Automatic object annotation in streamed and remotely explored large 3D reconstructions

Benjamin Höller¹, Annette Mossel¹, and Hannes Kaufmann¹ (✉)

© The Author(s) 2020.

Abstract We introduce a novel framework for 3D scene reconstruction with simultaneous object annotation, using a pre-trained 2D convolutional neural network (CNN), incremental data streaming, and remote exploration, with a virtual reality setup. It enables versatile integration of any 2D box detection or segmentation network. We integrate new approaches to (i) asynchronously perform dense 3D-reconstruction and object annotation at interactive frame rates, (ii) efficiently optimize CNN results in terms of object prediction and spatial accuracy, and (iii) generate computationally-efficient colliders in large triangulated 3D-reconstructions at run-time for 3D scene interaction. Our method is novel in combining CNNs with long and varying inference time with live 3D-reconstruction from RGB-D camera input. We further propose a lightweight data structure to store the 3D-reconstruction data and object annotations to enable fast incremental data transmission for real-time exploration with a remote client, which has not been presented before. Our framework achieves update rates of 22 fps (SSD Mobile Net) and 19 fps (Mask RCNN) for indoor environments up to 800 m³. We evaluated the accuracy of 3D-object detection. Our work provides a versatile foundation for semantic scene understanding of large streamed 3D-reconstructions, while being independent from the CNN's processing time. Source code is available for non-commercial use.

Keywords dense 3D reconstruction; object detection; CNN; distributed virtual reality

1 Introduction

1.1 Background

Combining low-cost RGB-D depth imaging sensors with algorithms for large-scale dense 3D scene reconstruction has been broadly studied in the past. Recent approaches have demonstrated remote exploration of a 3D scene while reconstruction is still ongoing (allowing exploration of the live reconstruction) using immersive virtual reality (VR) input and output devices [1]. It has been shown that remote exploration of the live scene is beneficial in terms of cost, speed, and safety. Combining immersive VR technology, value is added with regard to natural viewing and improved spatial scene understanding [2]. These properties are vital to numerous real-world use cases, such as rescue operations and remotely guided inspection.

1.2 Motivation

However, state-of-the-art approaches for large 3D reconstructions lack the ability to provide 3D object annotations at run-time, and further do not provide solutions to remotely explore the annotated 3D reconstruction.

Object annotation and classification are key to being able to automatically guide users to points of interest in a VR visualization, as well as to deriving an understanding of scene semantics. Such semantic scene understanding is of great importance in a disaster scenario, for instance, to determine not only the position of each casualty, but also their spatial distribution across the site, to plan the recovery in a well-informed manner. In the absence of automatic semantic scene understanding, the viewer needs to manually explore the entire 3D reconstruction to find points of interest, such as hazardous areas or

¹ Institute of Visual Computing and Human-Centered Technology, Vienna University of Technology, Favoritenstraße 9-11/193/06, A-1040 Vienna, Austria. E-mail: B. Höller, benjamin.hoeller@alumni.tuwien.ac.at; A. Mossel, annette.mossel@tuwien.ac.at; H. Kaufmann, hannes.kaufmann@tuwien.ac.at (✉).

Manuscript received: 2020-08-31; accepted: 2020-09-06

casualties. This results in a time consuming process.

Further, lack of automated understanding of which parts of the 3D reconstruction belong to individual scene objects leads to the usage of mesh colliders for immersive exploration. These are computationally expensive and thus easily lead to low update rates, and thus poor, error-prone immersive exploration. By exploiting annotation information to enable efficient computation of non-complex colliders at run-time, high update rates for live immersive exploration can be achieved, yielding a high degree of presence.

1.3 Contributions

Our proposed approach for semantic scene understanding in live captured and remotely explored dense 3D reconstructions asynchronously integrates the results of a pre-trained state-of-the-art convolutional neural network (CNN) with a simultaneous localization and mapping procedure (SLAM). Thereby, we achieve object annotations in large 3D reconstructions that are efficiently streamed with incremental lossless compression to enable remote exploration using immersive virtual reality.

Our approach, which partly builds upon prior art, is depicted in Fig. 1, and makes the following contributions:

1. Simultaneous computation of dense 3D reconstruction, processing live RGB-D camera input, and annotation of 3D scene objects by asynchronously integrating CNN results into the dense reconstruction process. Asynchronous computation of 3D reconstruction and object detection, both with respect to time and resources, avoids degradation of the reconstructed scene. Our approach achieves interactive frame rates and overcomes limitations of prior art [3] as it is independent of the network's

inferencing time and thus can use shallow to very deep CNNs.

2. A novel filter pipeline that efficiently optimizes the CNN results for object prediction and spatial accuracy, and a novel voting algorithm that fuses the 3D reconstruction data with the object annotations using a lightweight data structure. This makes our approach independent of networks requiring complex training, as it applies temporal and spatial filters to the standard detection output of a state-of-the-art network. In addition, our data structure enables incremental data transmission for remote exploration of the annotated 3D reconstruction in real time, which has not been presented before.
3. A universal communication interface that enables the versatile integration of any 2D box detection or segmentation network.
4. A distributed software framework that provides server-side 3D reconstruction and simultaneous object detection, network-based transmission of the annotated volumetric data as voxel representations to the client, automatic computation of cost-efficient colliders, and live exploration of the annotated 3D reconstruction using immersive virtual reality.

2 Related work

Our proposed approach incorporates methodologies from different research fields, in particular (i) object detection and semantic segmentation using deep neural networks, and (ii) dense 3D scene reconstruction using SLAM. Both fields have been extensively researched, so the following section only outlines the work most relevant to our paper. However, it also considers current research which combines both fields while providing real-time 3D scene reconstruction and accurate 3D object annotation.

2.1 3D scene reconstruction with distributed real-time exploration

Combining RGB-D depth imaging sensors, based on LIDAR (light detection and ranging), structured light or ToF (time of flight), with algorithms for large-scale dense 3D scene reconstruction is now widely available, and has been broadly studied in the past. In addition, several approaches [4–6] have

Fig. 1 Proposed framework: On the server side a SLAM implementation generates a dense 3D surface reconstruction in real time. A 2D CNN performs asynchronous object detection to annotate the 3D data which is incrementally streamed to the client, who can immersive explore and interact with it using virtual reality.

investigated streaming 3D surface reconstructions, aiming to improve spatial and temporal compression by employing different point cloud data structures as well as compression algorithms. In recent years, streaming of 3D reconstructions has been extended to enable the remote exploration of a 3D scene while the live reconstruction process is still ongoing, using immersive VR input and output devices [1]. However, work in this field lacks capabilities for object detection needed to provide a semantic scene at run-time as well as to efficiently compute colliders at run-time to allow object interaction in virtual reality.

