Virtual Occlusions Through Implicit Depth

Jamie Watson[1,3] Mohamed Sayed[1,3] Zawar Qureshi[1] Gabriel J. Brostow[1,3]
Sara Vicente[1] Oisin Mac Aodha[2] Michael Firman[1]
[1]Niantic [2]University of Edinburgh [3]UCL
https://nianticlabs.github.io/implicit-depth

Abstract

For augmented reality (AR), it is important that virtual assets appear to 'sit among' real world objects. The virtual element should variously occlude and be occluded by real matter, based on a plausible depth ordering. This occlusion should be consistent over time as the viewer's camera moves. Unfortunately, small mistakes in the estimated scene depth can ruin the downstream occlusion mask, and thereby the AR illusion. Especially in real-time settings, depths inferred near boundaries or across time can be inconsistent. In this paper, we challenge the need for depth-regression as an intermediate step.

We instead propose an implicit model for depth and use that to predict the occlusion mask directly. The inputs to our network are one or more color images, plus the known depths of any virtual geometry. We show how our occlusion predictions are more accurate and more temporally stable than predictions derived from traditional depth-estimation models. We obtain state-of-the-art occlusion results on the challenging ScanNetv2 dataset and superior qualitative results on real scenes.

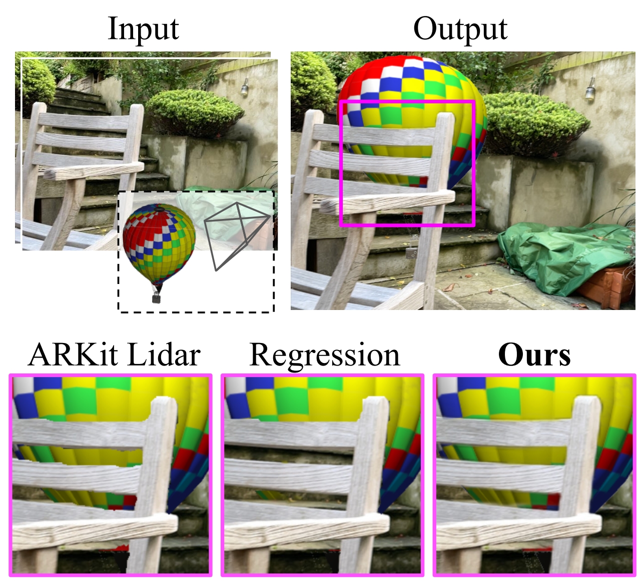


Figure 1. We address the problem of automatically estimating occlusion masks to realistically place virtual objects in real scenes. Our approach, where we directly predict masks, leads to more accurate compositing compared with Lidar-based sensors or traditional state-of-the-art depth regression methods.

1. Introduction

Augmented reality and digital image editing usually entail compositing virtual rendered objects to look as if they are present in a real-world scene. A key and elusive part of making this effect realistic is *occlusion*. Looking from a camera's perspective, a virtual object should appear partially hidden when part of it passes behind a real world object. In practice this comes down to estimating, for each pixel, if the final rendering pipeline should display the real world object there *vs*. showing the virtual object [53, 24, 36].

Typically, this per-pixel decision is approached by first estimating the depth of each pixel in the real world image [19, 90, 36]. Obtaining the depth to each pixel on the *virtual* object is trivial, and can be computed via traditional graphics pipelines [33]. The final mask can be estimated by comparing the two depth maps: where the real world depth value is smaller, the virtual content is occluded, i.e. masked.

We propose an alternative, novel approach. Given images of the real world scene and the depth map of the virtual assets, our network *directly estimates* the mask for compositing. The key advantage is that the network no longer has to estimate the real-valued depth for every pixel, and instead focuses on the binary decision: is the virtual pixel in front or behind the real scene here? Further, at inference time we can use the soft output of a sigmoid layer to softly blend between the real and virtual, which can give visually pleasing compositions [43], compared with those created by hard thresholding of depth maps (see Figure 1). Finally, temporal consistency can be improved using ideas from temporal semantic segmentation that were difficult to apply when regressing depth as an intermediate step.

We have three contributions:

1. We frame the problem of compositing a virtual object at a known depth into a real scene as a binary mask estimation problem. This is in contrast to previous approaches that estimate depth to solve this problem.
2. We introduce metrics for occlusion evaluation, and find our method results in more accurate and visually pleasing composites compared with alternatives.
3. We show that competing approaches flicker, leading to jarring occlusions. By framing the problem as segmentation, we can use temporal smoothing methods, which result in smoother predictions compared to baselines.

Our 'implicit depth' model ultimately results in state-of-the-art occlusions on the challenging ScanNetv2 dataset [17]. Further, if depths are needed too, we can compute dense depth by gathering multiple binary masks. Surprisingly, this results in state-of-the-art depth estimation.

2. Related work

Early approaches to occluding virtual assets in real scenes relied on user annotations of object boundaries [53, 65], precluding general real-time use. The typical approach for automated occlusion is to pixel-wise compare a depth map of the real scene with the virtual depth map [7]. The real image depth map can be estimated, e.g. using structured light [24], or from images [19, 90]. When sparse depths are available, they can be densified [36] or used to rescale relative depths to metric depth [1]. Direct estimation of an occlusion mask has been previously performed for segmentation for AR sky replacement [101] or hands grabbing virtual objects [85]. In contrast, our method enables general object compositing. A detailed review of occlusion handling in AR is outlined in [61, 8].

Depth estimation. Depth estimation is a key component of many AR occlusion systems. Depth can be estimated directly if binocular cameras are available at test time [34, 47, 11, 15]. This approach requires specialized hardware, as does depth estimation from structured light, Lidar, or time-of-flight devices [28]. Further, depth from such devices may not be accurate enough for realistic occlusions without further processing [91, 45].

It is attractive to estimate depth directly from color images, for example from a *single* image [27, 22, 21, 29, 95]. When a sequence of images is available, Multi-View Stereo (MVS) estimates depth for a reference image using one or more source images [25, 79], which assumes that the scene being observed is static. Recent MVS approaches match image pixels [93, 39] or deep features [41, 20] to create a cost volume, which can then be processed using convolutional layers. Other works have improved upon this basic setup by refining the final output [97], through injection of additional metadata [78], by handling occluded pixels between views [59], or with a Gaussian process prior [37]. Alternative approaches have dropped the reliance on supervised data through the use of self-supervision [29, 27, 30, 94]. There have been attempts to improve the quality of depth estimation around depth discontinuities, e.g. using a gradient or normal loss [71, 55, 38] or a learned network to post-process predictions [70]. Higher quality depths can also be computed via an offline optimization on test sequences [50, 60, 10, 14, 51], but this precludes online use.

In contrast to these depth estimation approaches, we *directly* estimate an occlusion compositing mask. However, in Section 4 we show that our models can also be adapted to predict depths equivalent to state-of-the-art methods without requiring retraining.

