Silhouette-Based Segmentation of 3D Objects for Industrial Use Cases

Category: Research

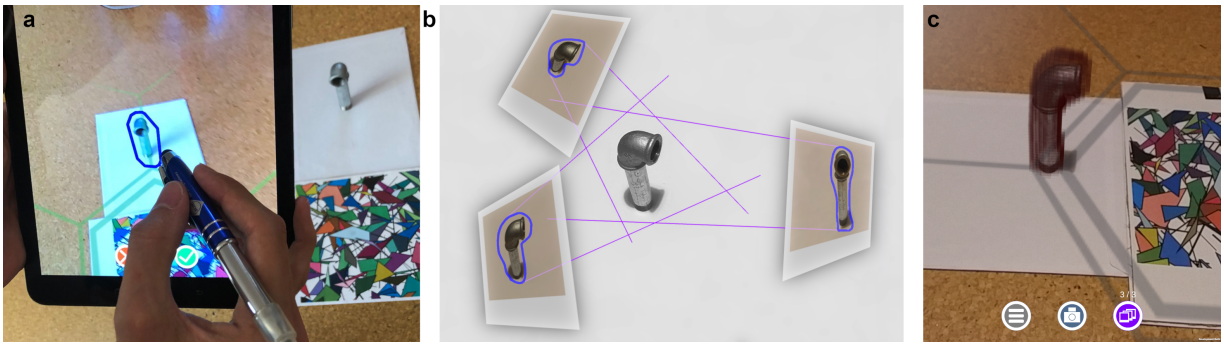

Figure 1: Overview of the proposed approach: a) The user surrounds the object on a small set of images from different perspectives. b) The algorithm computes an intersection. c) The volume is shown as a voxel grid.

Abstract

The process of scanning objects and creating digital copies of them gains increasing attention in the modern industry. But even though scanning itself is often automated, it is still necessary to define the volume which should be scanned manually. This poses various challenges in the environment of an industrial plant. A potentially hazardous environment with multifaceted structures requires a user with maybe no experience in 3D scanning to segment a volume accurately, which should then be comprehensible to the scanner. At least for this specific use case, common approaches to volume segmentation/selection fail to meet all necessary requirements for safety, efficiency, and functionality. Therefore, we developed an own algorithm, which segments the intended volume by intersecting the projection of user-drawn outlines on a small number of photos of the target object using augmented reality. Our implementation can successfully approximate various sizes of target volumes and delivers an appropriately detailed voxel structure. The pen-based input is easy and intuitive to use and requires the user to perform simple and clear tasks, while our algorithm takes the data, calculates the volume, and augments it into the scene. This gives visual feedback and shows flaws in prior annotations, which can then be resolved on the spot. Because of being designed around our use case, the application fulfills all requirements derived from the scenario. It delivers promising results and could be a simple, yet effective solution for our and similar problems.

Index Terms: Scenario-based design; Touch screens; Mixed / augmented reality

1 Introduction

Modern industrial facilities are characterized by constant change and development, not only in terms of goals and organization but also in their physical setting. Therefore, CAD models of industrial factories and plants can quickly become outdated. Concurrently, since companies are often distributed over extensive areas, responsible experts need to assess information about the facility remotely. This requires an up-to-date data basis.

In digital twins, a common first step to achieve an up-to-date data basis is to reconstruct the 3D shape of on-site machinery. This makes the required information available remotely with, depending on the scanning frequency, relatively current data. Moreover, scanning and segmenting scenes is a key part of enabling augmented reality (AR) navigation and guidance, which can be used to support workers on-site and improve the working process.

To perform the scanning of complex and huge machines or scenes two approaches exist. One is to capture the entire scene in a single scanning session and re-scan the entire scene on a regular basis. Alternatively, after an initial global scan, workers can scan small parts of the scene on demand and integrate them in the global scan whenever necessary. This paper discusses the iterative scanning approach henceforth.

For optical scene reconstruction, many approaches regarding scanning technology (i.e. active and passive sensors) and the performing actors (e.g. drones, robots or humans) exist. For all of these approaches, it is vital to have a human indicating which volume has to be scanned. Based on this volume, paths can be planned for autonomous scanning agents and visual feedback can be provided for humans. Therefore, the user needs to be able to select a volume in the scene which surrounds the intended object of interest. An interface is necessary which allows the spatial outlining of a volume and operates in the coordinate system of the real world. This interface has to be convenient to use, so that the worker performing the task does not have to be an expert in either 3D scanning or volume segmentation.

