Image Composition with Depth Registration

Zan Li[1,2] , **Wencheng Wang**[1,2*] and **Fei Hou**[1,2*]
[1]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences, Beijing, China
{lizhan, whn, houfei}@ios.ac.cn

Abstract

Handling occlusions is still a challenging problem for image composition. It always requires the source contents to be completely in front of the target contents or needs manual interventions to adjust occlusions, which is very tedious. Though several methods have suggested exploiting priors or learning techniques for promoting occlusion determination, their potentials are much limited. This paper addresses the challenge by presenting a depth registration method for merging the source contents seamlessly into the 3D space that the target image represents. Thus, the occlusions between the source contents and target contents can be conveniently handled through pixel-wise depth comparisons, allowing the user to more efficiently focus on the designs for image composition. Experimental results show that we can conveniently handle occlusions in image composition and improve efficiency by about 4 times compared to Photoshop.

1 Introduction

Image composition tries to merge selected contents from the source image (*source contents*) into the target image to produce a plausible image [Pérez *et al.*, 2003; Guo and Sim, 2009]. Till now, existing methods pay much attention to color blending for a seamless composition [Pérez *et al.*, 2003; Zhan *et al.*, 2020; Levin *et al.*, 2007; Yu *et al.*, 2021; Bao *et al.*, 2022], including a large quantity of matting methods [Levin *et al.*, 2007; Tang *et al.*, 2019; Huang *et al.*, 2019] and cloning-based methods [Farbman *et al.*, 2011; Pérez *et al.*, 2003]. As for the correct occlusion relationships required between the source contents and the contents in the target image (*target contents*), corresponding studies are insufficient [Niu *et al.*, 2021]. Existing methods always require the source contents to be completely in front of the target contents. For compositing the source contents that are partially occluded by the target contents, the user needs to remove the occluded parts of the source contents with careful manual interventions or use image editing software (e.g., Photoshop) to separate the contents into different layers and manually identify the orders of the layers to obtain the correct occlusions between the contents. These manual interventions are laborious, preventing image composition from applications.

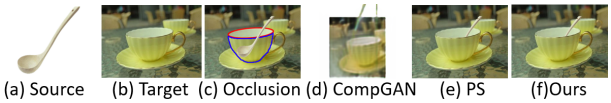

Figure 1: For the composition of placing a spoon into a cup, most existing methods cannot handle the occlusions here (c), as some parts of the spoon are occluded by the outer wall of the cup (inside the blue outlines) while some parts of the spoon occlude the inner wall of the cup (inside the red outlines). For example, the learning-based method compGAN [Azadi *et al.*, 2020] cannot handle the occlusions (d). When Photoshop was used for a plausible composition (e), it took us 78 manual operations and 169 seconds. Using our proposed method, the composition was fulfilled by only 4 manual operations and 32 seconds (f).

To promote image composition of partially occluded objects, Tan et al. [Tan *et al.*, 2019] proposed using two priors to infer occlusions. The first prior is that when a source content is completely inside the region of a target content, the source content must be in front of the target content. The second prior is that, when there is a common support plane for the source content and its overlapped target contents, the depth values of the sites for these contents to touch the support plane can be used to deduce the occlusions between them. Unfortunately, these priors cannot be applied in many cases; e.g., there is no common support plane for the source contents and their overlapped target contents. In particular, when the source content and the target content occlude each other, this method cannot be used, as illustrated in Fig. 1(c). In [Azadi *et al.*, 2020], learning techniques are exploited for handling complicated interactions between different real-world objects for image composition, including the occlusions between source contents and target contents. Unfortunately, it can only learn occlusions from the training data, and therefore its generalization potentials are much limited; e.g., it cannot deal with the example in Fig. 1(d).

This paper addresses the challenge of handling occlusions efficiently by presenting a depth registration method for seamlessly merging source contents into the 3D space represented by the target image. Thus, occlusions between source

*Corresponding authors

contents and target contents can be easily determined, reducing the amount of manual intervention for occlusion determination. This method benefits from the learning techniques that perform well in segmenting objects in images and estimating their depths, and from the observation in [Tan *et al.*, 2019] that image composition is always by mixing objects in an image, by which we can handle objects instead of pixels for image composition. Clearly, the estimated depths of objects in different images cannot be used directly for determining the occlusions between the objects in a composition image. Fortunately, the human visual system is very sensitive to the size of an object in the image with respect to its distance from the viewpoint. Thus, we can adjust the size of the source object to look as if it is standing at the same distance as a target object, by which the source object can be registered into the target image. For high-quality depth registration, in Section 3 we develop novel measures to construct a correspondence between the size of the source object and the depths of its pixels in the target image, so that complicated occlusions like those in Fig. 1 can be well handled with much fewer manual interventions than existing methods. As a result, without being distracted by occlusion determination in image composition, the user can more efficiently focus on design by adjusting the locations and sizes of source objects in the target image to achieve satisfactory results, as shown in experiments.