2.2 Object annotation with neural networks

The latest achievements in the field of neural networks enable fast and efficient object detection and semantic annotation. Convolutional neural networks (CNNs) are used to process spatial data, such as images. Most network structures [7–9] use a two-component system that separately performs classification and localization tasks. Some network architectures, such as YOLO V2 [10] and SSD [11], aim to solve this problem more efficiently in one step. So called fully convolutional networks, like Mask RCNN [12], use deconvolution to provide a pixel wise segmentation mask for each detected object. Due to their outstanding performance, the later two methods are used in our proposed approach as well as in most related research.

2.3 Object detection in 3D surface reconstructions

3D scene reconstruction and semantic object detection have been intensively investigated in the fields of computer vision and machine learning. In the following, we outline recent work that fuses these two research strands. We examine the approaches especially with regard to our objectives, enabling dynamic network inferencing time as well as live streaming and exploration of the annotated 3D data.

Sünderhauf et al. [3] combine ORB-SLAM2 with SSD [11]. They achieve precise detection of object boundaries by performing geometric 3D segmentation of the surface and matching the bounding boxes from the CNN with these segments. The segmented objects are stored as separate point clouds in a scene graph. This yields a complex data structure that permits straightforward streaming of the annotated point cloud data. They cannot cope with live camera input

due to the tight interlock between SLAM and the CNN, which only processes key frames.

In contrast, SemanticFusion by McCormac et al. [13] achieves object detection at interactive frame rates. Their work combines ElasticFusion SLAM [14] with a custom segmentation network that further considers the depth channel for object recognition. By interpreting every 10th frame this system achieves object detection at 25 fps. The class probability distribution is saved for each voxel and is updated via a Bayesian scheme, resulting in a high memory footprint. This approach does not provide streaming to distribute the 3D reconstruction for live exploration in VR.

The work of Runz et al. [15] presents MaskFusion, which combines RGB-D SLAM with Mask RCNN to detect, classify, and reconstruct moving scene objects. Object segmentation at pixel level is combined with a geometric segmentation method based on depth discontinuities. The presented system supports tracking and classification of multiple moving objects at 30 fps while interpreting every 12th frame with the CNN. No streaming of the 3D reconstruction is provided. Their work focuses on live reconstruction of a dynamic scene and thus does not provide large scale 3D scene reconstruction.

Fusion++ [16] follows a similar classification approach to Ref. [15] but focuses on 3D reconstruction and object detection in static room scenes. Therefore, the SLAM algorithm automatically segments the reconstructed 3D surface at object level, representing each scene object as a separate TSDF which is stored in a pose graph. This allows for different TSDF resolutions per object but also provides a complex data structure suitable for streaming. The classified 2D segments of the Mask RCNN are matched to the scene objects and the class probability distribution is saved per voxel, as in Ref. [13]. Instead of a Bayesian update scheme, the distributions are averaged to achieve a more even class probability distribution. The approach achieves an update rate of 4–8 fps and only performs detection in every 30th frame. It cannot process live camera input because the SLAM process is paused until the CNN delivers the detection result. This lack of asynchronicity in frame processing also prevents the usage of CNNs with long inference time.

Nakajima and Saito [17] use InfiniTAM v3 [18] for 3D scene reconstruction and YOLO V2 [10] for object

detection. During 3D reconstruction, the surface is incrementally segmented into geometric patches [19]. The class probability prediction of the bounding boxes is assigned to these patches, which results in precise object boundaries and a low memory footprint. After completion of the 3D reconstruction, object annotations are optimized by fusing geometrically adjacent patches with the same object class. Due to the short inference time of YOLO V2 and the efficient incremental segmentation method, the approach can process every frame while providing an update rate of 27 fps. However, its internal data processing architecture depends on CNNs with short inference time. In addition, the post-processing optimization results in complex streaming logic and high data transmission requirements.

Wang et al. [20] use locally convex connected patches (LCCP) for segmentation. Instead of segmenting the whole point cloud, as in Ref. [17], segmentation is performed on point cloud segments identified by Mask RCNN. After isolating the objects from the background, an accurate 3D object bounding box is estimated based on a Manhattan frame system. The objects are then fused and managed in an object database. Upon completion of the 3D reconstruction and detection, unlikely objects are removed from the database, based on prior knowledge of their actual size and volume ratios. However, they cannot cope with live camera input since processing a key frame requires about 5 ms.

Hou et al. [21] propose the combination of a 2D and a 3D CNN to achieve detailed object annotations. They use back-projected 2D CNN segmentation masks in combination with reconstructed 3D geometry as input to a 3D CNN that fuses and refines per voxel mask predictions. To perform fusion from multiple points of view, their network uses all RGB images of the pre-captured environment, so this approach is not capable of live reconstruction.

3 Methodology

While prior methods present promising approaches for precise object annotation in 3D reconstructions using live camera input, all of them lack capabilities for fast and efficient streaming of the annotated 3D data structure to ensure remote 3D scene exploration with short delays. Furthermore, none supports real-time 3D object annotation in large 3D scene

reconstructions independently of the inferencing time of the neural network used. In the following, we present our methodological approach to overcome both limitations.

We propose a framework for 3D reconstruction with object annotation, streaming, and remote immersive exploration that utilizes 2D CNNs to compute object annotations in large 3D scene reconstructions at run-time. Our approach for 3D reconstruction, streaming, and remote exploration builds upon [1]. As depicted in Fig. 2, the framework consists of two main modules. On the server side, live RGB-D camera input is simultaneously fed into decoupled pipelines for 3D reconstruction and object annotation using a CNN. For dense 3D surface reconstruction, the SLAM implementation of InfiniTAM 2 [22] is used which represents the 3D surface as a truncated signed distance function (TSDF) saved within a hashed voxel block data structure [23]. At the same time, the integrated CNN takes the camera input and performs semantic object annotation, using either an object detection network that outputs bounding boxes, or a semantic segmentation network that provides a pixel-wise segmentation mask per object. The network results are processed with our novel *filter pipeline* and then projected into the 3D reconstruction to label the voxels with the object class. Our *voting algorithm* resolves ambiguities in voxel classification that occur when voxels are differently annotated from multiple viewpoints. The 3D fusion pipeline significantly extends the work of Ref. [1] and allows efficient asynchronous processing of camera frames by decoupling the procedures for scene reconstruction and object detection. Thereby, our approach is independent of the inferencing time of the network employed and so even allows usage of very deep networks with long inference time.

Fig. 2 Main modules of our framework. On the server, an RGB-D camera stream is used for simultaneous 3D surface reconstruction and object annotation. The universal communication interface (UCI) is used to communicate with any CNN implementation. The annotated reconstruction result is incrementally transmitted to the client. There, the voxel block data structure is triangulated and efficient colliders are computed based on object annotations. The annotated mesh is then explored in virtual reality.