This research attempts to find a suitable interaction method for the use case of segmenting a volume for 3D reconstruction. After discussing a variety of existing mixed reality interaction examples in chapter 2, we describe the industrial requirements for volume segmentation in large facilities in chapter 3. In chapter 4, we illustrate our developed interaction method as well as possible limitations and future work. This is followed by a discussion of whether the approach fits the requirements in chapter 5.

2 Related Work

The field of 3D selection finds itself in various interactive scenarios. It ranges from selecting a single distinct object to selecting a subset of various objects to, as in the use case of digitally segmenting the space of a real world object, the selection of an area or volume. For selecting single, previously known objects in 3D space, Cashion and LaViola [1] propose an algorithm which selects an object based on a spotlight cursor. The spotlight's size increases with cursor speed and selects the nearest object to the camera within the spotlight cone. To correctly select any targets, this approach, however, needs prior information about the location of objects available.

Similarly, *Go'Then'Tag* [17] needs the entire data points being known beforehand. This algorithm intends to group the points into nodes according to their relation between each other, and then iteratively subdivide those nodes. For this, the authors originally

used an octree generation algorithm, which meant that the user first chooses between very large subsets and then refines the selection. This algorithm could be adapted to our use case by initializing the selectable objects as spatial voxels and then subdividing them iteratively. In theory, selecting and combining such voxel structures would allow arbitrary volumes.

As point clouds imply very similar selection methods compared to volumes, an algorithm by Dubois and Hamelin [2], called *Worm Selector*, could be adapted almost directly to our use case. It describes creating a mesh by iteratively extending a tube-like volume. Combining multiple of such worm meshes would also allow for arbitrary volumes to be selected. However, it requires the user to define an appropriate drawing plane in the scene to draw the contour on, which then extends the existing mesh.

Placing and rotating this drawing plane itself relates to another volume selection technique, which attempts to manipulate a volume to fit the desired target object. In practice, this could involve moving a bounding box in such a way that it eventually contains the area of interest. For this selection technique, it is necessary to find suitable algorithms to manipulate a 3D object in terms of the *canonical manipulation tasks* [5, Sec. 7.2.1] translation, rotation, and scaling. A considerable amount of research has been conducted on how to solve those tasks [3, 6, 8, 11, 13]. Those techniques deliver a limited approximation of the initial volume and are using multi-touch interaction metaphors [6, 11].

The previously mentioned algorithms operate with interaction techniques in 3d space. However, already since decades there is the approach "shape from silhouette" (e.g. [16]). This approach attempt to find the 2d silhouette of an object in multiple images and calculate the 3d shape of this object. Most of the used algorithms intend to work without or with minimal user interaction. In grabcut-alike segmentations, for example, the user gives stroke-based indications of background and foreground areas in an image [10]. Probabilistic approaches [4] allow for stable segmentation path refinements on parallel graphics cards, resulting in precise reconstructions for objects that are focused on the images and a sufficient distinction between foreground and background coloration.

3 Requirements

Our use case originates in a real-life setting with a concrete situational scenario: The user should be able to create a virtual 3D volume enclosing a real machine part within an industrial facility. From this, we can derive the following requirements for our application:

Visual guidance The user cannot be expected to know what type of input supports or obstructs a successful 3D reconstruction. Therefore, it is necessary to provide frequent visual feedback to the user, as this gives an estimate of which perspectives are expedient for taking photos and which ones are not. In general, visual feedback can strongly improve the quality of the resulting reconstruction when faced with a use case similar to ours [12]. This visual feedback can be achieved by adding a virtual overlay, similar to the drawn annotations introduced by Mohr, Mori, Langlotz, et al. [7]. The placement of this virtual overlay is the key scenario for our approach.

Speed The user's time in the facility is usually limited. Either due to economic reasons, for example, because the machinery needs to be stopped during that time, or because of the potentially hazardous environment and the avoidance of unnecessary risk for the user. Because of that, the application has to allow quick, yet reasonably accurate interaction. This means that the time required in direct proximity to the machinery should be reduced and tasks which can be done off-site can be delayed.