Depth via classification. Our implicit depth approach is related to classification-based depth estimation. Xie et al. [95] is an early approach which learned depth via classification, in the context of depth from stereo. In [23], the output domain is divided into discrete bins, and the final output head classifies each pixel as in front or behind each depth bin. Alternatively, [57, 9, 96, 88] classify the probability that the depth lands in a bin itself. Other works have relaxed the requirement for fixed bin centres, allowing them to be adapted on a per-image [5] or per-pixel [6] basis. Bi3D [3] pose stereo depth estimation as a binary classification task, but where a scalar query depth is provided to the network. Also related to depth classification approaches are works which decompose images into two or more layers, e.g. foreground and background [18, 89, 54].

We also frame geometry estimation as classification, but our approach classifies if a per-pixel virtual depth is in front or behind the real world object at that depth. We can therefore estimate a full compositing mask with a single forward pass, without a dense output tensor. We compare to classification approaches and show that our method is superior in terms of accuracy and compute.

Image and video segmentation. Our work is related to object segmentation [31, 99], salient object segmentation [44], and alpha-matting [56, 80, 12]. Like these, we estimate a binary mask, but our mask is conditioned on an input rendered virtual depth map. Similar to video segmentation [26, 68], we encourage temporal consistency across frames to prevent flicker.

Occlusion boundaries and regions. There are works which focus on detecting pixels which become occluded and disoccluded between frames in a video [40, 92, 35, 83, 84]. We differ as we recover the occlusion mask of a virtual object in real input images.

Implicit volumes. Finally, our approach is related to works on implicit volumes e.g. [58, 66, 62, 75, 76, 77, 16, 73, 67], where a 3D shape is represented by a trained multi-layer perceptron (MLP). When evaluated at each location in 3D space, the MLP's binary output indicates if that location is inside or outside an object. However, we operate in image space, and predict if a pixel in the real world scene is in front or behind a virtual target's depth map. Zhu et al. [100] used an implicit function for RGBD *completion*, operating on ray-voxel pairs. Neural radiance fields [63, 4, 64, 98] (NeRFs) are an alternative implicit approach which can estimate depths and color images from novel viewpoints. This can be used for AR effects, but NeRFs are typically not suited to online applications in novel scenes.

3. Method

Our goal is to automatically composite virtual objects into images of real scenes, respecting any real occluding objects that are 'in the way'. At inference time, we assume we have an RGB image I_{real} as input, together with a temporally preceding sequence of RGB source images and corresponding camera intrinsics and poses. We denote the full sequence of I_{real} together with source images as $\mathcal{I}_{\text{real}}$. We also assume knowledge of a 3D virtual object that we wish to place in the scene, which can change over time. From the 3D virtual object and camera poses, we extract for each frame a color rendering of the virtual object I_{virtual} with an associated virtual depth map D_{virtual}.

3.1. Our approach

Given the rendering of the asset, the job of an occlusion step is to estimate which virtual pixels should be shown, and which should be hidden, to create the final image I_{final}. This can be described by a compositing equation [69, 82], using the two images and a per-pixel compositing mask C, so

$$I_{\text{final}} = C I_{\text{real}} + (1 - C) I_{\text{virtual}}. \tag{1}$$

This compositing equation is only applied to pixels covered by the virtual object, outside of which we only show I_{real}.

Traditional occlusion methods, e.g. [70, 90], use a depth map to estimate C. The depth map D_{real} is the output of a network ψ, which takes $\mathcal{I}_{\text{real}}$ as input, so $D_{\text{real}} = \psi(\mathcal{I}_{\text{real}})$. Here the compositing mask C was formed using the relation

$$C = [D_{\text{real}} < D_{\text{virtual}}], \tag{2}$$

where [] is the Iverson bracket.

Training this network to instead *directly* predict C is a potentially attractive alternative solution. However, this direct prediction is not feasible without D_{virtual}, as the network has no context at inference time of where the virtual object should be positioned in the world and therefore would be unable to produce a plausible mask.

In our approach, we instead use a deep network ϕ to directly estimate C, conditioned on *both* $\mathcal{I}_{\text{real}}$ and D_{virtual} as input, so

$$C = \phi(\mathcal{I}_{\text{real}}, D_{\text{virtual}}). \tag{3}$$

The final image is then formed using Eqn. 1 from above.

Advantages of our depth informed mask prediction. We hypothesize that it is easier for our network to directly predict a *binary* 'in front vs. behind' value at each pixel location, compared with existing methods that predict a *continuous* value to regress the absolute depth.

3.2. Predicting an occlusion map

A natural choice for our network ϕ would be an image-to-image network. This would take the concatenation of $\mathcal{I}_{\text{real}}$ and D_{virtual} as input, and then output C. However, at training time such an architecture would need to see both realistic images and realistic *virtual* depth maps corresponding to the scene. Generating realistic virtual depths for a scene is difficult, as we do not know what the final use case of the system might be, and thus placing virtual objects in a scene automatically at training time is a non-trivial task. We instead take a different approach, and propose an architecture with two parts: (i) a backbone network for image encoding, followed by (ii) a per-pixel MLP (see Figure 2). Our virtual depths are only provided to the per-pixel MLP, meaning our training-time virtual depths do not need to be realistic virtual depth maps.

Backbone network for image encoding. Our backbone network maps the RGB image I_{real} to a pixel-aligned feature encoding F with K channels per pixel. While we could use any backbone to extract features, for most of our experiments we use a multi-view stereo approach as in [78]. This requires temporally preceding source frames and known camera poses. See Section 3.4 for details.

Predicting the occlusion mask with an MLP. The final prediction of the compositing mask at pixel location $\mathbf{p}$ relies on three inputs:

1. The image features at $\mathbf{p}$, i.e. $F(\mathbf{p})$. Inspired by [49], we interpolate features from F at arbitrary sub-pixel locations. This enables us to make final predictions at arbitrary locations and resolutions.
2. The virtual object depth at $\mathbf{p}$, i.e. $D_{\text{virtual}}(\mathbf{p})$. Again, we can sample D_{virtual} at arbitrary sub-pixel locations.
3. The warped previous temporal prediction at $\mathbf{p}$, as described next. This enables the network to use temporal information for more stable predictions.

At location $\mathbf{p}$, we concatenate the above three inputs to make a $K{+}2$-dimensional feature. This is given to an MLP to produce the final compositing output for that location $C(\mathbf{p})$. The final layer of the MLP has a sigmoid activation, so $C(\mathbf{p})$ is continuous $\in [0, 1]$. This can be interpreted as the probability that $I_{\text{real}}(\mathbf{p})$ is occluding the virtual object at depth $D_{\text{virtual}}(\mathbf{p})$.