2 Related Work

Various methods have been proposed for image composition. Unfortunately, most of them are focused on color blending [Cong *et al.*, 2021; Cong *et al.*, 2022; Hang *et al.*, 2022], and pay little attention to occlusion determination between the source contents and target contents, though correct occlusions are important for a plausible composition.

Color blending methods can be classified into two categories: matting-based methods [Levin *et al.*, 2007; Tang *et al.*, 2019; Huang *et al.*, 2019] and cloning-based methods [Farbman *et al.*, 2011; Pérez *et al.*, 2003; Fu *et al.*, 2008]. Matting-based methods try to produce an alpha matte to represent the source contents, whose alpha weights are used for pixel-wise linear color interpolation between the source and the target images. Cloning-based methods take the outlined source image patch containing interested contents as input and paste the input onto the target image for a composition. The color discrepancies between the source image and target image on the boundary of the input patch are propagated over the entire cloned area for a seamless color blending. In general, these methods require the source contents to be completely in front of the target contents in the composition. If some parts of the source contents are occluded by the target contents in the composition, the occluded parts of the source contents should be removed, and this generally requires substantial user interventions, such as drawing an excessive number of strokes or trimaps to generate suitable mattes [Levin *et al.*, 2007; Tang *et al.*, 2019] or laboriously drawing the boundary to exclude the occluded parts from the patch for cloning [Farbman *et al.*, 2011]. Intensive manual interventions are very troublesome, preventing image composition from applications.

Though there are some learning based methods proposed for image composition [Tang *et al.*, 2019; Huang *et al.*, 2019], they still require the source contents over the target image, which means that partially occluded source contents cannot be composited into the target image.

For compositing partially occluded source contents, image editing software (e.g. Photoshop) tries to separate the image contents into different layers and identify the orders of the layers for occlusion determination. However, such a procedure is very time-consuming.

To our knowledge, two priors are used in [Tan *et al.*, 2019] to facilitate occlusion determination to promote the composition of partially occluded objects. As discussed in Section 1, however, these two priors are not suitable in many cases and so this method is still prevented from compositing images with complicated occlusions. Learning techniques have also been exploited to determine the occlusions between objects and promoting image composition [Azadi *et al.*, 2020]. However, as the technique is dependent on the training data for learning occlusions, its potentials are limited and its handling is inconvenient, especially when handling complicated occlusions, as illustrated in Fig. 1(d).

3 Depth Registration

The proposed depth registration method places the source object into the 3D space represented by the target image. To achieve this, we first perform semantic segmentation and depth estimation in the source image and the target image to obtain objects and their related depths, to which end many existing learning techniques can be used. As the source image and the target image represent different 3D spaces, the estimated depths for the source object cannot be used directly in the target image. Fortunately, the depth differences between pixels of the source object are concerned with the 3D existence of the source object, and so keeping unchanged in the source image and target image. As a result, these depth differences can be exploited for depth registration. Our process of depth registration consists of the following three operations:

- **Registration initialization** places the source object into the 3D space of the target image (Subsection 3.1).
- **Adaptive depth adjustment** computes pixelwise depths to determine occlusions between the source object and the target objects for a composition (Subsection 3.2).
- **Registration refinement** optimizes depths and object segmentation to correct unsuitable occlusions (Subsection 3.3).

The pipeline for depth registration is illustrated in Fig. 2. Here, we generate a structure called *difference-template* for representing the 3D existence of the source object. The image patch of the extracted source object in the source image is called the *source image patch* in Fig. 2(a). Its *difference-template* in Fig. 2(b) is constructed to record the depth differences between the pixels of the *source image patch* and an *anchor point* of the source object; e.g., the red point P in Fig. 2(a), to be discussed in Subsection 3.1. Adaptive depth adjustment is then performed according to the location

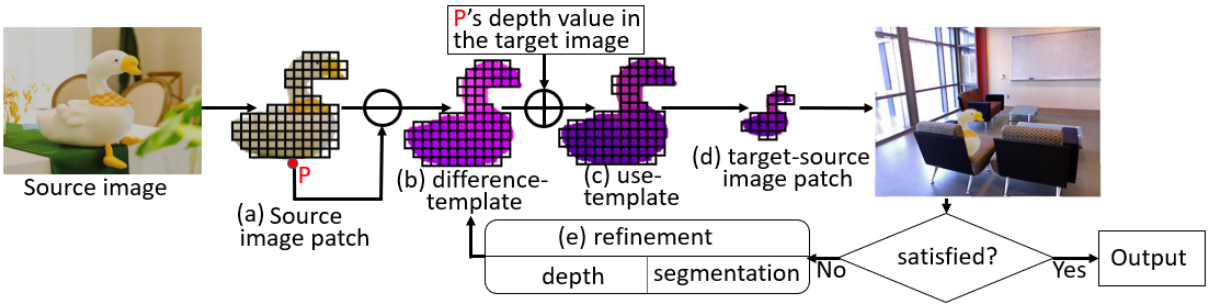

Figure 2: Pipeline for depth registration. The steps here are explained in the second paragraph of Section 3.

of the source object for its composition in the target image in Fig. 2(c) and Fig. 2(d), In Fig. 2(c), in order to compute the depth for each pixel of the *source image patch* in the target image, a *use-template* is formed from the *difference-template* and the depth of the anchor point in the target image. By mapping the *use-template* to the target image, we can obtain the depths for the pixels of the image patch of the source object in the target image, called the *target-source image patch*, as illustrated in Fig. 2(d).