Spatial limitation The user's health must not be put at risk with the application's interaction method. This presupposes the reduction of unnecessary movement and prolonged time spent in dangerous positions or locations. For example, the device's position should not be used for continuous object manipulation [6, 8, 13], as the user might have to concentrate too much on the result being displayed on the screen than the surroundings.

Complex three-dimensional setups Many existing approaches to comparable use cases are not designed for application in industrial fields. In particular, they often assume the object of interest to be lying on a table or similar plane surfaces with a clear view and no occlusions. For industrial use cases, we have to expect objects to be installed with complex machinery around it (to understand the complexity, see for example [9, Fig. 9]). The object might be partially hidden behind other parts of the scene, which cannot be removed. In all those cases, the user should be able to mark the object easily, regardless of its location with respect to the scene.

Pen-based input During maintenance and repair tasks, the user might have dirty hands or be required to wear gloves, among other protective garments. Therefore, the situation demands a pen-based interaction technique. This adds further restrictions to potential interactions, ruling out possible multi-touch interaction techniques.

4 Approach

General concept For the use case, we developed an interaction technique in which we photograph the scene from multiple angles and mark the area of interest in each photo (see 1.a). With the device saving its relative location with respect to the scene, the projected markings create volumes that can be intersected with each other (see 1.b). This approach is inspired by Szeliski's visual hull carving algorithm [16]. We have adopted the voxel grid as well as the octree as they simplify and accelerate the intersection process. Naturally, the proposed algorithm is different in terms of how the silhouette is acquired, and does not differentiate between voxels completely within or completely out of the volume. In our version, the resulting volume is then augmented into the scene. The user can decide whether or not to refine the volume by repeating the process (see 1.c). Ideally, a volume is created that approximates the original object and can then be used as input to a scanner to scan the object.

Technical realization We used a voxel grid to represent the potential scanning volume of the scene. The device's location is measured during usage by tracking a stationary marker within the scene. The location and orientation are stored together with each image taken and the potential marking added. Photos can also be saved without a marking added to them and then edited later-on. The intersection process takes a collection of images with known position and orientation, and a user-created marker. When the intersection process is started, the voxels are being projected onto each image plane based on the known camera projection matrix and the stored device position. It is then tested against the area defined by the added marking. If all 8 projected corners of a voxel are within all marked areas of each photo, they are considered as part of the volume. If all 8 projected corners are outside all areas, the voxel is considered outside the volume and discarded. The remaining voxels, here called *border voxel*, are subdivided and the process is repeated with the newly created voxels. The subdivisions end when a certain number of iterations is reached or when the number of border voxels grows above a certain threshold. The latter exit condition automatically adapts the level of detail to the size of the voxel structure. Large structures reach the border voxel limit quicker and are therefore not subdivided as often as small structures. This creates more precise small volumes and coarse large volumes in order to reduce the required amount of computation for big volumes. This approach shares some similarities with the algorithm *Go'Then'Tag* [17] and can even be interpreted as a variation of it. Essentially, we start with an evenly distributed, infinite point cloud and start with very coarse subsets. Different to Go'Then'Tag, it is then not the user who iteratively selects the relevant subsets, but our intersection algorithm.

