Occlusion Handling using Semantic Segmentation and Visibility-Based Rendering for Mixed Reality

Menandro Roxas
The University of Tokyo
Tokyo, Japan
roxas@cvl.iis.u-tokyo.ac.jp

Tomoki Hori
The University of Tokyo
Tokyo, Japan
hori@cvl.iis.u-tokyo.ac.jp

Taiki Fukiage
NTT Communication Science Laboratories
Tokyo, Japan

Yasuhide Okamoto
The University of Tokyo
Tokyo, Japan
okamoto@cvl.iis.u-tokyo.ac.jp

Takeshi Oishi
The University of Tokyo
Tokyo, Japan
oishi@cvl.iis.u-tokyo.ac.jp

ABSTRACT

Real-time occlusion handling is a major problem in outdoor mixed reality system because it requires great computational cost mainly due to the complexity of the scene. Using only segmentation, it is difficult to accurately render a virtual object occluded by complex objects such as vegetation. In this paper, we propose a novel occlusion handling method for real-time mixed reality given a monocular image and an inaccurate depth map. We modify the intensity of the overlayed CG object based on the texture of the underlying real scene using visibility-based rendering. To determine the appropriate level of visibility, we use CNN-based semantic segmentation and assign labels to the real scene based on the complexity of object boundary and texture. Then we combine the segmentation results and the foreground probability map from the depth image to solve the appropriate blending parameter for visibility-based rendering. Our results show improvement in handling occlusions for inaccurate foreground segmentation compared to existing blending-based methods.

CCS CONCEPTS

• **Human-centered computing** → **Mixed / augmented reality**; • **Computing methodologies** → **Image-based rendering**; **Mixed / augmented reality**;

KEYWORDS

ACM proceedings, LaTeX, text tagging

ACM Reference Format:
Menandro Roxas, Tomoki Hori, Taiki Fukiage, Yasuhide Okamoto, and Takeshi Oishi. 2018. Occlusion Handling using Semantic Segmentation and Visibility-Based Rendering for Mixed Reality. In *VRST 2018: 24th ACM Symposium on Virtual Reality Software and Technology (VRST '18), November 28-December 1, 2018, Tokyo, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3281505.3281546

VRST '18, November 28-December 1, 2018, Tokyo, Japan

ACM ISBN 978-1-4503-6086-9/18/11…$15.00
https://doi.org/10.1145/3281505.3281546

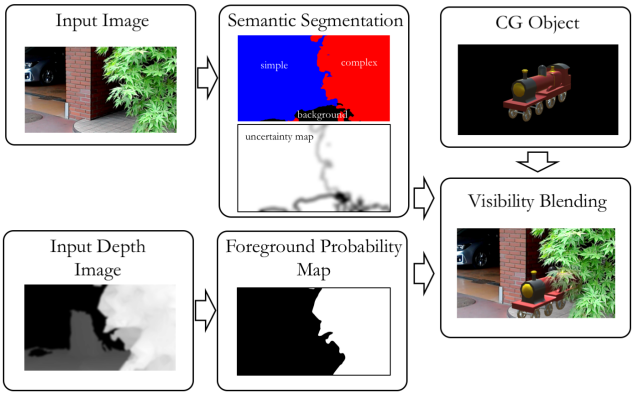

Figure 1: Overview of the proposed method.

1 INTRODUCTION

In mixed reality, contradictory occlusion problem happens when a foreground real object is partially or completely covered by a background virtual object. There are many ways to solve this problem that have been presented in previous work. The most straightforward solution is to create an accurate foreground mask of the real scene. In practice, foreground extraction is either an ill-posed problem (using only RGB images) or a sparse one (using RGBD images).

Several methods have been proposed that handles foreground extraction in a monocular camera. Methods presented in [23] and [17] can extract an accurate foreground region from still images. However, extending the method to videos is inherently difficult due to computational cost of the segmentation technique used. In [28] and [6], contours are cut by using alpha matting [23], however an accurate prior foreground estimation is still required and the method fails even for small inaccuracy in the estimation especially for complex scenes such as tree branches and vegetation.

Background subtraction methods [6][27] achieve real-time processing and can be applied for monocular cameras. In [11][16], the background subtraction technique are modified and applied on an arbitrary outdoor environment taking advantage of multipile cues

