2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

What the HoloLens Maps Is Your Workspace: Fast Mapping and Set-up of Robot Cells via Head Mounted Displays and Augmented Reality

David Puljiz[1], Franziska Krebs[2], Fabian Bösing[1], Björn Hein [1,3]

***Abstract*— Classical methods of modelling and mapping robot work cells are time consuming, expensive and involve expert knowledge. We present a novel approach to mapping and cell setup using modern Head Mounted Displays (HMDs) that possess self-localisation and mapping capabilities. We leveraged these capabilities to create a point cloud of the environment and build an OctoMap - a voxel occupancy grid representation of the robot's workspace for path planning. Through the use of Augmented Reality (AR) interactions, the user can edit the created Octomap and add safety zones. We perform comprehensive tests of the HoloLens' depth sensing capabilities and the quality of the resultant point cloud. A high-end laser scanner is used to provide the ground truth for the evaluation of the point cloud quality. The amount of false-positive and false-negative voxels in the OctoMap are also tested.**

I. INTRODUCTION

Knowledge of the robot environment is essential both in offline programming and for auto-generated trajectories as most manipulators lack external sensors needed to map their own environment.

Programming robotic manipulators is classically a time consuming process requiring expert knowledge. Offline programming is generally preferred to online, lead-through programming due to smaller downtimes [1]. Offline programming, however, requires a precise model of the working environment which is often a time-consuming undertaking requiring exact 3D models of the objects around the robot and precise measurements of their placement. Even then, the final program needs to be tested and verified inside the real workspace itself. As offline programming requires significant financial investment, expert knowledge and long delivery times, it is unsuited for small and medium enterprises which require intuitive and fast robot programming paradigms [2].

Likewise, setting up safety zones is a time consuming process, mostly done offline and then checked and rechecked until all safety zones are validated.

We propose a cost-efficient method to set-up the working environment of the robot that doesn't require any expert knowledge and combines exceedingly well with newer AR programming paradigms such as the one presented in [3]. It leverages the localisation and depth sensing capabilities of modern HMDs to map the workspace of the robot as the user walks around the robot cell. A mesh of the mapped environment is displayed to the user as feedback, so occlusions and unmapped areas can be noticed and remapped. The map is then represented as an OctoMap - a 3D occupancy grid of voxels [4]. The user adds safety zones and edits the OctoMap in situ using the AR capabilities. The user can, for example, add no-go zones or allow collisions where the use case requires contact with specific surfaces. This planning scene is then used for path planning software, in our case MoveIt!. The approach is agnostic in regards to the environment and the robot so long as a universal robot description file (urdf) and the link meshes are available.

[1]Intelligent Process Automation and Robotics Lab (IPR), Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Karlsruhe, Germany david.puljiz@kit.edu
[2] High Performance Humanoid Technologies Lab (H²T), Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Karlsruhe, Germany
[3] Karlsruhe University of Applied Sciences, Karlsruhe, Germany

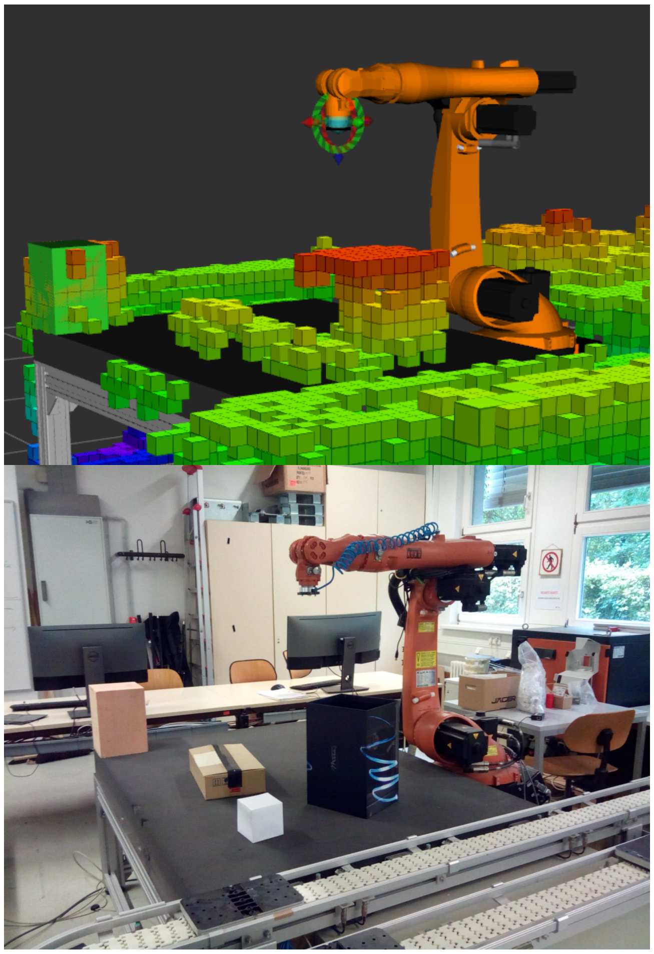
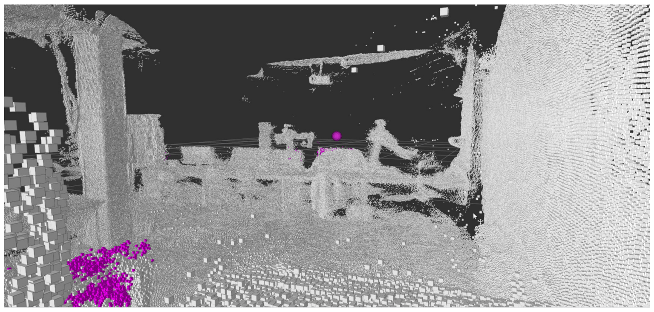