Temporal stability. Temporally stable occlusions are important for seamless and believable AR immersion as per-frame depth or semantic predictions can vary over time,

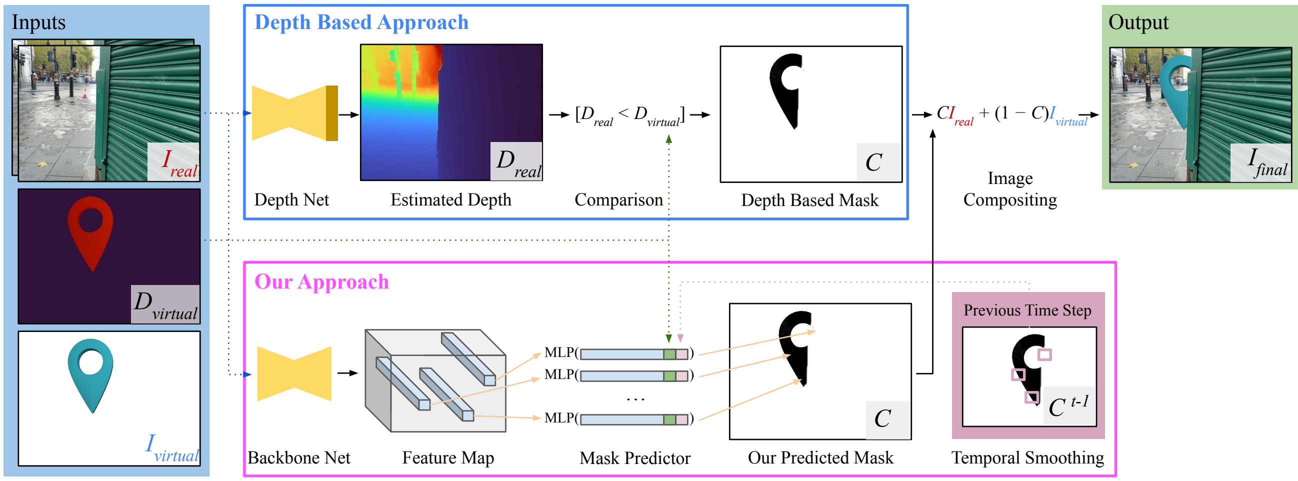


Figure 2. Left: Given RGB images of a real scene, and renderings of a virtual asset, our aim is to realistically composite the virtual asset into the scene. Top: Conventional approaches first estimate a depth map from the real image(s), before comparing each pixel with the virtual depth to generate a compositing mask C. Bottom: We instead directly estimate the mask given the real image(s) and virtual depth as input. Additionally, our method also employs a lightweight temporal smoothing input to generate more stable predictions.

leading to visual ‘flickering’. To combat this, we encourage the network to be temporally stable, inspired by [87, 68]. To achieve this, we feed to the MLP the previous prediction for the pixel at location $\mathbf{p}$, $\hat{C}(\mathbf{p})$, defined as $C^{t-1}(warp[\mathbf{p}])$. *warp*[] uses the known relative camera transform to backwards warp the pixels' locations at time t to time step $t-1$ using [42]. This warping requires known depth at time t, for which we use the rendered *virtual* depth. This is in contrast to [87] which does not use geometric information.

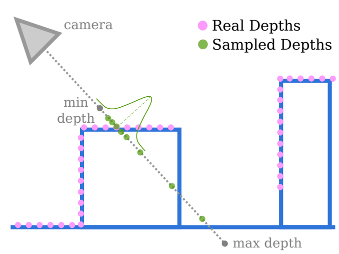
Relationship to prior work. While previous approaches (e.g. [3]) have trained a network which takes a single query depth as input, our approach is novel as our query depth can vary spatially per pixel, so each pixel can have a different query depth. Methods like [3] would require exhaustive evaluation of all depth values to be able to composite an object. Our approach can be seen as producing ‘implicit depths’, similar to prior work in 3D, e.g. [66].

3.3. Training our network

Our goal at training time is to update the weights of our network ϕ (i.e. both feature encoder and MLP) to accurately predict occlusions. We use training datasets, e.g. [17, 72], where we have access to sequences of training images $\mathcal{I}_{\text{real}}$ with pixel-aligned ground truth depth maps D_{real} and associated camera poses and intrinsics. However, these datasets do not come with augmented virtual depths D_{virtual}, so we need to synthesize these at training time.

Our training samples. We require training tuples with an image location $\mathbf{p}$, a virtual depth at that location $D_{\text{virtual}}(\mathbf{p})$, and a ground truth label $y_i \in \{0, 1\}$ stating if the virtual depth is in front (0) or behind (1) the real image depth map. Additionally, to encourage temporal stability, we require a previous prediction for this location, $\hat{C}(\mathbf{p})$.

To generate a training sample, we choose a single training sequence $\mathcal{I}_{\text{real}}$, with associated depth map D_{real}. We sample a 2D location $\mathbf{p}$ uniformly in image space, and subsequently sample a training-time feature $F(\mathbf{p})$. Given the image location $\mathbf{p}$, we have a choice to sample our synthesized virtual depth $D_{\text{virtual}}(\mathbf{p})$ anywhere along $\mathbf{p}$'s camera ray. Sampling a random depth means we might be far away from the difficult choices. The most difficult choices for depths are when D_{virtual} is near D_{real}, so we bias a fraction of our training-time samples to come from near the ground-truth depth surface D_{real}. Similar to [76, 77], with probability q we sample from a Gaussian with mean of the ground truth depth value at pixel $\mathbf{p}$, and variance 0.05. To ensure that we also make sensible predictions away from real surfaces, with probability $1 - q$ we sample a training depth uniformly between the minimum and maximum depth in D_{real}. The ground truth depth map determines the label y_i, which, in turn, is used to supervise the network with binary cross-entropy.



Training for temporal stability. As stated in Section 3.2, we encourage temporal stability by giving the warped previous prediction as an additional input to our MLP. During training, to avoid the need to run inference for multiple frames, we synthesize a previous prediction from the ground truth label y_i in a manner similar to [87]. We corrupt y_i to produce a pseudo previous prediction $\tilde{C}(\mathbf{p})$ by adding random noise, converting the binary labels into

Method	Occlusion evaluation			Depth evaluation					
	IoU All↑	IoU Surface↑	IoU Boundary↑	Abs Diff↓	Abs Rel↓	Sq Rel↓	RMSE ↓	$\delta < 1.05$ ↑
DPSNet [41]	46.17	21.68	23.50	.1552	.0795	.0299	.2307	49.36
MVDepthNet [93]	44.64	21.06	23.22	.1648	.0848	.0343	.2446	46.71
DELTAS [81]	48.48	23.37	25.51	.1497	.0786	.0276	.2210	48.64
GPMVS [37]	46.52	22.43	23.98	.1494	.0757	.0292	.2287	51.04
DeepVideoMVS, fusion [20]*	53.16	26.49	28.05	.1186	.0583	.0190	.1879	60.20
SimpleRecon (ResNet)	58.91	31.48	33.03	.0978	.0487	.0151	.1617	69.62
SimpleRecon [78]	<u>60.52</u>	32.44	34.52	<u>.0871</u>	**.0429**	<u>.0125</u>	<u>.1460</u>	**74.01**
SimpleRecon (ResNet) + **Ours**	60.14	<u>33.29</u>	<u>36.54</u>	.0988	.0498	.0149	.1595	68.52
SimpleRecon [78] + **Ours**	**62.61**	**35.48**	**38.01**	**.0862**	<u>.0436</u>	**.0123**	**.1426**	<u>73.74</u>

Table 1. **Occlusion and depth scores after converting our masks to depths compared with state-of-the-art prior works.** Evaluation is on the ScanNetv2 test set keyframes [20], and follows the evaluation protocol for *depth* from [20]. Our model is state-of-the-art on both occlusion and depth estimation. * indicates trained on additional data.