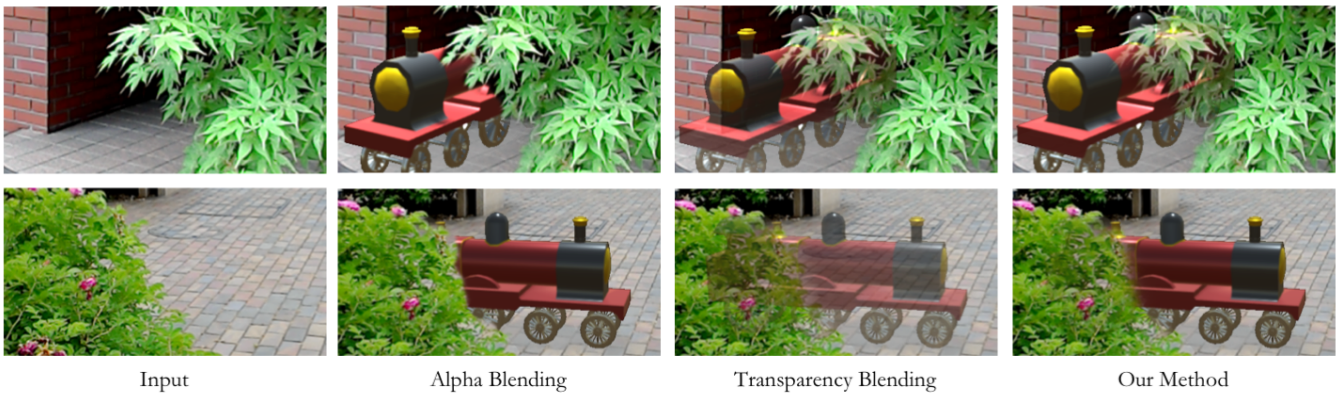

Figure 2: Results of alpha blending, transparency blending [8] and our method. In [8], the CG object becomes too transparent when the background and foreground real scene has very similar intensity levels. Our method achieves a better result by reasoning on the semantic information and setting different visibility levels for different scenes.

such as color, motion, and luminance change. However, these methods are constrained on a fixed camera and extension to moving camera applications is difficult.

Other methods use depth information [33][12][13][14] to effectively reason on the foreground-background relationship of the virtual and real objects. By adding additional hardware such as multiple cameras for stereo vision, time-of-flight cameras, and laser range sensors, depth estimation is a straightforward foreground detection method that can be done in real-time. However, existing hardwares have inherent problems such as limited range, dependency on lighting conditions and sparsity of data points. For example, sensors such as Microsoft Kinect and other devices that project light patterns have very limited range. To solve the range problem using these sensors, a tradeoff between real time sensing and accuracy is done by mapping a prior mesh model of the real scene, which can later be used in occlusion handling [2][20]. The sparsified depth map loses the information along object boundaries especially for complex objects such as trees. Lidar sensors, while having long range capabilities, still give sparse data and require interpolation to be useful in accurate foreground extraction.

Combining the depth and color images can bridge the shortcomings of the two modes. In [10] [7] RGBD images are used for accurate foreground estimation. In these methods, the RGB images are used to improve the appearance and accuracy of the sparse depth map. However, the method only solves the parts of the image where depth is not available, hence it still requires an already accurate foreground segmentation. On the other hand, in [34], sparse 3D model data from a GIS were used to infer a dense depth map of the real scene. However, these prior 3D models are not easily available in most cases.

In this work, we focus on improving occlusion handling when an inaccurate depth map of the current scene is given. The depth map can be created in real-time through different means such as mesh models [20][2], motion stereo [24] or sparse depth map through Lidars.

Several work have been proposed that also address the inaccuracy of the depth map. Alpha blending can improve the appearance of the occluded regions by gradually decreasing the visibility of the CG object along the boundary of the foreground object. However, this often results in non-realistic scene especially when the foreground region have complex edges. In [8], a transparency blending method which uses visibility predictor based on human vision system, was used to predict the transparency level of the CG object. The method has an advantage over alpha blending methods (Figure 2) because it does not require accurate estimation of complex boundaries. The method predicts the visibility of the CG object based on the intensity of the real scene and the foreground probability map inside a blending window. However, the method fails when the intensity of the foreground and background objects within the window are very similar, in which case the virtual object becomes too transparent.

In this paper, we propose to use semantic segmentation and a given depth map for handling occlusions. We assign different attributes (i.e. amount of visibility, or transparency) depending on the class of an object. The reasoning is straightforward: for outdoor augmented reality, the sky and ground are background, and therefore should be hidden behind the CG object. The rest could either be background or foreground. For objects that can be classified as both, we assign an experimentally determined visibility levels depending on the complexity of appearance of the object. For example, vegetation are considered complex and plane textures are not.

We use the given depth map to create a foreground probability map. By combining this and the semantic segmentation, we overlay the CG object onto the real scene by adapting a visibility-based rendering method from [9]. Instead of using a fixed visibility level for all objects as in [9], we use our proposed semantic classes to choose the amount of visibility for different type of objects. This allows us to control the appearance of the rendered object based on the type of the scene.

To summarize, this work has two main contributions (see Figure 1). First, we present a category scheme that uses semantics for assigning visibility values. We achieve this by first classifying the scene into specific categories using a real-time convolutional neural

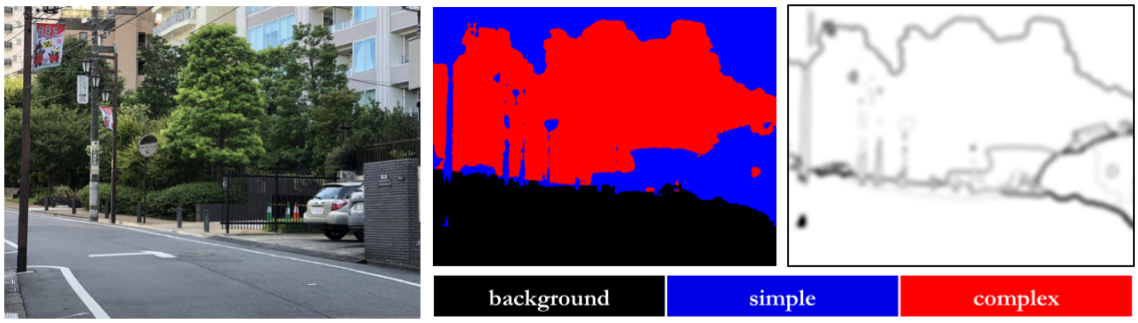

Figure 3: Our proposed semantic scheme and the uncertainty of class prediction.

network-based semantic segmentation method [31][32]. We then use our proposed scheme to group the segments into more usable categories for in visibility blending. Second, we present a visibility blending scheme that uses the foreground probability map and the semantic segmentation to create a visually pleasing augmented reality scene.