Fig. 1: Bottom: The real scene mapped with the HoloLens. Top: The resulting OctoMap after the HoloLens mapped the environment and the point cloud was filtered and converted to a voxel occupancy representation.


(a)

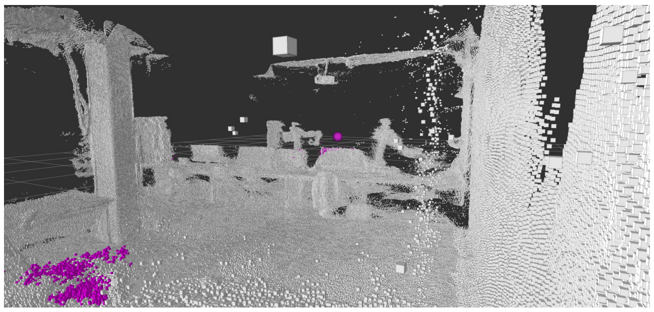

(b)

Fig. 2: Purple - points closer than 1 meter (a) Point cloud with points more than 3.3 meters distance removed. One can see the presence of sparse outliers that can be filtered out; (b) Point cloud with points more than 3.4 meters distance removed. The outliers are much denser requiring more aggressive filtering which may degrade the quality of the inliers.

II. RELATED WORK

Although most industrial robot manufacturing companies offer software for offline programming of robots, such as ABB's RobotStudio or KUKA's KUKA.Sim, these require precise CAD models of all objects in the environment as well as exact calibration between the virtual and the real robot cell.

Neto et al. [5] describe a more intuitive offline programming method based on the common CAD package Autodesk Inventor. The user inputs tool coordinates and a program is automatically created. This still requires precise CAD models and calibration. They note that calibration errors are a major source of inaccuracies. According to the authors, calibration requires expensive measurement hardware, software and expert knowledge. They also note that external sensing can help mitigate the errors of offline programming.

In [6] a trajectory is auto-generated for a spray-painting task by using range images of the part to colour. This eliminated the need for any programming on the part of the user. The collision-free trajectory generation, however, still required a model of the robot cell. Therefore the set-up step requires considerable input from the user and is not truly flexible or quick.

In the field of AR-based robotics, several approaches exist to plan robot motion in unknown environments. Ong et al. [7] use a tracked pointer tool to manually input trajectories and define collision-free volumes. However, no map of the environment is created and only a small part of the total collision-free volume is used. The authors themselves note that alternative methods for generating collision-free volumes should be explored.

Similarly, Quintero et al. [3] use holographic waypoints and B-spline interpolation to plan robot trajectories. The system relies on the user to manually modify trajectories to avoid obstacles. The mesh of the environment generated by the HoloLens is used to define waypoints on surfaces, yet the map itself is not used further.

In [8] the environment of a telepresence robot is mapped to allow the overlay of virtual fixtures - virtual objects for operator assistance. The motion of the robot arms, however, is guided by the user and no programming was implemented.

A. Contributions

This paper extends the previous approaches in several ways. Firstly, by mapping the environment with multi-purpose HMDs, we eliminate the need for any overhead equipment for cell setup or the need for CAD data of the surrounding objects. HMDs have been used for robot intention visualisation [9], collaborative task planning [10] and as previously mentioned programming [3] just to name a few.