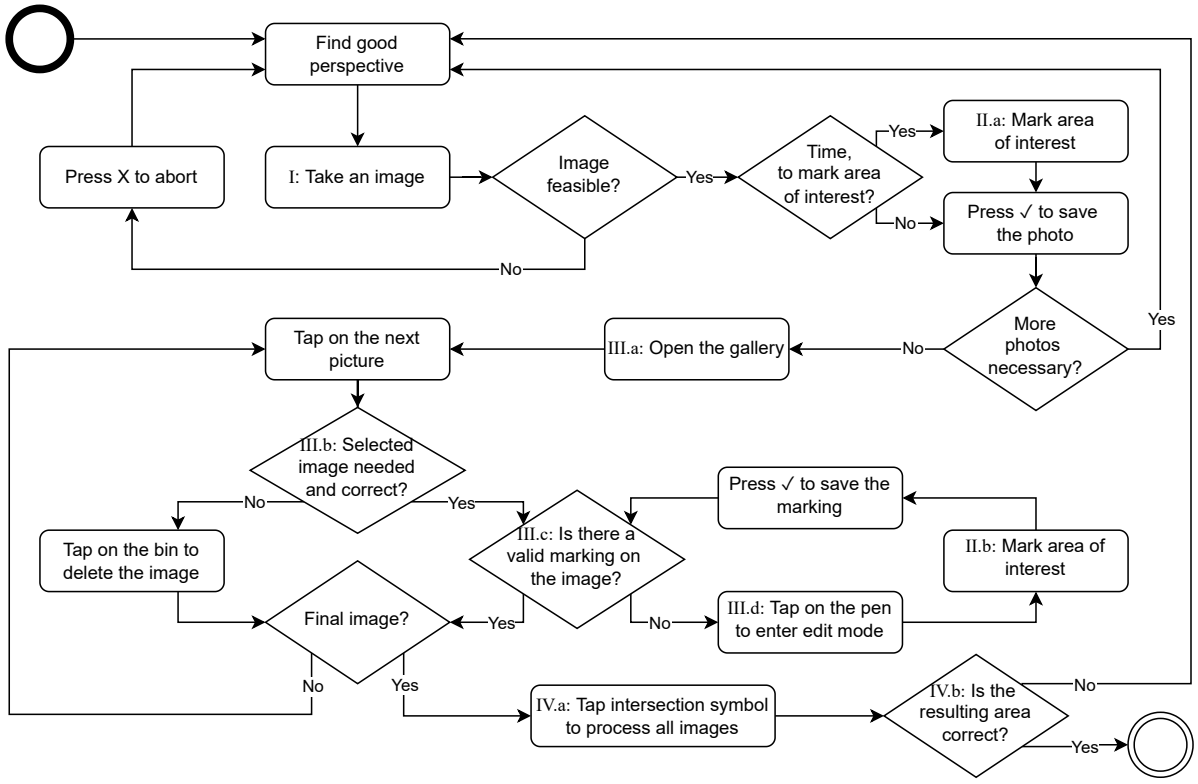

Figure 2: User-system interaction flowchart of the proposed approach

We basically automated the process based on the user's markings given in the images.

Application in the use case At the very beginning, the user is presented with the *AR view*, which consists of the live camera feed of the device and a small array of buttons. Once the marker is recognized, which is indicated by an aligned grid augmented into the scene, the user tries to get a good view of the object of interest. Via a corresponding button, the camera takes a photo for the user to review (see figure 2.I).

At this point, the user is in the *drawing view*, sees the taken photo, and can add a shape which outlines the object of interest in the image to it (see figure 2.II.a and 1.a). This marking is optional at this point and can be skipped. Unless the image is blurred or otherwise flawed, it can be saved and the user can continue with a new perspective. This process can be repeated several times. For example, three images might be created and stored that way.

Once the user deems the number of different perspectives sufficient, the *gallery view* can be opened (see figure 2.III.a). The gallery displays all stored images and the user is supposed to review all taken photos. For example, photos taken from very similar angles might be deleted (see figure 2.III.b). For photos with faulty or missing markings (see figure 2.III.c) the marking can be added and edited by pressing the edit button (see figure 2.III.d). Similar to the drawing view from the moment of taking the photo, the user can now add the marking retroactively to the image (see figure 2.II.b).

Once all images have been reviewed and edited or deleted, the intersection process can be started (see figure 2.IV.a). After the calculations are finished, the user returns to the AR view and can see the resulted volume in the scene. At this point, the volume can be visually compared to the real object of interest. Should there be parts of the augmented volume which do not contain any relevant element of the real object, they can be photographed and corrected with another added marking (see figure 2.IV.b and 3). This iteration can be repeated multiple times until the created volume fits the target well enough.

Limitations Several improvements are possible for the proposed implementation. For one, the algorithm could estimate a potential area to initialize the voxel grid automatically. The application requires a predefined area in which the initial coarse voxel grid is sampled and tested against the outlines. As of right now, this area's dimensions have to be defined with respect to the marker anchor by the user. A possible solution would be to calculate the approximated area filled by the intersection of all outlines explicitly. This would then require at least two images from an appropriately different angle to start the calculation process (see figure 4).