This paper is organized as follows. In Section 2, we propose a category scheme for foreground prediction using semantic segmentation. In Section 3, we introduce our visibility-based blending method which uses semantic classsification and the foreground probability map. In Section 4, we show our implementation, results, and comparison with existing methods. Finally, we conclude this work in Section 5.

2 SEMANTIC SCHEME FOR OCCLUSION HANDLING

Given an image of the real scene, we need to categorize each objects in it as either foreground or background. The overlayed CG object must be invisible if the real scene is foreground and highly visible otherwise. Instead of directly using these two labels, we classify the object into three main categories: Background, Complex and Simple. Objects that belong to the Background category are those that are always in the background such as the ground or the sky, therefore the CG object is visible when overlayed in these regions. On the other hand, objects that are classified as Complex or Simple can either be foreground or background, depending on the actual depth order of the CG object and the real scene. The visibility of the CG object is controlled differently for Complex and Simple objects using the method we will describe in the following sections.

In order to implement the above scheme, we first segment the scene into more specific categories. Based on this specific classification, we group the resulting classes into our proposed main categories (see Figure 3). We use these three categories but additional classes can be added depending on the type of the scene where the mixed reality system is deployed.

This choice of implementation (two stage) is done to avoid misclassification which is possible when the class size is very small. For example, roads and grass are visually different but they belong to the same Background category. Moreover, grass, which is often in the background, is visually closer to a tree, which can either be in the foreground or background region. Therefore, we opt to use the more refined classes first instead of immediately combining them into one semantic class.

We use the result of the semantic segmentation in two ways. First, the labeled segments are used for setting a visibility value of the CG object. Complex objects are usually highly textured objects and have complicated boundaries. Assuming the real scene is in the foreground, some sections of it (trees, vegetation etc.) especially along the object boundaries are see-through which means that the CG object must be slightly visible to create the effect of the scene being see-through. On the other hand, Simple objects are usually planar textures and the boundaries can be accurately estimated. In this case, if the real scene is in the foreground, the CG object behind it must be completely invisible.

We also use the uncertainty of prediction in setting the visibility of the CG object along the boundary of occlusion. In the classification stage, thesholding is applied to the the predicted values to fit the specified classes. We use the prediction values before this thresholding step in generating the uncertainty of prediction. This uncertainty allows us to gradually shift between two different categories without creating visual artifacts which can happen if the object boundaries are very sharp. In short, the uncertainty values allows for a smoothing effect for the visibility transition along the occlusion boundary for different categories.

Second, based on the class of the object, we either increase or decrease the width of transition between the visibility values from foreground to background. For example, the width of transition for Complex objects is higher than that of Simple objects because Complex objects have more complicated object boundaries. This allows us to handle the edges that cannot be accurately predicted by the segmentation method or the depth map. We further detail this approach in the next section.

3 VISIBILITY-BASED RENDERING

We extend a visibility-based blending method [9] to further utilize the semantic classification. This blending technique allows us to locally optimize the visibility of each region of the virtual object that can achieve arbitrarily targeted level. In [9], a visibility predictor based on human vision system is used in blending the virtual object. We extend this technique and use the semantic class and uncertainty of prediction in order to determine the desired visibility value.

We first define the overall intensity of the combined CG and real objects as:

$$I_{scene} = \alpha I_{cg} + (1 - \alpha) I_{real} \tag{1}$$

where α is the blending parameter and I_{cg} and I_{real} are the intensities of the CG object and the real scene. The blending parameter handles the transition between two different visibility levels. From the above equation, if α is zero, the CG is completely invisible. This is true if the CG object is in the background. If $\alpha >> 0$, the scene has a blended intensity that is based on the desired visibility of the CG object.

In our proposed method, the blending parameter is dependent on several properties: the desired visibility of the CG object, the texture of the overlapping scene, and the semantic class of the foreground real object. The texture of the real scene affects the perceived visibility of the CG object. For example, a highly textured surface is capable of conceiling the features of the blended CG object, thus makes it more invisible. On the other hand, if the real scene has planar texture, the CG is much more visibile. To address this issue, the intensity of the CG object is increased or decreased through the blending parameter α. We deteremine the texture of the real scene through our semantic segmentation scheme. Since we are only interested in the occluding regions, we only set a varying visibility value for the Complex and Simple scenes. Complex class characterizes the highly textured objects and Simple class characterizes the planar textured ones.