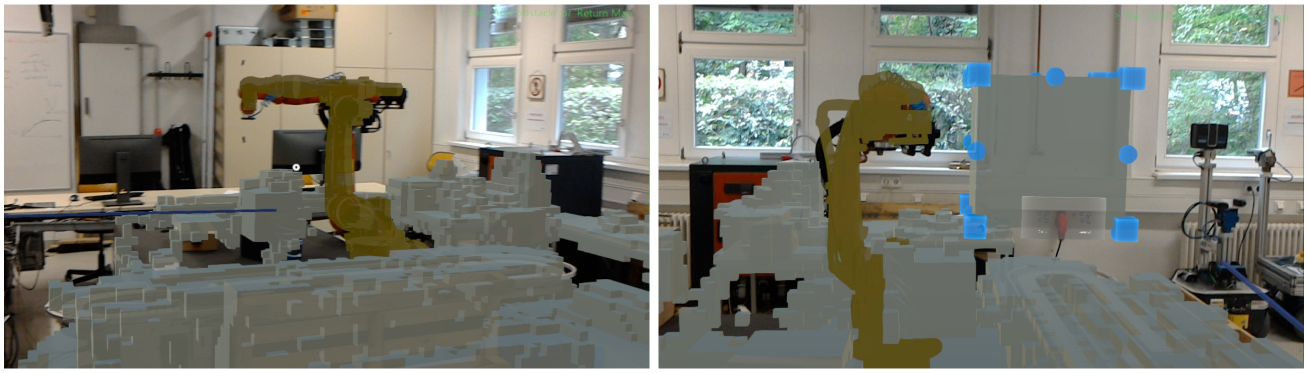
Secondly, the map created this way can be used both for offline programming or as an addition to AR-based approaches such as the one in [3]. In the latter case, it allows the use of higher-level motion planning to plan collision-free trajectories, such as MoveIT!. This significantly decreases the programming effort for the user.

Finally, we perform thorough tests of the depth sensing capabilities of the HoloLens. As of yet such tests have not been performed. This data will provide useful metrics and possible failure cases for future research.

III. METHODOLOGY

The system consists of two main components, the HoloLens HMD and a desktop computer connected to the robot and running the Robot Operating System (ROS) [11]. Communication between the desktop and the HoloLens is mediated via the ROSBridge package that allows seamless interfacing between ROS nodes and programs running on different systems. For point cloud editing and filtering the open source Point Cloud Library (PCL) [12] was used. The OctoMap representation and the path planning is done using the MoveIt! path planner.

On the HoloLens side, the AR interface was constructed using the Unity3D engine. For the capture and streaming of depth information the HoloLensForCV library package was used. Communication with ROS was done using the ROS# library which provides ROSbridge clients for .Net applications like Unity3D.



(a) (b)

Fig. 3: (a) Visualisation of the voxel grid representation of the environment in the HoloLens. The individual voxels can be added, moved and removed; (b) Adding safety zones in situ using the HoloLens. Such definition of safety zones is more intuitive and faster than classical offline robot programming.

A. *Mapping*

One of the main benefits of using HMDs for mapping is their mobility. To map the environment, the user simply walks around the robot cell. The mesh of the environment is displayed for feedback, thus the user knows if some part of the cell is left unmapped due to e.g. occlusions. Two different methods of obtaining the point cloud of the environment have been implemented.

The first approach uses the mesh of the environment already generated by the HoloLens. Randomly a mesh triangle is chosen, weighted by the size of the triangles. Then, using barycentric coordinates, a random point within the triangle is selected and saved to the point cloud. The number of iterations of this process, and therefore the size of the resulting point cloud, can be chosen. The resulting point cloud is filtered with voxel grid filtering to obtain a uniform point density.

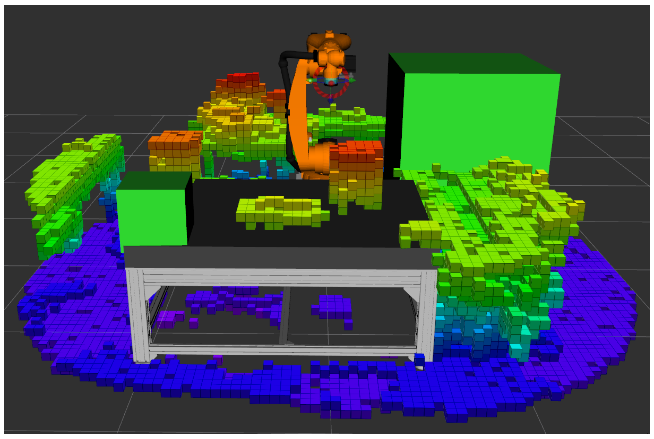


Fig. 4: The edited environmental voxel grid and safety zones as visualised in RViz. The table in this application was part of the urdf. If a CAD model of the object exists and the robot should interact with it, adding it to the robot model will filter out the unwanted voxels automatically.

