Volume xx (200y), Number z, pp. 1–10

Object Repositioning Based on the Perspective in a Single Image

S. Iizuka Y. Endo M. Hirose Y. Kanamori J. Mitani Y. Fukui

University of Tsukuba

Abstract

We propose an image editing system for repositioning objects in a single image based on the perspective of the scene. In our system, an input image is transformed into a layer structure that is composed of object layers and a background layer, and then the scene depth is computed from the ground region that is specified by the user using a simple boundary line. The object size and order of overlapping are automatically determined during the reposition based on the scene depth. In addition, our system enables the user to move shadows along with objects naturally by extracting the shadow mattes using only a few user-specified scribbles. Finally, we demonstrate the versatility of our system through applications to depth-of-field effects, fog synthesis and 3D walkthrough in an image.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Applications—

1. Introduction

Object repositioning is an image editing technique for reconstructing the scene structure in an image by relocating objects already existing in the image as the user intends. Several methods for object reposition have been introduced [BSFG09, SCSI08, CZM*10, CAF10] to achieve natural reconstruction of an input image. These techniques enable the user to rearrange objects or regions with simple user operations such as region selections or specifications of line constraints. However, these methods do not take into account the effect of perspective of the scene against the rearranged objects. For example, objects should become relatively smaller than the original size if the objects are replaced in the back from their original positions. Also, an object that has been moved behind other objects should be hidden by the foreground ones. Considering these changes makes the reconstructed image more natural as if it were a real scene.

In this paper, we introduce an interactive editing system for object repositioning based on the perspective in an image. In our system, the user can rearrange object positions with automatic adjustment of the object size and order of overlapping according to the scene perspective. This is accomplished after a simple interactive processing where the user specifies a ground region in an input image with a polygonal line and roughly specifies the objects with a bounding box and a few strokes. An input image is converted to a layer structure that is composed of object layers and a background layer using the user inputs, and then the depth of the scene is computed from a ground region. To construct object layers, we propose novel saliency detection which defines the likelihood of objects using a bounding box and nearly uniform regions, called *superpixels*, to improve the quality of object extraction. In the case that the object has a shadow, moving the shadow without changing its color often causes an unnatural result. The problem can be solved by extracting the shadow matte before repositioning and then synthesizing a shadow on the target ground according to the shadow matte. We present an efficient shadow matting method to compute the shadow matte with a few simple user inputs. We demonstrate that the result of the object reposition with shadow matting becomes more natural in the scene. To summarize, the main contribution of this paper is the introduction of an integrated workflow for interactive reposition of objects and shadows based on the perspective of the scene, without requiring technical skills on image editing.

2. Related Work

There are several methods for image reshuffling to reconstruct a desired image. Cho et al. [CAF10] proposed a patch-based image reshuffling technique called patch transform. This method breaks an image into small patches and then solves for the patch transform to reconstruct the image by formulating a Markov Random Field on the image patches

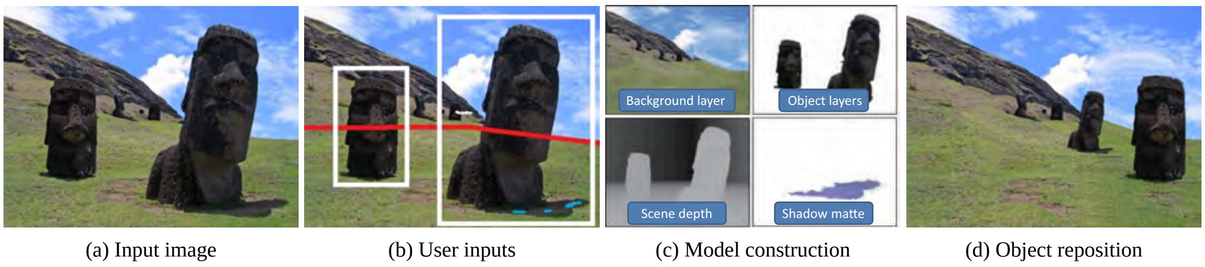


Figure 1: *Overview of our system. (a) Given a single image, (b) the user specifies a ground region with a polygonal line (red), objects with bounding boxes (white), and shadow regions with rough scribbles (blue) if shadows exist. (c) Then, the background layer (top left), object layers (top right), depth of the scene (bottom left), and shadow matte (bottom right) are generated and (d) the objects are rearranged based on the perspective of the scene.*

using belief propagation. Other methods define a global error function and minimize it by a similar patch synthesis [BSFG09] [SCSI08]. Unfortunately, these methods often cause large artifacts around rearranged objects and destroy the inherent scene structure after reconstructing the whole image from scratch.

Cheng et al. [CZM*10] proposed a system for extracting repeated objects that have similar shapes. In this method, the user specifies an object region and a background region with brush strokes. Then, the system detects all objects that have similar shapes of region boundaries and enables the user to edit the object positions, colors, and shapes simultaneously. This method does not cause distortions because it extracts the object regions previously and moves the region without changing the background. However, this method does not consider the scaling or order of overlapping of the objects caused by the scene perspective. In addition, shadows of objects are not taken into account.

Another important technique related to the object repositioning is object insertion. There are several methods for inserting objects seamlessly into an 2D image from other 2D images [PGB03, JSTS06] or into stereo images from other stereo images [LvBK*10]. In particular, the system for the object insertion considering the perspective of the scene, proposed by Lalonde et al. [LHE*07], is especially related to our work. This system enables the user to insert new objects into a photograph from an object library. At the insertion, the system adjusts the scale of the objects to match the scene based on the perspective. Our system incorporates this idea into object repositioning to achieve natural object synthesis. Note that, in our definition, simple object insertion is not object reposition because we define object reposition as to move objects already existing in an image and thus to include the accompanying editing such as object extraction and image completion, as demonstrated in this paper.

We require scene depth for adjusting object sizes and order of overlapping. There are several methods that automatically or semi-automatically compute depth of a scene from a single image. Although the automatic methods [HEH05, SCN08] do not require user efforts, they often fail to estimate depth of an image that has foreground objects on a ground. Therefore, we adopt an interactive technique for determining the 3D coordinates of the objects by defining the position of the vanishing line [KPAS01, IKMF11]. We simply define the position of the vanishing line as the top position of the ground region that is specified with a polygonal line.