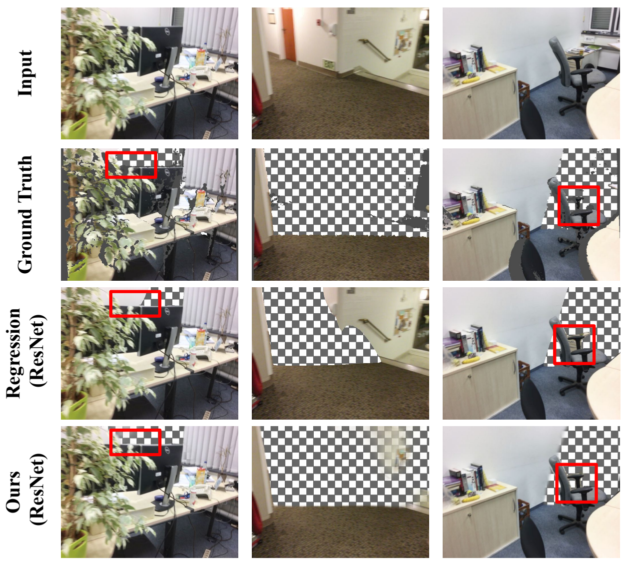


Figure 3. **Occlusion predictions on ScanNetv2.** Comparison of occlusion masks when a virtual plane sits at 3m from the camera. ScanNetv2's ground truth is noisy, especially near thin structures and distant points > 5m. Regression is crisp but makes mistakes affecting whole regions. Ours misclassifies part of a painting in the example in the second column, but is better overall.

Architecture / Method	IoU All↑	IoU Surface↑	IoU Boundary↑
SimpleRecon [78]	60.52	32.44	34.52
SimpleRecon [78] + **Ours**	**62.61**	**35.48**	**38.01**
SimpleRecon (ResNet)	58.91	31.48	33.03
SimpleRecon (ResNet) + **Ours**	**60.14**	**33.29**	**36.54**
ManyDepth [94]	55.68	29.18	30.47
ManyDepth [94] + **Ours**	**56.55**	**31.56**	**34.47**
MonoDepth2 [30]	46.06	18.62	23.37
MonoDepth2 [30] + **Ours**	**48.35**	**21.55**	**26.55**

Table 2. **Our method outperforms depth regression for the occlusion task,** regardless of the underlying architecture. All these architectures were trained and evaluated on ScanNetv2 sequences by us, using code published by the authors, and with automated virtual object insertion to evaluate binary occlusion masks.

floats $\in [0, 1]$, simulating the output of a sigmoid. To teach our model to be robust to incorrect previous predictions at inference time, we set $\tilde{C}(\mathbf{p}) = 1 - \tilde{C}(\mathbf{p})$ with probability p_1 during training. Additionally we use $\tilde{C}(\mathbf{p}) = -1$ to indicate the start of a sequence, and set this during training with probability p_2. For all experiments we use $p_1 = p_2 = 0.25$.

Regularization around depth discontinuities. Training a model as described in Section 3.3 can give rise to artefacts in predicted compositing masks near depth discontinuities. Due to the inherent ambiguity of occlusions in these regions, our model tends to predict values close to 0.5, which leads to less visually pleasing results in the final compositing. To combat this, during training we locate depth discontinuities in the ground truth depth map using a Sobel filter [46], and apply an L1 regularizer to penalize predictions near 0.5 in these regions. See the supplementary material for details.

3.4. Implementation details

We train all models and baselines using the Adam optimizer [48] with a batch size of 24 split across 2 GPUs. For speed of convergence, we initialize our backbone network with the weights of a depth regression network, and train for 40k steps, with an initial learning rate of 0.0001 dropping by a factor of 10 after 16k steps and 32k steps. Images are augmented with standard flip and color augmentations, as in [78]. We use $q = 0.25$ for our probability of sampling a virtual depth near the real depth surface. Similar to [30], we supervise the network at 4 output scales, using ground truth depth at a higher resolution than the feature maps, and at test time only use the highest resolution output.

Backbone. Our backbone network is based on [78], where a shallow feature extractor feeds a metadata infused plane sweep volume, followed by a U-Net [74]. See the supplementary material for full details.

MLP. Our compositing mask prediction uses an MLP with three fully-connected layers, with $K+2$ input channels and a single channel final output. Both hidden layers have 128 dimensions. We use ELU activations after the first two layers, with the final activation being a sigmoid to map our outputs to the range $[0, 1]$. Based on our backbone [78], our full-scale feature map has $K = 64$.

Timings. Inference with our backbone (from [78]) takes 64ms and our MLP takes 0.5ms on an A100 GPU.

4. Experiments

Datasets. We train and evaluate models using ScanNetv2 [17], with the standard train/val/test split. This allows direct comparison, of occlusions or depths, with most prior depth estimation methods that also train on ScanNetv2. For qualitative comparisons, we also train a model on the synthetic Hypersim [72] dataset. This synthetic data has better aligned edges in the training data, so yields a model with high edge fidelity. In visual occlusion comparisons, we compare that model against Lidar depth sensing [2] and a previous method which does not use a trained model [36].

Backbone variants. We present two variants of our model; 'SimpleRecon + **Ours**' which uses the architecture of [78] as a backbone, and a faster lower compute version using a ResNet-18 [32] encoder, a lightweight decoder, and a simple dot-product cost volume, referred to as 'SimpleRecon (ResNet) + **Ours**'. We also train a ResNet variant *without* a cost volume inspired by [30]: 'MonoDepth2 + **Ours**'.

4.1. Evaluating virtual asset occlusion

We directly evaluate performance on the task of virtual object insertion. We report scores using the standard segmentation metric, intersection-over-union (IoU), to measure occlusion quality. We compare our variants against state-of-the-art depth estimation on the standard ScanNetv2 test set.

Since rendered virtual assets would add noise to the evaluation process, we use infinite planes that lie ahead of the camera at each frame. These planes are placed at depths $d \in \{0.5\text{m}, 1.0\text{m}, \cdots, 5.0\text{m}\}$ along the look-at vector of the camera, where each plane's virtual depth map is D^d_{virtual}. We obtain a ground truth binary occlusion mask, Y^d_{GT}, for each plane using the ground truth depth map, obtained from depth sensors.

