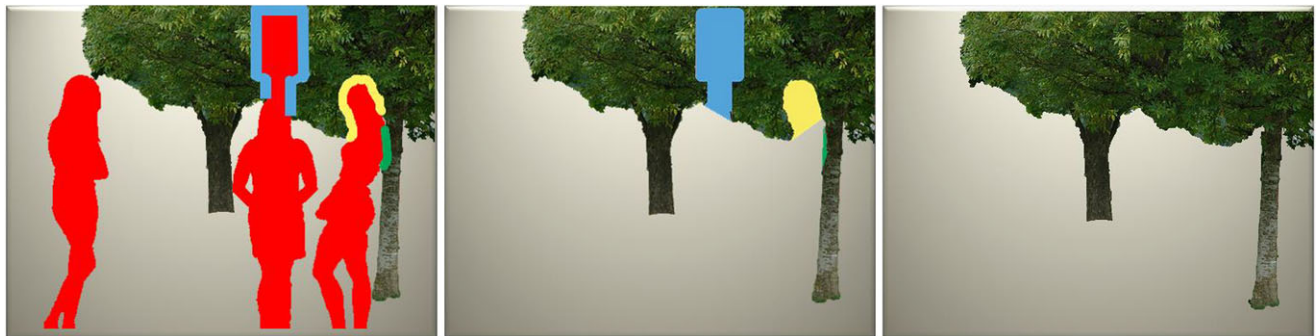


Fig. 13 Convex filling layers. Left: the red region indicates potential concealment by front layers. The blue and yellow regions are common boundary of the tree layer and frontal layers. Middle: the blue and yellow regions are the intersection of the convex borders with the front

layers designated to be completed. Right: an illustration of layer completion. Note that the image completion is carried out only for missing parts of image and not for each layer

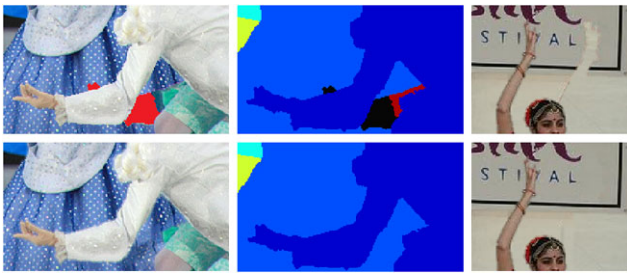


Fig. 14 Filling missing regions by utilizing the color from the correct layer (*left*). Nonmissing regions are also filled, such as the dancer hand silhouette on the sign (*right*)

Algorithm 3 The layer retargeting algorithm

```

1: procedure LAYERRETARGETING( $I =$ 
   ( $L_1, L_2, \dots, L_n$ ),  $r$ )
2:   for  $i = 1 \rightarrow n$  do
3:      $m_i \leftarrow \text{avg}(L_i)$ 
4:      $s_i \leftarrow (\frac{I_{\text{width}}}{2} - m_i) \frac{r}{2}$ 
5:      $L_i \leftarrow L_i + s_i$ 
6:   end for
7:    $B \leftarrow \text{BoundaryAnalysis}(I)$ 
8:    $J \leftarrow \text{JitterLayers}(I)$ 
9:    $B \leftarrow B + J$ 
10:  LayerCompletion( $I, B$ )
11: end procedure

12: function LAYERCOMPLETION( $I =$ 
   ( $L_1, L_2, \dots, L_n$ ),  $B$ )
13:  for  $i = n \rightarrow 1$  do
14:    place convex  $B_i$  (with value  $-i$ ) onto Image  $I'$ 
15:    place RGB value layer  $L_i$  onto image  $I'$ 
16:  end for
17:  for  $i = 1 \rightarrow n$  do
18:    Complete regions  $R \in I'$ 
19:    with value  $-i$  from layer  $L_i$ 
20:  end for
21:  Complete region  $R \in I'$  with no assignment
22:  Using layer  $L_i$  with the largest boundary
23: end function

```

surrounding the null region. This layer, again, must be concealed at that location (the grass layer in Fig. 12). The full algorithm is outlined in Algorithm 3.

Once every pixel has been associated with a layer as demonstrated in Fig. 11(d), we preform a patch-base image completion technique of [12] to fill in the designated parts. We use the patch base image completion. More results of our methods for retargeting are shown in Figs. 16 and 17.



Fig. 15 Layer operations. Cloning (*left*): a user drags and drops a layer to a new location in the image. The layer is scaled and inserted automatically utilizing the depth information. Layer transfer (*right*): a new layer from a different image replaces a previous one

We also compared our results to some examples from RetargetMe benchmark [17]. Note that as no depth information was available for these images, the layers were separated manually. As illustrated in Fig. 18, for images where there are many condensed details, our retargeting scheme creates better results by using overlaps.

7 Limitations and conclusions

We have presented a method that combines two image modalities, depth and color, to achieve an enhanced segmentation. The two channels differ in resolution, and both contain noise and missing pieces. We showed how to carry information from one modality to the other to enhance the segmentation of the image into layers. We also demonstrated a novel image retargeting algorithm that utilizes such layered segmentation for resizing complex images that are difficult for previous retargeting algorithms.