As the estimated depths and object segmentation are not very accurate, some occlusions by the registered depths are not correct. To remedy this, we refine the registration. We allow the user to manually correct unsuitable occlusions, so that the *difference-template* has its depth differences refined correspondingly. As a result, the subsequent depth registration for other designs of compositing the source object into the target image will be facilitated as refined depths and segmented objects would have fewer and fewer incorrect occlusions generated.

3.1 Registration Initialization

We initialize depth registration by resizing the *target-source image patch* of the source object against a target object in the target image until the two objects look as if they are standing at the same distance from the viewpoint, as illustrated in Fig. 3.

At this time, it is generally regarded that the lowest pixels of these two objects in the target image have the same distance from the viewpoint, meaning the same depth value. Thus, the lowest pixel of the *target-source image patch* for the source object has its depth value in the target image; i.e., the depth of the lowest pixel of the target object, as illustrated in Fig. 3(c). Clearly, this lowest pixel has a corresponding point in the *source image patch*, called the *anchor point*. If there are several pixels at the lowest sites in the target-source image patch, any one of them can be selected as the *anchor point*, without preventing image composition.

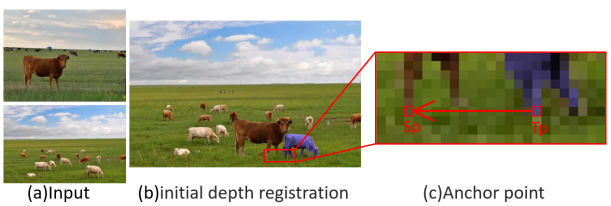

Figure 3: Place a source object (a tan cow) near a target sheep (marked in blue). Depth registration is initialized in (b), and the anchor point (in red) is identified in (c).

By depth interpolation of the neighboring pixels around the *anchor point* in the *source image patch*, the *anchor point* has its depth in the source image. Thus, the pixels of the *source image patch* compute their depth differences from the *anchor point*, forming the *difference-template*.

Clearly, having obtained the *difference-template* and the depth of the *anchor point* in the target image, we can now construct the *use-template* of the source object, and place it into the 3D space of the target image.

3.2 Depth Determination for Composition

After initializing depth registration, the user designs the composition by progressively adjusting the location and size of the source object until satisfactory results are produced. In this procedure, the depths of the overlapped pixels for the target objects and the source object are compared to determine the occlusion between them, where the pixels of the source object have their depths determined adaptively by investigating the size variation of the *target-source image patch*.

As Fig. 4 shows, for a 3D object projected onto the image plane, its two points P_1 and P_2 lie on a plane, which is parallel to the image plane, and their projection points on the image plane, P_1' and P_2', have the following equations,

$$h_1/m_1 = d/f. \tag{1}$$

$$h_2/m_2 = d/f. \tag{2}$$

$$(h_1 - h_2) \times f = (m_1 - m_2) \times d. \tag{3}$$

where h_1 and h_2 are the heights of P_1 and P_2 in the 3D space, m_1 and m_2 are the heights of P_1' and P_2' in the image plane, f is the focal distance from the viewpoint to the image plane

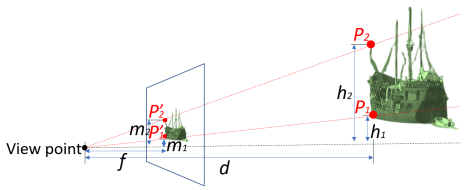

Figure 4: The diagram for a 3D object projected onto the image plane to generate an image with respect to the viewpoint.

and d is the distance from the plane of P_1 and P_2 to the viewpoint in the 3D space.

As we know, when the user places a source object into the target image for image composition, the user may shrink, enlarge, or rotate the source object. With these operations, the target-source image patch for the source object varies . Some variations are caused by the changes in the distance from the source object to the image plane, while some are not; e.g., making the 3D shape of the source object wider or shorter. Thus, we should distinguish whether the user's interventions are intended to resize the 3D shape of the source object or to change its depth. Fortunately, it seldom occurs that the user simultaneously resizes the source object and changes its depth. Thus, when the user's intention is to resize the source object, the user should add an intervention as an indication. Then, the other cases of size variation in the *target-source image patch* all stem from the user's interventions to change the depth of the source object. Under this assumption, the proposed measure of depth computation is as follows.