For our method, we compute the probability of occlusion for each depth plane, C^d, which we threshold with τ to produce Y^d_{pred}. We pick τ for each depth bin using a mini-val set of 100 scans. We compute IoU for the occluded asset fragments, IoU^d_-, and the visible parts of the asset, IoU^d_+. For depth estimation methods, we obtain the predicted occlusion mask directly by comparing D^d_{pred} and D^d_{virtual} to compute IoU^d_+ and IoU^d_-. We compute IoU All^d for each plane using the harmonic mean of IoU^d_+ and IoU^d_-. We average IoUs for each keyframe from [20] and then for each depth plane. As regions near depth boundaries tend to be difficult,

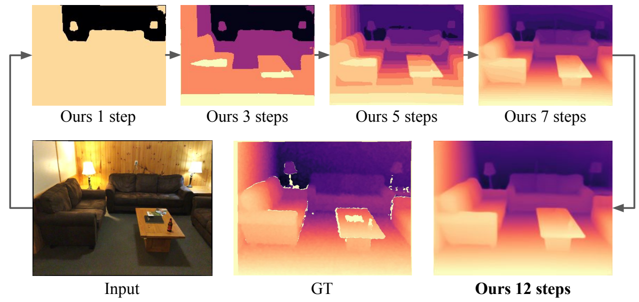


Figure 4. **Our binary search converts our predictions to a full depth map.** First the backbone is run only once to extract features, then the much faster binary prediction MLP is run once for each step of binary search, iteratively refining a depth map.

following [13] we evaluate *IoU Boundary*, and separately, regions near the geometry's surface (*IoU Surface*).

In Table 2, we combine our method with existing backbones by first training with regression losses [78], and then finetuning with our approach on ScanNetv2. We compare with several recent MVS methods, including SimpleRecon [78], ManyDepth [94], as well as a single frame depth method MonoDepth2 [30]. In all cases, our method improves occlusion scores, most notably in difficult cases near surfaces. Additionally, our lightweight ResNet variant yields strong performance (sometimes exceeding [78]) while operating at a fraction of the compute time (20ms vs 64ms on an A100 GPU). Note that for fair comparison to regression baselines, all results (except those in Table 4) are without our temporal stability contribution.

4.2. Evaluating depth estimation

While our method is focused on estimating occlusions for virtual assets, we can leverage our binary predictions to iteratively refine a binary-searched depth map. We compare against depth estimation methods on the ScanNetv2 test set using the protocol from [20], presenting results in Table 1.

We can convert our binary predictions to depths by making the observation that along each ray from every pixel location $\mathbf{p}$ there lies a depth d where the prediction from our network, $C(\mathbf{p})^d$, is at the decision threshold τ. Generally τ would be 0.5, but we use the best thresholds from Section 4.1. Our optimal estimated depth map D_{pred} is one where $C(\mathbf{p})^d = \tau$ for all $\mathbf{p}$.

It is time consuming to naively iterate different depth values to find d for which $C(\mathbf{p})^d = \tau$. Instead we binary search along the ray to find the optimal depth, relying on directional predictions that signify if the current depth on the ray is ahead or behind the real depth. We initialize our min. and max. depths to 0.5m and 8m respectively. These are updated each iteration, for each location $\mathbf{p}$, as we search. We take $M = 12$ binary search steps, achieving an effective granularity of 4096 for each $\mathbf{p}$ (see Figure 4). Only the final MLP head is run at each step, with the backbone only run

Method	Occlusion evaluation			Depth evaluation			
	IoU All↑	IoU Surface↑	IoU Boundary↑	Abs Rel↓	Sq Rel↓	RMSE ↓	$\delta < 1.05$ ↑
Ours (no edge-regularization)	59.79	33.03	36.02	.0503	.0155	.1617	68.31
Ours (no high resolution supervision)	59.94	33.09	36.31	.0501	.0157	.1628	68.44
Ours (monocular backbone)	48.24	21.46	26.72	.1220	.0533	.2740	34.71
DORN-style classification	56.81	29.54	31.29	.0535	.0171	.1743	65.41
Classification	55.95	29.49	31.44	.0617	.0220	.1937	60.75
Regression	58.91	31.48	33.03	**.0487**	.0151	.1617	**69.62**
Ours	**60.14**	**33.29**	**36.54**	.0498	**.0149**	**.1595**	68.52

Table 3. **Ablating our method, showing our contributions lead to better depth and occlusion scores.** All are trained equivalently on the ScanNetv2 dataset using the SimpleRecon (ResNet) backbone network.

Method	Temporal Score↓	IoU A.↑	IoU S.↑	IoU B.↑
Regression	233.1	77.84	43.44	41.29
Ours (w/o temporal)	235.1	79.09	**44.73**	42.44
Ours (with temporal)	**164.5**	**79.28**	44.50	**42.90**

Table 4. **Evaluating temporal stability on ScanNetv2**, by comparing predictions on temporally adjacent frames. Our temporal approach leads to significantly less flicker, as seen in the large reduction in the temporal score, without impacting IoU.

once to produce feature maps that are reused.

Notably, when using [78] as a backbone, our method achieves a new state-of-the-art on ScanNetv2 in depth estimation, alongside our core occlusion-IoU evaluation.

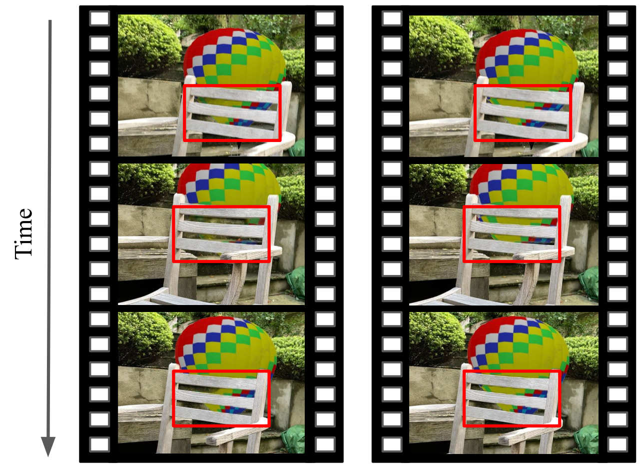
4.3. Temporal evaluation

Occlusion systems should exhibit *temporal coherence* to ensure visually compelling results [36]. In the spirit of [60, 36], we place a fixed virtual AR asset into a scene (here a plane), and keep track of the change in predicted visibility of the groundtruth scene mesh, provided in ScanNetv2, across a window of frames. Specifically, we use an infinite vertical plane at a fixed position in front of the first camera in a sequence, and compute a compositing mask, C^t, for each subsequent frame. We project scene mesh vertices to the camera and store the visibility prediction from C^t for that vertex w.r.t the vertical plane. We tally the number of times the visibility prediction changes for each vertex over 13 frames (i.e. 0.43 seconds). We normalize the count by the number of frames to compute a temporal score. Evaluation is performed on the ScanNetv2 test scenes.

'Ours (with temporal)' results in a significant boost of almost 30% in temporal stability while achieving IoU scores comparable to our non-temporal variant (see Table 4). We also show a qualitative example of our more temporally stable approach in Figure 5. Please see our video for examples.