Next, the labeled voxel blocks are compressed and incrementally transmitted to the client, where the data is triangulated to mesh blocks that incorporate the semantic annotations. The annotated 3D mesh can be immersively explored with a VR setup. In addition, our approach automatically generates computationally efficient colliders that represent the coarse geometry of the annotated 3D scene objects. This enables on-the-fly interaction with the 3D scene objects while providing VR exploration at high update rates.

3.1 CNN integration via a universal interface

The usage of any 2D state-of-the-art CNN that has been pre-trained for object detection or semantic segmentation is enabled via a universal communication interface (UCI). It manages the communication between the framework and the CNN module, and provides the camera's RGB-D image as input to the CNN. The CNN module processes the camera frames by solving the classification and localization tasks. For classification, the network computes a *label-ID* and the corresponding *probability*; localization predicts a *bounding box* if an object detection network is used, or a *pixel-wise segmentation mask* when using a semantic segmentation network. This raw network output is then further optimized and transformed by our novel filter pipeline. This UCI design enables the implementation of the CNN module as a network-based micro service and allows the usage of a variety of different neural network types, architectures, and machine learning frameworks. Furthermore, it enables straightforward and quick network replacement to use pre-trained 2D CNNs from any domain, making our proposed framework highly versatile.

3.2 Projecting network results onto the 3D reconstruction

To integrate the network output into the voxel block data structure, the raw CNN results first need to be transformed. Therefore, our filter pipeline optimizes the result and transforms it into two bitmaps, the *label-ID bitmap* and a *probability bitmap*, which are then projected onto the reconstructed 3D surface. To provide a lightweight data structure for the annotated 3D surface, our voting algorithm incrementally fuses multiple labels from different viewpoints to determine a discrete *label-ID* for each voxel.

3.2.1 Filter pipeline

To optimize the CNN results in terms of object prediction and spatial accuracy of the object detection, we have developed the filter pipeline depicted in Fig. 3. The filter pipeline applies our own numerical and visual filters to the raw network output, making our approach independent of complex trained networks. The object detection results of simple image-based CNNs (that do not consider time) are improved by filtering them in temporal and spatial relation to each other. First the numerical filters determine whether a semantic annotation is plausible and next, visual filters modify the geometric representation of the bounding box or segmentation mask. The pipeline's thresholds have been empirically determined, as described in detail in Section 4. Our filter pipeline outputs two bitmaps which annotate pixels with label-ID and probability; examples are depicted in Fig. 4.

Numerical filters. First, the *probability filter* discards neural network results with insufficient class probability. Next, our novel *persistence filter* determines the plausibility $p(O)$ of an object annotation O , in a temporal dimension. By buffering the recent

Fig. 3 Our filter pipeline generates the *label-ID bitmap* and *probability bitmap* from the CNN results. Numerical filters determine the plausibility of a semantic annotation. Visual filters modify the geometric representation of the bounding box or segmentation mask.

Fig. 4 Left: bounding boxes (result of SSD Mobile Net) enclosing detected objects are input to the filter pipeline. Center: the resulting label-ID bitmap represents the class of each object. The shape of the bounding box has been optimized. Right: the probability bitmap describes the local probability of the detection. Note how the value decreases towards the outside.

history of network results $[B]$, a threshold t can be applied to discard objects that did not appear a sufficient number of times $n(O) = \text{number of } O \text{ in } [B]$ as described in Eq. (1).

$$p(O) = \begin{cases} 0, & n(O) < t \\ 1, & n(O) > t \end{cases} \quad (1)$$

This temporal awareness drastically reduces the number of false-positive detections, as further described in detail in Section 5.1.3.

Finally, the *IOU filter* calculates the 2D intersection over union between all detection results of one frame to analyse their spatial relations. Bounding boxes are an inherent result of object detection networks. Some segmentation networks also provide the bounding boxes apart from the segmentation mask (i.e., Mask RCNN). If they do not, the bounding boxes can easily be computed by determining convex hulls in the segmentation mask. When the IOU of two bounding boxes exceeds the threshold, the IOU filter discards the object detection with lower class probability.

Visual filters. As second step, visual filters reduce and modify the shape of the numerically filtered network result to optimize the spatial coverage of an object. Our pipeline applies the visual filters differently for results from object detection networks and segmentation networks. For object detection networks, our filter pipeline processes the resulting *bounding boxes*. A bounding box usually includes areas other than the actual detected object, such as part of other objects or background. To filter these irrelevant areas, the filter pipeline modifies the shape of the bounding box. First, a *margin* is applied that reduces the bounding box size by a pre-defined fraction. If $\text{margin} = 0$ the box does not change; with $\text{margin} = 0.5$ the area is reduced to zero. *Corner cutoff* further reduces the area of the bounding box to approximate an octagonal shape. This filter is applied to the area already reduced by the margin. Shrinking the bounding box's area guides the class prediction to cover the object's center, thereby achieving a significant reduction in falsely classified voxels, as described in detail in Section 5.1.1. Note that excluded object areas may still be labeled in the final reconstruction by accumulation of several recognitions from widely diverging viewpoints. Finally, the *center weight* filter weights the probability towards the object's center by decreasing the probability towards the edges, as

shown in Fig. 4 (image to the right). The probability value in the center equals the initial class probability of the CNN. The value is then decreased relative to the square of the Manhattan distance to the center until it reaches zero at the corner.

For semantic segmentation networks, our filter pipeline processes the resulting *segmentation mask*. First, the margin filter is applied, and then the area of the segmentation is reduced with an *erosion* operation. This is beneficial to avoid falsely classified pixels at the object's edges. Instead of applying the filters for corner cutoff and center weight, the shape of the mask is used for subsequent processing. Thus, the probability does not decrease towards the edges.

In a final step, our filter pipeline incorporates the *depth information* from the camera frame into both network result types. Therefore, a histogram of the depth image is calculated within the reduced bounding box region (including the margin). Only pixels within a defined range around the peak of the histogram are drawn. This filters near and far regions in relation to the camera position, further reducing the number of falsely classified voxels and improving the spatial accuracy. The effect of the depth filter can be seen in Fig. 4: the detected plant is accurately segmented from the background.

3.2.2 Projection of optimized network results

After the filter pipeline has computed the label-ID bitmap and the probability bitmap, they are projected onto the reconstructed 3D surface. To allow real-time capability, this step is efficiently performed on a GPU using CUDA [24]. For projection, we modified the algorithm of Ref. [25] that proposed a color fusion approach during surface reconstruction. While color is a continuous value and can be averaged over time, the label-ID is a discrete value and thus needs special integration logic. We solve this with our novel voting algorithm, as described in Section 3.2.3. The bitmaps need to be projected from their original point of view. Thus, the camera transformation as well as the depth image need to be buffered to allow asynchronous integration of (slow) network results, as described in Section 3.2.4.