The second approach was made possible after Microsoft allowed access to the depth stream. The depth sensor on the HoloLens provides two depth streams, the short-throw depth stream, with 30 frames per second update rate and a range of 0.2-1 meters, and a long-throw depth stream, with 1-5 frames per second update rate and a range of 0.5-4 meters.

We discard points that are below the minimum cut-off distance $D_{cut-off\_min}$ to eliminate points that may belong to the user's hand, and above the maximum cut-off distance $D_{cut-off\_max}$ to eliminate low quality points. The maximum cut-off distance was experimentally determined to be 3.3 meters (Fig. 2). The minimum cut-off distance was taken to be one meter, around the reach of the user's arms. Therefore, we use only the long-throw stream and discard points further away than 3.3 meters.

With the localisation capabilities of the HoloLens, the different depth frames can be fused into a single point cloud of the environment. Point clouds of different frames are registered to the main point cloud using ICP [13], with the HoloLens' own localisation as the initial guess. It was found, however, that the HoloLens' localisation is precise enough that ICP does not significantly increase the precision. Therefore, the registration step may be skipped. This shall be demonstrated in the experiments in Section IV-A.

The resulting point cloud is down-sampled using voxel-grid filtering to ensure uniform point density. It is then filtered with an outlier removal filter, removing any point that had less than 9 neighbours in a radius of 5 cm, and smoothed with moving least squares [14]. Finally RANSAC plane detection is used to detect planes and map all the points near the plane to the plane itself. This improves the resolution of objects on floors and tables.

B. *Referencing*

To get a robust coordinate transform between the HoloLens and the robot world coordinate system, a semi-automatic referencing approach is used [15]. The user is asked to position a seed hologram near the robot base and rotate it approximately towards the front of the robot. Using the urdf and the link meshes of the robot, a point cloud of the robot is created. The model, together with the map of the environment and the position of the seed hologram, are used as the input to an ICP registration algorithm. As the ICP is highly sensitive to local minima, the seed cube is paramount to get a robust coordinate transform. As shown in [15] the positioning of the seed hologram doesn't have to be precise but merely near the base of the robot. The referencing itself is agnostic to the type of the robot and has been tested with the Kuka KR 5, KR 16 and KR 120 robots.

C. Workspace Representation and Editing

The point cloud is once again down-sampled with a voxel-grid filter with the voxel size of the OctoMap, D_{leaf}. A kD-tree representation of the point cloud is then constructed. The kD-tree is a data structure that facilitates radius and nearest-neighbour searches. Knowing the pose of the robot from the referencing step, all points belonging to the robot are removed as well as all points further away than the maximum reach of the robot D_{reach}. The resulting point cloud is used to generate an OctoMap voxel grid representation of the occupancy as seen in Fig. 1.

The voxel occupancy grid is sent to the HoloLens where a user may edit the occupancy grid. This step allows the user to correct errors in the map if needed or to remove voxels from parts of the environment where contact with the environment is needed to perform the robot's task. Finally, safety zones can be defined in situ, drastically reducing the set-up and test times. In Fig. 3 the overlayed robot model, the rendered OctoMap, and the set up of the safety zones can be seen.

When the user is done, the safety zones and the edited OctoMap are sent back to the computer. As the user can freely move and add voxels through AR on the HoloLens, these voxels must be snapped back to the voxel grid. The OctoMap environmental representations can be saved, loaded in MoveIt! and edited with the HoloLens as many times as necessary. Likewise, one could save different voxel grids and safety zones depending on the task for future uses. A representation of an edited map and safety zones in RViz can be seen in Fig. 4.

An algorithmic representation of the proposed method from the desktop computer's side can be found in Algorithm 1. The point cloud frames P_i are generated directly on the HoloLens from the depth frames and the transformation matrix between the frames. The x_{guess} is the position of the seed hologram that the user positioned to indicate the pose of the robot. The point cloud of the robot P_r is automatically generated from the urdf and the link meshes of the robot on the desktop computer itself. The urdf and the link meshes can be freely changed depending on the robot used.