4.4. Ablation

We validate our approach by training variants of our model with our contributions turned off, and show in Table 3 that these models achieve worse scores. We train a model without our edge-based regularization; without high resolution supervision, where our MLP is supervised at the native model output resolution as in [78]; a non-MVS monocular method; a DORN-style [23] classification network, where we output a classification head with 80 bins each with a BCE-trained sigmoid activation; and a discretized depth-classification loss.



(a) Ours without temporal stability

(b) Ours with temporal stability

Figure 5. **Temporal stability.** (Left) The basic method without our temporal stability input displays prominent flickering i.e. big changes in the predictions for each frame of this sequence. (Right) Our predictions are more temporally stable over time, enabling more immersive AR. Please also see the accompanying video.

4.5. Qualitative comparisons

Figure 6 shows qualitative results trained on Hypersim [72] on a range of real-world scenes, comparing our approach to alternative state-of-the-art methods. Surprisingly, our method produces visually equivalent, or better-quality, predictions compared to those from on-device Lidar from an iPhone 12 Pro. We also compare to the sparse-point densification approach from [36]. Their approach relies on sparse points as input. We found that ARKit's SLAM points are too sparse for their method, so we instead randomly sampled 2,000 Lidar depth points for each test frame, and fed these into their publicly available code. Visually, our re-

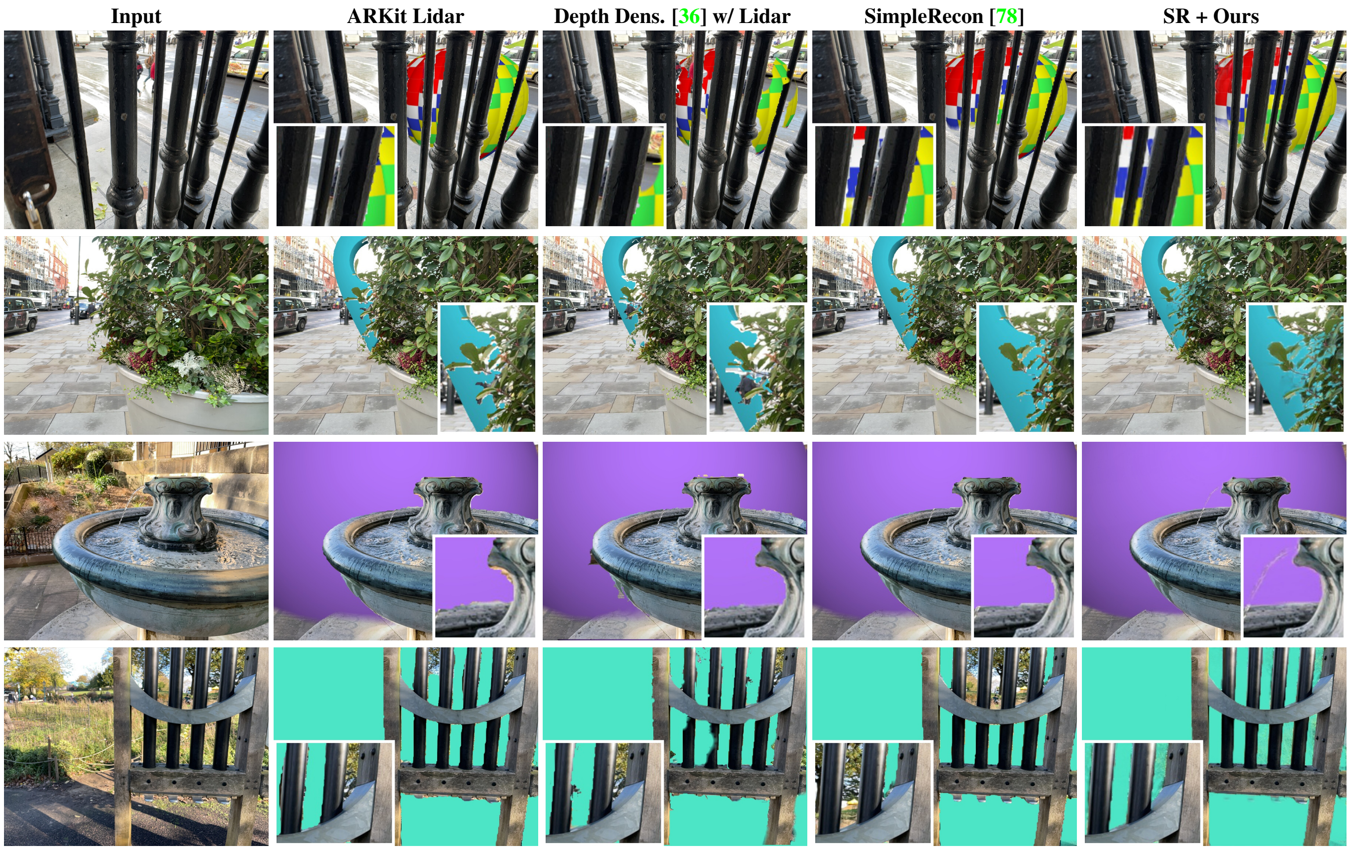


Figure 6. **Qualitative occlusion comparisons using our own casually captured footage.** We occlude virtual assets (rows 1-2) and a fixed plane at 2m depth (rows 3-4). Our occlusions are typically more realistic than baselines, in particular around soft edges, e.g. leaves. We also avoid catastrophic failures, e.g. around the bars in the final row. Please see the supplementary material for videos.

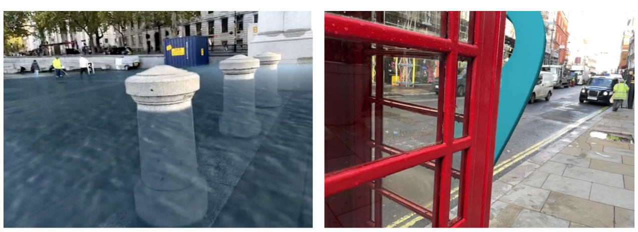


Figure 7. **Additional qualitative results.** On the right we see a failure mode, where transparency through glass is not handled correctly. This is due to limitations in our training data [72]. Please see our video for more results.

Figure 8. **Moving objects.** While our model was trained on static scenes, we achieve surprisingly robust results on moving objects.

sults have better edge fidelity than their Lidar-guided predictions. We also found that our predictions are surprisingly good on moving objects, given that our training data comes from static scenes (see Figure 8). More results are shown in Figures 3 and 7, and the supplementary video.

5. Conclusion

We presented a novel approach for inserting virtual objects into real scenes. In contrast to existing depth-based methods, we directly estimate compositing masks. We introduce metrics for evaluating occlusion mask quality, and showed that our approach allows for greater temporal stability than previous methods. Qualitative results highlight that our method produces more realistic object insertions. A natural continuation of our work is to train a model in a fully 'end-to-end' fashion, where the network directly outputs the final composited image I_{final}. This could allow the network to reason about lighting and shadows [52, 86] and object positioning [7].