3. System Overview

Figure 1 shows an overview of our system. The input of our system is a single outdoor image in which objects are placed perpendicular to the flat ground. Following this assumption, the user divides the input image into two regions; a ground region and the other regions such as buildings or the sky. Our targets for reposition are objects which are attached to the ground and their depth values are determined based on their bottom positions where they contact the ground.

User interface. In our system, the user performs two or three simple operations: (1) specifying a boundary of a ground region with a polygonal line to estimate depth of the scene; (2) setting bounding boxes around objects to extract target objects; and (3) specifying shadow regions with rough scribbles if shadows exist. After the processing, the user rearranges the position of objects by drag and drop with automatic adjustment of the object size and order of overlapping according to the scene perspective.

The processing flow of our system is as follows. First, an image is segmented into nearly uniform regions called superpixels. Then, the image is converted into a layer structure that is composed of multiple object layers and a background layer using a boundary line and a bounding boxes specified by the user. Object layers are generated based on regions of human interest called salient regions which are computed from bounding boxes and superpixels. Then, the region behind the object is filled automatically by an image patch-based completion method constrained with the polygonal line. Furthermore, if an object has a shadow, the system

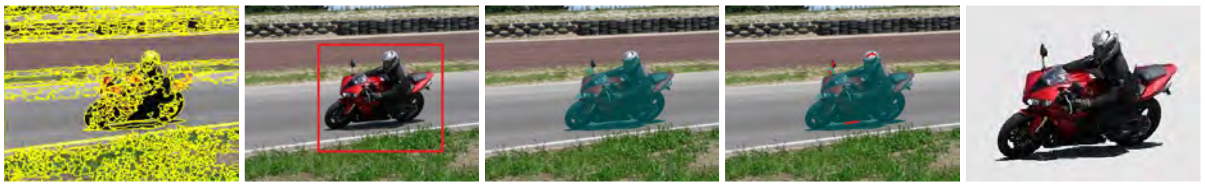

(a) Superpixels (b) User input (c) Object extraction (d) Extra input (e) Object region

Figure 2: *Overview of the object extraction. (a) An input image is segmented into superpixels in preprocessing. (b) The user sets a bounding box around the object, then (c) the object region is detected. (d) Further inputs (red) can be applied to refine the detected region and then (e) the correct object region is obtained. The object region contains the shadow region, which is then converted into a shadow matte, as described in Section 5.*

performs shadow matting based on only a few user scribbles (Figure 1(b), blue scribbles) for achieving natural shadow reposition. Finally, the system estimates the depth of the scene from the ground region to decide the size and order of overlapping of objects according to the scene. The details are described in the following sections.

4. Construction of Layer Structure

In this section, we describe the method of constructing a layer structure from an input image. A layered representation of an image is used for several image processing techniques such as image encoding [WWEA94], image retargeting [MGVGR], and object detection [YHRF12]. In our system, the layer structure composed of multiple object layers and a background layer is used for efficient object repositioning. Section 4.1 shows the algorithm to create object layers efficiently using saliency based on a bounding box and superpixels. Section 4.2 presents the way to construct a background layer.

4.1. Object Layer

To construct object layers, we extract object regions from the input image. There are several techniques for extracting a foreground region effectively from the background, e.g., paint-based techniques [LSTS04, LSS09] which paint an object region directly, or boundary-based techniques [MB95, KWT88] which select a boundary of an object. These methods are effective but often require detailed and relatively-frequent user operations in an outdoor scene where objects have complex boundary shapes. In our work, we adopt a bounding box-based approach to specify a target object roughly, as done in GrabCut [RKB04]. Lempitsky et al. use a tightness prior of a bounding box to improve the accuracy of the object extraction [LKRS09], but the method requires more computational time (up to 15 times slower than the GrabCut). Instead, we incorporate the foreground likelihood into the GrabCut framework by computing contrast-based saliency with superpixels. A salient region denotes a

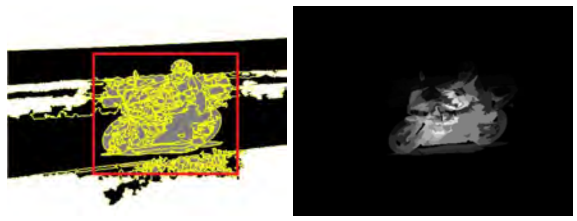

(a) Superpixels of use (b) Saliency map

Figure 3: *Computation of a saliency map. (a) We compute saliency values using black superpixels that intersect a boundary of a bounding box and gray superpixels within the box. (b) The result of our saliency map.*

region of human interest, and the superpixels are homogeneous regions obtained by a mean shift-based image segmentation [CMM02] (Figure 2(a)). Our main contribution in object extraction is an introduction of a novel saliency detection for computing the foreground likelihood based on the bounding box. For the object extraction, we optimize the following energy function:

$$E(I) = \sum_p R(I_p) + \lambda \sum_{(p,q)\in C} [I_p \neq I_q]\, B(I_p, I_q) \qquad (1)$$

where I is an input image, I_p is a pixel value at pixel p, C is a set of neighboring pixel pairs p and q, R is a data term, B is a smoothness term, λ is a constant value to define a relative importance of the smoothness term, and $[\cdot]$ denotes the indicator function. We use a smoothness term $B(I_p, I_q) = \exp(-\frac{\|I_p - I_q\|^2}{2\sigma^2}) \cdot dist(p,q)^{-1}$ as described in [BJ01], where σ is a constant value and $dist(p,q)$ is a spatial distance between p and q. To define the data term R, we use color Gaussian Mixture Models (GMM) and saliency values. The detail is shown in Section 4.1.1.