We set two levels of visibility for each of the semantic categories, V_f if the CG object is in the foreground and V_b if it is in the background. For the background class, these two values are the same. We calculate these values based on the uncertainty g and semantic class:

$$V_f = \frac{1}{2} V_{f1} + \frac{1}{2}\{(1 - g) V_{f1} + g V_{f2}\} \tag{2}$$
$$V_b = \frac{1}{2} V_{b1} + \frac{1}{2}\{(1 - g) V_{b1} + g V_{b2}\}$$

where V_{f1}, V_{f2}, V_{b1}, and V_{b2} are arbitrary values based on the desired appearance of the augmented scene depending on the class of the segment and the texture of the real scene.

V_{f1} and V_{b1} are the desired maximum visibility and V_{f2} and V_{b2} are the fallback minimum visibility. For the Background class, $V_f 1$ and $V_b 1$ are set to a high value such that the foreground probability map will be ignored. This is due to the fact that background object should not be visible.

For the Simple class, V_{f1} is set to a very low value (almost zero), where as V_{b1} is set to high value. In contrast, the Complex class has V_{f1} also set to a high value, which should mean that when the Complex object is in the foreground the CG object will still be visible. This is not the case. In our observation, the Complex class tend to always appear in the foreground due to its texture complexity. Hence, when we solve I_{scene} within the square window containing a Complex object, the CG object appears more transparent. We avoid this case by setting a high value for V_{f1}.

Equation 2 allows gradual shifting from different categories and visibility levels through the uncertainty value. This scheme is particularly effective along object boundaries. For example, if the uncertainty is very low (i.e. $g = 0.01$) for a background object in the Simple category (i.e. Tree trunk), the visibility V_f and V_b is almost equavalent of the maximum visibility. In this case, if the foreground probibility map is high (i.e. $P_f = 0.95$), the total visibility of the CG object V_{cg} approaches the maximum visibility for foreground V_f. On the other hand, if the uncertainty is high, (i.e. g = 0.85) which usually happens along boundaries, then V_f and V_b become weighted averages of the maximum visibility V_{b1} and the fallback minimium visibility V_{b2}.

Using the visiblity values V_f and V_b, we then define the blending parameter as:

$$\alpha = V_f - \frac{V_f - V_b}{1 + e^{-k(\omega - \omega_0)}} \tag{3}$$

which is a logistic function with maximum value of V_f and minimum value of V_b. The parameter k handles the steepness of the sigmoid curve and ω_0 gives the midpoint. The parameters k and ω_0 are solved based on the semantic class.

The value ω is dependent on the foreground probability map, P_f. Given the depth map of the real scene and the CG object, we define P_f (probability that the real scene is a foreground) as:

$$P_f = \frac{1}{1 + e^{-(d_{cg} - d_{real})}} \tag{4}$$

where d_{cg} is the depth of the CG object and d_{real} is the depth of the real scene. Equation 4 is a straightforward computation of the foreground probability map. The value is high if the depth of the real scene is smaller than that of the virtual object, which means that the real scene is closer to the camera. As the depth difference becomes smaller, however, the probability only decreases gradually so as not to suffer from inconsistency in depth estimation.

To solve for omega, the map is accumulated and normalized within a square window centered at the solved pixel with a total size of w:

$$\omega = \frac{1}{w} \sum_{i}^{w} P_{f_i} \tag{5}$$

The transition between two levels of visibility needs to be smooth in order to avoid artifacts. While the probability map can handle this, we further utilize the semantic information to adapt the transition depending on the predicted class. For Complex objects, the transition width should be inceased such that the switch between two visibility levels is more gradual that extends further into the foreground region (see Figure 4) compared to that of Simple objects. This technique allows us to handle regions of the foreground where the probability map is inaccurate.

The transition width can be adjusted by setting the steepness of the sigmoid curve and the value at the midpoint ω_0 of the blending parameter α. For Simple class, the curve should be steep and

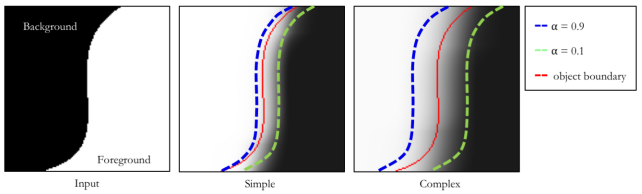

Figure 4: Blending boundary dilation and skew for Complex and Simple objects.

