Long-Range Augmented Reality with Dynamic Occlusion Rendering

Mikhail Sizintsev* Niluthpol Chowdhury Mithun Han-Pang Chiu Supun Samarasekera
Rakesh Kumar

SRI International, Princeton, NJ, USA

(a) Raw image (b) Segmented object masks wih depth (c) AR with occlusions

Fig. 1. Example Augmented Reality (AR) with occlusion rendering of real and virtual objects: (a) Input color image; (b) Object instances detected in the input color images and their depth inferred using outdoor terrain model; (c) Final AR rendering where closer synthethic excavator partially occludes row of real cars, while real cars properly occlude farther synthetic excavator.

Abstract— Proper occlusion based rendering is very important to achieve realism in all indoor and outdoor Augmented Reality (AR) applications. This paper addresses the problem of fast and accurate dynamic occlusion reasoning by real objects in the scene for large scale outdoor AR applications. Conceptually, proper occlusion reasoning requires an estimate of depth for every point in augmented scene which is technically hard to achieve for outdoor scenarios, especially in the presence of moving objects. We propose a method to detect and automatically infer the depth for real objects in the scene without explicit detailed scene modeling and depth sensing (e.g. without using sensors such as 3D-LiDAR). Specifically, we employ instance segmentation of color image data to detect real dynamic objects in the scene and use either a top-down terrain elevation model or deep learning based monocular depth estimation model to infer their metric distance from the camera for proper occlusion reasoning in real time. The realized solution is implemented in a low latency real-time framework for video-see-though AR and is directly extendable to optical-see-through AR. We minimize latency in depth reasoning and occlusion rendering by doing semantic object tracking and prediction in video frames.

Index Terms—Augmented reality, occlusion reasoning, depth inference, object tracking

1 Introduction

1.1 Motivation

Augmented reality (AR) became extremely popular in recent years and continuously moving from research labs to the industrial and consumer application areas. While it is possible to purchase very compelling AR systems for indoor short-range use, outdoor geo-located AR solutions still pose a significant number of challenges. Specifically, outdoor AR systems require better quality and brighter displays, high-accuracy head tracking capable of robust operation in geo-location coordinate space and in the open area; system itself must be robust to various temperature, inclement weather, dust and mechanical impacts.

Another important aspect of AR that is still overlooked by many contemporary systems (especially, outdoor systems) is the proper treatment of occlusions by real objects. A typical challenge of AR is to properly render synthetic objects that may be partially or fully occluded by real objects in the scene that are *not* a part of the model, as depicted in Fig. 1. The occlusion-based reasoning for rendering must be accurate with respect to object boundaries and stable over time. Occlusion reasoning is directly guided by depth ordering and naturally requires some knowledge of distance at all image areas that are to be augmented. This knowledge can come either from modeling of the scene and interaction objects or directly retrieving depth information. Direct dense depth measuring can be conducted via stereo cameras, but they generally have very poor depth resolution at far distances. For compact systems, baselines of 5-15 centimeters are practically possible which cannot guarantee good depth estimation farther than 10-20 meters. At the same time, outdoor AR operates on a larger scale and longer range estimations on the order of hundreds of meters are desirable. The latter can be achieved by LiDARs, but their cost, power consumption, bulkiness and lack of resolution make it challenging to use for head-worn applications at present moment. In this work, we explore yet another technique to estimate the scene depth using semantic reasoning and deep learning methods and provide a method on how to implement it within the realtime AR pipeline.

*email:mikhail.sizintsev@sri.com

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

1.2 Previous Work

Even for earlier AR systems it was widely acknowledged that proper occlusion rendering is necessary to achieve compelling experience [2]. At the same time, extracting clean occlusion boundaries proved to be quite difficult and proposed solutions followed either static model-based or depth-based approaches, or some combination [25]. Many earlier endeavors concentrated mostly on very contrived occlusion examples, such as static background or occlusions of specific types [5, 16, 29, 32]. Simultaneously, various real-time and near-real-time

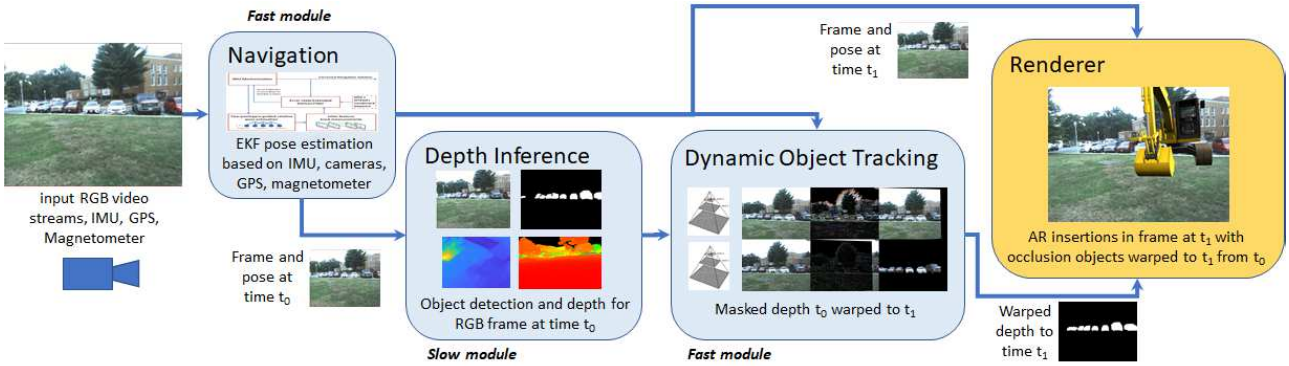


Fig. 2. Realtime Augmented Reality with dynamic occlusions processing pipeline. Raw data comes from the sensor block into the fast *Navigation* module that produces low-latency head pose estimates. Occlusion masks with depth maps are generated by the slow *DepthInference* module and its outcome is warped toward the last-tracked head pose using the fast *ObjectTracking* module. *Renderer* module receives the input from *Navigation* and *ObjectTracking* modules to generates final AR output in real time with low latency.

stereo solutions were proposed over the years to extract 3D objects boundaries [14, 15, 24, 26, 30, 33]. There more recent methods proved to be quite effective in situations where the range is limited and relatively short (on the order of dozen of meters). Additionally, methods that exploit object tracking in possible combination with the stereo [9, 27] or object's image projection on a simple terrain model [31] have also emerged. Nowadays, availability of commercial depth cameras (e.g., Intel Realsense, Etron, ZED) can also provide a good quality stereo solution that is very beneficial, especially for near-range indoor applications. Finally, there exist methods like [13] that efficiently densify the sparse point reconstruction coming as byproduct from simultaneous localization and mapping (SLAM) output used for pose tracking and allow for close range occlusion reasoning.

On the other hand, for long-range outdoor applications, where insertions can be required up to be hundreds of meters to a kilometer, stereo-based depth estimation is not practically feasible. Recent advances in LiDAR technology made these sensors available and more affordable to general public and very promising methods for densifying the LiDAR depth measurement using color images have emerged [18, 23, 28]. Despite that, LiDAR sensors are still more expensive and too bulky for head-worn systems. Furthermore, some scenarios may even have restriction on IR light emission (interference with other devices, like light-sensitive cameras). Thus, we are currently interested in the a solution that would make most of current AR setups capable of occlusion handling without non-trivial hardware upgrade.