4.1.1. Calculating Saliency

A region of human interest, also called a salient region, can be regarded as an object region in many scenes. Existing

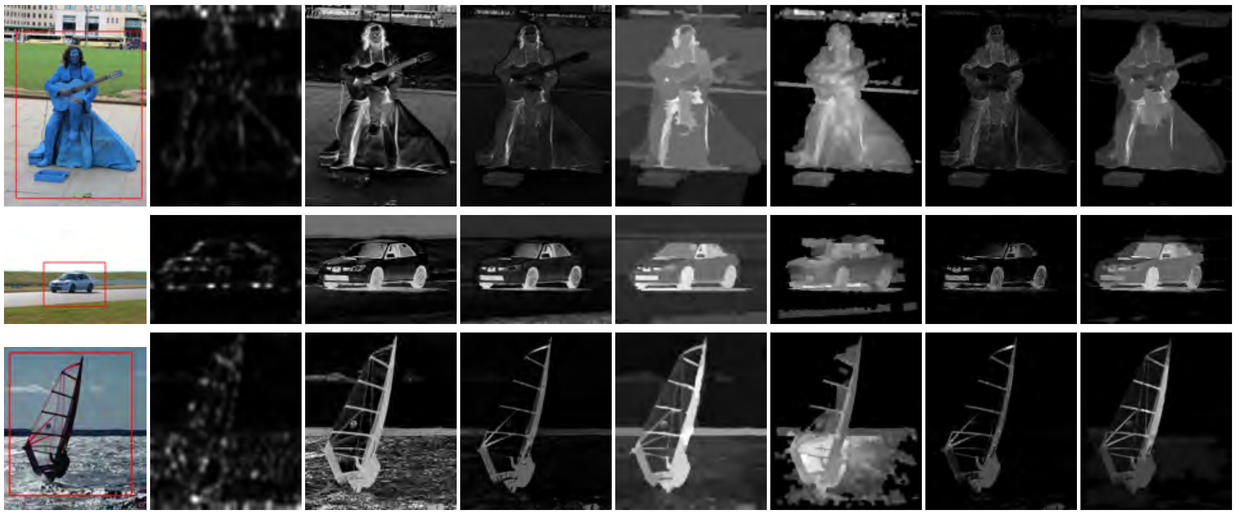


Figure 4: *Comparison with other saliency maps. (a) A bounding box is specified by the user and the saliency is computed within the red box. Compared to the existing methods, the proposed method produces high saliency values only in the object region.*

methods compute saliency in an image globally based on a biologically-plausible architecture. However, these methods often produce high saliency values not only in object regions but also in background regions. We propose a local contrast-based saliency detection using superpixels and a bounding box which is specified by the user. Our method is inspired by the global contrast-based saliency detection [CZM*11] with which the saliency is computed in the entire region in an image. In contrast to their methods, we calculate saliency locally based on the following assumptions:

1. Saliency values of superpixels that intersect a boundary of the bounding box are low because the superpixels are categorized as the background region.
2. The color distance between an object and the background region is large as described in [CZM*11].
3. Spatially-nearby superpixels are more important than the distant superpixels.
4. Large superpixels that intersect a boundary of the bounding box are more likely to be the background region because a background region typically contains large continuous chunks.
5. Superpixels close to the bounding box produce low saliency values because they are likely to be the background.

Following these assumptions, as shown in Figure 3(a), we compute saliency values based on a comparison between superpixels which intersect with the bounding box (black) and superpixels within the bounding box (gray). The other superpixels (white) are excluded from the computation for avoiding false positives.

Let r_i be a superpixel within the bounding box. The saliency value $S(r_i)$ of the superpixel r_i is calculated using a set of superpixels that intersect a boundary of the bounding box, Ω_b, as follows:

$$S(r_i) = \sum_{j \in \Omega_b} e^{-\frac{d_s(r_i,r_j)}{\sigma_1^2}} (1 - e^{-\frac{d_b(r_i)}{\sigma_2^2}})\, f(r_j)\, d_c(r_i,r_j) \quad (2)$$

where $d_s(r_i,r_j)$ is the distance between the centroids of r_i and r_j with pixel coordinates normalized to $[0,1]^2$, $d_b(r_i)$ is the distance between the centroid of r_i and the bounding box b, and σ_1 and σ_2 are constant values. In our current implementation, we set 0.5 and 0.5 respectively. The first term enhances the contrast of nearby superpixels (Assumption 3) and the second term reduces saliency values of superpixels close to the bounding box (Assumption 5). $f(r_j)$ is the ratio of the number of pixels in the superpixel r_j to the number of pixels in all superpixels that intersect the boundary of the bounding box (Assumption 4). $d_c(r_i,r_j)$ is the distance between the mean colors of r_i and r_j in the *Lab* color space (Assumption 2). Saliency values of pixels included in superpixels intersecting the boundary of the bounding box is set to zero (Assumption 1). Figure 3(b) shows the example of the saliency map produced by our method.

Using the saliency values $S(p)$, we define the data term of Eq. (1) as follows:

$$R_p(\text{'}obj\text{'}) = -\log\left(Pr(I_p|obj)S(I_p)\right) \quad (3)$$

$$R_p(\text{'}back\text{'}) = -\log\left(Pr(I_p|back)(1 - S(I_p))\right) \quad (4)$$

where $Pr(I_p|\cdot)$ is the likelihood computed using the color GMMs in the same manner as the existing method [RKB04].