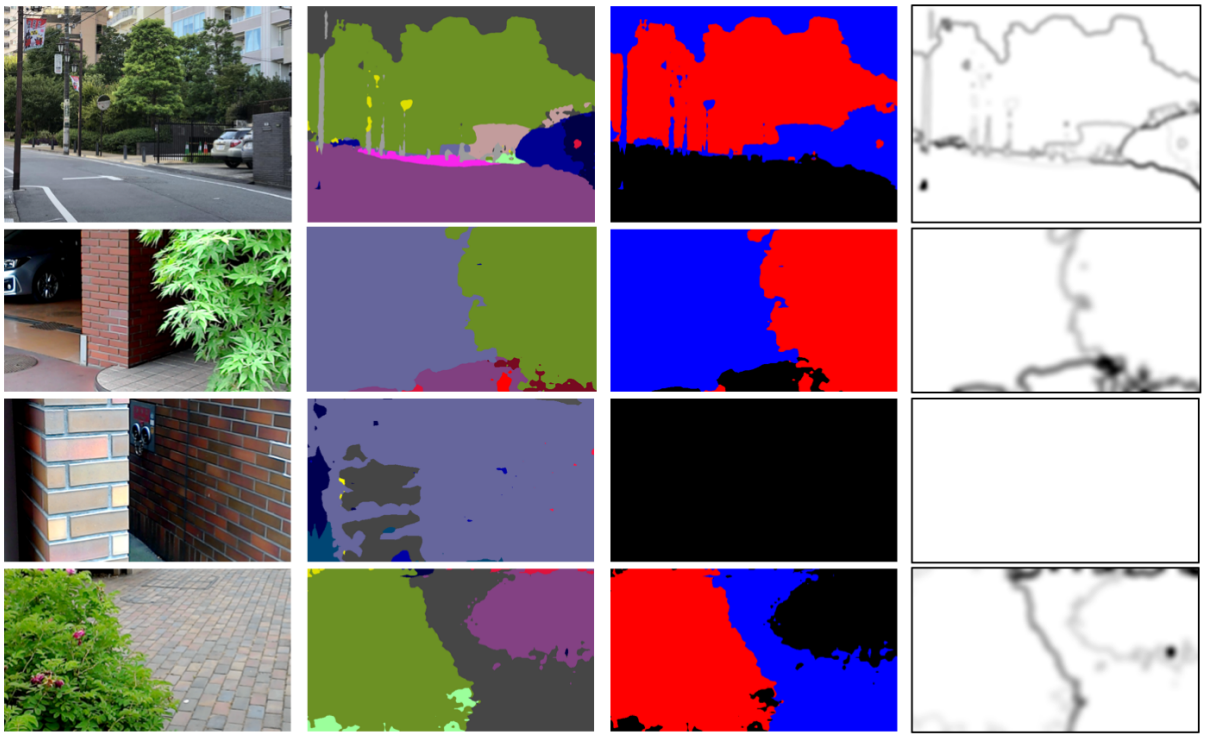

Figure 5: Re-classification of the Cityscape labels into our proposed categories and uncertainty of prediction.

road	sidewalk	building	wall	fence
pole	traffic light	traffic sign	vegetation	terrain
sky	person	rider	car	truck
bus	train	motorcycle	bicycle	

Figure 6: Labels used for specific classes.

midpoint should be skewed towards the center of the occluding object. For Complex, the curve should be smoother and the midpoint should lie along the natural boundary of the occlusion.

To set the steepness and the midpoint that is dependent on the classes, we define two control points in the curve $\alpha_d = \alpha(\omega \approx 0.5)$ and $\alpha_m = \alpha(\omega \approx 1.0)$. We set α_d to high when the class is Simple and low when the class is Complex. On the other hand, α_m is set to low ($\alpha_m < \alpha_d$) when the class is Simple and high when the class is Complex.

The steepness of the curve k is then solved by:

$$k = \frac{log(\frac{\alpha_m}{1-\alpha_m}) - log(\frac{\alpha_d}{1-\alpha_d})}{\omega_d - \omega_m} \tag{6}$$

The value ω_0 is then given by:

$$\omega_0 = \frac{log(\frac{\alpha_m}{1-\alpha_m})}{k + \omega_m} \tag{7}$$

4 IMPLEMENTATION AND RESULTS

4.1 Semantic Segmentation

Semantic segmentation methods have been ubiquitous in both indoor and outdoor applications. Specifically, convolutional neural network-based methods achieves high accuracy and real time results [31] [32]. To implement our own semantic scheme, we utilize a publicly available implementation [1] using the method described in [31] for outdoor applications trained on the Cityscapes dataset [5].

The specific classes are: Road, Sidewalk, Building, Wall, Fence, Pole, Traffic Light, Traffic Sign, Vegetation, Terrain, Sky, Person, Rider, Car, Truck, Bus, Train, Motorcycle, and Bicycle. We then regroup these classes to fit our semantic scheme as: Background (Road, Sidewalk, Terrain, and Sky), Complex (Vegetation, Person, Rider, Motorcycle, and Bicycle) and Simple (Building, Wall, Fence, Pole, Traffic Light, Traffic Sign, Car, Truck, Bus, and Train). Figure 5 shows the recategorized classes using this scheme.

The semantic segmentation method is implemented in Python using Tensorflow and run on an NVIDIA GTX 1060 GPU with 6GB of video RAM. The input frames are fed through an input stream and processed per frame. We achieved a 10ms (10fps) processing time for an image size of 634x360 which is sufficient for real time applications.

Category	V_f [8]	V_b [8]	V_{f1}	V_{f2}	V_{b1}	V_{b2}
Background	1.0	1.0	1.0	1.0	1.0	1.0
Simple	0.0005	0.5	0.0005	0.001	0.5	0.4
Complex	0.15	0.4	0.15	0.1	0.4	0.25

Table 1: Visibility parameters setting for [8] and our method.