Depth Computation

In image composition, no matter what happens to cause size variations in the *target-source image patch*, we observe that for two points P_1 and P_2 on a plane parallel to the image plane, they still remain on a plane parallel to the image plane. In addition, when the source object has its corresponding 3D shape unchanged, $(h_2 - h_1)$ remains unchanged for P_1 and P_2, as shown in Fig. 4. As f is also fixed, we have

$$(m_2' - m_1') \times d' = (m_2 - m_1) \times d = (h_2 - h_1) \times f, \quad (4)$$

when the source object is placed at another location with d' for P_1 and P_2, and their m_1' and m_2' on the image plane. As we know, $(m_2' - m_1')$ and $(m_2 - m_1)$ can be easily detected on the target image, so that d' for producing m_2' and m_1' can be easily determined, assuming that $(m_2 - m_1)$ is the result for P_1 and P_2 in the initial registration.

As we know, a pixel is a rasterized point, which can be approximated by a small rectangle parallel to the image plane. For a pixel of the *source image patch*, its related rectangle varies in size in the target image when the distance from the source object to the image plane changes. Thus, according to the size variations of the rectangle, we can compute the depth of the pixel by Equation (4). On the other hand, with the changes in distance, all rectangles of the pixels for the source object vary in the same way, meaning these rectangles have their heights changed by the same rate d_{rate}. Thus, we can use the variations in the *target-source image patch* to compute the depth of the pixels of the source object respectively, using the following equations,

$$d_p = d_{ini} \times d_{rate}, \quad (5)$$

$$d_{rate} = height_{ini}/height_p, \quad (6)$$

where d_p is the depth of pixel p of the source object when the source object is placed at a new location in the target image, d_{ini} is the depth of pixel p when the source object is initially depth registered into the target image, d_{rate} is the variation rate, $height_{ini}$ is the height of the *target-source image patch* in the initial depth registration, and $height_p$ is the height of the *target-source image patch* at the new location.

Note that when the 3D shape of the source object is not changed in composition, the size variations in its target-source image patch are caused only by distance changes. Thus, the variation rate along any direction that is parallel to the image plane for the *target-source image patch* is the same, as it is only determined by f and d. Therefore, we compute d_{rate} by the height variation here, without loss of generality.

When the source object has its 3D shape changed in the composite, we compute d_{rate} by investigating the changes, as discussed in the following "Computation of d_{rate}".

With d_{rate} determined, each pixel can have its depth computed using Equation (5). As discussion in Subsection 3.1, the *difference-template* represents the 3D existence of the source object. Thus, after computing the depth for the *anchor point* for the source object at the new location, we can obtain the depths of the pixels of the *use-template* from the *difference-template*, where only one addition is required for each pixel. This can save a lot of time compared with using Equation (5) directly, which needs multiplications. As a result, we obtain the depths of the pixels of the *target image patch* by mapping the *use-template*.

Computation of d_{rate}

In Equation (6), the variation rate d_{rate} is computed by the variation in height of the *target-source image patch*. However, in adjusting the size of the *target-source image patch* for composition using a series of manual interventions, the variation rates along different directions may be different. To solve this, we need to investigate the variation rates along all directions and select the most suitable for depth computation.

Note that only depth change can cause size variations in the target-source image along all directions and at the same variation rate, whereas other changes have not such an effect; e.g., widening the target-source image. Therefore, we investigate the variation rates of the target-source image along all directions and select the one that is nearest to 1.0. In implementation, we investigate only the variation rates along the horizontal direction and the vertical direction, as the user generally manipulates the *target-source image patch* along the horizontal or the vertical.

3.3 Registration Refinement

In image composition, when the source objects are registered into the target image, the occlusions between the source object and the target objects can be determined by their pixelwise depths. Unfortunately, estimating pixelwise depths and object segmentation by learning techniques is not very accurate and may cause unsuitable occlusions, requiring the user to perform manual corrections. Therefore, the pixelwise depths and object segmentation are refined, which helps facilitate the subsequent composition attempts.

Considering that image composition actually combines some meaningful parts of objects rather than entire objects, we further segment the source objects and the target objects into semantic parts respectively, where many methods can be used for this purpose. We use the Watershed Algorithm [Kornilov and Safonov, 2018], and correct occlusions by parts instead of pixels. In this way, interventions are easy and more efficient. Fig. 5 shows the process of depth refinement for

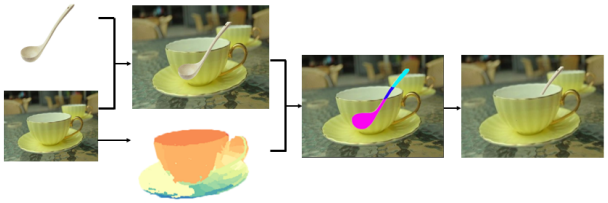

Figure 5: Depth refinement by semantic subdivision for occlusion correction.

occlusion correction by subdividing an object into semantic parts.The wall of the cup is subdivided into the inner and the outer; therefore, the spoon can be compared with the subdivided parts for refining occlusion determination. As a result, when there is an incorrect occlusion, the user need only click the corresponding parts for occlusion correction.