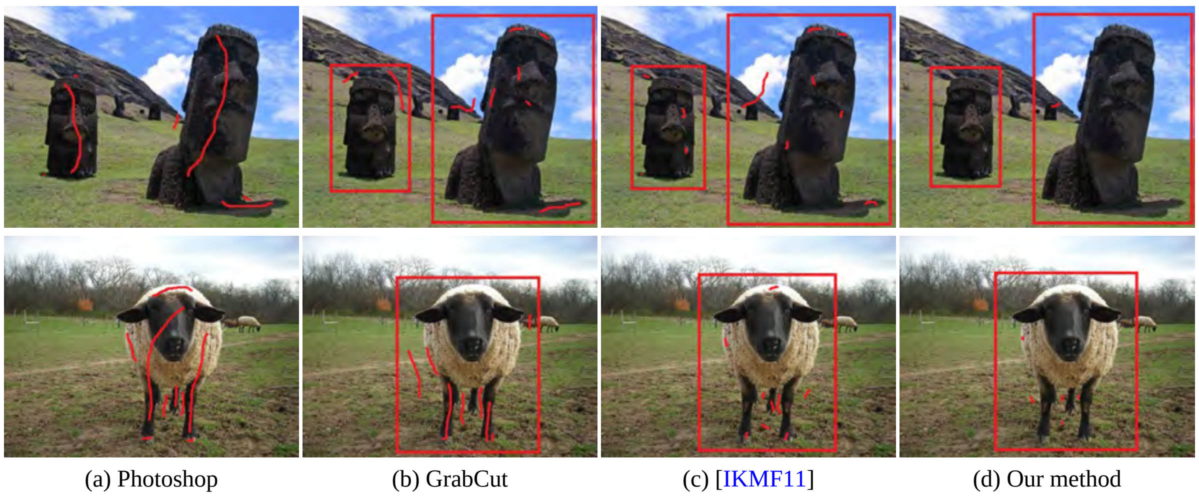


Figure 5: *Comparison with other object extraction methods. The user inputs are shown in red. Compared to (a) Photoshop Quick Selection, (b) GrabCut [RKB04], and (c) the region-based extraction [IKMF11], (d) our method requires only a few rough inputs.*

After optimizing the energy by graph cut, the object region is extracted from the image. For miss-labeled pixels, the user can assign correct labels in a superpixel basis using rough scribbles, as done in [IKMF11]. The user can also modify the miss-labeled pixels in a pixel basis if superpixels do not accurately straddle boundaries. Finally, we apply alpha matting [SJTS04] to the object boundary to achieve more natural object blending.

4.1.2. Comparisons

Figure 4 shows a comparison of saliency maps with existing methods; Hou and Zhang [HZ07], Zhai and Shah [ZS06], Achanta et al. [AHES09], Cheng et al. [CZM*11], Wei et al. [WWZS], and Perazzi et al. [PKPH12]. For a fair comparison, saliency maps of these methods are computed only from the pixels within the user-specified bounding box in our experiment. While the existing methods fail to distinguish the object region from the background as the background regions have high saliency values, the proposed method yields high saliency values in the object region while suppressing those of the background.

Figure 5 illustrates a comparison with other object extraction methods (i.e., Photoshop Quick Selection in CS5, GrabCut [RKB04], and the region-based extraction proposed by Iizuka et al. [IKMF11]). Compared to the existing methods, our method extracts the object region by more rough and less user interaction, e.g., the number of scribbles are roughly halved compared to the region-based extraction [IKMF11].

4.2. Background Layer

For rearranging positions of objects in an image, missing regions behind objects should be completed. In our system, the hole region is filled by sampling image patches from the background texture fully automatically based on the image completion method of Wexler et al. [WSI07]. For an interactive editing, we speed up the similar patch search using the randomized sampling method [BSFG09]. In the patch search, we use the polygonal line between a ground and its upper region as a constraint of the patch search space, similar to [IKMF11], to improve the quality of the background texture.

5. Shadow Extraction and Synthesis

In the case that an object has a shadow, relocating the shadow region directly will cause an unnatural result because the hue of the shadow does not change depending on the ground color. Based on the *natural shadow matting* [WTBS07], we describe an efficient shadow matting by specifying shadow regions of objects with a few scribbles. Figure 7 compares results of shadow reposition with and without shadow matting.

5.1. Natural Shadow Matting

For natural shadow reposition, we should extract the shadow matte in advance and then blends the matte with the target ground. Wu et al. [WTBS07] proposed an interactive method called the natural shadow matting which extracts a shadow matte from a single image. In their method, the user first specifies four regions to make a *quadmap*. The four regions are as follows; a *shadowed region* which is definitely shadowed, a *nonshadowed region* which is definitely unshadowed, an *uncertain region* which includes shadowed and unshadowed regions, and an *excluded region* which has

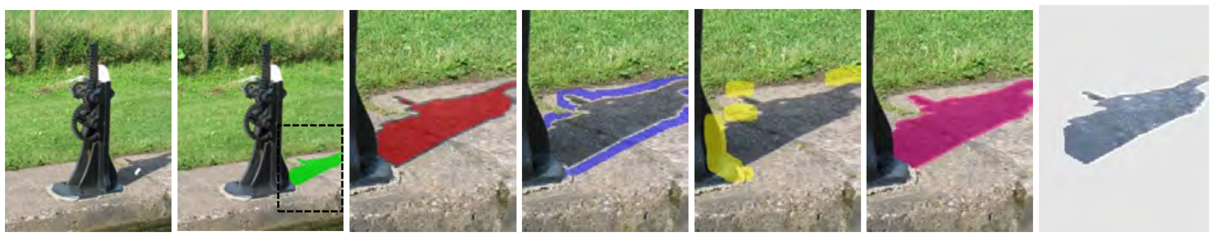


Figure 6: *Construction of a quadmap. Given an image, (a) the user specifies the shadow mask with scribbles (white) and then (b) the shadow mask are extracted based on superpixels (green). (c)-(g) are zoom-in images of the region marked with the dotted black rectangle in (b). Using the shadow mask, the four regions are extracted by the system: (c) definitely shadowed (red), (d) definitely unshadowed (blue), (e) excluded (yellow), and (f) uncertain (pink) regions. Using the quadmap, (g) the shadow matte is computed by the natural shadow matting.*

irrelevant colors in shadow calculation. Using the quadmap, the shadow matte is generated based on energy minimization that utilizes color transfer, gradient of the texture, and smoothness of the shadow.

5.2. Generating Quadmap Using GMM

Although the natural shadow matting can extract a fine shadow matte, it is a tedious task to specify the four regions on every object manually. In our system, the shadow matte can be extracted by specifying only a single region, i.e., the shadow mask of an object (Figure 6). Here we use the term *shadow mask* to differentiate from a definitely *shadowed region* in a quadmap; a user-specified shadow mask is likely to contain nonshadowed pixels at the boundary, and thus should not be used for a shadowed region as it is.