As a radical departure from standard methods, authors of [8] proposed to re-use the terrain model of the scene to approximate structural scene depth, then use semantic segmentation to outline possible occluders (objects like cars, people, trees, etc.) and then estimate their depth from the terrain rendering by assuming detected objects reside on the ground plane (depth value is assigned as corresponding to the lowest intersection point with the terrain model). Authors [8] demonstrated their approach is capable of generating proper occlusion arrangements; however, it is mostly suitable for offline processing. Since depth estimation and occlusion detection rely on computationally intensive inference and rendering mechanisms, non-trivial adjustments must be made to allow such architecture to run at high framerate, low latency, and in dynamic real scenarios especially on efficient small form-factor low-power hardware.

1.3 Contribution

To satisfy the objectives and taking previous research into account, this paper offers the following contributions that allow to achieve accurate and efficient occlusion rendering as depicted in Fig. 1. We present an extended AR pipeline in Sec. 2.1 that is designed to perform AR augmentation with dynamic occlusions in real-time taking into consideration the generally slower nature of depth inference mechanism. Specifically, we propose to extract candidate occluder entities via semantic instance segmentation (Sec. 2.3) and infer their metric distance either from the terrain model rendered from specific camera viewpoint or neural depth estimation (Sec. 2.4). The important contribution is two-fold: (i) instead of semantic segmentation used in [8], we rely on instance segmentation which provides precise occlusion boundaries and allows the system to distinguish objects even if they self-occlude each other in the image rather than incorrectly treating them as one entity; (ii) we effectively compensate for the higher latency of object detection and depth rendering components by adding additional tracking module that efficiently adjusts occluder masks to the latest AR image and pose (Sec. 2.5); furthermore, we can extrapolate occluder masks to future time and compensate for optical-see through latency requirements. This ability to extrapolate effectively also allows us to process fewer frames and for the pipeline to run on more power-efficient hardware. The framework is currently implemented and executed on a backpack VR computer and we have demonstrated applicability and robustness of the proposed framework of AR with dynamic occlusion reasoning.

2 Technical Approach

2.1 AR with occlusions pipeline

The conceptual workflow of the realtime AR with dynamic occlusions pipeline is depicted in Fig. 2. Raw data (camera images, IMU, GPS, magnetometer) comes from the sensor block into the *Navigation* module. In a classical AR, the *Navigation* module produces very low-latency head pose estimate and feeds it into the *Renderer*.

In case of occlusion, depth map as well as its corresponding color image is passed on *Renderer* to generate texture masks to occlude proper portions of the synthetic avatars. The naive system architecture would be a sequential connection from *Navigation* to *DepthInference* to *Renderer* and that is how previous work [8] has it set up. However, depth estimation process of *DepthInference* is much slower than *Navigation* module, which means the system would not run in real time and would exhibit huge latency – its performance would be limited by the throughput of the slowest node *DepthInference*.

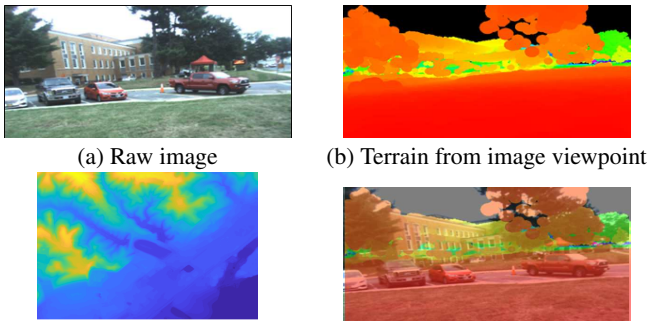
In order to enable real time performance and hide the latency produced by *DepthInference* module, we introduce another *ObjectTracking* module to *warp* last-inferred depth map at time t_0 to the latest output pose from low-latency *Navigation* module at time t_1. *ObjectTracking* module tracks the objects of interest from time t_0 to time t_1 in an efficient manner, allowing proper occlusion reasoning by this object in the latest AR frame t_1, as explained in Sec. 2.5 below.

2.2 Navigation module

Our *Navigation* module is a visual-inertial tracking system that uses several sensors to allow for geo-located real-time pose estimation: 6-axis IMU (3-axis gyroscope and 3-axis accelerometer), monochrome global shutter stereo camera for tracking, high-resolution RGB color camera for AR augmentation and optional geo-landmark matching, GPS device and magnetometer for global location and orientation updates. The tracking algorithm itself is adapted from [20, 22] that is based on error state Extended Kalman Filter (EKF) for visual-inertial

(a) Raw image (b) Detected objects

Fig. 3. Example raw images (a) and corresponding object detection results (b). Each object silhouette is modeled as a planar entity in 3D space, which serves as efficient occlusion mask for AR insertion.



(a) Raw image (b) Terrain from image viewpoint

(c) Terrain database (top view) (d) Image and depth overlaid

Fig. 4. Depth estimation from terrain model. Color image and its estimated pose (a) is used to render the terrain data (b) from the image viewpoint as in (c). The quality of pose estimation is critical for proper terrain rendering and consequent depth prediction for each input color image, as demonstrated in overlay (d).

odometry and is an extension of the seminal work of [19] with several important differences and additions: (i) reliance on a small form-factor lightweight MEMs IMU of inferior performance (in comparison to the one used in [19]) which requires more often filter updates to limit higher drift rate (ii) simple but computationally efficient Harris corner tracker that allows to perform filter update at every frame and use of robust two-point gyro guided relative pose estimation constraint (iii) feature tracks over three time-consecutive frames enforce a more robust geometric pose estimation constraint (iv) measurement model is extended to include monocular feature tracks from both left and right of the stereo cameras as well as the stereo 3D tracks constraints which allows to utilize stereo depth information while being robust to significant occlusion and saturation in a stereo pair (v) dynamic geo-landmark database update for global orientation correction that relies on gravity-aligned feature descriptor matching.

The employed navigation solution is comparable to other state-of-the-art visual-inertial systems and achieves low dead-reckoning positional drift rates around 0.1%, global heading angle error of 0.1° and global position error within roughly 2.5 meters (constrained by GPS). We also make use of the terrain map either in the form of top down LiDAR scan or digital elevation model (DEM) for reliable global height estimation as well as proper placement of AR insertions on the ground plane.

The *Navigation* module produces global geo-located pose P with minimal latency at the color camera video rate for video see-through (VST) AR, and can also yield IMU-rate poses (or higher) for Optical see-through (OST) AR.

2.3 Occlusion mask segmentation

Occlusion masks provide very pertinent information for the development of effective augmented reality systems. The most obvious aspect is model-based occlusions, such as from terrain, buildings or trees – entities that could be modeled via 3D reconstruction beforehand and effectively used in scene interaction. The outstanding challenge is presented by dynamic objects (e.g., vehicles, pedestrians) that are not part of static scenes. In this regard, we propose to apply an instance segmentation model to infer the instances of these classes of objects from the camera image and use the corresponding segmented masks to enrich the occlusion information. Specifically, we rely on a fully convolutional deep neural network-based model YOLACT [6] (trained on COCO dataset [17]) for real-time instance segmentation. The YOLACT model is capable of efficiently detecting different classes of objects that are not modeled in the scene (people, cars, bicycles, traffic signs) and provide the associated binary mask in the form of a silhouette. We also evaluated Mask R-CNN based instance segmentation model [12], and empirically choose YOLACT due to better speed with similar accuracy.

