Deleting 3D Objects in Augmented Reality using RGBD-SLAM

Sharvani Chandu, Bhavan Jasani, George Joseph, Aashi Manglik

***Abstract*— We present an Augmented Reality application where the user can remove an existing object in the scene and visualize how the scene looks in absence of that object. There are two main challenges to remove an existing object from the scene. First, it is difficult to acquire the scene information which is occluded by the object to be removed. Second, after acquiring the occluded scene information, it is not straight-forward to get a natural looking image with object removed. We propose to solve the first task by capturing a 360 degrees view of the scene to minimize occlusion and reconstruct a globally consistent 3D map using RGB-D SLAM in an online incremental fashion. The second task is solved by projecting the augmented 3D point cloud onto the desired image via the estimated camera pose and replace the object pixels by the projected points. We observe that a post-processing step of applying traditional inpainting algorithm further refines the image to look more natural.**

I. INTRODUCTION

Apart from robotic exploration, SLAM techniques have recently received interest in the field of Augmented Reality (AR). AR targets to interact with objects in the unknown map adding virtual annotations. Accurate tracking and 3D reconstruction of unknown scene achieved via SLAM thus boosts the feasibility of AR experience. Current AR applications take a 3D model of the object, often from graphics engines such as Unity and Unreal Engine, and registers it on the current scene respecting the physical constraints of the world and camera geometry. One of the ways it is done is to align the base of virtual object with a planar surface in the global scene. Then, use the transformation between the virtual object points in global frame to camera pose in the global frame followed by a pinhole camera projection.

While there exists ways to place and register new 3D models in the scene, it is nontrivial to remove the objects already present in the scene. For example, let us consider the scenario shown in Fig. 1 where a person wants to remove the existing chair in the living room to see how it looks. The emptied scene can then be used to insert a new virtual object in the scene as shown in Fig. 2 which otherwise would have interfered with the previously existing chair. The latter problem of adding a new 3D object, i.e., Fig. 2, is solved by estimating the 3D location of the underlying surfaces via SLAM and then providing an accurate location to place the new geometric model. In this work, we aim to focus on the former and novel task of deleting the existing object as illustrated in Fig. 1.

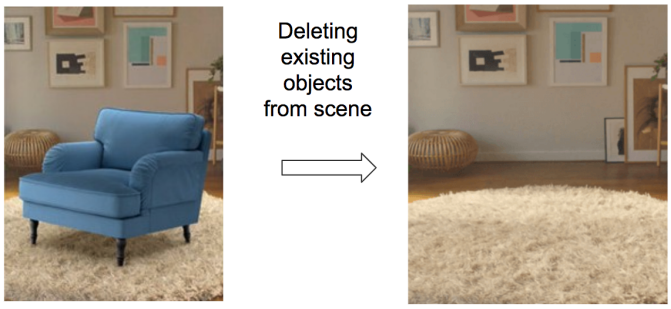

Fig. 1. Deleting the existing chair in scene to give a natural image as output with no artifacts

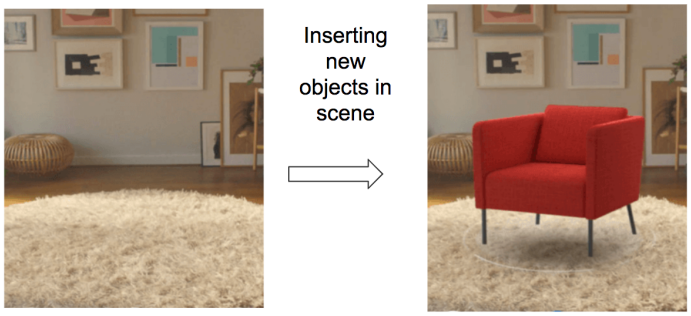

Fig. 2. After emptying the scene, a new virtual can be inserted easily without any unrealistic interference with the previously existing object

II. RELATED WORK

Loop closure is an integral part of SLAM algorithms. In sparse feature-based SLAM, small scale loop closure can be done via bundle adjustment, i.e., joint optimization of camera poses and triangulated features. Whereas large scale loop closure can be achieved by partitioning the map into keyframes and applying pose graph optimization [1]. For maintaining a globally consistent dense map, bundle adjustment is computationally infeasible and approaches such as Kinect-Fusion [2] targets a very small area requiring extremely loopy motions.