Our quadmap is calculated as follows. First, a shadow mask of an object is specified by the user in a superpixel basis using a few scribbles. Then, we first reduce the size of the shadow mask and define it as a definitely *shadowed region*. Then, we dilate the shadow mask in twice and categorize the difference region into two regions based on the color distribution. In our system, the first dilation size is 5 pixels and the second size is 17 pixels. For the categorization, we fit a two-component GMM to the color distribution of the pixels and make two clusters based on the fitted Gaussian probability distributions. We label pixels that belong to the larger cluster as *nonshadowed regions*. Pixels that belong to the other cluster are dilated in the image space and labeled as *excluded regions*. Finally, the user-specified region is dilated and defined as an *uncertain region*. Now, we can obtain the shadow matte by the natural shadow matting technique using the quadmap.

6. Estimating Depth of Scene

We estimate depth of a scene from a top position of a ground region that is specified with a polygonal line [KPAS01, IKMF11] as discussed in Section 2. Then, the order of the overlapping objects is determined based on the depth coordinates of the bottom position of the object region. We calculate the image height of repositioned object i, h'_i, as follows. Following Hoiem et al. [HEH06], the world height y_i of the object i is computed by the relationship $y_i = \frac{y_c h_i}{v_0 - v_i}$, where h_i is the original image height of the object, y_c is the camera height, v_0 is the y coordinate of the vanishing line, and v_i is y coordinate at object's bottom. Let v'_i be the repositioned y coordinate at the object's bottom. The world height y_i is then expressed as $y_i = \frac{y_c h'_i}{v_0 - v'_i}$ as well. From these two expressions we can eliminate y_c and obtain

$$h'_i = \frac{v_0 - v'_i}{v_0 - v_i} h_i \tag{5}$$

Although the calculated coordinates are not accurate, we can obtain a convincing result of an object reposition based on the perspective.

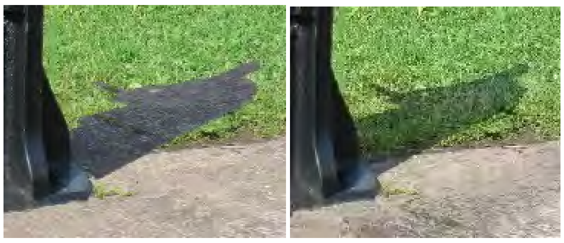


Figure 7: *Reposition of the shadow together with the object in Figure 6(a). (a) The color of the shadow does not conform to the color of the ground without shadow matting. In contrast, (b) our system achieves a plausible result by using shadow matting.*

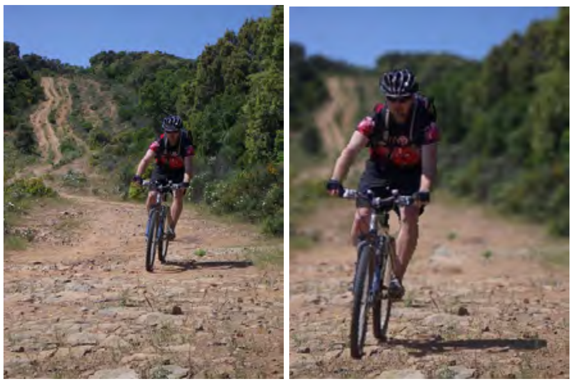
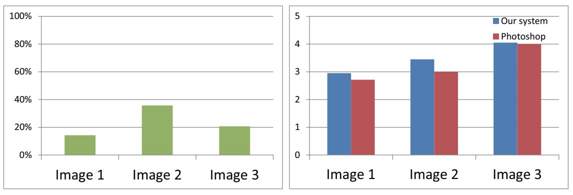


Figure 8: *Object repositioning and depth-of-field effect by blurring the background based on the perspective of the scene.*



(a) Time (b) Quality

Figure 9: *Comparisons of the editing time and the quality of object repositioning using different tools. (a) The average editing time by novice users of our system is 3-6 times shorter than the time by experienced users of Photoshop (100%). (b) The quality of resultant images using our system is comparable to that of Photoshop.*

7. Results

We implemented our prototype system with C++, OpenGL and GLUT, and ran the program on a PC with Intel Core i7 620M 2.67GHz CPU and 4.00GB RAM. Most of the input images are taken from the image database [EVGW*] and the sizes are all in the range of 0.3 to 1 megapixels. In the result, the average processing time of a computation of superpixels is about 1.5 seconds, a computation of saliency is about 0.02 seconds, and a shadow extraction is about 0.5 seconds. The total processing time including manual operations is within 2 minutes. Note that superpixels are computed as preprocessing, thus the computation time is negligible in an interactive editing for object reposition. For much larger images (more than 10 megapixels), GPU-based acceleration is desirable.

Object Repositioning. After all the preprocessing, the user can rearrange each object intuitively by dragging it with automatic adjustment of the size or order of overlapping based on the perspective. Figures 1, 11, and 12 show the results of object repositioning using our system. The rearranged objects look fine in the scene. For example, in the middle of Figure 11, the relocated cow is occluded by the foreground cow.

Depth-of-field Effect. To enhance a target object, our system can produce a depth-of-field effect by blurring the background region in an image. We blur the background and object layers except the target one by changing the standard deviation σ of the Gaussian filter based on the depth value d, i.e., $\sigma = \frac{\sigma_{max}-\sigma_{min}}{d_{max}-d_{min}}(d-d_{min})+\sigma_{min}$, where d_{min} and d_{max} are the minimum and maximum depths respectively, and σ_{min} and σ_{max} are constant values for adjusting the strength of blur. Figures 8 and 13 middle show the results of depth-of-field effects that focus on objects.

Fog Synthesis. Our system can also simulate aerial perspective to enhance the perceived depth by synthesizing fog or haze according to the scene depth. Our system can synthesize fog using the standard model of a fog image [Fat08]:

$$\mathbf{I}_{out} = t(\mathbf{x})\mathbf{I}_{in} + (1-t(\mathbf{x}))\mathbf{A} \tag{6}$$

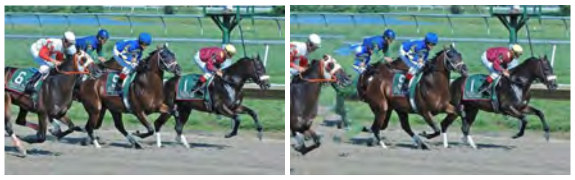
$$t(\mathbf{x}) = e^{-\frac{b}{z}} \tag{7}$$

where $\mathbf{A}$ is the color of fog, b is a constant value, and z is the depth value of position $\mathbf{x}$. In our current implementation, we set $\mathbf{A} = (0.9, 0.9, 0.9)$ and $b = 0.07$. The result of Figure 13 right shows a visually plausible fogged scene based on the scene depth.