Algorithm 1 Proposed Method of Cell Set-up

```
Input: Point Cloud Frames from Hololens $P_i$, Robot Point Cloud $P_r$, Robot Pose Guess from HoloLens $x_{guess}$
Output: MoveIt! Planning Scene $M$
    Parameters : $D_{voxel}$, $D_{cut-off\_min}$, $D_{cut-off\_max}$
 1: while no "Send Map" voice command do
 2:     Get $P_i$ from HoloLens
 3:     Remove points $p_i$ from $P_i$ s.t. $p_i \leq D_{cut-off\_min} \,\|\, p_i \geq D_{cut-off\_max}$
 4:     Add $P_i$ to scene point cloud $P$
 5: end while
 6: Filter and downsample $P$
 7: Get $x_{guess}$ from the HoloLens
 8: HoloLens to robot transform $T_{rH}$ = ICP(source: $P_r$, target: $P$, first guess: $x_{guess}$)
 9: Remove $P_r$ from $P$
10: Convert $P$ to Octomap with voxel size $D_{voxel}$ and send to HoloLens
11: Use "Get Scene" voice command to load and edit Octomap in HoloLens
12: Edit Octomap and Safety Zones on the HoloLens
13: Wait until "Send Scene" voice command
14: Create $M$ using edited Octomap and safety zones from Hololens
15: return $M$
```

IV. EXPERIMENTS AND RESULTS

Here we present in-depth tests of the HoloLens' depth sensor precision and the resultant point cloud (Section IV-A) as well as the quality of the generated OctoMap by counting the amount of false-positive and false-negative voxels in the occupancy grid of a test scene (Section IV-B). Further tests with the robot and the MoveIt! motion planer itself were conducted in which we showed that such an approach is indeed valid and works well with motion planners. However the motion planner tests also revealed a particular failure case which will be addressed in Section IV-C. The results of the previous tests shall also be discussed there.

A. Evaluation of Depth Sensing Capabilities

The first set of experiments was aimed at calculating spatial deviations of the HoloLens' depth sensor, while the second set was aimed at temporal deviation. Both use the same set-up where a flat cardboard surface was positioned at 1 and 2 meters respectively from the HoloLens' depth sensor.

For the first set, to measure spatial variation, the HoloLens was rotated so that one of the designated five points pictured in Fig. 5(a) lies on the cardboard surface. A total of fifteen consecutive depth frames were taken for each pixel and each distance for a total of 150 measurements. The results are depicted in Table I. The standard deviation of the depth measurement fluctuations around the average was found to be 3 mm and the maximum fluctuation around the average 5 mm with no major difference between pixels and distances.

In the second set of experiments, to measure temporal variations, a 5x5 pixel square in the centre of the depth image was selected and the values measured. For each distance 5 repetitions were carried out to average out human positioning error. For each repetition 5 consecutive frames were used for a total of 50 measurements. The setup can be seen in Fig. 5(b). The averages can be seen in Table II. The maximum error of the averages of each square is 11.2 mm and the total average error is 6 mm.

TABLE I: The distances with smallest and highest error as well as the average distance and standard distance deviations for one and two meters respectively. Measured in meters.

	Center	Top	Right	Bottom	Left
Minimum Error Distance 1m $[m]$	1.050	1.016	1.012	0.885	1.016
Maximum Error Distance 1m $[m]$	1.056	1.024	1.015	0.893	1.020
Average Distance 1m $[m]$	1.05233	1.01907	1.0138	0.88813	1.018
Standard Deviation Distance 1m $[m]$	0.00171	0.00228	0.00063	0.00269	0.00106
Minimum Error Distance 2m $[m]$	2.006	2.005	2.001	2.042	2.004
Maximum Error Distance 2m $[m]$	2.013	2.012	2.006	2.049	2.008
Average Distance 2m $[m]$	2.00907	2.00907	2.00387	2.0458	2.00567
Standard Deviation Distance 2m $[m]$	0.00200	0.00222	0.00130	0.00231	0.00100

TABLE II: The observed averaged depth values for each repetition of experiment 2. Measured in Meters

	Repetition 1	Repetition 2	Repetition 3	Repetition 4	Repetition 5
1 Meter	0.998400	1.007512	0.998728	1.005440	1.003424
2 Meters	2.011248	2.006576	2.005984	2.008064	2.000224

In the third set of experiments we took scans of our entire laboratory using a Faro Focus^S laser scanner with 1 mm precision as the ground truth. We compared it to a point cloud generated by the HoloLens, with the user walking around the laboratory to map it. All four different permutations using ICP for registration or not and using a post-processing step (consisting of MLS smoothing and RANSAC plane detection and projection) or not were tested. First, we removed discrepancies between the two point clouds - areas not covered by the laser scanner or due to the changing environment (as the point clouds were captured at slightly different times). To evaluate the difference between the two point clouds we used CloudCompare. CloudCompare gives the percentile distribution of distances between the two point clouds and therefore offers a good insight into the quality of the point cloud generated from the HoloLens. The results are shown in Table III. One can see that the best performance is the mapping without ICP, meaning that the HoloLens localisation is as precise as the point cloud, and with post-processing. In this case 75 percent of points have an error of 3.6 cm or lower. A visual comparison of the four point clouds to the ground truth can be seen in Fig. 6.