3.2.3 Voting algorithm

A single voxel may have numerous classifications due to different detection results in multiple camera views. To handle the fusion of these results, we developed a novel voting algorithm that determines a

discrete *label-ID* for each voxel, saved as $v(ID)$, and a corresponding *probability*, stored as $v(P)$. This results in a lightweight annotation structure, compared to the complex structures of prior art [13, 16] that save the entire class probability distribution for each voxel. Our voting algorithm fuses multiple detection results by incrementally incorporating results into each voxel's local probability. Therefore, it applies our defined voting rules as follows. As soon as a voxel receives a classification, its annotation data is modified. If the voxel has not yet been classified so that $v(ID) = 0$, it is assigned the received annotation result, $v(ID) = v_{ID}^r$, $v(P) = v_P^r$. If a classified voxel receives a new annotation result from the projection, the voxel's annotation data is modified according to Eq. (2)

$$v(P) = \begin{cases} \min(\max(v_P^r + s, v_P^{r+1}), 1), & v_{ID}^r = v_{ID}^{r+1} \\ v_P^r - s, & v_{ID}^r \neq v_{ID}^{r+1} \end{cases} \quad (2)$$

where s denotes the step size, empirically set to $s = 2.55\%$. If $v_P^{r+1} - s \leq 0$, $v(ID)$ and $v(P)$ are reset to zero. Thus, the voxel's probability $v(P)$ does not represent the certainty of the neural network result but is defined as the accumulated probability over time, determined by the voting algorithm. This improves the demarcation of object boundaries when multiple different label-IDs are projected onto the same area of the 3D reconstruction.

3.2.4 Object annotation in live 3D reconstruction

Object detection or semantic segmentation with a CNN is time intensive: depending on the depth and capabilities of the CNN, the computation may even take several seconds. Performing SLAM with live camera input requires real-time integration of color and depth information into the 3D reconstruction. To combine this with object annotations at run-time, we asynchronously integrate the CNN results into the 3D reconstruction process. In the existing pipeline of Ref. [1], all voxel blocks that are within the current view frustum are swapped into GPU memory for efficient computation. Voxel blocks that fall outside the view frustum are swapped out of the GPU to CPU memory. However, this strategy makes it impossible to integrate delayed CNN results into the 3D reconstruction.

Thus, we extended the pipeline in two ways.

1. For each frame processed by the CNN, the corresponding camera's projection matrix is stored

to enable correct projection of asynchronous CNN results into the 3D reconstruction at a later point in time. The frame's depth channel is saved to be further processed by our filter pipeline.

2. All voxel blocks belonging to the camera frame currently being processed by the CNN are buffered in GPU memory until the CNN result has been projected. We thus modified the existing memory swapping algorithm to be able to separately perform swapping in (CPU to GPU) and swapping out (GPU to CPU). After integrating the CNN result of this buffered frame, all voxel blocks outside the view frustum of the current frame are swapped out to the CPU memory.

Our extended 3D fusion pipeline is depicted in Fig. 5. The first five steps are derived from the original InfiniTAM 3D-reconstruction as implemented in Ref. [1]. The unmodified *tracking stage* calculates the current camera transformation. In contrast to the original implementation, our modified *allocation stage* does not mark voxel blocks outside the view frustum as *not visible*. This prevents the swapping engine from moving those blocks to CPU RAM. New blocks within the view frustum are initialized on the GPU, while existing blocks in CPU memory are marked for swap-in. The *integration stage* updates the TSDF and color values of voxels by integrating the RGB and depth information from the current camera frame. Our modified swapping

Fig. 5 Our extended 3D fusion pipeline integrates asynchronous CNN results into the live 3D reconstruction.

stage performs a *swap in* of previously marked voxel blocks. *Raycasting* renders the next frame's depth image, which is used for camera pose estimation. The *CNN feed stage* sends the current RGB-D image to the CNN via the UCI. If the image is accepted for interpretation, the *fill buffer* saves the current camera's projection matrix and depth image. The CNN processes the current RGB image while 3D reconstruction continues to integrate all upcoming frames. Once the CNN results are available, the filter pipeline computes the bitmaps for label-ID and probability, and the *load buffer stage* restores the saved projection matrix and depth information, and buffers the values of the current frame. During *NN integration* the bitmaps for label-ID and probability are projected onto the 3D reconstruction from their original viewpoint by applying the restored projection matrix. During projection, the voting algorithm incrementally determines each voxel's label-ID and probability. The *swap buffer* restores the projection matrix (for the current live camera pose) and depth image that were changed in the load buffer stage. *Allocation* marks all voxel blocks outside the current view frustum as not visible. Finally, the *swap out stage* swaps voxel blocks marked as not visible from the GPU into CPU RAM. Figure 6 provides a visual example of the modified memory swapping strategy.

3.3 Incremental streaming and meshing

After the neural network results have been integrated into the voxel block data structure, existing modules from Ref. [1] are used to compress the semantically

annotated voxel blocks with a lossless DEFLATE compression algorithm [26] and incrementally stream them to the client over TCP/IP. A slow network connection does not influence the reconstruction and annotation result since all voxel blocks are buffered on the server. The only noticeable effect are less frequent mesh updates in the VR visualization.

On the client, the transmitted voxel blocks are triangulate into dense mesh blocks using marching cubes [27]. We extended the Marching Cubes algorithm to additionally extract the label-ID for each vertex to use the semantic annotations in 3D. As in Ref. [1], the annotated mesh is transmitted from the client application to a 3D game engine (Unreal) via shared memory. This allows immersive exploration and interaction with detected objects in virtual reality.

3.4 Automatic collider generation

To exploit all the benefits of VR for exploration of the triangulated 3D reconstruction, while providing a high level of presence, the scene objects require physical properties (*colliders*) to enable interaction as well as to prevent the user from walking through objects, for example. However, colliders that directly use the mesh information are computational expensive, especially in large 3D reconstructions. Thus, we exploit the semantic annotations of the vertices to automatically generate box colliders within our incrementally triangulated 3D reconstruction at run-time. Thereby, we enable on-the-fly interaction with the annotated 3D scene objects while providing

Fig. 6 Extended allocation and GPU buffering approach. Left: frame 0: voxel blocks are generated or updated on the GPU during 3D reconstruction. Center: frame 10, pipeline of Ref. [1]: all voxel blocks that fell out of the view frustum are swapped to the CPU memory. Right: frame 10, our extended pipeline: the image of frame 0 was sent to the CNN, the current frame's camera projection matrix and depth channel were buffered, all voxel blocks of frames < 10 were kept in GPU memory until the CNN results were integrated by projection. Then all voxel blocks outside the view frustum are swapped out.