Prior work [8] relied on SegNet semantic segmentation model [3] to estimate the depth of objects in the scene. However, due to the inherent limitations of semantic segmentation models, the model is not able to treat multiple objects of the same class as distinct objects. Hence, the system [8] would suffer significantly when multiple objects of the same classes are occluded in the scene (as in Fig. 1). On the contrary, our system would be able to reliably estimate the depth of occluder objects in such scenarios, as the instance segmentation model can distinguish instances within categories. A couple of output examples are shown in Fig. 3. Each object silhouette is modeled as a planar entity in 3D space. This simplifying assumption greatly speeds up depth and occlusion procession which effectively serving for the purpose of AR – very good 3D boundary delineation and proper depth ordering, rather than depth accuracy within the object itself that barely affects occlusion reasoning. The error analysis of this method is presented in Sec. 3.2 below.

2.4 Depth estimation

It is argued in Sec. 1.2 that dense depth estimation via explicit sensing (e.g., stereo, LiDARs) is currently unfeasible for most outdoor head-worn AR systems. Therefore, inspired by [8], we propose *DepthInference* module that generates a full range dense depth map for camera frames based on geo-referenced terrain model and occlusion mask of objects, as depicted in Fig. 4. Since we utilize the terrain model for proper AR vehicle and effect placement, for every frame at time t we could also render the 3D point cloud from the viewpoint defined by navigation pose P_t and camera intrinsic matrix K (e.g., as in Fig. 4b). The terrain model itself could be a minimal digital elevation model (DEM) that is publicly available for the majority of the Earth locations, or top-down aerial lidar scans that have ground structures, or even full 3D reconstruction from the aerial drones that have the most details for the static scenes, like buildings, trees and even foliage. In this work, we utilize aerial LIDAR point cloud collected from United States Geological Survey (USGS) website as the geo-referenced data.

Once the described-above depth-map is rendered, the distance for each of the objects silhouettes detected as in Sec. 2.3 is inferred as the depth value corresponding to the lowest point on the silhouette according to the gravity vector deduced from the rotation of the navigation pose P – here, for simplicity and speed, we rely on an assumption that objects reside on the ground plane, which is almost always true for cars, trucks, bicycles, people, dumpsters, etc.

We also considered an alternative depth estimation technique from a single color image. Specifically, we used the DenseDepth inference network [1] trained on KITTI dataset [10] that is fairly similar to our outdoor scenarios. Furthermore, we calibrate the output depth values based on the corresponding sparse stereo matches obtained from the tracking camera. Specifically, we use the sparse stereo matches from our tracking camera to measure "true" depth of some points in the AR camera image – based on these samples, we can adjust a linear mapping from network output values to metric depth values. We found this adjustment useful because while the scenarios covered in KITTY dataset generally match our outdoor scenario, the camera parameters of the AR system (focal length, aspect ratio and hence fields of view) are typically different from the camera parameters of the training dataset. Some depth inference examples are depicted in Fig. 10 including comparison to DfT technique which is detailed in section Sec. 3.2.

2.5 Dynamic occlusion prediction

Due to intentionaly asynchronous nature of the proposed AR pipeline in Fig. 2, *Navigation* and *DepthInference* run with different rates and

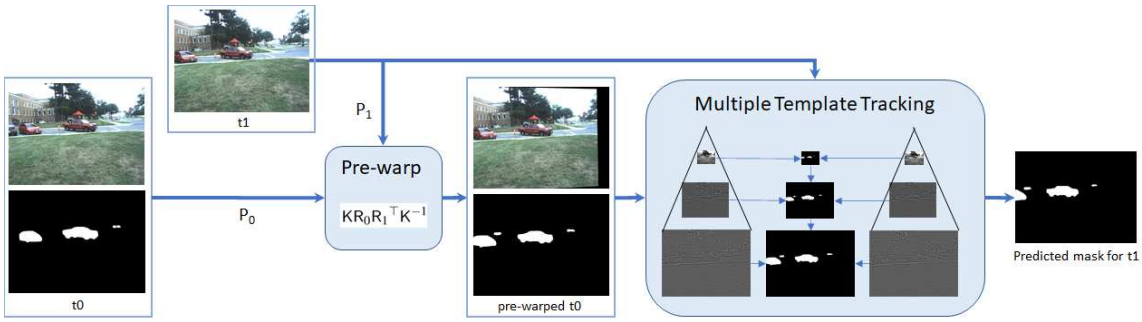


Fig. 5. Dynamic object tracking via *ObjectTracking* module. The objective is to track each occluder mask detected at time t_0 to a more recent time of AR rendering t_1. Initially, camera motion is partially compensated using rotation-based homography pre-warp; then each object template is tracked from frame t_0 to t_1 using hierarchical affine model-based tracking over Laplacian pyramid. See Sec. 2.5 for details.

latency (latter is much slower). Still, the rendering of background cplor image produced at some time t_1 and occlusion mask produced at slightly earlier time t_0 generally results in a compelling visual experience for *static* objects, provided that computed corresponding poses P_0 and P_1 are accurate. The AR illusion only breaks down either when planar silhouette is too simplistic to capture object structure, or objects themselves are moving with respect to the scene. The proposed *ObjectTracking* module takes care of these shortcomings by performing necessary adjustment of occlusion masks detected at time t_0 toward a later time t_1.

The purpose of *ObjectTracking* module is to track individual occlusion masks from time t_0 to time t_1 and is schematically visualized in Fig. 5. Specifically, each silhouette mask $\mathscr{M}(t)$ is treated as a planar template and is tracked via affine constraint [11] that can encapsulate a variety of transforms including translation, rotation and sheering:

$$\mathscr{M}(t_1) = \mathsf{A}.\mathscr{M}(t_0), \text{ where } \mathsf{A} = \begin{bmatrix} 1+a_{11} & a_{12} & a_{13} \\ a_{21} & 1+a_{22} & a_{23} \end{bmatrix}. \quad (1)$$

To minimize the processing time, we need a simple yet robust and effective method – we rely on a modified version of the inverse Lucas-Kanade template tracking [4]. Specifically, we compute the gradient and error-image components for the whole image, while updating the motion model parameters for each silhouette separately. In order to capture larger displacement and be more resilient to frame-to-frame lighting variations, the iterative updates are also performed in the coarse-to-fine manner using Laplacian pyramid [7]. Furthermore, if at any particular level a template is of small size, we fallback to using a simpler 2-parameter $\begin{bmatrix} a_{13} & a_{23} \end{bmatrix}^\top$ image translation-only motion model.