In our treatment, we first compute the overlapped regions between the source object and the target objects by parts. Thus, each overlapped region is related to only a part of a target object, and in general only an occlusion relationship is required in each overlapped region. The exceptional case is when the parts are concave, in which case such parts could be further decomposed into convex subparts for convenient implementation. Therefore, we expect all pixels in each overlapped region to satisfy $depth_p(t) >= depth_p(s)$ or $depth_p(t) <= depth_p(s)$, where $depth_p(t)$ and $depth_p(s)$ refer to the depths of pixel p for the target object t and the source object s respectively. Based on this, we correct unsuitable occlusions and refine depths and object segmentation correspondingly, as discussed below. The refinement result is then transferred to the *difference-template* of the source object for refining the representation of its 3D existence.

Refinement Measures

The refinement process updates the depths of the related objects or their segmentation. This is achieved by firstly refining the source object, as we expect the target image to remain unchanged if possible. When desired results cannot be obtained by refining the source object, we will further refine the related target objects that have unsuitable occlusions with the refined source object. By refining the source object and the related target objects iteratively, the desired results can be obtained as the estimated depths and object segmentation using learning techniques provide a good base for our implementation. In practical tests, refinement generally takes 1 or 2 iterations.

Using our method, overlapped regions are investigated respectively. For each overlapped region, we first deduce its occlusion relationship by investigating the pairs of $(depth_p(t), depth_p(s))$. If most pixels here have $depth_p(t) >= depth_p(s)$, it means the source object occludes the overlapped target object; otherwise, the source object is occluded by the overlapped target object. For the unsuitable occlusions, we allow the user to manually click the corresponding regions. Thus, the expected occlusions in all the overlapped regions are obtained, for which the refinement process is as follows.

We first refine depths and then refine object segmentation. For depth refinement to reduce depth conflicts in occlusion determination, we deduce a depth offset β and a parameter $\alpha \in [0, 1]$ to flatten the object. In general, the occlusions in the overlapped regions can be obtained correctly with depth refinement, but some pixels in the overlapped regions may still have unsuitable occlusions. For these pixels, we investigate whether they are outliers of the object and exclude such outliers for refining the object segmentation here. If there are still some pixels with unsuitable occlusions after object refinement, we resort to iterative refinement of the source object and the related target objects alternatively.

In sum, the refinement process involves the following steps:

1) Occlusions in overlapped regions are deduced respectively.

2) The user interactively clicks the regions with unsuitable occlusions.

3) The pixelwise depths and object segmentation are refined iteratively by the following sequence of steps until expected occlusions in all the overlapped regions are achieved.

3.1) Refine depths for the source object;

3.2) Refine object segmentation for the source object;

3.3) Refine depths for the overlapped target objects;

3.4) Refine object segmentation for the overlapped target objects.

Without loss of generality, we now discuss the measures for depth refinement and object refinement.

- **Depth refinement.** Depth refinement is defined by the following formula:

$$depth'_i = depth_{AcPoint} + \alpha \times (depth_i - depth_{AcPoint}) + \beta, \quad (7)$$

where $depth'_i$ is the refined depth for pixel i from its previous depth $depth_i$, and $depth_{AcPoint}$ is the depth of the anchor point. Clearly, β can be derived by investigating the depth differences between $depth_p(t)$ and $depth_p(s)$ for the pixels in the regions with wrong occlusions; e.g., $\beta = max(|depth_p(t) - depth_p(s)|)$ if the source object is expected to be occluded, and $\beta = -max(|depth_p(t) - depth_p(s)|)$ otherwise. As for α, it should be as close as possible to 1.0 for retaining the originally estimated 3D shape of the object. With all overlapped regions considered, α and β can be computed by dynamic programming. Two examples are illustrated in Fig. 6 and Fig. 7.

- **Object refinement.** Refining object segmentation is computed by finding and excluding the outliers of the object in an overlapped region. First, we check the colors and depths of the pixels of the object under investigation in an overlapped region; e.g., $(color_p, depth_p)$ for pixel p. In general, these pixels should be clustered very well by their $(color_p, depth_p)$. The pixels that are on the boundary of the object and have much different $(color_p, depth_p)$ from their neighboring pixels in this overlapped region are regarded as outliers. Here, for each pixel p, we compute its maximum depth difference and color difference with its neighboring pixels in the overlapped region; e.g., $dep\text{-}diff(p)$ and $clr\text{-}diff(p)$. We then obtain the average values for these depth differences and color differences; e.g., $average\text{-}dep\text{-}diff$ and $average\text{-}clr\text{-}diff$. For each pixel p, if $dep\text{-}diff(p) \notin [average\text{-}dep\text{-}diff \times (1.0 -$

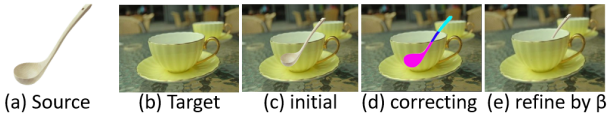

Figure 6: Depth refinement by the depth offset β. With initial depth registration, the spoon is not well placed into the cup (c). The user clicks the region with a wrong occlusion, the purple part of the spoon (d), and the pixels of the spoon have their depths refined by β to obtain a composition with correct occlusions (e).