Not all images could be separated into discrete layers. Some objects such as trees or hair as well as transparent objects call for more sophisticated methods of combining layers. Shadows also create difficulties for good segmentation and retargeting. It is possible to use alpha-matting while trying to separate and combine such layers. In this case each pixel will have soft assignments to several layers instead of a single one. These situations could be distinguished by utilizing depth information as well: pixels where the depth value is not consistent over time can lie on the edges of objects or on fuzzy or transparent objects. Using several images or video may assist in recognizing these pixels. Our work only begins to explore the opportunities of combining the two modalities. There are also many other applications that could benefit from segmentation with depth data such as restructuring of recomposition and cloning. As an example, we illustrate restructuring based on layers in Fig. 15. In the future we would like to explore these extensions both for better classification of multiple layers, for extension to video, and for utilizing depth in more applications.

Fig. 16 Retargeting using depth information: (a) image, (b) depth image, (c) layers extracted, (d) retargeted image

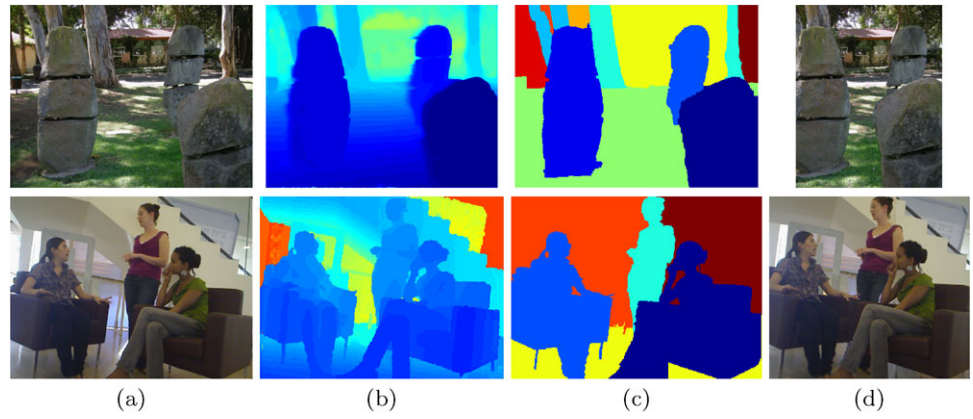


Fig. 17 Comparison of retargeting by 30%: (a) original image, (b) seam carving, (c) shift map, (d) optimized scale-and-stretch, (e) our method. Using layers (in this case artificially created) can alleviate artifacts in complex images



Fig. 18 Comparing our results (*bottom right*) to various retargeting algorithm from RetargetMe benchmark: the original image and layers are shown *on the top*. The results shown are *from top left* (see details in [17]): cropping, streaming video, multioperator, seam carving,

nonuniform scaling with bi-cubic interpolation, shift-maps, scale-and-stretch, nonhomogeneous warping, and finally at the *bottom right*, our method. As can be seen, these images contain many details that are either lost or deformed in previous approaches

Acknowledgements We would like to thank Vadim Konushin, Guofeng Zhang, Yael Pritch, and Yu-Shuen Wang for their assistance to produce and compare the results in this paper.

References

- Avidan, S., Shamir, A.: Seam carving for content-aware image resizing. *ACM Trans. Graph.* **26**(3), 10 (2007)
- Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(9), 1124–1137 (2004)
- Becker, J., Stewart, C., Radke, R.J.: Lidar inpainting from a single image. In: 3DIM2009 (2009)
- Comaniciu, D., Meer, P., Member, S.: Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **24**, 603–619 (2002)
- Carlbom, I., Paciorek, J.: Planar geometric projections and viewing transformations. *ACM Comput. Surv.* **10**(4), 465–502 (1978)
- Crabb, R., Tracey, C., Puranik, A., Davis, J.: Real-time foreground segmentation via range and color imaging. In: TOF-CV08, pp. 1–5 (2008)
- Cheng, M.-M., Zhang, F.-L., Mitra, N.J., Huang, X., Hu, S.-M.: Repfinder: finding approximately repeated scene elements for image editing. *ACM Trans. Graph.* **29**(4), 83 (2010) (SIGGRAPH 2010)
- Feldman, D., Weinshall, D.: Motion segmentation and depth ordering using an occlusion detector. *IEEE Trans. Pattern Anal. Mach. Intell.* **30**(7), 1171–1185 (2008)
- Gonzalez, R.C., Woods, R.E.: *Digital Image Processing*. Addison-Wesley, Reading (1992)
- Kopf, J., Cohen, M.F., Lischinski, D., Uyttendaele, M.: Joint bilateral upsampling. In: SIGGRAPH '07: ACM SIGGRAPH 2007 papers, New York, NY, USA. ACM, New York, p. 96 (2007)
- Krähenbühl, P., Lang, M., Hornung, A., Gross, M.: A system for retargeting of streaming video. *ACM Trans. Graph.* **28**, 126 (2009) (Proceedings Siggraph'09)
- Kwatra, V., Schödl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph.* **22**(3), 277–286 (2003)
- Leens, J., Piérard, S., Barnich, O., Droogenbroeck, M., Wagner, J.-M.: Combining color, depth, and motion for video segmentation. In: ICVS '09: Proceedings of the 7th International Conference on Computer Vision Systems, Berlin, Heidelberg, 2009, pp. 104–113. Springer, Berlin (2009)
- Mansfield, A., Gehler, P., Van Gool, L., Rother, C.: Scene carving: scene consistent image retargeting. In: European Conference on Computer Vision (ECCV) (2010)
- Pritch, Y., Kav-Venaki, E., Peleg, S.: Shift-map image editing. In: ICCV'09, Kyoto, Sep–Oct 2009
- Riemens, A., Gangwal, O., Barenbrug, B., Berretty, R.-P.: Multi-step joint bilateral depth upsampling. *Vis. Commun. Image Process.* (2009). doi:10.1117/12.805640