The described-above mask warping Eqn. (1) is suitable for the Video See-Through (VST) insertion into images at time t_1 but can also be extrapolated into some arbitrary short time in future t_2 to avail prediction for Optical see-through rendering. Specifically, since $(t_2 - t_1)$ is expected to be relatively small for low-latency live system, the same transform A properly scaled by the time adjustment $\Delta t = (t_2 - t_1)$ would correspond to the constant velocity assumption:

$$\mathscr{M}(t_2) = \mathsf{A}(\Delta t)\mathsf{A}.\mathscr{M}(t_0), \text{ where } \Delta t = (t_2 - t_1) \text{ and} \quad (2)$$

$$\mathsf{A}(\Delta t) = \begin{bmatrix} 1+a_{11}\Delta t & a_{12}\Delta t & a_{13}\Delta t \\ a_{21}\Delta t & 1+a_{22}\Delta t & a_{23}\Delta t \end{bmatrix}. \quad (3)$$

Finally, prior to performing the described-above template model-based motion computation, we pre-warp the initial image $Im(t_0)$ toward target image $Im(t_1)$ using rigid estimated poses P_0 and P_1. This step of subtracting out camera motion before template tracking is quite important to make sure recovered transform A (1) subsumes the object motion only and constant velocity model is properly applied in (2). Considering that we have dense estimated depth for time t_0, we can perform proper new view synthesis to time t_1. However, since we are not interested at rendering the full image, but only aiding the template tracking, a much simpler and faster operation of homography-based warping based on frame-to-frame suffices:

$$\mathbf{p}_0 = \mathsf{K}\mathsf{R}_0\mathsf{R}_1{}^\top\mathsf{K}^{-1}\mathbf{p}_1, \quad (4)$$

where R_0 and R_1 are rotation components of P_0 and P_1, respectively. The parallax resulting from depth variation in the scene is generally small and its compensation will be subsumed the template adjustment computation itself.

3 Experimental Evaluation

The system is implemented in C++ (renderer is Unity-based) and executed on a HP VR backpack with Intel Core i7-8850H CPU, 16 GB of RAM and Nvidia RTX 2080 GPU. Intel Realsense D435i stereo camera with IMU and high-resolution RGB image for augmentation was used as a main sensor. The average EKF estimation epoch of *Navigation* takes less than 30 msecs. and satisfies the speed requirement for the video see-through augmented reality. Instance segmentation and terrain rendering with various overhead inside *DepthInference* module take about 120 msecs. per frame while *ObjectTracking* module takes less than 15 msecs. on average to complete.

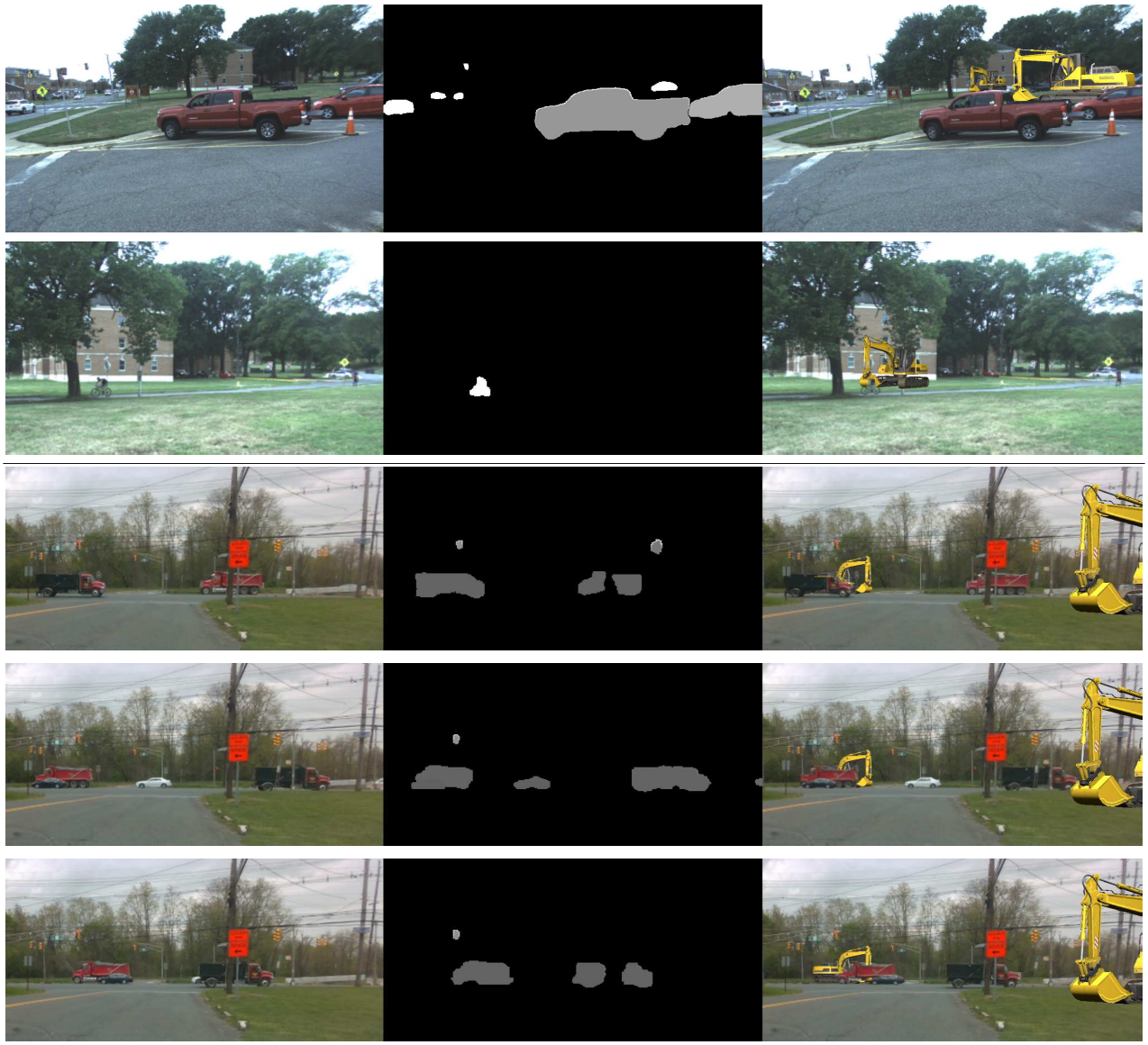
Our experimental evaluation will show example insertions in two different scenarios with static and moving objects that are *not* part of the terrain model. We also give an example of intermediate processing outcomes for *ObjectTracking* and evaluate its performance.

3.1 AR rendering with occlusions

Figures 1 and 6 show example frames from two different sequences with inserted synthetic excavator and occlusion objects being mostly cars, bicycles and people: incoming color images (Fig. 6a) together with detected object instances and inferred depth (Fig. 6b) are shown as well. The supplementary materials present full video snippets. Visual inspection of augmentation results (Fig. 6c) show that object-on-the-ground assumption works well in practices because cars/trucks properly occlude far-inserted avatars and being occluded by close-inserted avatars; both static and dynamic object outlines are consistent in the majority of cases and result in more compelling AR experience.

Additionally, to exemplify necessity of introducing *ObjectTracking*, Fig. 7 shows augmentation example with *ObjectTracking* module disabled in the example of **Seq 2**. Zoomed-in portion concentrates on car and truck moving in opposite directions. As expected, it is quite clear that while occlusion rendering remains correct for static objects, noticeable artifacts are introduced for dynamic objects that are proportional to their displacement, as depicted in Fig. 7c. Specifically, we observe the synthetic excavator is incorrectly occluded while the car and the track look elongated – that happens when the occluder is rendered at the location of frame in the past (time t_0 in the context of Fig. 2) but background image is the latest (time t_1 in the context of Fig. 2) when both vehicles have moved relatively to the global sence.