Acknowledgements. Many thanks to Daniyar Turmukhambetov, Jamie Wynn, Clément Godard, and Filippo Aleotti for their valuable help and suggestions.

References

[1] Filippo Aleotti, Giulio Zaccaroni, Luca Bartolomei, Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. Real-time single image depth perception in the wild with handheld devices. *Sensors*, 2020.

[2] Apple. ARKit. Accessed: 12 July 2022.

[3] Abhishek Badki, Alejandro Troccoli, Kihwan Kim, Jan Kautz, Pradeep Sen, and Orazio Gallo. Bi3d: Stereo depth estimation via binary classifications. In *CVPR*, 2020.

[4] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021.

[5] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *CVPR*, 2021.

[6] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Localbins: Improving depth estimation by learning local distributions. In *ECCV*, 2022.

[7] David E Breen, Ross T Whitaker, Eric Rose, and Mihran Tuceryan. Interactive occlusion and automatic object placement for augmented reality. In *Computer Graphics Forum*, 1996.

[8] Jacky Cao, Kit-Yung Lam, Lik-Hang Lee, Xiaoli Liu, Pan Hui, and Xiang Su. Mobile augmented reality: User interfaces, frameworks, and intelligence. *ACM Computing Surveys*, 2021.

[9] Yuanzhouhan Cao, Zifeng Wu, and Chunhua Shen. Estimating depth from monocular images as classification using deep fully convolutional residual networks. *Transactions on Circuits and Systems for Video Technology*, 2017.

[10] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *AAAI*, 2019.

[11] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *CVPR*, 2018.

[12] Guowei Chen, Yi Liu, Jian Wang, Juncai Peng, Yuying Hao, Lutao Chu, Shiyu Tang, Zewu Wu, Zeyu Chen, Zhiliang Yu, et al. PP-Matting: High-accuracy natural image matting. *arXiv:2204.09433*, 2022.

[13] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *PAMI*, 2017.

[14] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *ICCV*, 2019.

[15] Xinjing Cheng, Peng Wang, and Ruigang Yang. Learning depth with convolutional spatial propagation network. *PAMI*, 2019.

[16] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3D shape reconstruction and completion. In *CVPR*, 2020.

[17] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, 2017.

[18] Helisa Dhamo, Keisuke Tateno, Iro Laina, Nassir Navab, and Federico Tombari. Peeking behind objects: Layered depth prediction from a single image. *Pattern Recognition Letters*, 2018.

[19] Ruofei Du, Eric Turner, Maksym Dzitsiuk, Luca Prasso, Ivo Duarte, Jason Dourgarian, Joao Afonso, Jose Pascoal, Josh Gladstone, Nuno Cruces, et al. DepthLab: Real-time 3D interaction with depth maps for mobile augmented reality. In *Symposium on User Interface Software and Technology*, 2020.

[20] Arda Duzceker, Silvano Galliani, Christoph Vogel, Pablo Speciale, Mihai Dusmanu, and Marc Pollefeys. DeepVideoMVS: Multi-view stereo on video with recurrent spatio-temporal fusion. In *CVPR*, 2021.

[21] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.

[22] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NeurIPS*, 2014.

[23] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, 2018.

[24] Henry Fuchs, Mark A Livingston, Ramesh Raskar, Kurtis Keller, Jessica R Crawford, Paul Rademacher, Samuel H Drake, Anthony A Meyer, et al. Augmented reality visualization for laparoscopic surgery. In *MICCAI*, 1998.

[25] Yasutaka Furukawa and Carlos Hernández. Multi-view stereo: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2015.

[26] Raghudeep Gadde, Varun Jampani, and Peter V Gehler. Semantic video cnns through representation warping. In *ICCV*, 2017.

[27] Ravi Garg, Vijay Kumar BG, and Ian Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *ECCV*, 2016.

[28] Silvio Giancola, Matteo Valenti, and Remo Sala. *A survey on 3D cameras: Metrological comparison of time-of-flight, structured-light and active stereoscopy technologies*. 2018.

[29] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[30] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. In *ICCV*, 2019.

[31] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[33] Donald Hearn, M Pauline Baker, and M Pauline Baker. *Computer graphics with OpenGL*. Pearson, 2004.

[34] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *PAMI*, 2007.

[35] Derek Hoiem, Andrew N Stein, Alexei A Efros, and Martial Hebert. Recovering occlusion boundaries from a single image. In *ICCV*, 2007.

[36] Aleksander Holynski and Johannes Kopf. Fast depth densification for occlusion-aware augmented reality. *SIGGRAPH Asia*, 2018.

[37] Yuxin Hou, Juho Kannala, and Arno Solin. Multi-view stereo by temporal nonparametric fusion. In *ICCV*, 2019.

[38] Junjie Hu, Mete Ozay, Yan Zhang, and Takayuki Okatani. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. In *WACV*, 2019.

[39] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning multi-view stereopsis. In *CVPR*, 2018.

[40] Ahmad Humayun, Oisin Mac Aodha, and Gabriel J. Brostow. Learning to Find Occlusion Regions. In *CVPR*, 2011.

[41] Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon. DPSNet: End-to-end deep plane sweep stereo. In *ICLR*, 2019.

[42] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NeurIPS*, 2015.

[43] Varun Jampani, Huiwen Chang, Kyle Sargent, Abhishek Kar, Richard Tucker, Michael Krainin, Dominik Kaeser, William T Freeman, David Salesin, Brian Curless, et al. SLIDE: Single image 3D photography with soft layering and depth-aware inpainting. In *ICCV*, 2021.

[44] Huaizu Jiang, Jingdong Wang, Zejian Yuan, Tie Liu, Nanning Zheng, and Shipeng Li. Automatic salient object segmentation based on context and shape prior. In *BMVC*, 2011.

[45] Joaquim Jorge, Rafael Kuffner Dos Anjos, and Ricardo Silva. Dynamic occlusion handling for real-time AR applications. In *International Conference on Virtual-Reality Continuum and its Applications in Industry*, 2019.

[46] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. Design of an image edge detection filter using the sobel operator. *Journal of solid-state circuits*, 1988.

[47] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *ICCV*, 2017.

[48] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[49] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. PointRend: Image segmentation as rendering. In *CVPR*, 2020.

[50] Johannes Kopf, Xuejian Rong, and Jia-Bin Huang. Robust consistent video depth estimation. In *CVPR*, 2021.

[51] Yevhen Kuznietsov, Marc Proesmans, and Luc Van Gool. CoMoDA: Continuous monocular depth adaptation using past experiences. In *WACV*, 2021.

[52] Chloe LeGendre, Wan-Chun Ma, Graham Fyffe, John Flynn, Laurent Charbonnel, Jay Busch, and Paul Debevec. DeepLight: Learning illumination for unconstrained mobile mixed reality. In *CVPR*, 2019.

[53] Vincent Lepetit and M-O Berger. A semi-automatic method for resolving occlusion in augmented reality. In *CVPR*, 2000.