4.2 Comparison with existing methods

We compare the results of our method with simple alpha blending and transparency blending [8] methods. For all three methods, we use the same depth map to solve the foreground probability map. For the alpha blending method we solve the color of the pixel as:

$$RGB = Real_{RGB} \times P_f + Cg_{RGB} \times (1 - P_f) \tag{8}$$

For [8], we set the V_f and V_b as fixed based on the region class (see Table 1), and solve the visiblity as in Equation 2.

We show the comparison of the output from the three methods in Figure 8. The first column correspond to the frame seen by the HMD. The second, third and fourth column are the output of the alpha blending, transparency blending and our method. In all cases, the alpha blending method achieves the highest visibility value. However, it is apparent along the more complex contours of vegetation that the alpha blending fails. The method results in an insufficient segmentation of the foreground region.

In contrast, the transparency blending achieves more visually pleasing blending along the complex contours. However, the visibility of the virtual object suffers when the background and foreground are of the same brightness or intensity. This results in the virtual object being almost invisible.

Our method achieves the best tradeoff between visibility and accurate segmentation. Along the regions of the complex contours of the foreground, our method outperforms the simple alpha blending. When the background is flat, our method outperforms the transparency blending.

Our rendering pipeline is implemented using shader language (HLSL) and can run on a PC (Intel Core i7-6700HQ at 2.60GHz and NVIDIA GTX 1060 GPU) at more than 60fps.

4.3 Implementation for Optical See-Through HMD

Our method can be used in both video see-through and optical see-through devices. The difference between the two devices is that for optical see-through devices, the real scene cannot be modified. Therefore, we need to slightly change our visibility blending equation:

$$I_{scene} = \alpha I_{cg} \tag{9}$$

The above equation is modified because the visibility of the real-scene is always 1.0. An input camera is also needed in order to properly sense the texture of the environment since the blending parameter α is still dependent on the visibility values based on the texture of the real scene. To handle the parallax between the eye view and the camera view, the position of the camera needs to be calibrated to the relative position of the see-through display and the eyes. Since there is only one camera, we re-position the optical center of the camera image in between the eyes. Then, we reproject the image by changing the focal length of the projection matrix to match that of the see-through display. We show the result of our implemenation in Figure 7 and our supplementary video.

Figure 7: Results of our method on an optical see-through device (Microsoft Hololens)

We tested our implementation on a Microsoft Hololens [2] device and achieved a frame rate of 60fps for rendering. However, the overall processing is bottlenecked by the 15fps video stream input from the built-in camera.

4.4 User Evaluation

Using the same settings for the three methods as in the previous section, we conducted an experiment with users (6 male and female, ages 23-48). Five scenes of 10-second video each were randomly shown to the users. The scene consist of a combination of Background, Complex and Simple objects from a moving camera and a static CG object.

We performed a pairwise comparison (total of 6 combinations) among the three methods. We showed one sequence first and then another and asked the users to compare the two sequence based on three categories: 1)Visibility of virtual object (Is it easy to see the virtual object?), 2)Realistic occlusion of the virtual object (Does the virtual object appear to be realistically occluded?) and 3) Realistic appearance of the rendered scene (Does the scene look realistic?). Each of the sequence is graded from -3 to +3 (+3 if the second video has maximum preferrence, -3 if the first video has maximum preference). We also randomly show each video pairs in reverse order, resulting in 30 pairs of evaluation dataset.

Based on the evaluation, we plot the total preference scores for each scene and questions in Figures 9. In all the tests, our method achieved highest preferential scores compared to the other two methods.

5 CONCLUSION AND FUTURE WORK

In this work, we demonstrated how to use visibility-based blending method and semantic segmentation in handling occlusion problem in mixed reality. We incorporated a foreground probability map solved from a given inaccurate depth map together with semantic classsification using convolutional neural network. Our results