3D Walkthrough. We also show an application to virtual walkthrough in an image. Based on the scene depth, we construct a simple 3D model similarly to Iizuka et al. [IKMF11]. In their method, the model is represented as simple planar polygons that are composed of a background plane and foreground planes. This method can be used in our system by replacing the layers with the polygons. Figure 14 shows the result of 3D walkthrough which achieves a convincing 3D effect.

7.1. User study

We conducted a user study to evaluate usefulness of our system. We requested five novice users of our system and two experienced users of the commercial image editing tool (Adobe Photoshop CS6) to perform object repositioning using each tool. The three images used in this study are Figures 1 left, 11 top left, and 11 middle left. The requested task is to rearrange positions of objects as naturally as possible by moving objects near the camera backward and vice versa while adjusting the object sizes and the order of overlapping according to the scene perspective. The editing time was recorded, and the resultant images were evaluated by four evaluators using a subjective score ranging from 1 to 5, as summarized in Figure 9. Thanks to the simple and few user operations in our system, the editing times of our system are typically around 1-2 minutes, which are about 3-6 times



(a) Input image (b) After repositioning

Figure 10: *Typical failure case of our system. When the left-most horse and jockey in (a) were repositioned, (b) texture completion for the second-left failed because of the large occlusion.*

shorter than the times of the commercial tool, while the quality of our system is comparable to that of the commercial tool. However, some of the users mentioned that specifying the boundary line of a ground is slightly difficult because it is sometimes obscure as in Figure 1(a).

7.2. Limitations

Our current system has the following limitations. As shown in Figure 10, texture completion often fails in case of objects largely occluded by others because estimating the shape and texture of the hidden parts is difficult, which can be handled to some extent by, for example, a data-driven approach [GCZ*12]. Our depth estimation does not work well for indoor images because we need to specify a vanishing line, which is assumed to lie sufficiently far from the camera. Also, we do not handle *relighting*, i.e., changes of objects' shading when the lighting environment changes, e.g., from a shady area to a sunny area and vice versa. For relighting, we should estimate the object shape, reflectance and the lighting environment.

8. Conclusions

We have proposed an interactive image editing system for object reposition based on the perspective of the scene. Our system can adjust the object size and order of overlapping based on the perspective by constructing a layer structure and estimating the scene depth. Furthermore, the system can achieve a natural shadow reposition by specifying only the shadow region with a few scribbles. Our system can also be applied to depth-of-field effects, fog synthesis and 3D walk-through in an image.

In future work, we would like to improve the quality of a background layer and change the shading of repositioned objects by employing relighting. Also, we would like to allow the user to modify the shadow similarly to the method of [SCRS] for more natural object rearrangement.

Acknowledgment

We would like to thank the following Flickr users for Creative Commons imagery: David Jones, Alfonso, Alice Popkom, Lali Masriera, and Florida Fish and Wildlife's photostream.

References

[AHES09] Achanta R., Hemami S., Estrada F., Susstrunk S.: Frequency-tuned Salient Region Detection. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2009)* (2009), pp. 1597 – 1604. 4, 5

[BJ01] Boykov Y., Jolly M.-P.: Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *Proceedings of ICCV* (2001), vol. 1, pp. 105–112. 3

[BSFG09] Barnes C., Shechtman E., Finkelstein A., Goldman D. B.: PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. of SIGGRAPH) 28*, 3 (aug 2009). 1, 2, 5

[CAF10] Cho T. S., Avidan S., Freeman W. T.: The patch transform. *IEEE Trans. Pattern Anal. Mach. Intell. 32*, 8 (2010), 1489–1501. 1

[CMM02] Comaniciu D., Meer P., Member S.: Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24* (2002), 603–619. 3

[CZM*10] Cheng M.-M., Zhang F.-L., Mitra N. J., Huang X., Hu S.-M.: RepFinder: Finding approximately repeated scene elements for image editing. *ACM Transactions on Graphics 29*, 4 (2010), 83:1–8. 1, 2

[CZM*11] Cheng M.-M., Zhang G.-X., Mitra N. J., Huang X., Hu S.-M.: Global contrast based salient region detection. In *Proceedings of CVPR* (2011), pp. 409–416. 4, 5

[EVGW*] Everingham M., Van Gool L., Williams C. K. I., Winn J., Zisserman A.: The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. 7

[Fat08] Fattal R.: Single image dehazing. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 72:1–72:9. 7

[GCZ*12] Goldberg C., Chen T., Zhang F.-L., Shamir A., Hu S.-M.: Data-driven object manipulation in images. *Computer Graphics Forum 31*, 2pt1 (2012), 265–274. 8

[HEH05] Hoiem D., Efros A. A., Hebert M.: Automatic photo pop-up. *ACM Trans. Graph. 24*, 3 (July 2005), 577–584. 2

[HEH06] Hoiem D., Efros A. A., Hebert M.: Putting objects in perspective. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2* (2006), CVPR '06, IEEE Computer Society, pp. 2137–2144. 6

[HZ07] Hou X., Zhang L.: Saliency detection: A spectral residual approach. In *Proceedings of CVPR* (2007), pp. 1–8. 4, 5

[IKMF11] Iizuka S., Kanamori Y., Mitani J., Fukui Y.: Efficiently modeling 3D scenes form a single image. *IEEE Computer Graphics and Applications 32*, 6 (2011), 18–25. 2, 5, 6, 7

[JSTS06] Jia J., Sun J., Tang C.-K., Shum H.-Y.: Drag-and-drop pasting. *ACM Trans. Graph. 25*, 3 (July 2006), 631–637. 2

[KPAS01] Kang H. W., Pyo S. H., Anjyo K., Shin S. Y.: Tour into the picture using a vanishing line and its extension to panoramic images. *Computer Graphics Forum 20*, 3 (2001), 132–141. 2, 6