Figure 7: Depth refinement by flattening the source object via α. β is adjusted to make the bird's depth shift as a whole, but the left wing of the bird is still wrongly occluded at this time (d). Subsequent refinement by α achieves the final result (e).

$\lambda)$, $average\text{-}dep\text{-}diff \times (1.0 + \lambda)]$, or $clr\text{-}diff(p) \notin [average\text{-}clr\text{-}diff \times (1.0-\mu), average\text{-}clr\text{-}diff \times (1.0+\mu)]$, the pixel is regarded as an outlier. Many tests show that setting λ=0.7 and μ=0.7 can always obtain good results.

4 Algorithm for Image Composition

Using the measures in Section 3, we present an image composition algorithm that can handle complicated occlusions conveniently and efficiently. The algorithm has the following steps:

1) Semantic segmentation and depth estimation are executed in the source image and the target image using learning techniques.

2) The user selects an interested source object and places it onto the target image to test designs. A satisfactory composite is achieved by the following steps:

2.1) The source object is initially depth registered into the 3D space of the target image.

2.2) The user tries a design by placing the source object with a suitable size onto the target image for a composition. Here, occlusions are determined by pixelwise depths, and color blending methods are then used for the visible pixels of the source object.

2.3) When there are incorrect occlusions in the composite obtained in 2.2), the user manually clicks the related regions for occlusion correction.

2.4) Step 2.2) and 2.3) are repeated until a satisfactory composite is obtained.

4.1 Implementation

In implementing our method, we use the following techniques for semantic segmentation, depth estimation, and color blending.

Semantic segmentation uses primarily the popular method DeepLabV3+ [Chen *et al.*, 2018]. We also use GrabCut [Rother *et al.*, 2004] to further optimize the extracted objects.

Depth estimation uses NeWCRFs [Yuan *et al.*, 2022] to obtain a high-resolution depth map given a single RGB image.

Color blending uses the enhanced matting method of [Wang *et al.*, 2016], which computes the color at pixel i by the following formulae,

$$f_i = (1-w_i)\times t_i + w_i \times g_i + w_i \times \sum_{j=1}^{N} W_j S(i,j)(t_j - g_j), \quad (8)$$

where f_i, t_i, and g_i are the colors of the pixel i in the composited image, the target image, and the source image respectively, and N is the set of pixels in the source object. The weight w_i is set to 0.8 for the pixels on the boundary of the source object, and 0.9 for the pixels inside the source object, which has always produced very good composites in our tests, as detailed in Section 5. W is the normalized weight for distinguishing smoothing effects in different regions, computed as $W_j = (1 - w_j)/\sum_{k=1}^{N}(1 - w_k)$ for pixel j, and $S(i,j) = exp(-||p_i - p_j||^2)$ for pixels i and j, where p_k denotes the pixel position.

5 Results and Discussion

The proposed method was implemented in Python 3.8.9. For comparison, we implemented the method in [Tan *et al.*, 2019] in Visual C++ and downloaded the learned network (*CompGAN*) of [Azadi *et al.*, 2020]. Comparisons with Adobe Photoshop 2020 were also performed. Experimental results were collected on a personal computer installed with an Intel(R) Core(TM) i7-6700 CPU, 16GB RAM, and Windows 10. Due to limit of the paper length, we list only a few results here for discussion. More results are provided in the supplementary materials.

Quality. As shown in Fig. 8, it is clear that our method and Photoshop can well handle complicated occlusions to produce plausible composition images. However, the results produced by the [Tan *et al.*, 2019] and compGAN[Azadi *et al.*, 2020] methods are implausible as they cannot handle many occlusions, like the occlusions with off-ground objects in the 3rd row in Fig. 8 and the cyclic occlusions in the 6th row. This is attested by the quantitative results in Table 1, where accuracy refers to the percentage of pixels with correct occlusion labels in the source image patch, and the Photoshop compositions are used as the ground truth as they are manually produced. Though Photoshop can produce results as plausible as ours, it is inefficient and very troublesome to use.

Efficiency. In Table 2, we list the required interventions and times for composing the results shown in Fig. 8, comparing our method with Photoshop. These statistics show that we can considerably reduce the interventions and time costs, achieving an acceleration of 145.6/36.3=4.01 times over Photoshop on average.