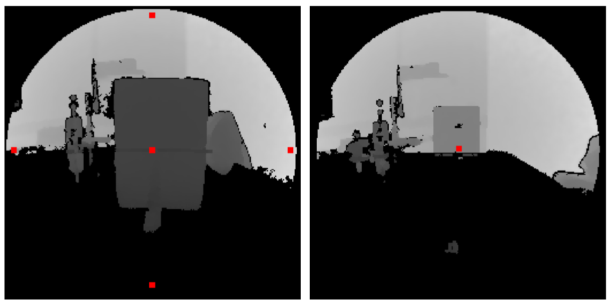

(a) (b)

Fig. 5: Experiments to calculate the standard deviation of depth measurements. A flat cardboard surface was placed at one and two meters respectively. (a) Set 1 to calculate the spatial deviation, we used the 5 points depicted here with the surface at one and two meters. Pictured here the setup the middle pixel. (b) Set 2 to calculate the time deviation i.e. noise, of the centre 5x5 pixel area for 5 consecutive frames, here at 2 meters distance

TABLE III: The percentiles of the distances of each spatial map combination to the laser scan.

Percentile	Without ICP, not postprocessed	With ICP, not postprocessed	Without ICP, postprocessed	With ICP, postprocessed
10th $[m]$	0.00591	0.00629	0.00588	0.00513
25th $[m]$	0.01177	0.01254	0.01135	0.01099
50th $[m]$	0.02192	0.02503	0.02150	0.02582
75th $[m]$	0.04105	0.04573	0.03673	0.04183
90th $[m]$	0.06447	0.06994	0.05275	0.06057

B. Evaluation of the OctoMap

To evaluate the quality of the OctoMap generated from the HoloLens' point cloud we placed a wooden 20x20x30 cm cuboid on the front-left of the table. We assumed the worst case scenario of using the sampled environmental mesh of the HoloLens. Khoshelham et al. [16] found that the average global error of the HoloLens' environmental mesh is around 5 cm. We counted the total number of false-positives, i.e. the voxels that are detected as occupied by the object that are in fact not, and false-negatives, i.e. voxels detected as free that are in fact part of the object. The results presented in Table IV show that on average there are 9.58 false-negatives and 61.33 false-positives with 12 point clouds tested. Point clouds were generated at different times with the user walking a different path around the robot cell. Worth noting is that the false-negatives are much more critical as they can cause crashes while false-positives only slightly limit the collision-free volume. Also worth noting is that some false-negatives are hidden behind false-positive voxels or near the table and are therefore unreachable.

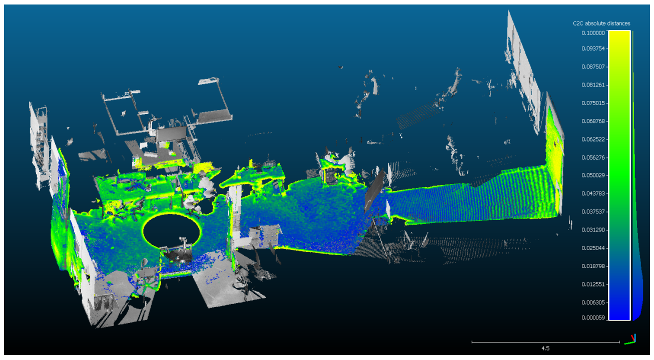
We also carried more than 50 tests with the MoveIt! path planner to try and provoke collisions in the cluttered scene shown in Fig. 1. The tests showed that there are indeed edge cases were a collision might happen i.e. when objects are positioned diagonally. A solution for these edge cases are presented in the next subsection.

C. Discussion

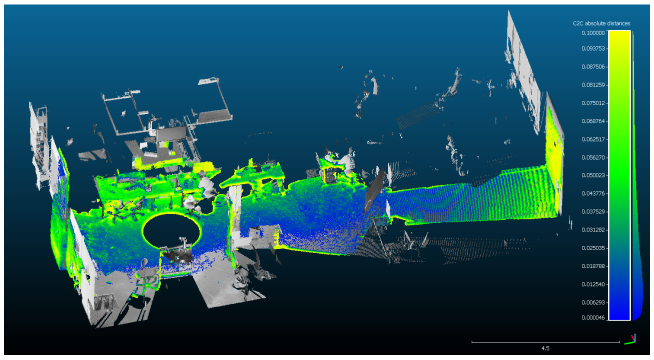
We have shown that the HoloLens's environment mapping capabilities, with a MLS smoothing and RANSAC plane finding and fitting stage can produce a point cloud where 75 percent of the points lie within 3.6 cm of the ground truth captured with a high-end laser scanner.