[54] Jiaxin Li, Zijian Feng, Qi She, Henghui Ding, Changhu Wang, and Gim Hee Lee. MINE: Towards continuous depth MPI with NeRF for novel view synthesis. In *ICCV*, 2021.

[55] Zhengqi Li and Noah Snavely. MegaDepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018.

[56] Shanchuan Lin, Andrey Ryabtsev, Soumyadip Sengupta, Brian L Curless, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Real-time high-resolution background matting. In *CVPR*, 2021.

[57] Chao Liu, Jinwei Gu, Kihwan Kim, Srinivasa G Narasimhan, and Jan Kautz. Neural RGB$\rightarrow$D sensing: Depth and uncertainty from a video camera. In *CVPR*, 2019.

[58] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3D supervision. In *NeurIPS*, 2019.

[59] Xiaoxiao Long, Lingjie Liu, Christian Theobalt, and Wenping Wang. Occlusion-aware depth estimation with adaptive normal constraints. In *ECCV*, 2020.

[60] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *SIGGRAPH*, 2020.

[61] Marcio C de F Macedo and Antonio L Apolinario. Occlusion handling in augmented reality: Past, present and future. *Transactions on Visualization and Computer Graphics*, 2021.

[62] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR*, 2019.

[63] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[64] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. RegNeRF: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, 2022.

[65] Kiem Ching Ong, Hung Chuan Teh, and Tiow Seng Tan. Resolving occlusion in image sequence made easy. *The Visual Computer*, 1998.

[66] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019.

[67] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020.

[68] Federico Perazzi, Anna Khoreva, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. Learning video object segmentation from static images. In *CVPR*, 2017.

[69] Thomas Porter and Tom Duff. Compositing digital images. In *Annual Conference on Computer Graphics and Interactive Techniques*, 1984.

[70] Michael Ramamonjisoa, Yuming Du, and Vincent Lepetit. Predicting sharp and accurate occlusion boundaries in monocular depth estimation using displacement fields. In *CVPR*, 2020.

[71] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *TPAMI*, 2020.

[72] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *ICCV*, 2021.

[73] Thomas Roddick, Benjamin Biggs, Daniel Olmeda Reino, and Roberto Cipolla. On the road to large-scale 3D monocular scene reconstruction using deep implicit functions. In *ICCV*, 2021.

[74] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[75] Martin Runz, Kejie Li, Meng Tang, Lingni Ma, Chen Kong, Tanner Schmidt, Ian Reid, Lourdes Agapito, Julian Straub, Steven Lovegrove, et al. FroDO: From detections to 3D objects. In *CVPR*, 2020.

[76] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. PIFu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *ICCV*, 2019.

[77] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. PIFuHD: Multi-level pixel-aligned implicit function for high-resolution 3D human digitization. In *CVPR*, 2020.

[78] Mohamed Sayed, John Gibson, Jamie Watson, Victor Prisacariu, Michael Firman, and Clément Godard. SimpleRecon: 3D reconstruction without 3D convolutions. In *ECCV*, 2022.

[79] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016.

[80] Soumyadip Sengupta, Vivek Jayaram, Brian Curless, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Background matting: The world is your green screen. In *CVPR*, 2020.

[81] Ayan Sinha, Zak Murez, James Bartolozzi, Vijay Badrinarayanan, and Andrew Rabinovich. Deltas: Depth estimation by learning triangulation and densification of sparse points. In *ECCV*, 2020.

[82] Alvy Ray Smith and James F Blinn. Blue screen matting. In *Conference on Computer Graphics and Interactive Techniques*, 1996.

[83] Andrew N Stein and Martial Hebert. Occlusion boundaries from motion: Low-level detection and mid-level reasoning. *IJCV*, 2009.

[84] Patrik Sundberg, Thomas Brox, Michael Maire, Pablo Arbeláez, and Jitendra Malik. Occlusion boundary detection and figure/ground assignment from optical flow. In *CVPR*, 2011.

[85] Xiao Tang, Xiaowei Hu, Chi-Wing Fu, and Daniel Cohen-Or. GrabAR: occlusion-aware grabbing virtual objects in AR. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, 2020.

[86] Joanna Tarko, James Tompkin, and Christian Richardt. Real-time virtual object insertion for moving 360° videos. In *International Conference on Virtual-Reality Continuum and its Applications in Industry*, 2019.

[87] Andrei Tkachenka, Gregory Karpiak, Andrey Vakunov, Yury Kartynnik, Artsiom Ablavatski, Valentin Bazarevsky, and Siargey Pisarchyk. Real-time hair segmentation and recoloring on mobile gpus. In *CVPR Workshop on Computer Vision for Augmented and Virtual Reality*, 2019.

[88] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *CVPR*, 2020.

[89] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3D scene inference via view synthesis. In *ECCV*, 2018.

[90] Julien Valentin, Adarsh Kowdle, Jonathan T Barron, Neal Wadhwa, Max Dzitsiuk, Michael Schoenberg, Vivek Verma, Ambrus Csaszar, Eric Turner, Ivan Dryanovski, et al. Depth from motion for smartphone AR. *Transactions on Graphics*, 2018.

[91] David R Walton and Anthony Steed. Accurate real-time occlusion for mixed reality. In *ACM Symposium on Virtual Reality Software and Technology*, 2017.

[92] Chaohui Wang, Huan Fu, Dacheng Tao, and Michael Black. Occlusion boundary: A formal definition & its detection via deep exploration of context. *PAMI*, 2020.

[93] Kaixuan Wang and Shaojie Shen. MVDepthNet: Real-time multiview depth estimation neural network. In *3DV*, 2018.

[94] Jamie Watson, Oisin Mac Aodha, Victor Prisacariu, Gabriel J. Brostow, and Michael Firman. The temporal opportunist: Self-supervised multi-frame monocular depth. In *CVPR*, 2021.

[95] Junyuan Xie, Ross Girshick, and Ali Farhadi. Deep3D: Fully automatic 2D-to-3D video conversion with deep convolutional neural networks. In *ECCV*, 2016.

[96] Gengshan Yang, Peiyun Hu, and Deva Ramanan. Inferring distributions over depth from a single image. In *IROS*, 2019.

[97] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth inference for unstructured multi-view stereo. In *ECCV*, 2018.

[98] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021.

[99] Gang Zhang, Xin Lu, Jingru Tan, Jianmin Li, Zhaoxiang Zhang, Quanquan Li, and Xiaolin Hu. Refinemask: Towards high-quality instance segmentation with fine-grained features. In *CVPR*, 2021.

[100] Luyang Zhu, Arsalan Mousavian, Yu Xiang, Hammad Mazhar, Jozef van Eenbergen, Shoubhik Debnath, and Dieter Fox. RGB-D local implicit function for depth completion of transparent objects. In *CVPR*, 2021.

[101] Zhengxia Zou, Rui Zhao, Tianyang Shi, Shuang Qiu, and Zhenwei Shi. Castle in the sky: Dynamic sky replacement

and harmonization in videos. In *Transactions on Image Processing*, 2022.