VR exploration at high update rates. The generation is performed in two steps, as depicted in Fig. 7. First, small *label-cubes* are generated enclosing the classified vertices. These cubes can grow to a size of max. 20 cm along each axis to enclose as many adjacent vertices of the same label into one cube. Second, adjacent *label-cubes* of the same label are merged to larger *object-cubes* that finally enclose the entire scene object. The amount and type of the label cubes, and thus the object cubes, are determined by the numerical filter of the server-sided *Filter Pipeline*, while the visual filters influence their spatial accuracy. Each *label-cube* is used as a box-collider, enabling fast collision detection with the coarse geometry of an annotated scene object.

3.5 Implementation

Our methodological approach was developed based on the streaming and reconstruction framework from Ref. [1], which is implemented in C++ using InfiniTAM 2 [28]. To integrate semantic object annotations, the existing data structures and framework modules (mainly the 3D fusion and 3D rendering) were fundamentally modified and extended. We implemented the CNNs using the Tensorflow 1.8.0 machine learning framework with Python 3.6.3 and cuDNN 7.0.

The UCI handles the communication between the neural network (Python) and the reconstruction pipeline (C++) and was developed with the standard Python/C API. Although the UCI design enables the implementation of the CNN as a network-based micro service, it was implemented without a network connection for this paper to avoid erroneous evaluation results due to network failure. Unreal Engine v4.19 was used for visualization and immersive exploration with a VR-setup.

Fig. 7 Visualizing annotated scene objects using the Unreal engine. Left: Label-cubes are generated enclosing classified vertices, representing their coarse geometry. Right: Object-cubes completely enclose annotated scene objects.

4 Evaluation

4.1 Discussion

To examine our proposed framework, we evaluated the processing performance of 3D reconstruction and the asynchronous integration of the network results by testing with an object detection network and a semantic segmentation network.

Furthermore, we analyzed the accuracy of the semantic object annotations computed by an object detection network. The objectives of our evaluation were two-fold:

1. The first aim was to determine optimal settings for our numerical and visual filters with regard to the combination of tested scene and employed CNN. While needing further testing, these filter settings can be used as a baseline for future research in the field.
2. Second, we aimed to deriving universal findings from our results regarding the influence of the individual filters on the detection results, to optimize the accuracy of the semantic 3D object annotations computed by other CNNs.

The filter values presented in the following were empirically obtained sequentially, since examining the entire filter parameter space is not manageable in terms of time and resources. Furthermore, we deliberately removed parameters that were not the focus of our study to avoid distortion of the results, principally a client–server connection between two physical machines. Instead, we tested our proposed approach using a client and server both running on the same machine, simulating a perfect network connection. An unstable network connection would in fact have no influence on the reconstruction and annotation results, since the voxel blocks are buffered on the server side, as explained in Section 3.3. Further we removed variations in the asynchronous integration pipeline by performing synchronous integration of network results to increase repeatability, as explained in detail in Section 4.4.

4.2 Test setup

Evaluation was performed on a Windows 10 workstation with a 4.20 GHz Intel i7 7700 CPU, 32 GB RAM and a GeForce GTX 1080 Ti GPU with 11 GB RAM. The client and server application were executed on the same workstation. Since the CNN and the scene reconstruction run on the same

GPU, Tensorflow was configured to only allocate 50% of the available CUDA cores to leave enough processing power for the 3D fusion. We used two publicly available pre-trained neural networks. As object detection network, we integrated the fast SSD Mobile Net [11, 29, 30] box detection network, while for semantic scene segmentation we integrated the fully convolutional Mask RCNN [12, 31] segmentation network. Both networks were pre-trained on the COCO data-set [32]. For our evaluation we used the *Lounge* scene *lounge.oni* [33] which provides a standard indoor scene with an average sized room containing several objects from the COCO data-set. It should be noted that the scene is biased for the persistence filter due to the large number of chairs.

4.3 Performance measures for object detection

To evaluate the quality of 3D object detection using varying filter settings, we manually created some ground truth by placing labeled *object-cubes* around the 3D scene objects. Next, we used 3D intersection over union (3D IOU) to measure the spatial overlap between the object-cubes generated by our pipeline and the object-cubes of our ground truth. This provides the spatial accuracy of each reconstructed 3D bounding box, not the detailed shape of the annotation. Employing an empirically determined 3D IOU threshold of 0.25, we categorized the reconstructed object-cubes either as false-positives (< 0.25) or true-positives (≥ 0.25). Next, we calculated *precision* as the fraction of detected objects that are part of the ground truth, and *recall* as the fraction of ground truth cubes found. To generate an overall scene score, precision and recall were combined to the F_β score, with $\beta = 0.5$.

4.4 Repeatability of scene reconstruction

The dimensions of the generated object-cubes for the evaluation scene can vary between evaluation runs because of the high amount of parallel processing (3D fusion, marching cubes) and algorithms depending on the applied sequence (voxel block streaming, marching cubes), both in 3D fusion and the meshing module. To measure variability of our test results, we ran 15 similar evaluations and compared all generated object-cubes with each other by calculating their 3D IOU. During live reconstruction, with the CNN and 3D Fusion running asynchronously, we measured a mean IOU $\mu = 89.83$ with a standard deviation

of $\sigma = 15.26$.

For our evaluation of accuracy and quality of 3D object annotation, we minimized the IOU variations by deactivating asynchronous processing of the 3D fusion pipeline, thereby forcing the CNN to process every camera frame. This increases the mean IOU to $\mu = 97.40$ with a standard deviation of $\sigma = 4.54$. This provided better comparability of results, and was therefore used for the evaluation.

5 Experimental results

Our proposed framework performs live 3D reconstruction with asynchronous object annotation at 21.7 fps using SSD Mobile Net, and 18.6 fps when using Mask RCNN. When asynchronous processing is deactivated, our approach achieves 19.3 fps with SSD Mobile Net and 18.8 fps with Mask RCNN.

5.1 Accuracy of 3D object annotation

As explained in Section 3.2.1, the filters of our filter pipeline define the content of the label-ID and probability bitmaps that are projected onto the reconstructed 3D surface. Based on our evaluation objective to determine optimal filter settings, we individually evaluated the filters to determine their influence on the accuracy of 3D object annotation. The value range and the determined optimum of these filters are listed in Table 1.

We determined the optimal filter settings in a sequential way to obtain an empirical optimum. Future work includes crossing of all filter values to analyze all correlations separately. In the following, we outline the filters with the highest influence on accuracy as well as their correlations.