This could be improved further. We used the in-built HoloLens image processing pipeline for distortion correction that, according to the official documentation, gives a maximum error of 10 pixels. Calculating a better undistortion matrix for further rectification would give a further increase in precision.

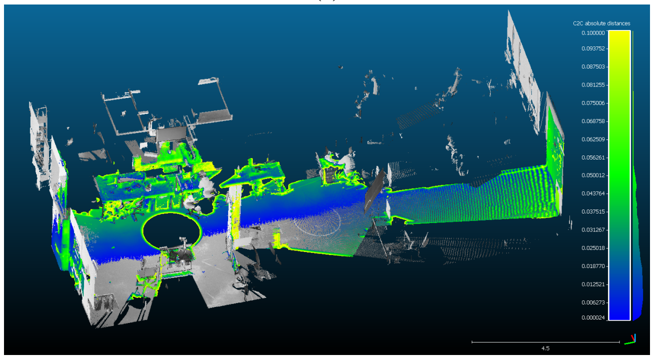
The OctoMap voxel occupancy grid performs adequately even in the worst case scenario where a sampled environmental mesh of the HoloLens was used with a mean square error of around 5cm.



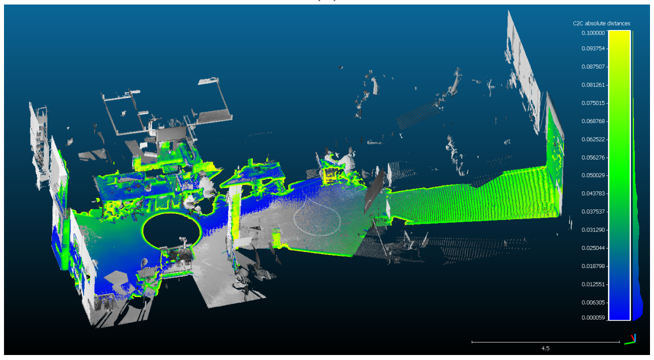
(a)



(b)



(c)



(d)

Fig. 6: Comparison of the HoloLens point cloud with the ground truth obtained via laser scan; (a) Point cloud without ICP registration between frames and without the post-processing step of MLS smoothing and RANSAC plane detection and projection; (b) Point cloud with ICP registration and without postprocessing; (c) Point cloud without ICP registration but with post processing; (d) Point cloud with ICP registration and with post-processing.

To mitigate the edge cases, a padding algorithm was developed. Each new level pads the surface of the starting voxels iteratively with voxels half the size of the starting voxels, as illustrated in Fig. 7. Tests with the motion planer have shown that the edge cases were eliminated and no collisions occurred anymore even with only one level of padding.

V. CONCLUSION AND FUTURE WORK

For robotic applications to really become ubiquitous in enterprises of any size, easy set-up and programming of robots is crucial. In this paper, we presented a robot cell modelling approach that relies on the Microsoft HoloLens to reference and map the previously unknown environment of the robot. As research in AR-based human-robot interaction and programming has seen major growth in recent years, a plethora of such programs can be combined with our approach to reduce user workload and extend the area of possible applications.

We have proven that the mapping and localisation capabilities of the HoloLens are more than adequate. Even if errors do occur, the interactive editing of the environmental map and the safety zones allows the user to quickly remove such errors. Furthermore the map can be edited for different task quickly and efficiently.

There is still plenty of room for improvement. We developed several programming approaches, one of them based on the work of Quintero et al. [3], which should be integrated with the cell set-up algorithm presented here to make an end-to-end system. User tests should then be made to see if the perceived workload of the users is indeed lowered which such a system compared to more classical approaches. The effect of the density of the environmental point cloud in regards to the precision of robot referencing and the quality of the final OctoMap will be explored further in follow up papers.