Applications. In general, the user needs to try many designs to achieve a satisfactory composition result. As our method inserts the source object into the 3D space represented by the target image, the refined results for depths and object segmentation can be used repeatedly for facilitating occlusion determination. Thus, when the user tries designs, the required interventions for occlusion correction are reduced

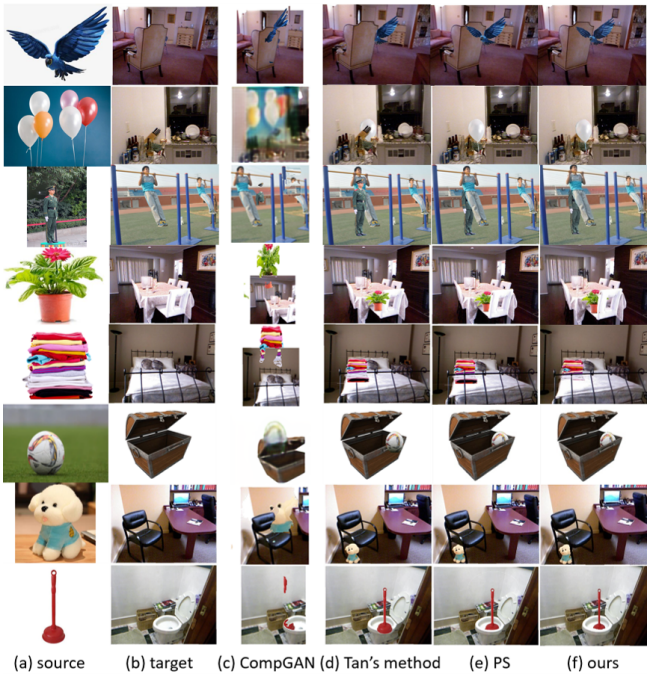

Figure 8: Compositions compared.

and image composition is more efficient. In comparison, Photoshop needs respective interventions to determine occlusions for each composition, which is inefficient. The example in Fig. 9 shows that the user needs four interventions to obtain a satisfactory composition using our method, whereas many more are required by Photoshop (Fig. 10).

Limitation. Our method relies on the experiences of human stereoscopy in the real 3D world. If the user lacks such experiences, a good result would not be produced. Fortunately, the user is always familiar with the objects selected for image composition in applications. Thus, our method can help promote corresponding applications.

	Accuracy			
	compGAN	Tan's	ours	our*
Row1	0.2100	0.7001	**0.9826**	-
Row2	0.0000	0.7951	**0.9896**	-
Row3	0.1674	0.8163	**0.9799**	-
Row4	0.4322	0.6304	0.7846	**0.9995**
Row5	0.4503	0.7913	**0.9687**	-
Row6	0.5222	0.5786	**0.9731**	-
Row7	0.0000	0.9178	0.9218	**0.9887**
Row8	0.1998	0.8357	**0.9809**	-
Average	0.2477	0.7582	0.9477	0.9941

Note: "our*" refers to the results with some required refinements; "-" means no refinement required.

Table 1: Quantitative comparisons for the composition results in Fig. 8.

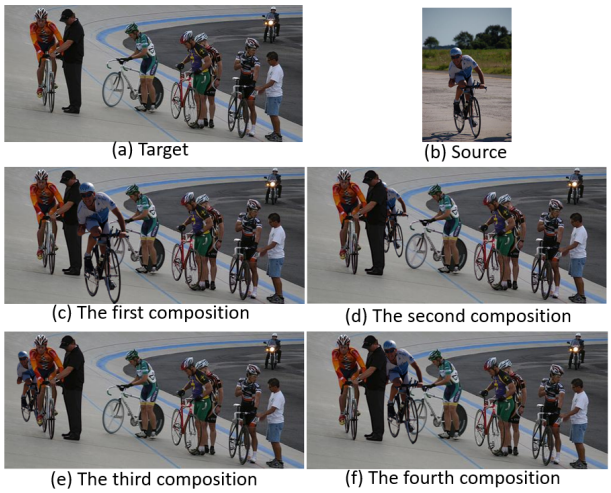

Figure 9: Four interventions achieve a satisfactory composition using our method.

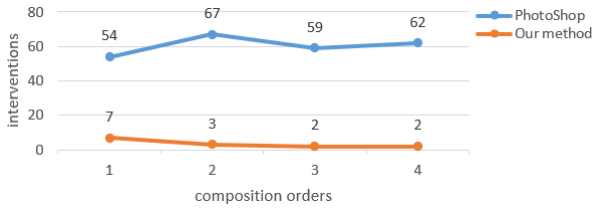

Figure 10: Interventions compared for the results in Fig. 9.

6 Summary

This paper has presented a novel method for depth registration of the source object into the 3D space that is represented by the target image. Thus, the occlusions between the source object and the target objects can be easily handled. As a result, we can handle the occlusion problem in image composition more conveniently compared to existing methods. Experimental results show that we can conveniently composite images with complicated occlusions and considerably reduce manual interventions, making efficiency improved by about 4 times compared to Photoshop.