A. Elastic-Fusion

Elastic Fusion [3] creates a dense surfel-based map of room scaled environment captured using an RGB-D camera in an incremental online fashion. Unlike ORB-SLAM [4], it does not use pose graph optimization. Instead, it uses dense frame-to-frame camera tracking and windowed surfel-based fusion coupled with the frequent model refinement through non-rigid surface deformations. The authors [3] state that the pose graph SLAM systems primarily focus

on optimizing the camera trajectory, whereas elastic fusion utilizes a deformation graph to focus on optimizing the map.

B. Mask RCNN

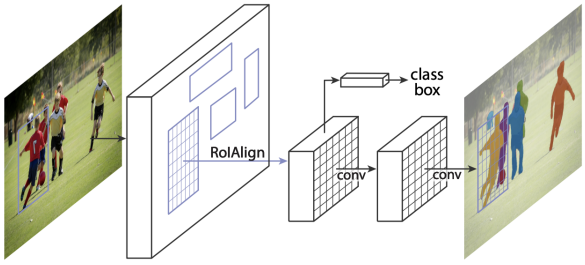

Fig. 3. The Mask RCNN framework for image instance level segmentation. Image taken from [5]

Mask RCNN [5] is a deep neural network based image instance segmentation algorithm. It can classify objects at pixel level by outputting binary masks for various objects in the image. Important thing to mention is that it does instance segmentation as opposed to semantic segmentation. In semantic segmentation objects of same category are all grouped with a single mask where as in instance segmentation different instances of the same object type are associated with different masks, which is something important for the augmented reality applications. At the back bone Mask RCNN is built open the RCNN family of object detection algorithm, in particular the Faster RCNN. To Faster RCNN it adds a head which generates binary masks to the.

C. Mask-Fusion

Mask-Fusion [6] is a real-time, object-aware, semantic and dynamic SLAM system. Mask-Fusion can detect different 3D objects, segment and track them even when they are moving. Mask-Fusion consists of the following two major components.

1) Mask RCNN [5] which runs on images to provide 2D segmentation of 80 different object classes
2) Elastic fusion which provides a 3D point cloud map and camera poses

It fuses traditional RGBD-SLAM with powerful image level segmentation from Mask-RCNN. This gives us a reach semantic information about the scene in comparison to traditional SLAM systems which only provides a pure geometric map. Since it can segment objects in the 3D point cloud, it allows one to remove points corresponding to particular objects from the point cloud, essentially allowing one to delete certain objects from the scene.

D. Co-Fusion

Co-Fusion [7] is earlier work from the authors of the Mask-Fusion. It is again a dynamic SLAM system and so can identify and segment moving objects in point cloud map, in comparison to static SLAM systems which consider moving objects as outliers and remove them from the map. The crucial difference with Mask-Fusion is that in order to find moving objects, they use motion clue of the moving objects to differentiate them from the background. So they can only detect moving objects but not there rich semantic information. In our case finding the semantic information is necessary so we build upon Mask-Fusion.

III. Data Collection

As shown in Fig. 1, there are two main challenges to remove an existing object from the scene and get a natural image as output.

1) We need information about the part of 3D scene which is occluded by the object
2) Project this part of 3D scene in the 2D image without generating any artifacts

To tackle the first challenge, we have to capture a sequence where the camera moves 360 degrees around the object. This would guarantee information from all possible viewpoints around the object. However, the standard TUM dataset [8] used for RGBD benchmarks do not satisfy this criteria. Hence we recorded our own sequence with the Intel RealSense Structured Light Camera. This gives us RGB images and depth maps which are not aligned with each other. After rectification, the sequence is converted to the KLG format which is the default input format used by open source implementations of MaskFusion and ElasticFusion.

Multiple sequences were collected from Smith Hall, with a bottle as the object of interest. A sample image from a sequence is shown in Fig. 4.