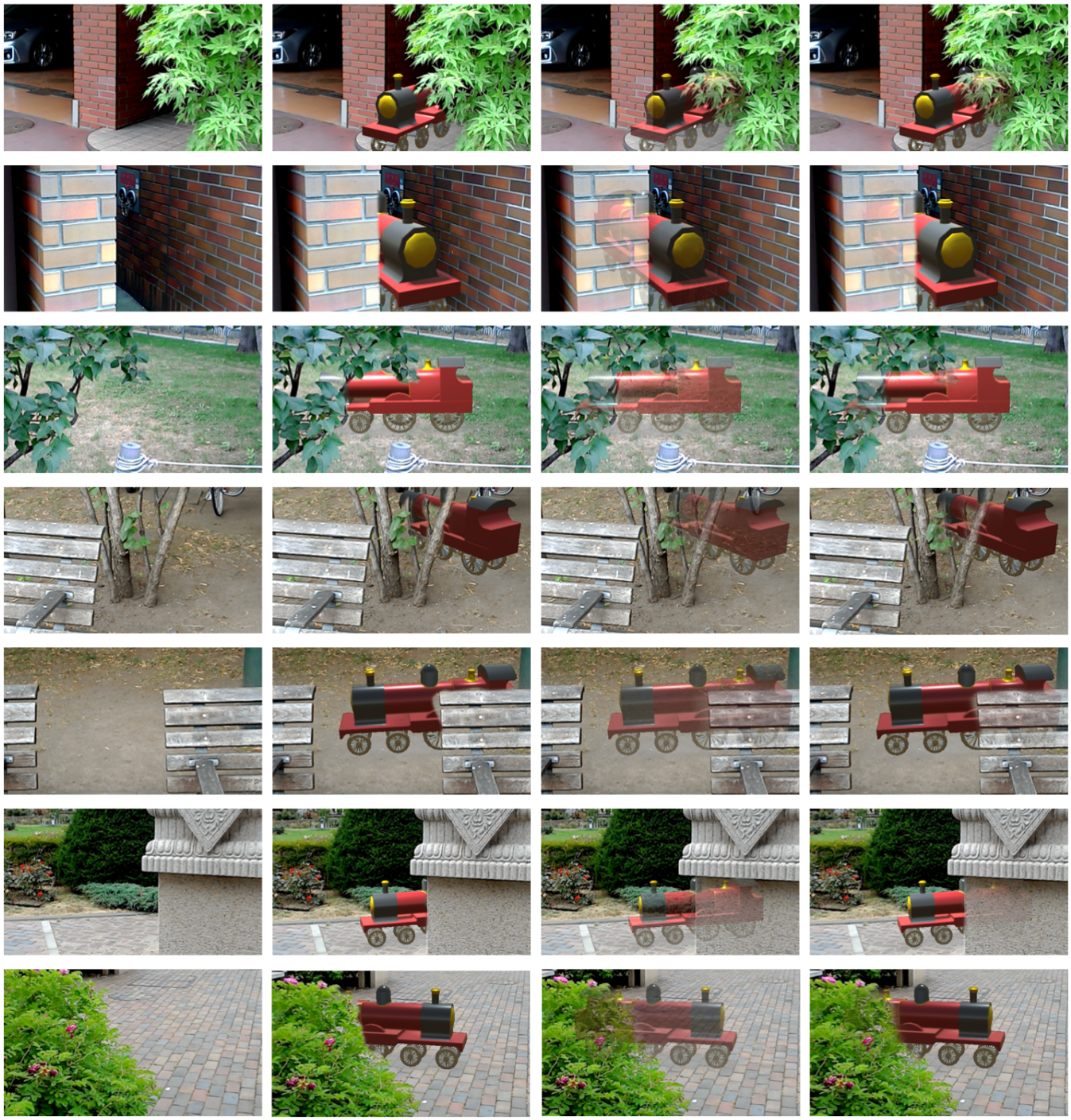

Figure 8: Comparison of rendering results from alpha blending, transparency blending, and our method.

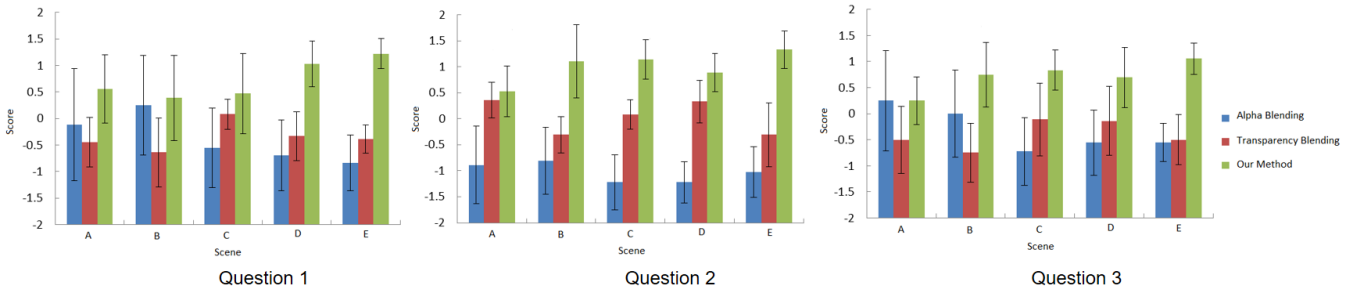

Figure 9: Comparison of preference scores for Question 1 (visibility of virtual object), Question 2 (realistic occlusion of the virtual object) and Question 3 (realistic appearance of the rendered scene).

shows that compared to existing alpha blending and transparency blending based techniques, our method achieves better visibility in flat background areas and better occlusion handling along complex foreground objects. Our implementation is achieved in real time using modern GPUs.

However, there are limitations in our implementation that can be improved in the future. First, the semantic segmentation is performed on a per frame basis, which is time consuming and naive. Since a continuous stream of input images are used in augmented reality, the semantic segmentation can be modified to utilize temporal information. Second, our semantic classification is narrow such that a bunch of object appearances and textures exists that can either be considered Simple or Complex. For example, some vegetation are dense enough that they can be considered as SImple objects. However, in our current implementation, all vegetation are considered Complex. By using additional classes and more ground truth data for training the semantic segmentation network, we can further control the blending parameter to handle more specific situations and extended to more complex scenes. Finally, to make the method complete, a real time depth estimation method can be added.

REFERENCES

[1] [n. d.]. An implementation of ICNet (Real-time image segmentation) in tensorflow. https://github.com/hellochick/ICNet-tensorflow. Accessed: 2018-08-15.

[2] [n. d.]. Microsoft Hololens. https://www.microsoft.com/en-us/hololens. Accessed: 2018-08-15.