After presenting AR insertion results, we want to exemplify how the proposed work is different from its predecessor [8]. First, the architecture of [8] is not designed for real-time AR pipeline and cannot operate at a reasonable framerate with low latency on the current hardware to provide satisfactory AR experience. Second, the use of instance segmentation [6] rather than traditional semantic segmentation [3] results in better 3D boundary outlines and occlusion reasoning – representative examples of segmented frames from both datasets and corresponding



(a) Raw image (b) Detected object masks (c) AR with occlusions

Fig. 6. Augmented reality example for **Seq 1** (top 2 rows) and **Seq 2** (bottom 3 rows) were synthetic excavators are inserted in the scene respecting occlusions with real objects: (a) input color image; (b) detected objects intances with inferred depth (c) augmented image with occlusions. Synthetic excavator closer to camera occlude real objects, while farther excavator is properly occluded by objects in front of it. See supplementary videos.

object instance frames are depicted in Fig. 8 (we only retain certain classes like cars, trucks, bicycles in order to obtain hypothetical occlusion mask from the semantic segmentation output) . Specifically, the instance segmentation [6] employed in the current solution produces crisper object silhouettes (as in Fig. 8d) and does not merge several objects that are close to each other into one blob (as in Fig. 8c) and naturally produces distinctive templates for *ObjectTracking* module without requiring an additional grouping step. All of the above attributes directly affect the quality of depth inference (since the estimated terrain intersection points are more reliable) and the quality of the occlusion rendering output.

3.2 Error analysis

While correct boundary delineation and correct depth ordering are the key factors for proper occlusion treatment, the *absolute* depth estimation is still important because virtual objects operate in physical metric space. For the subsequent analysis of accuracy of the depth-from-terrain (DfT) prediction method described in Sec. 2.4, let us consider a general case of observer residing on the flat ground plane looking straight at the horizon and analyze the depth error based on projection rays. Without loss of generality, if the camera oriented at some non-zero roll and pitch angles, we can always rectify the image with homography operation similar to Eqn. (4).

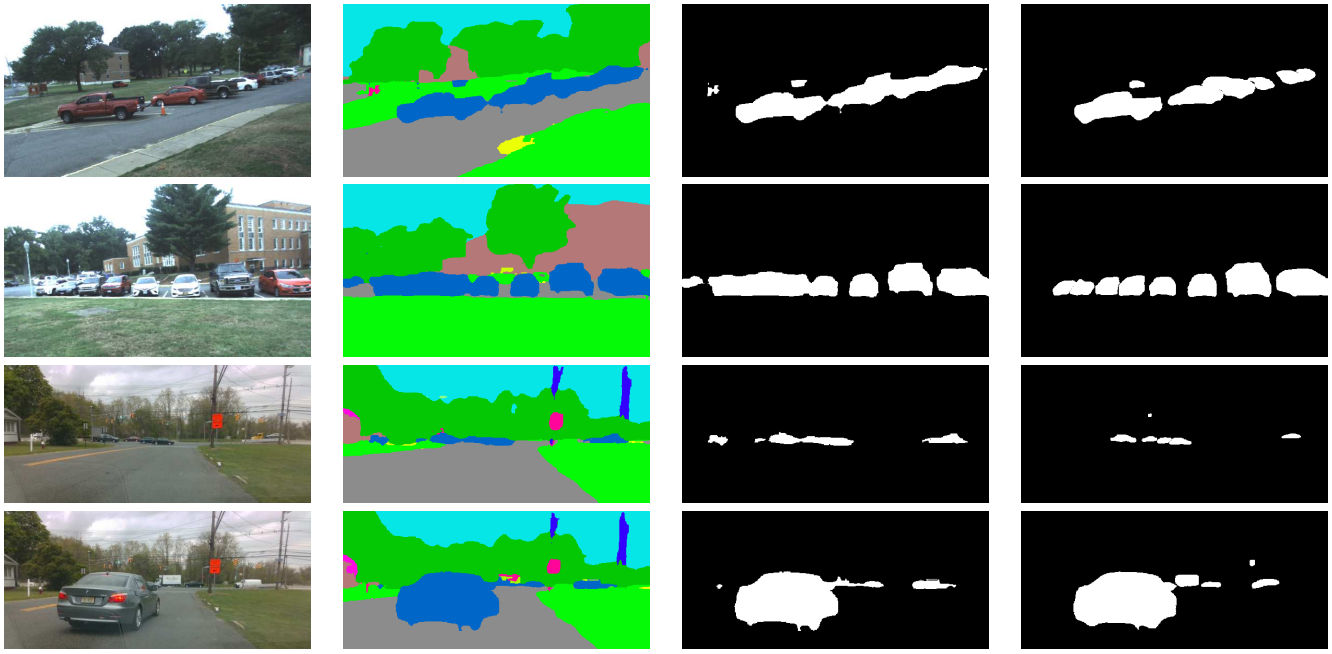
Figure 9 depicts the side view of an observer with a camera with focal length f residing at height $\mathbf{h}$. Then using basic geometry we can deduce the relationship between depth and depth uncertainty, which turns out to be quadratic:

$$D_{err} = \frac{p}{2}\frac{D^2}{f\mathbf{h}}, \tag{5}$$

with p corresponding to the underlying image intersection [pixel] precision. This error profile is very similar to stereo and any other depth-from-triangulation method. In fact, the DfT technique resembles stereo paradigm where depth is inferred from image [vertical] disparity from horizon line $\mathbf{y}$, baseline being $\mathbf{h}$ and focal length f. The quality of occluder object outline (errors at object boundaries) depends on the instance segmentation algorithm in use. The instance segmentation methods we rely on typically yield masks with pixel precision (e.g., $p \approx 1$). In the example of Fig. 9, we considered an observer standing on the ground with head-worn sensor at $\mathbf{h} = 1.8$ m height;

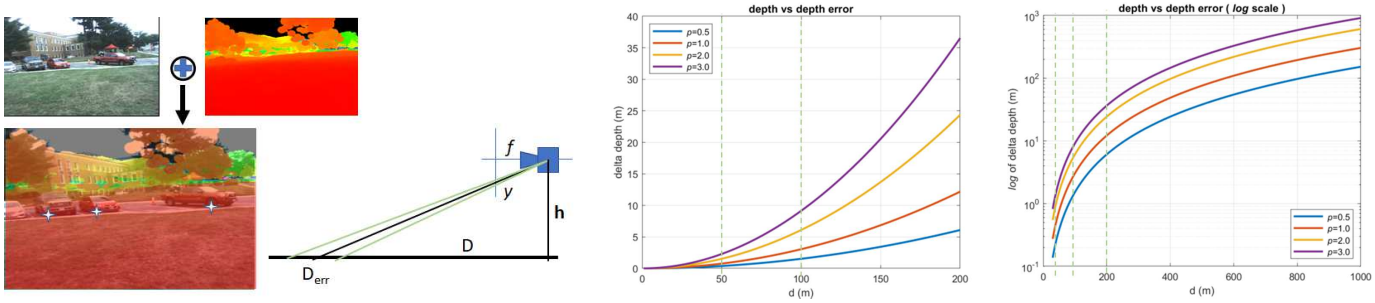
(a) Raw image (b) AR with *ObjectTracking* (c) AR without *ObjectTracking*

Fig. 7. Augmented reality example for **Seq 2** when *ObjectTracking* module is enabled and disabled: (a) Zoomed-in central image portion of the raw color image; (b) real-time AR insertion with all active modules in the proposed architecture (including *ObjectTracking*); (c) real-time AR-insertion without *ObjectTracking* module – moving objects tend to create "stretching" artifacts due to latency in *DepthInference* module (over few frames) and must be compensated via engaging *ObjectTracking* module, as explained in Sec. 3.1.