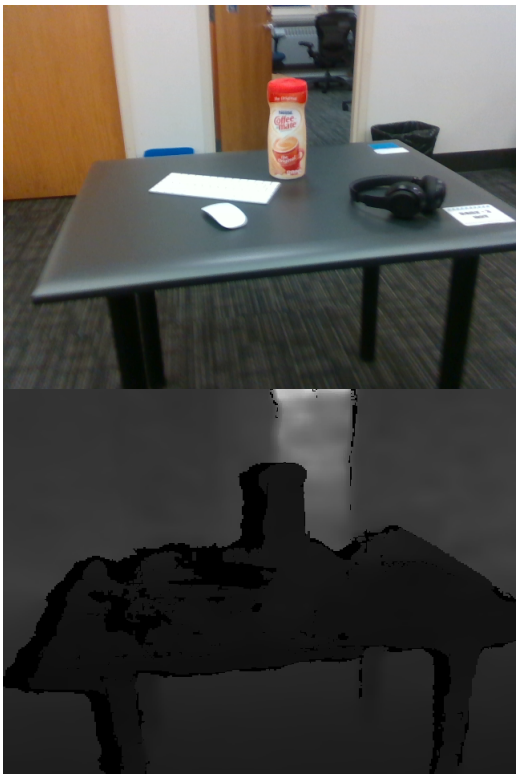

Fig. 4. Aligned RGB image and the corresponding depth map

The recorded data can be found here. Also, only objects that can be segmented using Mask-RCNN can be deleted. Mask-RCNN is trained to segment 80 object classes such as teddy bear, bottle and person.

IV. Approach

We build our system based on Mask-Fusion. The system consists of Mask-RCNN which is implemented in Python using Tensorflow library and RGBD-SLAM implemented in C++.

The approach consists of following steps.

A. 2D image segmentation via Mask-RCNN

The RGB frame is sent through Mask-RCNN network to segment out the object in image space. An illustration is shown in Fig. 5 where the bottle gets segmented in the image space. Mask RCNN outputs a segmentation mask, but the mask is diffused with the background and hence doesn't have perfect object boundary.

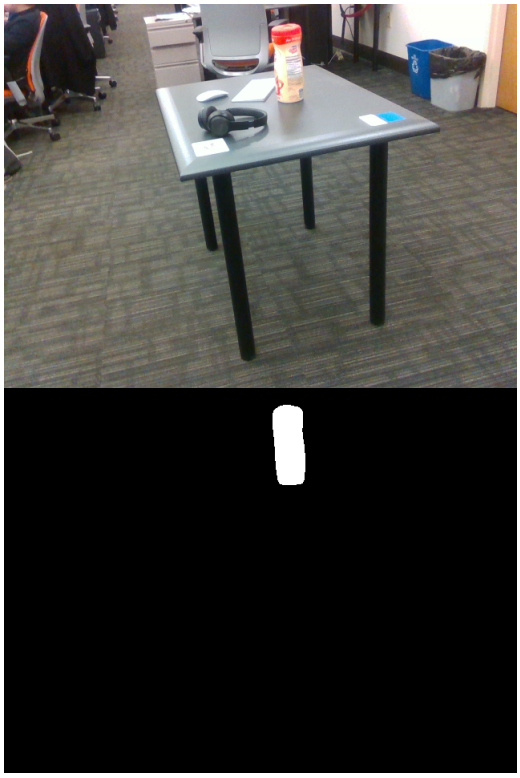

Fig. 5. Image segmentation using Mask RCNN

B. Geometric segmentation on depth information

It is important to have a perfect pixel level mask of the object which we want to remove from the scene. So to counteract the issue of diffused mask provided by Mask RCNN, we also do geometric segmentation on the depth information.

Our geometric segmentation approach follows from Mask-Fusion [6] comprising of two steps.

- Generate an edginess-map as shown at top of Fig. 6 based on depth discontinuity and concavity measure. These are calculated using the vertex positions and normal directions stored in the reconstructed point cloud.
- This edginess score is reprojected from 3D vertex to 2D pixel using the estimated camera pose. A connected component algorithm is then applied on the projected edge map to get a geometric labelling where each pixel maps to an object class excluding the background as shown at the bottom of Fig. 6. Please note that some edges are filtered by applying a threshold on the edginess score to get better connected components as shown in Fig. 6.