[3] S. Uchiyama andK. Takemoto, K. Sato, H. Yamamoto, and H. Tamura. 2002. MR Platform: A basic body on which mixed reality applications are built. In *ISMAR*.

[4] Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla. 2015. SegNet: A Deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. In *CVPR*.

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[6] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov. 2006. Bilayer segmentation of live video. In *Computer Vision and Pattern Recognition*. 53–60.

[7] C. Du, Y. Chen, M. Ye, , and L. Ren. 2016. Edge snapping-based depth enhancement for dynamic occlusion handling in augmented reality. In *ISMAR*.

[8] Taiki Fukiage, Takeshi Oishi, and Katsushi Ikeuchi. 2012. Reduction of contradictory partial occlusion in mixed reality by using characteristics of transparency perception. In *ISMAR*. 129–139.

[9] Taiki Fukiage, Takeshi Oishi, and Katsushi Ikeuchi. 2014. Visibility-based blending for real-time applications. In *ISMAR*. 63–72.

[10] A. Hebborn, N. Hohner, and S. Muller. 2017. Occlusion Matting: Realistic Occlusion Handling for Augmented Reality Applications. In *ISMAR*.

[11] T. Kakuta, L.B.Vinh, R. Kawakami, T.Oishi, and K.Ikeuchi. 2008. Detection of moving objects and cast shadows using spherical vision camera for outdoor mixed reality. In *VRST*. 219–222.

[12] Masayuki Kanbara, Takashi Okuma, Haruo Takemura, and Naokazu Yokoya. 1999. Real-time composition of stereo images for video see-through augmented reality. In *International Conference on Multimedia Computing and Systems*. 213–219.

[13] Hansung Kim, Seung jun Yang, and Kwanghoon Sohn. 2005. 3D reconstruction of stereo images for interaction between real and virtual objects. In *Signal Processing and Image Communication*. 61–75.

[14] Tae Hoon Kim, Hoyub Jung, Kyoung Mu Lee, and Sang Uk Lee. 2008. Segmentation based foreground object disparity estimation using Zcam and multiple-view stereo. In *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. 1251–1254.

[15] LabelMe. [n. d.]. LabelMe. The Open Annotation Tool. http://labelme.csail.mit.edu.

[16] L.B.Vinh, Tetsuya Kakuta, Rei Kawakami, Takeshi Oishi, and Katsushi Ikeuchi. 2010. Foreground and shadow occlusion handling for outdoor augmented reality. In *ISMAR*. 13–16.

[17] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. 2006. Lazy Snapping. In *ACM Transaction on Graphics*, Vol. 23. 22–32.

[18] B. Liu and X. He. 2015. Multiclass Semantic video segmentation with object-level active inference. In *CVPR*.

[19] J. Long, E. Shelhamer, and T. Darrell. 2015. Fully convolutional networks for semantic segmentation. In *CVPR*.

[20] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. 2011. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *ISMAR*.

[21] George Papagiannakis and Sebastien Schertenleib. 2005. Mixing virutal and real scenes in the site of ancient Poppeii. In *Journal of Computer Animation and Virtual Worlds*. 11–24.

[22] Point Grey Research. [n. d.]. Lady Bug Technical Reference. https://www.ptgrey.com/ladybug5-360-degree-usb3-spherical-camera-systems.

[23] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. 2004. Grab-cut - Interactive foreground extraction using iterated graph cuts. In *ACM, SIGGRAPH '04*. 309–314.

[24] M. Roxas and T. Oishi. 2017. Real-Time Simultaneous 3D Reconstruction and Optical Flow Estimation. In *WACV*.

[25] SegNet. [n. d.]. SegNet: A Deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. http://mi.eng.cam.ac.uk/projects/segnet.

[26] A. Sharma, O. Tuzel, and D. Jacobs. 2015. Deep Hierarchical parsing for semantic segmentation. In *CVPR*.

[27] J. Sun, W. Zhang, X. Tang, and H.Y.Shum. 2006. Background Cut. In *ECCV*. 628–641.

[28] S.W.Hasinoff, S.B. Kang, and R.Szeliski. 2006. Boundary Matting for View Synthesis. In *Computer Vision and Image Understanding*. 53–60.

[29] D. Varas, M. Alfaro, and F. Marques. 2015. Mutiresolution hierarchy co-clustering for semantic segmentation in sequences with small variations. In *ICCV*.

[30] C. Zach, T. Pock, and H. Bishof. 2007. A duality based approach for real-time TV-L1 optical flow. In *Lecture Notes in Computer Science: Pattern Recognition*.

[31] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. 2017. ICNet for Real-Time Semantic Segmentation on High-Resolution Images. *arXiv preprint arXiv:1704.08545* (2017).

[32] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. 2017. Scene Parsing through ADE20K Dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[33] Jiejie Zhu, Liang Wang, Ruigang Yang, and James Davis. 2008. Fusion of time-of-flight depth and stereo for high accuracy depth maps. In *CVPR*. 23–28.

[34] S. Zollmann and G. Reitmayr. 2012. Dense Depth Maps from Sparse Models and Image Coherence for Augmented Reality. In *VRST '12*.