Table 1 Range, unit, and optimal value for each filter. These settings were determined by empirical evaluation using the SSD Mobile Net in combination with the *Lounge* scene

Filter	Range	Optimum	Unit
Probability	0–1	0.65	
Persistence size	1– ∞	100	Frames
Persistence threshold	1–max	50	Frames
IOU threshold	0–1	1	
Margin	0–0.5	0.1	
Corner cutoff	0–1	0.5	
Center weight	0–1	1	Boolean
Depth fraction	0–1	0.2	
Erosion	0– ∞	16	Pixels

5.1.1 Correlation between margin and corner cutoff

The margin and corner cutoff filters are most correlated. They have the highest influence on the overall shape and size of the area that is projected onto the 3D surface reconstruction. The margin filter reduces the size of the bounding box, while the corner cutoff removes the corners of the box according to the Manhattan distance to the center. With a corner cutoff value of 0.75, a rectangle will be reduced to an octagon, while a value of 0.5 results in a rhombus. Our experimental results indicate a correlation between the size of the projected area and the achieved F_β score, as visualized in Fig. 8. The left column illustrates the evaluation with the 3D IOU threshold set to the standard value of 0.25 (see Section 4.3): a generated object-cube is classified as true positive if 3D IOU with a corresponding ground truth object-cube is ≥ 0.25 . In Fig. 8, two peaks can be observed. The underlying data shows that precision peaks at a corner cutoff of 0.0 and margin of 0.25, while recall peaks at a corner cutoff of 0.5 (rhombus) and margin of 0.0.

To improve the balance between precision and recall, we increased the 3D IOU threshold to 0.4 to only classify object-cubes with higher dimensional accuracy as true positives. With this increased threshold we observe that the optimal F_β score for corner cutoff is 0.5 and for margin is 0.1, as shown in Fig. 8(right).

5.1.2 Depth filter

The depth filter removes regions within the bounding box that are too near or far from the current camera position by calculating a depth histogram, as explained in Section 3.2.1, so that only pixels within a defined range around the histogram's mean are incorporated into the probability bitmap. Our

evaluation shows the best F_β score for the depth filter is at 0.2, so only pixels within 20% of the mean (40th to 60th percentile) of this histogram are taken into account for the bitmap. This narrow band helps to remove most foreground and background obstructions for an object.

5.1.3 Probability and persistence filter

Our evaluation clearly showed that the probability filter is the most influential factor in obtaining accurate 3D object annotations. The higher the probability threshold, the more certain are the predictions made by the object detection network. This obviously results in a trade-off between the total number of detections and the quality of predictions. Our evaluation shows that the best scene score was reached with a probability threshold of 65%.

Due to the continuous motion of the camera while capturing RGB-D data, detectable objects will appear over several consecutive frames. Despite extensive training, neural networks may still produce false positive detections. This can happen if an object's unique properties are not visible in the current image frame, as well as due to insufficient lighting and image noise. For temporal classification consistency, the persistence filter maintains a history of recent detection results and evaluates their numbers of occurrences. Thus, the persistence filter is defined by the size of the temporal window (persistence size) and the minimum number of occurrences in this window (persistence threshold). The configurations evaluated are listed in Table 2, while experimental results are plotted in Fig. 9.

Our experiments reveal increasing F_β scores with increasing window size, which reaches its natural limit with the average number of frames for which an object is visible within the view frustum. The effect of this

Fig. 8 Correlation between values for margin and corner cutoff filters. The heat maps show F_β for 3D IOU thresholds of 0.25 (left) and 0.4 (right).

Table 2 Evaluation settings for the persistence filter. The peak threshold is the value where F_β reached maximum. The relative threshold gives the peak threshold in relation to the window size

Window size	Peak threshold	F_β	Relative threshold
10	10	0.461538	1
20	12	0.461538	0.6
30	22	0.500000	0.73
50	35	0.439024	0.63
100	50	0.545454	0.5
200	120	0.518519	0.6

Fig. 9 Evaluation of 6 values for persistence size. The number in the legend represents the window size.

filter is therefore influenced by the movement speed of the camera.

Figure 9(top left) shows a rapid drop in F_β value after reaching the peak. When considering all evaluated window sizes, F_β reaches its peak on average at a relative threshold of 61.2%. From these two observations we define the recommended persistence threshold at 50% of the window size. During live reconstruction, the effect of the persistence filter heavily depends on the relation between the scanning speed of the camera and the inference time of the CNN. Therefore, the persistence filter is considered to be the main parameter to:

- (i) control the total number of classified scene objects,
- (ii) adjust the fraction of correctly classified objects,
- (iii) adapt to the scanning speed of the camera,
- (iv) adapt to the inferencing time of the neural network.

5.1.4 Persistence filter detail

To test our claims regarding the effect of the persistence filter, a second scene was evaluated. In the 80 m² *Flat* scene *home_small.oni* from Ref. [1], objects are not as visually delimited and the camera movement is slightly faster than in the *Lounge* scene. Again the scene was processed synchronously to

increase repeatably of the results. Due to the lack of ground truth, the result was verified manually. In this test only classification results of the object cubes were considered, not the dimensional accuracy. An object cube is counted as correct if it encloses part of the correctly labeled object. Small, visually imperceptible object cubes were ignored.

With a persistence filter setting of 25/50 the fraction of correctly classified object cubes was 23 out of 44. Using the optimal filter parameters 50/100, as described in Table 1, reconstruction provided 31 reconstructed object cubes with 20 correctly classified.

This confirms the findings regarding the overall number and proportion of correctly classified objects with increasing persistence filter size. Detailed evaluation of the effects of scanning motion patterns, camera movement speed, and neural network inference time requires further research.

5.1.5 Other filters

Our evaluation of the IOU filter did not reveal a significant improvement in 3D annotation accuracy. However, the filter may be beneficial in a cluttered scene with many overlapping objects. Furthermore, our experiments indicate that the center weight filter only provides minor improvements in spatial accuracy.

We further found that the current implementation of the erosion filter, applied to segmentation bitmaps, is error prone, because pixel-based erosion does not consider the size and shape of an object. The values stated in Table 1 are given in pixels and relate to a image size of 640×480 px.