Since geometric segmentation exploits depth channel it provides very sharp boundaries of the objects. The downside of it is, it results into over-segmentation as can be seen in Fig 6 top image.

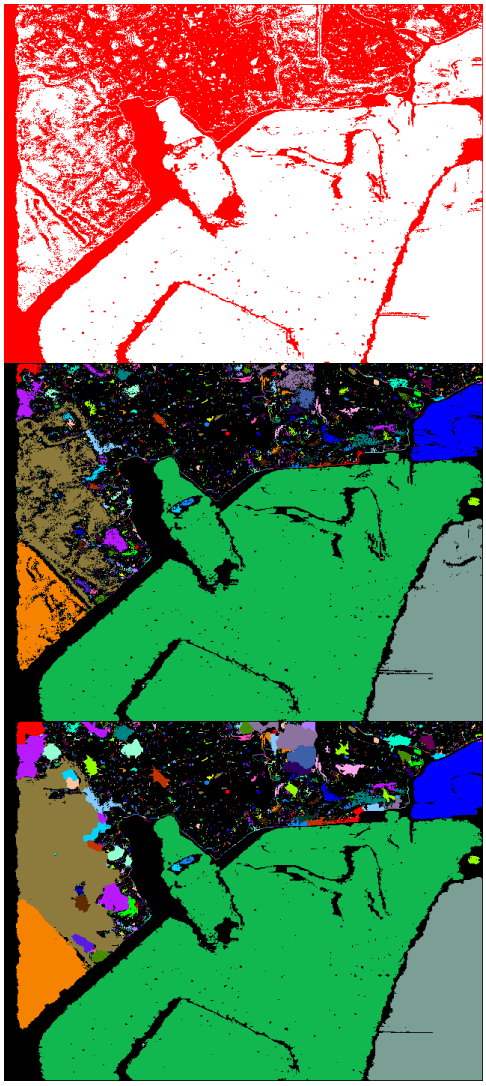

Fig. 6. Geometric Segmentation in 3D Point Cloud. **Top:** Edge Detection; **Middle:** Connected Components before edge removal; **Bottom:** Connected Components after edge removal

C. Combining geometric segmentation and image segmentation

Mask RCNN provides diffused object masks but has full knowledge of the objects, whereas geometric segmentation does sharp segmentation but doesn't incorporate the knowledge of the objects in the scene. Hence as per Mask Fusion [6] paper, these both are combined to provide better image segmentation masks.

Fig. 7. Map after removing the object

D. Deleting the subset of point cloud associated with the object of interest

We first build a small map (point cloud) of the scene by moving the camera. Use the combined segmentation mask from Mask RCNN and geometric segmentation we know pixels in RGB image as well in the point cloud for the object which we want to delete. So while building the point cloud map, we ignore the pixels from object we plan to delete in the live image.

E. Projecting the point cloud onto the image and replacing the mask pixels with the point cloud

After the map is built without the object, the view is rendered from the same location of the camera. This gives us information about the background. Using the mask, the object is cut out and replaced with the rendered background. An important point to emphasis is that the point cloud is generated from different view points of the scene and hence it contains information of the points behind the object which we are removing, although it won't be able to fill in the full background scene which is done by the next part of our system.

F. Refinement using inpainting

To fix minor artifacts like missing point, in-painting is applied for refinement.
Traditional inpainting techniques were used for filling in lines or small blocks of an image, like restoring old photographs. In recent times, there are generative techniques developed which fill in large blocks of the image. These methods require a lot of training data and do not generalize well to unseen images. But in the problem we wish to solve, we have the information in the point cloud and have to replace the small holes that exist due to the sparse points (sparsity is due to occlusion) generated in the cloud.
With the data from the point cloud, we using OpenCV inpainting technique INPAINT_NS based on fluid dynamics.