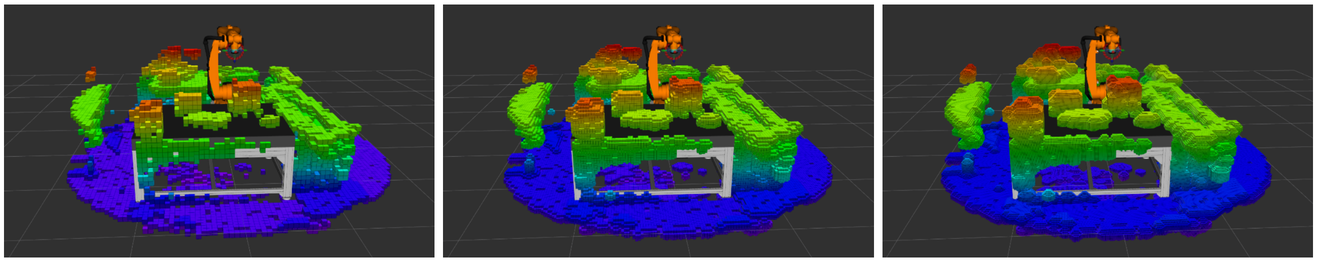
Another exciting research direction is using a dynamic map for the interaction with the robot as well as sharing and fusing of sensor data in the case the robot is also equipped with sensors. In the authors opinion such on-line sharing of data could be a great benefit in proximal human-robot collaboration.

ACKNOWLEDGMENT

This work has been supported from the European Union's Horizon 2020 research and innovation programme under grant agreement No 688117 "Safe human-robot interaction in logistic applications for highly flexible warehouses (SafeLog)".

TABLE IV: The number of false-positives and false-negatives in the OctoMaps generated from 12 different mappings

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	$\varnothing$	σ
false positive	83	60	67	71	52	75	78	61	41	35	58	55	61.33	14.48
false negative	6	16	10	5	5	8	14	7	12	15	11	6	9.58	3.99



(a) (b) (c)

Fig. 7: The padding process to remove the edge cases that might result in collisions. (a) The original OctoMap; (b) First level padding with voxels of size 2.5 cm; (c) Second level padding with voxels of size 1.25 cm on top of the level 1 padding.

References

[1] Z. Pan, J. Polden, N. Larkin, S. V. Duin, and J. Norrish, "Recent progress on programming methods for industrial robots," in *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, June 2010, pp. 1–8.

[2] R. D. Schraft and C. Meyer, "The need for an intuitive teaching method for small and medium enterprises," *VDI BERICHTE*, vol. 1956, p. 95, 2006.

[3] C. P. Quintero, S. Li, M. K. Pan, W. P. Chan, H. F. M. V. der Loos, and E. Croft, "Robot programming through augmented trajectories in augmented reality," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1838–1844.

[4] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com

[5] P. Neto and N. Mendes, "Direct off-line robot programming via a common cad package," *Robotics and Autonomous Systems*, vol. 61, no. 8, pp. 896 – 910, 2013.

[6] M. Vincze, A. Pichler, and G. Biegelbauer, "Detection of classes of features for automated robot programming," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 1, Sep. 2003, pp. 151–156 vol.1.

[7] S. K. Ong, J. W. S. Chong, and A. Y. Nee, "A novel ar-based robot programming and path planning methodology," *Robotics and Computer-Integrated Manufacturing*, vol. 26, no. 3, pp. 240–249, 2010.

[8] D. Lee and Y. S. Park, "Implementation of augmented teleoperation system based on Robot Operating System (ROS)," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 5497–5502. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8594482

[9] M. Walker, H. Hedayati, J. Lee, and D. Szafir, "Communicating robot motion intent with augmented reality," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '18. New York, NY, USA: ACM, 2018, pp. 316–324. [Online]. Available: http://doi.acm.org/10.1145/3171221.3171253

[10] T. Chakraborti, S. Sreedharan, A. Kulkarni, and S. Kambhampati, "Alternative modes of interaction in proximal human-in-the-loop operation of robots," *CoRR*, vol. abs/1703.08930, 2017. [Online]. Available: http://arxiv.org/abs/1703.08930

[11] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, May 2009.

[12] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[13] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb 1992.

[14] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 1, pp. 3–15, Jan 2003.

[15] D. Puljiz, K. S. Riesterer, B. Hein, and T. Kröger, "Referencing between a head-mounted device and robotic manipulators," in *Proceedings of the 2nd Workshop on Virtual, Mixed and Augmented Reality Human.Robot Interaction, HRI 2019*, 2019. [Online]. Available: http://arxiv.org/abs/1904.02480

[16] K. Khoshelham, H. Tran, and D. Acharya, "Indoor Mapping Eyewear: Geometric Evaluation of Spatial Mapping Capability of Hololens," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 4213, pp. 805–810, Jun. 2019.