5.2 General findings

From our results on the accuracy of 3D object annotations, we can generalize three major findings that serve a broader research context:

1. Our evaluation clearly indicates a correlation between the spatial accuracy of a reconstructed 3D bounding box and the size and shape of the area projected. Thus, the number of misclassified background voxels is already minimized by reducing the size of a bounding box. Removing the corners of the box (to approximate a rhomboid shape) further improves the spatial accuracy (see Section 5.1.1).
2. We observe a significant increase in accuracy as more frames are interpreted by the CNN. While these observations are well aligned with related work, we further found that better results were obtained with SSD Mobile Net (fast and accurate bounding boxes) than with Mask RCNN (accurate but expensive segmentation mask), since the greater number of network results outweighs the advantage of segmentation accuracy. Thus, fusion of multiple modified bounding boxes from different viewpoints yields better results than fusion of a few frames that are (more accurately) segmented.
3. During scene capture, objects are usually visible over a series of camera frames. Observing the occurrences of network results over time can be exploited to determine the plausibility of object detection. This can drastically reduce the number of false positive predictions (see Section 5.1.3). Since the effect of this filtering approach is correlated with the speed of the camera and the inferencing time of the network, consistent camera movement is beneficial.

6 Conclusions

In this paper, we introduced a novel framework for 3D reconstruction with simultaneous scene object annotation, streaming, and remote exploration using a virtual reality setup. Our framework employs pre-

trained 2D CNN-based networks to compute object annotations in large 3D scene reconstructions at run-time, achieving update rates of 21.7 fps with SSD Mobile Net and 18.6 fps with Mask RCNN. The framework partly builds upon readily available components [1, 30, 31], extends existing modules, and integrates novel approaches for asynchronous processing of dense 3D reconstruction and object annotation, filtering of network output, and on-the-fly computation of colliders in large triangulated 3D reconstructions.

Our asynchronous processing of dense 3D reconstruction and object annotation supports the usage of CNNs with long and varying inference time. Thereby, we improve the state of the art, as related work either requires hard-coding of the number of frames that can be interpreted [13, 16, 20], or only achieves interactive frame rates when using networks with short inference time [17]. Furthermore, our presented approach avoids degradation of the 3D reconstruction due to the asynchronous computation of 3D reconstruction and object detection, both with respect to time and resources.

With our novel filter pipeline, our framework achieves fast and efficient object classification by highly efficient, generic modification and filtering of the predicted 2D bounding boxes. This makes our approach computationally more lightweight than related works, which use fully convolutional networks [13, 15, 20] or geometry-based segmentation algorithms [3, 15, 17] to optimize the network results. Furthermore, our filter pipeline makes our approach independent of complex trained networks as it applies temporal and spatial filters to the standard detection output of a state-of-the-art network. To store the 3D reconstruction, our framework uses voxels that are assigned a label-ID and an accumulated class probability, determined by our voting algorithm. The voxels are combined into blocks which are efficiently stored with a voxel block hashing data structure. Thereby, our framework's annotated 3D data is much less complex than that of related work [13, 16], improving the memory footprint and enabling fast incremental streaming of the annotated 3D reconstruction to a client for real-time exploration, which has not yet been presented in prior art.

Our framework's incremental streaming combined with incremental triangulation of the transmitted

voxel blocks enables live exploration of the annotated 3D reconstruction using a virtual reality setup. The CNN detection results are represented as 3D bounding boxes that enclose the annotated scene objects. We further exploit the annotation information to automatically generate computational-efficient colliders that represent the coarse geometry of the annotated 3D scene objects. Thereby, we enable on-the-fly interaction with the 3D scene objects while providing VR exploration at high update rates.

We have evaluated the framework's accuracy on 3D object detection using SSD Mobile Net and Mask RCNN. Since this study focused on the influence of filter parameters on system functionality, only one 3D scene was used for evaluation. Further experiments using the NYUv2 dataset [34] are the subject of future research.

We first investigated the influence of the different stages of the filter pipeline on detection accuracy, to obtain optimal filter settings for the tested scene, which can be used as a baseline for other networks and scenes. Furthermore, our results also yield three major findings that can be generalized to serve a broader research context.

First, modification of the bounding boxes' size and shape significantly increases the spatial accuracy of the annotated objects. While our approach is not as accurate as the geometric object segmentation of Refs. [17, 20] it still provides accurate 3D bounding boxes in the virtual 3D reconstruction. Further, we observed a significant increase in accuracy as more frames are interpreted by the CNN. Therefore, our approach supports findings of related work and further reveals that more accurate results are obtained using a box detection network than a semantic segmentation network. Finally, a temporal persistence filter significantly reduces the number of false classifications. Our promising results can be further improved in future by additionally tracking locations and sizes of bounding boxes for the persistence filter.

Our framework provides a universal communication interface that enables the versatile integration of any 2D box detection or segmentation network. In future, we plan to implement the CNN module as a network-based micro service to enable 3D scene reconstruction with a mobile device while still obtaining object annotations at run-time from a CNN running on a powerful machine. Furthermore, we plan to extend

and test our framework with different neural network types, architectures, and machine learning frameworks to expand its application domain.

The source code of this work has been released by Vienna University of Technology at the following link: <https://gitlab.cg.tuwien.ac.at/amosse/semantic-3d-reconstructions>.

Acknowledgements

The work was solely supported by Vienna University of Technology.

Electronic Supplementary Material Supplementary material is available in the online version of this article at <https://doi.org/10.1007/s41095-020-0194-4>.

References

- [1] Mossel, A.; Kroeter, M. Streaming and exploration of dynamically changing dense 3D reconstructions in immersive virtual reality. In: Proceedings of the IEEE International Symposium on Mixed and Augmented Reality, 43–48, 2016.
- [2] Ruddle, R. A.; Lessels, S. The benefits of using a walking interface to navigate virtual environments. *ACM Transactions on Computer-Human Interaction* Vol. 16, No. 1, Article No. 5, 2009.
- [3] Sünderhauf, N.; Pham, T. T.; Latif, Y.; Milford M.; Reid, I. Meaningful maps with object-oriented semantic mapping. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 5079–5085, 2017.
- [4] Kammerl, J.; Blodow, N.; Rusu, R. B.; Gedikli, S.; Beetz, M.; Steinbach, E. Real-time compression of point cloud streams. In: Proceedings of the IEEE International Conference on Robotics and Automation, 778–785, 2012.
- [5] Golla, T.; Klein, R. Real-time point cloud compression. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 5087–5092, 2015.
- [6] Morell, V.; Orts, S.; Cazorla, M.; Garcia-Rodriguez, J. Geometric 3D point cloud compression. *Pattern Recognition Letters* Vol. 50, 55–62, 2014.
- [7] Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 580–587, 2014.
- [8] Girshick, R. Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, 1440–1448, 2015.