(a) Raw image (b) Segmentation output [3] (c) Dynamic classes of [3] (d) Occlusion mask from [6]

Fig. 8. Semantic segmentation examples for frames from **Seq 1** and **Seq 2** sequences and comparison to corresponding instance segmentation outputs: (a) raw color image; (b) segmentation output from [3] and (c) mask for "dynamic" object classes (vehicle, bycicle, pedestrian); (d) instance segmentation result [6] for the same frame. Smaller far vehicles are almost always segmented into one big blob with hairy boundaries by [3], while individual cars are picked up by [6] used in the proposed approach which allows to better reason about occlusions for AR.

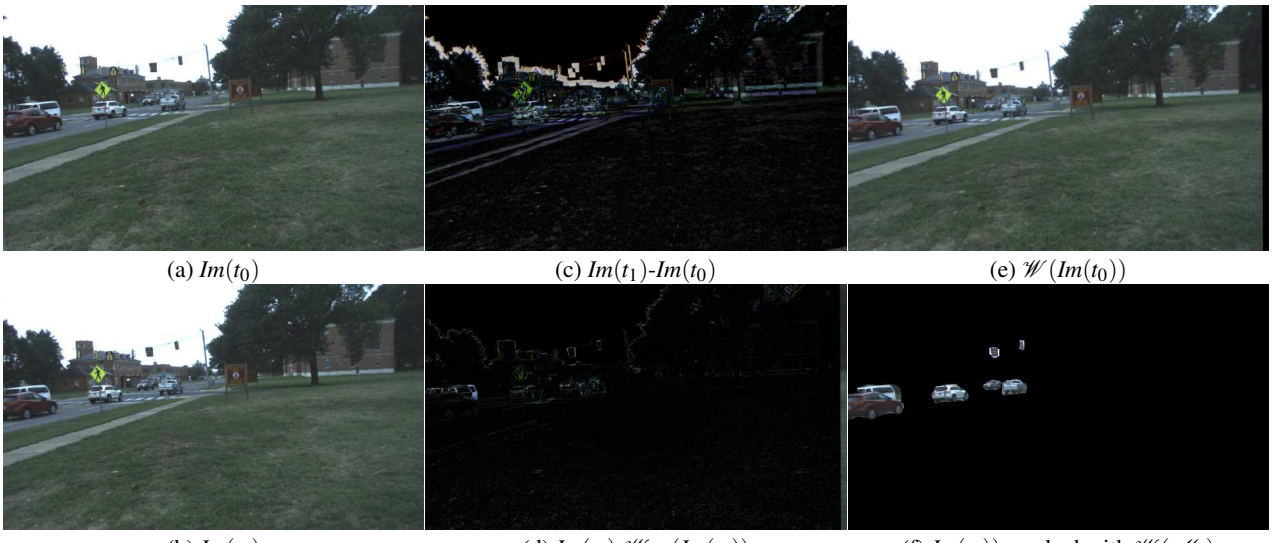


(a) DfT example and geometry (b) Depth error vs. depth (c) Depth error vs. depth (log scale)

Fig. 9. Error analysis for Depth-from-Terrain (DfT) technique. (a) Example input for depth analysis and geometric sketch of the "rectified" depth-from-terrain scenario – color frame which is used to find object instances is overlaid with the terrain model rendered from the AR insertion viewpoint. "Lowest" object silhouette points are marked as intersections with terrain using white stars. (b) Expected depth estimation error vs. depth (distance from the camera) for camera with $f = 900$ at height $\mathbf{h} = 1.8$ for different silhouette boundary detection pixel precisions p. (c) Same analysis as in (b) depicted in log scale to capture larger depth range D.

(a) Raw image (b) Depth-from-Terrain (c) Depth from single image

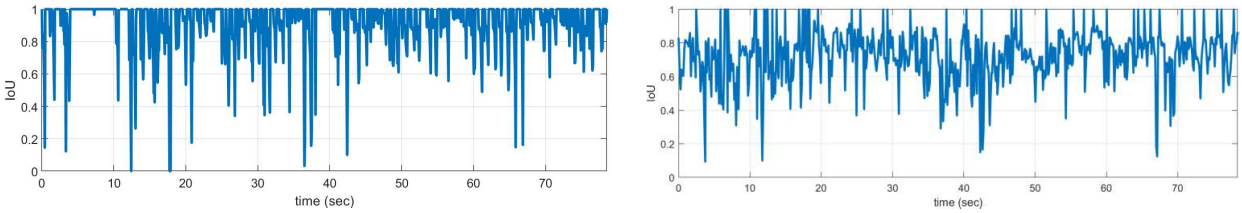
Fig. 10. Depth inference examples for depth from terrain and depth from single image methods. (a) Raw color image (b) Depth inferred by Depth-from-Terrain technique (c) Depth inferred from the color image [1] for the same input image. Depth estimates for both methods are very similar for close and middle ranges, whereas depth values for very far regions in depth from single image tend to be underestimated in comparison to the depth-from-terrain estimated depicted in (b). Further, depth from single image currently requires running another neural network which further increases the computation delay of the whole pipeline in comparison to a more efficient Depth-from-Terrain technique.



(a) $Im(t_0)$ (c) $Im(t_1)$-$Im(t_0)$ (e) $\mathscr{W}(Im(t_0))$

(b) $Im(t_1)$ (d) $Im(t_1)$-$\mathscr{W}_{pre}(Im(t_0))$ (f) $Im(t_1))$ masked with $\mathscr{W}(\mathscr{M}_0)$

Fig. 11. Example template tracking between images at different times t_0 and t_1 for **Seq 1**. The objective of *ObjectTracking*is to "predict" the object instance masks from image at time t_0 (i.e. $Im(t_0)$) in the image at a target later time t_1 (i.e. $Im(t_1)$). (a) Reference image $Im(t_0)$ at time t_0; (b) Target image $Im(t_1)$ at time t_1; (c) Initial difference between images $Im(t_1)$-$Im(t_0)$; (d) Significant portion of this image difference can be compensated by pre-warping the target image based on the delta pose $\mathsf{P}_{01} = \mathsf{P}_1\mathsf{P}_0^{-1}$, i.e. $Im(t_1)$-$\mathscr{W}_{pre}(Im(t_0))$, as explained in Sec. 2.5, which exemplifies mostly depth parallax and non-rigid parts of the scene; (e) Resulting warping of the reference image $\mathscr{W}(Im(t_0))$ toward target image $Im(t_1)$ after affine model-based template tracking procedure (f) target image $Im(t_1))$ superimposed with masks tracked from the reference frame $\mathscr{W}(\mathscr{M}_0)$. Refer to Sec. 3.3 for details.