	Photoshop		Ours	
	Time	Interventions	Time	Interventions
Row1	142.0s	71	33.0s	4
Row2	158.0s	84	34.0s	5
Row3	139.0s	87	32.0s	4
Row4	160.0s	93	44.0s	5
Row5	144.0s	77	45.0s	5
Row6	137.0s	75	33.0s	4
Row7	133.0s	82	35.0s	4
Row8	152.0s	73	34.0s	4
Average	145.6s	80.3	36.3s	4.4

Table 2: Time costs and interventions for the results in Fig. 8

Acknowledgements

This work is partially supported by the National Natural Science Foundation of China under Grant. 62072446 and 61872347. We highly appreciate the constructive suggestions of anonymous reviewers.

References

[Azadi *et al.*, 2020] Samaneh Azadi, Deepak Pathak, Sayna Ebrahimi, and Trevor Darrell. Compositional gan: Learning image-conditional binary composition. *International Journal of Computer Vision*, 128(10):2570–2585, 2020.

[Bao *et al.*, 2022] Zhongyun Bao, Chengjiang Long, Gang Fu, Daquan Liu, Yuanzhen Li, Jiaming Wu, and Chunxia Xiao. Deep image-based illumination harmonization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18542–18551, 2022.

[Chen *et al.*, 2018] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[Cong *et al.*, 2021] Wenyan Cong, Li Niu, Jianfu Zhang, Jing Liang, and Liqing Zhang. Bargainnet: Background-guided domain translation for image harmonization. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2021.

[Cong *et al.*, 2022] Wenyan Cong, Xinhao Tao, Li Niu, Jing Liang, Xuesong Gao, Qihao Sun, and Liqing Zhang. High-resolution image harmonization via collaborative dual transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18470–18479, 2022.

[Farbman *et al.*, 2011] Zeev Farbman, Raanan Fattal, and Dani Lischinski. Convolution pyramids. *ACM Trans. Graph.*, 30(6):175, 2011.

[Fu *et al.*, 2008] Xinyuan Fu, He Guo, Yuxin Wang, Tianyang Liu, and Han Li. Arbitrary image cloning. In *Signal Processing for Image Enhancement and Multimedia Processing*, pages 289–300. Springer, 2008.

[Guo and Sim, 2009] Dong Guo and Terence Sim. Color me right–seamless image compositing. In *International Conference on Computer Analysis of Images and Patterns*, pages 444–451. Springer, 2009.

[Hang *et al.*, 2022] Yucheng Hang, Bin Xia, Wenming Yang, and Qingmin Liao. Scs-co: Self-consistent style contrastive learning for image harmonization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19710–19719, 2022.

[Huang *et al.*, 2019] Han Huang, Yihui Liang, Xiaowei Yang, and Zhifeng Hao. Pixel-level discrete multiobjective sampling for image matting. *IEEE Transactions on Image Processing*, 28(8):3739–3751, 2019.

[Kornilov and Safonov, 2018] Anton S Kornilov and Ilia V Safonov. An overview of watershed algorithm implementations in open source libraries. *Journal of Imaging*, 4(10):123, 2018.

[Levin *et al.*, 2007] Anat Levin, Dani Lischinski, and Yair Weiss. A closed-form solution to natural image matting. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):228–242, 2007.

[Niu *et al.*, 2021] Li Niu, Wenyan Cong, Liu Liu, Yan Hong, Bo Zhang, Jing Liang, and Liqing Zhang. Making images real again: A comprehensive survey on deep image composition. *arXiv preprint arXiv:2106.14490*, 2021.

[Pérez *et al.*, 2003] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *ACM SIGGRAPH 2003 Papers*, pages 313–318. 2003.

[Rother *et al.*, 2004] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "GrabCut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004.

[Tan *et al.*, 2019] Xuehan Tan, Panpan Xu, Shihui Guo, and Wencheng Wang. Image composition of partially occluded objects. *Computer Graphics Forum*, 38(7):641–650, 2019.

[Tang *et al.*, 2019] Jingwei Tang, Yagiz Aksoy, Cengiz Oztireli, Markus Gross, and Tunc Ozan Aydin. Learning-based sampling for natural image matting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3055–3063, 2019.

[Wang *et al.*, 2016] Wencheng Wang, Panpan Xu, Xiaohui Bie, and Miao Hua. Enhanced use of mattes for easy image composition. *IEEE Transactions on Image Processing*, 25(10):4608–4616, 2016.

[Yu *et al.*, 2021] Bing Yu, Youdong Ding, Zhifeng Xie, and Dongjin Huang. Stacked generative adversarial networks for image compositing. *EURASIP Journal on Image and Video Processing*, 2021(1):1–20, 2021.

[Yuan *et al.*, 2022] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. Newcrfs: Neural window fully-connected crfs for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[Zhan *et al.*, 2020] Fangneng Zhan, Shijian Lu, Changgong Zhang, Feiying Ma, and Xuansong Xie. Adversarial image composition with auxiliary illumination. In *Proceedings of the Asian Conference on Computer Vision*, 2020.