- [9] Ren, S.; He, K.; Girshick, R.; Sun, J.; Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 39, No. 6, 1137–1149, 2017.
- [10] Redmon, J.; Farhadi, A. Yolo9000: Better, faster, stronger. In: Proceedings of the IEEE Conference On Computer Vision and Pattern Recognition, 7263–7271, 2017.
- [11] Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C. Y.; Berg, A. C. SSD: Single shot MultiBox detector. In: *Computer Vision—ECCV 2016. Lecture Notes in Computer Science, Vol. 9905*. Leibe, B.; Matas, J.; Sebe, N.; Welling, M. Eds. Springer Cham, 21–37, 2016.
- [12] He, K.; Gkioxari, G.; Dollfari, P.; Girshick, R. Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, 2961–2969, 2017.
- [13] McCormac, J.; Handa, A.; Davison, A.; Leutenegger, S. Semanticfusion: Dense 3D semantic mapping with convolutional neural networks. In: Proceedings of the IEEE International Conference on Robotics and automation, 4628–4635, 2017.
- [14] Whelan, T.; Leutenegger, S.; Salas Moreno, R.; Glocker, B.; Davison, A. ElasticFusion: Dense SLAM without a pose graph. In: Proceedings of the Robotics: Science and Systems, 2015.
- [15] Runz, M.; Bufer, M.; Agapito, L. MaskFusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In: Proceedings of the IEEE International Symposium on Mixed and Augmented Reality, 10–20, 2018.
- [16] He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, 2980–2988, 2017.
- [17] Nakajima, Y.; Saito, H. Efficient object-oriented semantic mapping with object detector. *IEEE Access* Vol. 7, 3206–3213, 2019.
- [18] Prisacariu, V. A.; Kähler, O.; Golodetz, S.; Sapienza, M.; Cavallari, T.; Torr, P. H.; Murray, D. W. InfiniTAM v3: A framework for large-scale 3D reconstruction with loop closure. *arXiv preprint arXiv:1708.00783*, 2017.
- [19] Tateno, K.; Tombari, F.; Navab, N. Realtime and scalable incremental segmentation on dense slam. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 4465–4472, 2015.
- [20] Wang, L.; Li, R.; Sun, J.; Liu, X.; Zhao, L.; Seah, H. S.; Quah, C. K.; Tandianus, B. Multi-view fusion-based 3D object detection for robot indoor scene perception. *Sensors* Vol. 19, No. 19, 4092, 2019.
- [21] Hou, J.; Dai, A.; Niefiner, M. 3D-sis: 3D semantic instance segmentation of RGB-D scans. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4421–4430, 2019.
- [22] Prisacariu, V. A.; Kähler, O.; Cheng, M. M.; Ren, C. Y.; Valentin, J.; Torr, P. H.; Reid, I. D.; Murray, D. W. A framework for the volumetric integration of depth images. *arXiv preprint arXiv:1410.0925*, 2014.
- [23] Nießner, M.; Zollhöfer, M.; Izadi, S.; Stamminger, M. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics* Vol. 32, No. 6, Article No. 169, 2013.
- [24] Nickolls, J.; Buck, I.; Garland, M.; Skadron, K. Scalable parallel programming with CUDA. *Queue* Vol. 6, No. 2, 40–53, 2008.
- [25] Newcombe, R. A.; Izadi, S.; Hilliges, O.; Molyneaux, D.; Kim, D.; Davison, A. J.; Kohi, P.; Shotton, J.; Hodges, S.; Fitzgibbon, A. KinectFusion: Real-time dense surface mapping and tracking. In: Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality, 127–136, 2011.
- [26] Deutsch, P. DEFLATE Compressed Data Format Specification version 1.3. RFC 1951. DOI: 10.17487/RFC1951. 1996.
- [27] Lorensen, W. E.; Cline, H. E. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics* Vol. 21, No. 4, 163–169, 1987.
- [28] Kähler, O.; Adrian Prisacariu, V.; Yuheng Ren, C.; Sun, X.; Torr, P.; Murray, D. Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Transactions on Visualization and Computer Graphics* Vol. 21, No. 11, 1241–1250, 2015.
- [29] Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [30] Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C. Y.; Berg, A. C. SSD mobilenet v2 COCO 2018 03 29. Available at <https://github.com/openai/openai-flixtra/blob/master/testdata/dnn/ssd/fmobilenet/fiv2/ficoco/fi2018/fi03/fi29.ptxt>.
- [31] Abdulla, W. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. 2017. Available at <https://github.com/matterport/Mask-RCNN>.
- [32] Lin, T. Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollfari, P.; Zitnick, C. L. Microsoft COCO: Common objects in context. In: *Computer Vision—ECCV 2014. Lecture Notes in Computer Science, Vol. 8693*. Fleet, D.; Pajdla, T.; Schiele, B.; Tuytelaars, T. Eds. Springer Cham, 740–755, 2014.
- [33] Zhou, Q. Y.; Koltun, V. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics* Vol. 32, No. 4, Article No. 112, 2013.

- [34] Silberman, N.; Hoiem, D.; Kohli, P.; Fergus, R. Indoor segmentation and support inference from RGBD images. In: *Computer Vision–ECCV 2012. Lecture Notes in Computer Science, Vol. 7576*. Fitzgibbon, A.; Lazebnik, S.; Perona, P.; Sato, Y.; Schmid, C. Eds. Springer Berlin Heidelberg, 746–760, 2012.

Benjamin Höller is a postgraduate student with the Interactive Media System Group at the Institute of Visual Computing and Human-Centered Technology at Vienna University of Technology, where he received his M.Sc. degree with distinction in 2019. His research interests lie at the intersection

of virtual and augmented reality, and 3D computer vision, with a strong focus on machine learning.

Annette Mossel is a post-doctoral researcher at the Institute of Visual Computing and Human-Centered Technology at Vienna University of Technology, and a scientific entrepreneur. She received her Ph.D. degree in 2014 from Vienna University of Technology. During her studies, she worked as a

visiting researcher at the Fraunhofer Institute for Computer Graphics and the MIT Media Lab. She has 12 years of experience in mixed reality with strong expertise in vision-based self-localization, dense 3D mapping, and 3D human computer interaction (HCI). She has authored or co-authored more than 20 scientific publications and participated and lead multiple nationally funded scientific projects on wide-area indoor localization, multi-user VR, and dense 3D surface reconstruction.

Hannes Kaufmann is full professor of virtual and augmented reality at the Institute of Visual Computing & Human-Centered Technology at TU Wien. He has conducted research into virtual reality, tracking, mobile augmented reality, training spatial abilities in AR/VR, tangible interaction, medical

VR/AR applications, real time ray-tracing, redirected walking, geometry, and educational mathematics software. His habilitation (2010) was on “applications of mixed reality” with a major focus on educational mixed reality applications. He has acted on behalf of the European Commission as a project reviewer, participated in EU projects in FP5, FP7, and Horizon2020, managed over 30 research projects and published more than 100 scientific papers.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made.

The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Other papers from this open access journal are available free of charge from <http://www.springer.com/journal/41095>. To submit a manuscript, please go to <https://www.editorialmanager.com/cvmj>.