(a) IoU between $\mathscr{M}_1$ and $\mathscr{W}(\mathscr{M}_0)$ (b) IoU between $\mathscr{M}_2$ and $\mathscr{W}(\mathscr{M}_0, t_2)$

Fig. 12. Quantitative evaluation of occluder mask prediction accuracy for dataset **Seq 2**. (a) Plot shows the intersection-over-union (IoU) measure between instance segmentation mask $\mathscr{M}_1$ computed at time t_1 and the mask tracked by *ObjectTracking* module from previous time t_0 and warped $\mathscr{W}(\mathscr{M}_0)$. (b) Plot shows the IoU measure between mask $\mathscr{M}_2$ at time t_2 and mask $\mathscr{W}(\mathscr{M}_0, t_2)$ extrapolated into time t_2 from t_0 using the affine warp parameters recovered from t_0 to t_1 tracking.

for the sensor itself we used Intel Realsense D-435i[1] with 1280x720 color camera and focal length around $f = 900$ pixels. As shown in Fig. 9b, for $p = 1$, that results in depth error of less than one meter for closer 50 m distances with growing to 3 m at 100 m distances and more than 10 m for 200 m; distance up to a 1 km are also detectable but the triangulation uncertainty of 300 m becomes quite high for reliable occlusion reasoning. This error doubles if $p = 2$. In general, higher pixel precision for ground intersection point (lower p) results in lower depth estimation error, and vice versa. Consequently, following Eqn. 5, a good option to improve depth analysis at farther distances is to take a higher vantage point (increase **h**) or use telescopic system as in [21] (increase f).

Importantly, since the scale of the object is inversely proportional to its depth, the amount of pixels will be much less for the farther object which will reduce the quality of their segmentation and eventually make some of them undetectable. This implies that real objects will have difficulty interacting with even farther synthetic objects that they suppose to occlude, which are even smaller in size. On the contrary, the most noticeable flaws for AR are introduced by *close* objects that occlude large portion of the scene and for which incorrect delineation and occlusion reasoning are the most visually striking – the proposed method works really well exactly with this kind of objects and brings a significant value to the overall AR experience.

Interestingly, some objects can be so close that their true intersection with the terrain would be below the image boundary – that makes the DfT prediction to assume the intersection point resides on the last image row and the resulting depth is overestimated (intersection point is deduced to be further on the ground). We argue this overestimation is *not* crucial because most of the intended synthetic insertions in our long-range applications are at a further distance from the user and fully visible by camera including their point of intersection with terrain – thus, such AR insertions will be properly occluded by aforementioned close real object anyway. Another solution is to use dense stereo estimation or other existing techniques (like [9, 13, 26, 27]) to directly sample the depth of such close range objects.

Another source of error is the pose estimation itself: specifically, height component of location and especially pitch component of orientation. Nowadays the accuracy of the most contemporary AR systems is quite good, including the variation of [20] that we rely on. For simplicity, the tracking error analysis could be subsumed into silhouette boundary analysis by expressing the orientation angular error in terms of pixels. For example, error of 1 pixel in images with focal length of 900 roughly corresponds to $\arctan(1/900) = 0.0637$ deg angular error, which is similar to the overall orientation error of 0.10° exhibited by *Navigation* module [22]. Importantly, typical pitch (and roll) estimation errors are much smaller than total error that is dominated by yaw component because roll and pitch angles are determined by the gravity vector and are directly observable via IMU, i.e. they do *not* drift. In the context of Eqn. 5 that would mean a small addition to the uncertainty of typical instance segmentation module p and total depth inference error would still be dominated by the instance segmentation error.

An important point to remember is that the error in pose estimation will directly result in wrong AR avatar placement irrespectively of correctness of occlusion reasoning. Even in this case, considering that we use the same terrain model for AR insertion and depth inference, the incorrect AR insertion can still be properly occluded by real object with incorrect depth as long as it appears *lower* on the terrain in comparison to synthetic one.

On the contrary, the alternative depth inference method from a single color image plus metric stereo mapping does NOT directly depend on the pose accuracy because it does not require terrain alignment. Instead, it depends on other factors like on the generality of the training dataset, image lighting conditions, etc. Also, as Fig. 10 depicts, even though we calibrate the depth scale factor using metric stereo matches, depth values for very far regions tend to be underestimated in comparison to the depth estimated from the terrain-rendered models. Finally, depth from single image requires running a neural network which further increases the computation delay of the whole pipeline in comparison to a more efficient terrain renderer.

In conclusion, the presented analysis was concerned with the absolute depth inference. In turn, proper occlusion reasoning relies on *relative* depth between occluder and occluded – that means the proposed method will perform great in case when distance between those objects is significant (i.e. close large objects in the foreground occlude small far objects in the background) and most of the challenges occur when objects are close to each other. In the latter case occluder-occluded relation may even swap across different frames and our current architecture described in Sec. 2.1 does not allow to easily disambiguate this situation because there is no feedback loop between the renderer and the *ObjectTracking* module. Proper treatment of this case would involve reasoning about 3D bounding volumes of virtual and real objects (e.g., as defined defined by uncertainty of the estimated depth) and falls outside of the scope of the current work which keeps AR rendering separate from head tracking.

3.3 Template tracking and occlusion prediction

To explicitly show the effect of the template tracking, a more detailed analysis of intermediate processing is visualized in Fig. 11 for dataset **Seq 1**. The objective of *ObjectTracking*is to "predict" the object instance masks from image at time t_0 (i.e. $Im(t_0)$, Fig. 11a) to a later time t_1 (i.e. $Im(t_1)$, Fig. 11b). Typically, the significant part of image motion $Im(t_1)$-$Im(t_0)$ (Fig. 11c) can be compensated by warping based on delta pose $\mathsf{P}_{01} = \mathsf{P}_1 {\mathsf{P}_0}^{-1}$, as explained in Sec. 2.5 and visualized in Fig. 11d. Ideally, the resulting warped $\mathscr{W}(Im(t_0))$ should be very similar to the target image $Im(t_1)$ except possible ghosting effect near image boundaries of the moving objects, as near the cars in Fig. 11e. The resulting warping of the mask $\mathscr{W}(\mathscr{M}_0)$ from time t_0 should be in very good alignment with the target image t_1, as visualized in Fig. 11f.

Furthermore, it is possible evaluate the accuracy of template tracking (1) by comparing the warped object detection mask $\mathscr{W}(\mathscr{M}_0)$ at time t_0 with the analogous mask $\mathscr{M}_1$ computed offline for time t_1. We use a well-known intersection-over-union (IoU) measure [6] which must be 1 when $\mathscr{W}(\mathscr{M}_0)$ is identical $\mathscr{M}_1$. Figure 12a shows the time profile for IoU measure for **Seq 2** (it predominantly consists of moving cars) and generally results in high values. Note that we cannot expect this value to be always 1 due to uncorrelated object boundary delineation errors introduced in $\mathscr{M}_0$ and $\mathscr{M}_1$. Also, occasionally IoU measurement drops noticeably because of novel objects are detected at time t_1 that were

[1] https://www.intelrealsense.com/depth-camera-d435i/

not yet present at time t_0.

Similar evaluation has been conducted to analyze quality of the predicted frame warping (2) to future frame time t_2 from estimated warp $\mathscr{W}(\mathscr{M}_0, t_2)$. Figure 12b shows similar IoU measure for the same **Seq 2** dataset comparing $\mathscr{M}_2$ to $\mathscr{W}(\mathscr{M}_0, t_2)$ and we can notice that predicted mask are of slightly inferior quality but still will be adequate to provide occlusion experience in OST setups.