17. Rubinstein, M., Gutierrez, D., Sorkine, O., Shamir, A.: A comparative study of image retargeting. *ACM Trans. Graph.* **29**(5), 160 (2010) (Proc. SIGGRAPH Asia)
18. Rother, C., Kolmogorov, V., Blake, A.: “grabcut”: interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.* **23**(3), 309–314 (2004)
19. Rubinstein, M., Shamir, A., Avidan, S.: Improved seam carving for video retargeting. *ACM Trans. Graph.* (2008) (Proceedings SIGGRAPH 2008). doi:[10.1145/1399504.1360615](https://doi.org/10.1145/1399504.1360615)
20. Smith, P., Drummond, T., Cipolla, R.: Layered motion segmentation and depth ordering by tracking edges. *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(4), 479–494 (2004)
21. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vis.* (2002). doi:[10.1145/356893.356896](https://doi.org/10.1145/356893.356896)
22. Schuon, S., Theobalt, C., Davis, J., Thrun, S.: High-quality scanning using time-of-flight depth superresolution. In: TOF-CV08, pp. 1–7 (2008)
23. Vezhnevets, V.: “GrowCut”—interactive multi-label N-D image segmentation by cellular automata, 2005
24. Wolf, L., Guttman, M., Cohen-Or, D.: Non-homogeneous content-driven video-retargeting. In: Proceedings of the Eleventh IEEE International Conference on Computer Vision (ICCV-07) (2007)
25. Wexler, Y., Shechtman, E., Irani, M.: Space-time video completion. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 120–127 (2004)
26. Wang, Y.-S., Tai, C.-L., Sorkine, O., Lee, T.-Y.: Optimized scale-and-stretch for image resizing. *ACM Trans. Graph.* **27**(5), 1–8 (2008)
27. Yang, Q., Yang, R., Davis, J., Nister, D.: Spatial-depth super resolution for range images. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2007)
28. Zhang, G., Dong, Z., Jia, J., Wan, L., Wong, T.-T., Bao, H.: Refilming with depth-inferred videos. *IEEE Trans. Vis. Comput. Graph.* **15**(5), 828–840 (2009)
29. Zhang, G., Jia, J., Wong, T.-T., Bao, H.: Consistent depth maps recovery from a video sequence. *IEEE Trans. Pattern Anal. Mach. Intell.* 974–988 (2009)



Meir Johnathan Dahan is a researcher in a leading security company. He received his B.Sc. summa cum laude in computer science (2004) from Tel-Hai Academic College. He received his M.Sc. degree from Tel-Aviv University (2010) in the field of computer graphics and image processing. His main research area and interests are image synthesis and machine learning, in particular retargeting, probabilistic algorithms, and clustering.



Nir Chen received his B.Sc. degree in Computer Science and Mathematics from Bar-Ilan University and his M.Sc. degree cum laude in Computer Science at IDC Israel. Nir has an extensive practical industry background with hi-tech companies such as National Semiconductor and artNet NetCom, where he worked as a computer engineer. His main research interests are computer graphics and specifically image segmentation and retargeting algorithms.



Ariel Shamir is an associate Professor at the school of Computer Science at the Interdisciplinary Center in Israel. Prof. Shamir received his Ph.D. in computer science in 2000 from the Hebrew University in Jerusalem. He spent two years at the center for computational visualization at the University of Texas in Austin. During 2006, he held the position of visiting scientist at Mitsubishi Electric Research Labs in Cambridge MA. Prof. Shamir has numerous publications in journals and international refereed conferences, he has a broad commercial experience working with and consulting numerous companies. He is a member of the ACM SIGGRAPH, IEEE Computer and Eurographics societies.



Daniel Cohen-Or is a professor at the School of Computer Science. He received his B.Sc. cum laude in both mathematics and computer science (1985), an M.Sc. cum laude in computer science (1986) from Ben-Gurion University, and his Ph.D. degree from the Department of Computer Science (1991) at State University of New York at Stony Brook. He received the 2005 Eurographics Outstanding Technical Contributions Award. His research interests are in computer graphics, in particular, rendering and modeling techniques. His main interest right now is in few areas: image synthesis, motion and transformations, shapes and surfaces, and surface reconstruction.