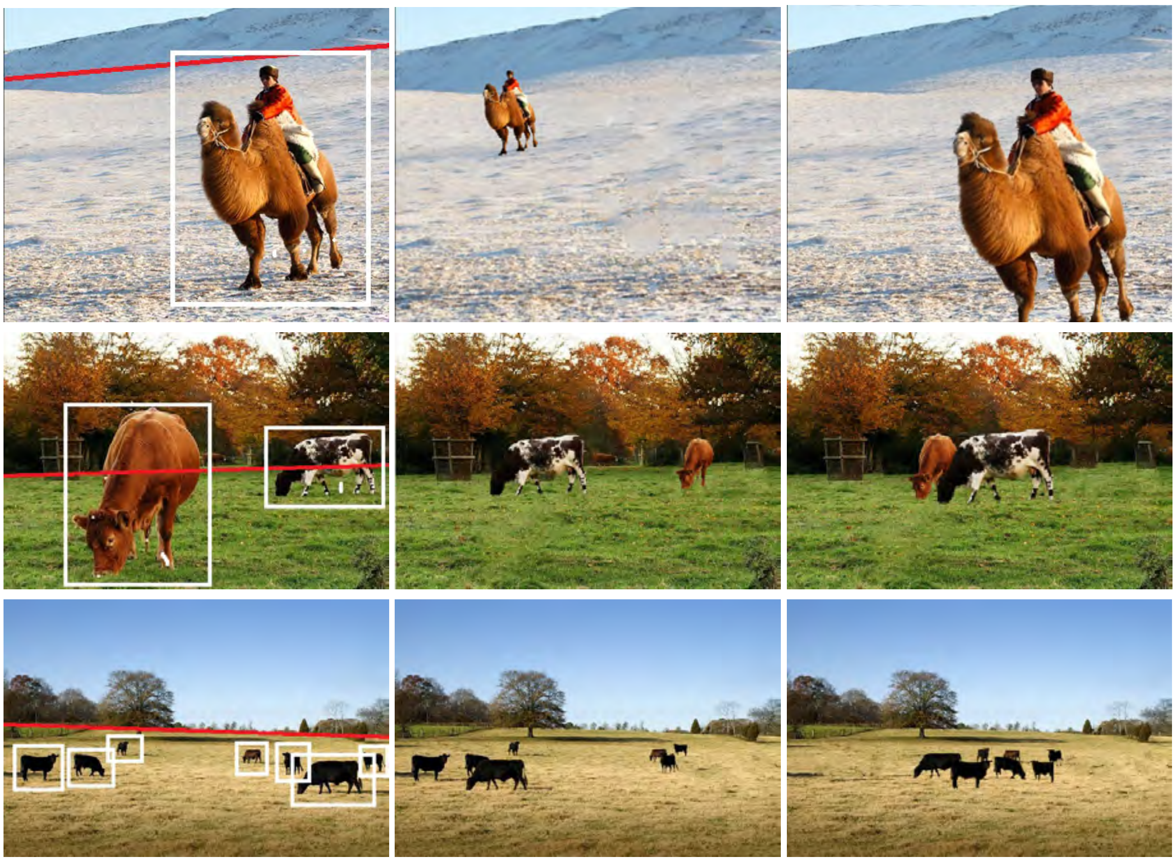


Figure 11: *Results of object repositioning. The input images with user inputs are shown in the left-most column. The red lines specify boundaries of ground regions, and white bounding boxes and scribbles are used for the objects.*

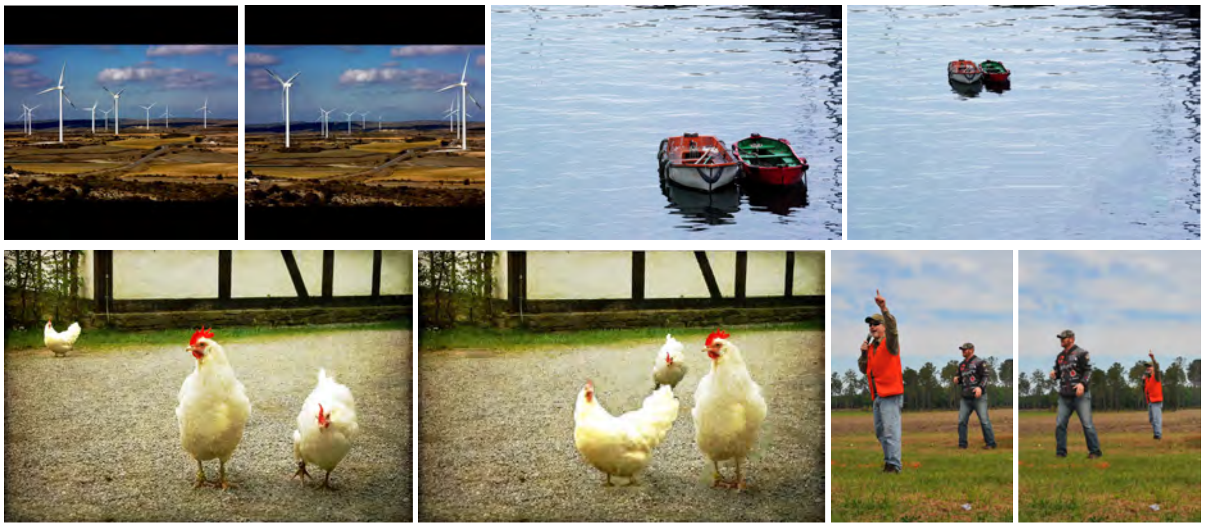


Figure 12: *Other examples of object repositioning.*

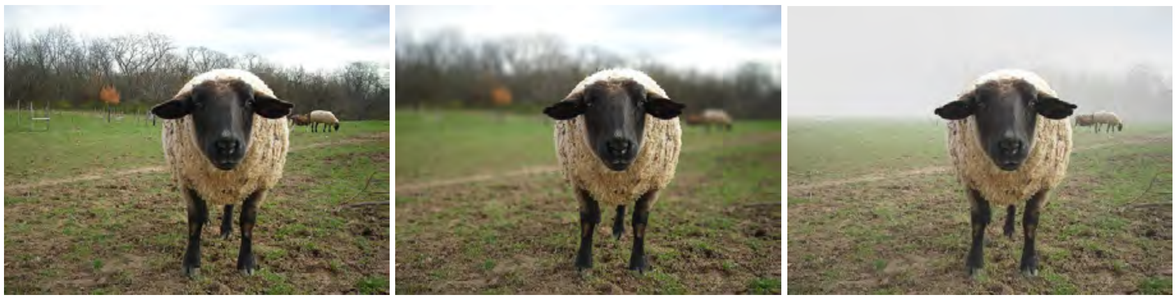


Figure 13: *Depth-of-field effect and fog synthesis. The left column is the input image, the middle column is the result of background blur, and the right column is the result of synthesis of fog.*