4 Discussion

We presented an extended Augmented Reality pipeline capable to perform real-time occlusion reasoning in the long-range large-scale outdoor scenarios. Our solution is based on semantic object detection and depth inference from terrain model rendering. Furthermore, we derived the strategy to mask the latency introduced by object detection and depth rendering by introducing a low-latency module that explicitly adjusts the occlusion mask to the latest augmented image for video-see-through mode and extrapolates to future time frame for optical see-through. The proposed solution was realized on a commercial backpack computer to evaluate the applicability of the solution to outdoor AR with occlusion reasoning.

Despite its efficacy and practicality, the proposed technique has certain limitations. The assumption of observing terrain point intersection is reasonable but lacks robustness in situations when terrain is bumpy with a lot of self-occlusions. The *ObjectTracking* module performs affine matching and that can fail for significantly deformable or articulated objects (like close-up moving pedestrians), but more specific human tracking techniques can be employed in this case. The instance segmentation is not very reliable when objects are only partially visible in the field of view of the camera which occasionally results in wrong occlusion reasoning near image borders.

The future work will address the listed-above shortcomings in the systematic fashion. Specifically, next steps concentrate on further increasing robustness and versatility of occlusion reasoning by improving both object detection and depth inference components. Specifically, we intend to alleviate planar object assumption by employing recent advances of deducing full 3D pose from single image, which should result in better occlusion reasoning for objects that are very close to each other. Also, we plan to increase the robustness of the underlying depth estimation technique by combining the deep learning depth-from-monocular-image techniques with our terrain model viewpoint renderer to be more resilient to errors in pose estimation (and possible alignment errors between image and rendered terrain model) as well as get better quality depth estimate for static detailed structures not captured by the terrain model. Finally, we also consider to organizing a feedback loop between *ObjectTracking* and *DepthInference* modules to produce more temporally-consistent depth maps, occlusion masks and ultimately improve occlusion reasoning for real and synthethic objects that are close to each other.

Acknowledgements

The work for this program was conducted with the support of the Office of Naval Research (ONR) Contracts N00014-19-C-2025. We thank ONR for their support. The views, opinions, and findings contained in this report are those of the authors and should not be construed as an official position, policy, or decision of the United States government.

References

[1] I. Alhashim and P. Wonka. High quality monocular depth estimation via transfer learning. *arXiv e-prints*, abs/1812.11941, 2018.

[2] R. Azuma, B. Hoff, H. Neely, and R. Sarfaty. A motion-stabilized outdoor augmented reality system. In *IEEE Virtual Reality*, 1999.

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *TPAMI*, 39(12):2481–2495, 2017.

[4] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *IJCV*, 56:221–255, 2004.

[5] M. O. Berger. Resolving occlusion in augmented reality: A contour based approach without 3D reconstruction. In *CVPR*, page 91–98, 1997.

[6] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee. YOLACT: real-time instance segmentation. In *ICCV*, pages 9157–9166, 2019.

[7] P. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 3:532–601, 1983.

[8] H.-P. Chiu, V. Murali, R. Villamil, G. D. Kessler, S. Samarasekera, and R. Kumar. Augmented reality driving using semantic geo-registration. In *VR*, pages 423–430, 2018.

[9] P. Fortin and P. Hebert. Handling occlusions in real-time augmented reality: Dealing with movable real and virtual objects. In *CRV*, 2006.

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[11] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision. Second Edition.* Cambridge University Press, Cambridge, UK, 2004.

[12] K. He, G. Gkioxari, and P. D. R. Girshick. Mask R-CNN. In *ICCV*, pages 2961–2969, 2017.

[13] A. Holynski. Fast depth densification for occlusion-aware augmented reality. *AMC Transactions on Graphics (Proc. SIGGRAPH Asia)*, 37(6):2481–2495, 2018.

[14] M. Kanbara, T. Okuma, H. Takemura, and N. Yokoya. A sterescopic video see-through augmented reality system based on real-time visionbased registration. In *ISMAR*, page 255–262, 2000.

[15] K. Kiyokawa, Y. Kurata, and H. Ohno. An optical see-through display for mutual occlusion with a real-time stereovision system. *Computers and Graphics*, 25(5):765–779, 2001.

[16] L. Li, T. Guan, and B. Ren. Resolving occlusion between virtual and real scenes for augmented reality applications. In *LNCS 4551*, 2007.

[17] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, pages 740–755, 2014.

[18] F. Ma, G. V. Cavalheiro, and S. Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from LiDAR and monocular camera. In *ICRA*, 2019.

[19] A. Mourikis and S. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *IEEE ICRA*, 2007.

[20] T. Oskiper, S. Samarasekera, and R. Kumar. Multi-sensor navigation algorithm using monocular camera, IMU and GPS for large scale augmented reality. In *IEEE ISMAR*, 2012.

[21] T. Oskiper, M. Sizintsev, V. Branzoi, S. Samarasekera, and R. Kumar. Augmented reality binoculars. In *IEEE Transactions of Visualization and Computer Graphics*, volume 21, pages 611–623, 2015.

[22] T. Oskiper, M. Sizintsev, V. Branzoi, S. Samarasekera, and R. Kumar. Augmented reality scout: Joint unaided-eye and telescopic-zoom system for immersive team training. In *IEEE ISMAR*, 2015.

[23] J. Qiu, Z. Cui, yinda zhang, X. Zhang, S. Liu, B. Zeng, and M. Pollefeys. DeepLiDAR: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. In *CVPR*, 2019.

[24] J. Schmidt, H. Niemann, and S. Vogt. Dense disparity maps in realtime with an application to augmented reality. In *WACV*, page 225–230, 2002.

[25] M. M. Shah, H. Arshad, and R. Sulaiman. Occlusion in augmented reality. In *8th International Conference on Information Science and Digital Content Technology (ICIDT)*, 2012.

[26] M. Sizintsev, S. Kuthirummal, S. Samarasekera, R. Kumar, and H. S. Sawhney. Gpu accelerated realtime stereo for augmented reality. In *3DPVT*, 2010.

[27] Y. Tian, T. Guan, and C. Wang. Real-time occlusion handling in augmented reality based on an object tracking approach. *Sensors*, 10(4):2885–2900, 2010.

[28] W. van Gansbeke, D. Neven, B. D. Brabandere, and L. V. Gool. Sparse and noisy lidar completion with rgb guidance and uncertainty. In *MVA*, pages 1–6, 05 2019.

[29] J. Ventura and T. Hollerer. Depth compositing for augmented reality. In *ACM SIGGRAPH posters*, 2008.

[30] M. M. Wloka and B. G. Anderson. Resolving occlusion in augmented reality. In *I3D*, page 5–12, 1995.

[31] Y. Zhang, J. Pettré, J. Ondrej, X. Qin, Q. Peng, and S. Donikian. Online inserting virtual characters into dynamic video scenes. *Computer Animation and Virtual Worlds*, 22(6):499–510, 2011.

[32] J. Zhu and Z. Pan. Occlusion registration in video-based augmented reality. In *Virtual Reality Continuum And Its Applications*, 2008.

[33] Z. Zhu, V. Branzoi, M. Sizintsev, N. Vitovitch, T. Oskiper, R. Villamil, A. Chaudhry, S. Samarasekera, and R. Kumar. Ar-weapon: Live augmented reality based first-person shooting system. In *WACV*, pages 618–625, 2015.