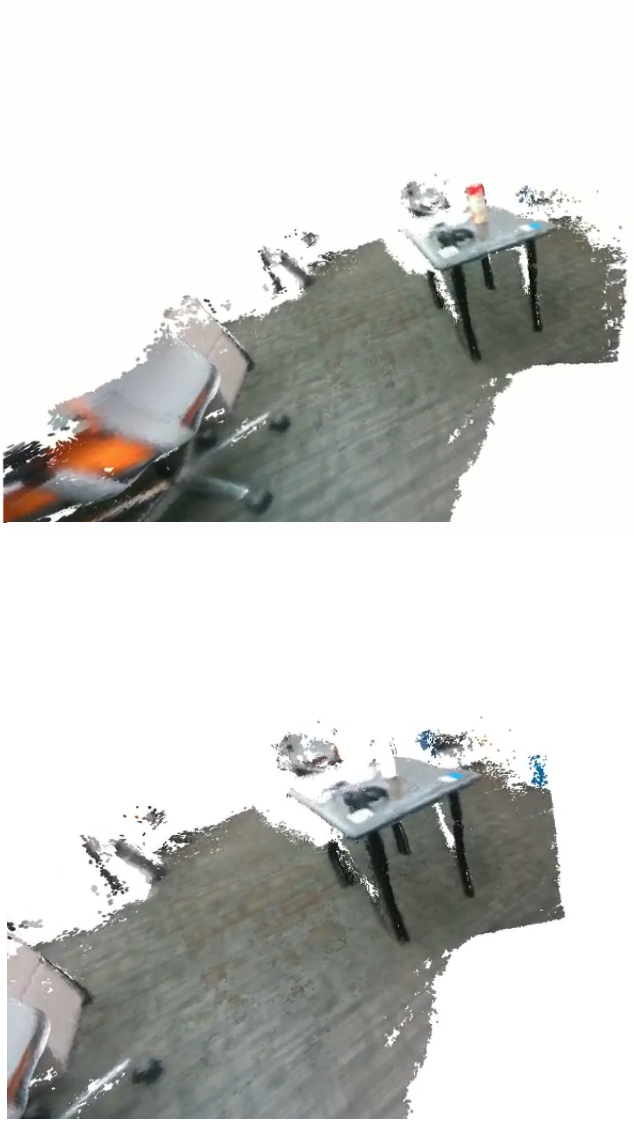

Fig. 8. Map built with bottle (top) and without (bottom)

V. Results

The Fig 8 shows a rendering of the point cloud after removing the bottle. The background information is used to fill up the area occupied by the bottle.

The point cloud is used to fill up the area cut out by the mask and finally, small artifacts like missing region in the point cloud are fixed with in-painting. The results are shown in the Fig 9. Notice that the background is visible through the image, which makes the deleting more realistic.

Fig. 9. Image after filling up the bottle (top) and after in-painting (bottom)

VI. Conclusions

Our approach leverages the information from the 3D point cloud to replace the masked portion in the image. However, since the mask area is replaced with the scene, we can notice that the portion from the point cloud is not well blend in. To improve this, some blending techniques can be used to make it look more seamless.

The lack of a 360-degree view dataset was an issue for our project, however, data was not a bottleneck for making our initial modules to work. So, we think we did well with our proposed timeline.

Another major bottleneck of our method is the huge computational cost. The system needs two GPUs, one for Elastic Fusion and another for Mask-RCNN. The quality of the final output depends on the number of points on the map, which we are constrained by the maximum GPU memory.

The code is split into three parts, the modified MaskFusion can be found here, the data collection, and post-processing codes are attached with this report.

References

[1] R. Kmmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 3607–3613.

[2] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, Oct 2011, pp. 127–136.

[3] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "Elasticfusion: Dense slam without a pose graph." Robotics: Science and Systems.

[4] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[5] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: http://arxiv.org/abs/1703.06870

[6] M. Rünz, M. Buffier, and L. Agapito, "Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects," *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 10–20, 2018.

[7] M. Rünz and L. Agapito, "Co-fusion: Real-time segmentation, tracking and fusion of multiple objects," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4471–4478.

[8] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.