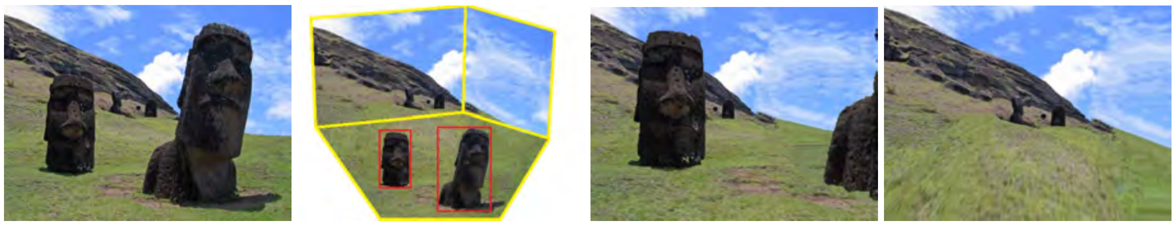


Figure 14: *Example of 3D walkthrough. Left to right: the input image, the 3D model, and the novel views.*

[KWT88] Kass M., Witkin A., Terzopoulos D.: Snakes: Active contour models. *INTERNATIONAL JOURNAL OF COMPUTER VISION 1*, 4 (1988), 321–331. 3

[LHE*07] Lalonde J.-F., Hoiem D., Efros A. A., Rother C., Winn J., Criminisi A.: Photo clip art. *ACM Transactions on Graphics (SIGGRAPH 2007) 26*, 3 (August 2007), 3. 2

[LKRS09] Lempitsky V., Kohli P., Rother C., Sharp T.: Image segmentation with a bounding box prior. pp. 277–284. 3

[LSS09] Liu J., Sun J., Shum H.-Y.: Paint selection. *ACM Trans. Graph. 28*, 3 (jul 2009), 69:1–69:7. 3

[LSTS04] Li Y., Sun J., Tang C.-K., Shum H.-Y.: Lazy snapping. *ACM Trans. Graph. 23*, 3 (aug 2004), 303–308. 3

[LvBK*10] Lo W.-Y., van Baar J., Knaus C., Zwicker M., Gross M.: Stereoscopic 3D copy & paste. *ACM Trans. Graph. 29*, 6 (Dec. 2010), 147:1–147:10. 2

[MB95] Mortensen E. N., Barrett W. A.: Intelligent scissors for image composition. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), SIGGRAPH '95, ACM, pp. 191–198. 3

[MGVGR] Mansfield A., Gehler P., Van Gool L., Rother C.: Scene carving: Scene consistent image retargeting. In *Proceedings of the 11th European Conference on Computer Vision: Part I*, ECCV'10, Springer-Verlag, pp. 143–156. 3

[PGB03] Pérez P., Gangnet M., Blake A.: Poisson image editing. *ACM Trans. Graph. 22*, 3 (July 2003), 313–318. 2

[PKPH12] Perazzi F., Krähenbühl P., Pritch Y., Hornung A.: Saliency filters: Contrast based filtering for salient region detection. In *CVPR* (2012), pp. 733–740. 4, 5

[RKB04] Rother C., Kolmogorov V., Blake A.: GrabCut: interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph. 23*, 3 (aug 2004), 309–314. 3, 4, 5

[SCN08] Saxena A., Chung S. H., Ng A. Y.: 3D depth reconstruction from a single still image. *Int. J. Comput. Vision 76*, 1 (Jan. 2008), 53–69. 2

[SCRS] Shesh A., Criminisi A., Rother C., Smyth G.: 3D-aware image editing for out of bounds photography. In *Proceedings of Graphics Interface 2009* (Toronto, Ont., Canada, Canada), GI '09, pp. 47–54. 8

[SCSI08] Simakov D., Caspi Y., Shechtman E., Irani M.: Summarizing visual data using bidirectional similarity. In *Proceedings of CVPR* (2008), pp. 1–8. 1, 2

[SJTS04] Sun J., Jia J., Tang C.-K., Shum H.-Y.: Poisson matting. *ACM Trans. Graph. 23*, 3 (aug 2004), 315–321. 5

[WSI07] Wexler Y., Shechtman E., Irani M.: Space-time completion of video. *IEEE Trans. Pattern Anal. Mach. Intell. 29*, 3 (2007), 463–476. 5

[WTBS07] Wu T.-P., Tang C.-K., Brown M. S., Shum H.-Y.: Natural shadow matting. *ACM Trans. Graph. 26*, 2 (jun 2007). 5

[WWEA94] Wang J., Wang J. Y. A., Edward, Adelson H.: Representing moving images with layers. *IEEE Transactions on Image Processing 3* (1994), 625–638. 3

[WWZS] Wei Y., Wen F., Zhu W., Sun J.: Geodesic saliency using background priors. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part III*, ECCV'12, Springer-Verlag, pp. 29–42. 4, 5

[YHRF12] Yang Y., Hallman S., Ramanan D., Fowlkes C. C.: Layered object models for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*, 9 (2012), 1731–1743. 3

[ZS06] Zhai Y., Shah M.: Visual attention detection in video sequences using spatiotemporal cues. In *Proceedings of the 14th annual ACM international conference on Multimedia* (2006), MULTIMEDIA '06, ACM, pp. 815–824. 4, 5