Another improvement could be to visualize the voxel structure in a better way. Right now, the voxels along the border of the area are visualized with semi-transparent cubes. This leads to occlusions, as several voxels can align with each other from certain viewing angles and are thus rendered on top of each other. With enough voxels in a line, the background cannot be seen anymore. To avoid this, only those sides of the voxel facing out should be rendered.

Furthermore, it can be hard for the user to realize that a tracking error has taken place. If, for example, the AR software calculates the device's position with a slight shift, parts of the target area might be removed, which the user intended to include. Naturally, it is difficult for the program itself to detect such errors. A possible solution in this case might be to provide additional information about the mesh creation to the user: Recalculating the intersected area after each newly added outline might be helpful to see which outline and corresponding photo cause wrong behavior. However, this would require a much faster calculation. Otherwise, this causes disruptive pauses after each drawing action. Another possibility would be to log for each border voxel for which annotated photo it was registered as a border voxel. This would allow the user to see for voxels, which would normally be expected to be inside the structure, by which

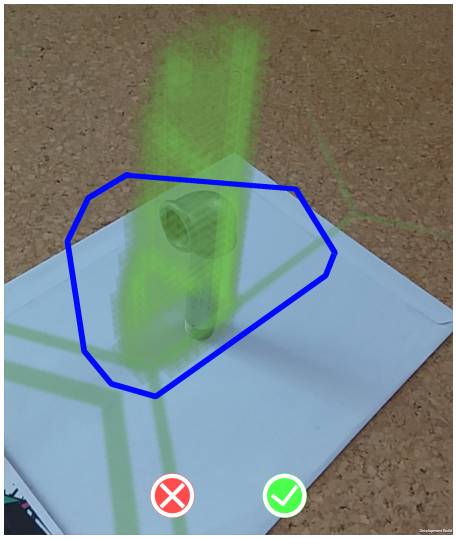

Figure 3: In this image, it is clear to see that the first silhouette marking(s) did not create a well-fitting volume (green voxel cloud) around the object of interest (metal pipe). By adding another image with a drawn outline (blue shape) which excludes the unwanted upper half of the volume, this can be corrected. At this point, another intersection calculation can be started, and the new marking will be incorporated together with the prior photos.

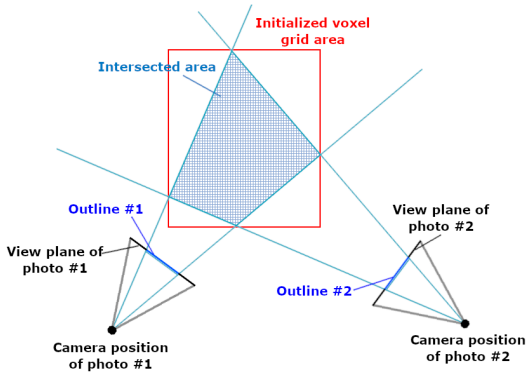

Figure 4: A possible improvement to our algorithm would be the automatic initialization of the voxel grid area by taking the intersected area of the projected outlines into account, here simplified to a 2D visualization.

photo outline they got intersected (see figure 6). This intersection must then be the same, which caused the voxels next to the false border voxel to be outside the intersected outline frustum.

5 Discussion

In order to visualize the segmentation process for the user, we display the resulting voxel structure directly over the target object in AR. This gives a direct estimate of how well the selected volume fits the desired form. Because the user can easily add corrections based on this visualization, it offers a direct guidance during the process. With the refinements being intuitive by *cutting unwanted parts off* (see figure 3), it suits user with and without a background in 3D reconstructions.

Due to our use case involving potentially hazardous machinery and environment, the short duration required to perform the corresponding task was an important requirement. To address this, our approach allows the user to separate tasks that require on-site presence from others. The user can enter the area with the object of interest, quickly take an arbitrary number of pictures, and then leave the area to perform the remaining tasks. The annotation on the images outlining the target object and the selection of valid images can be done at any point, as well as the final intersection step. Only to verify the result and to perhaps take additional photos, the site needs to be reentered.

Another safety hazard was the space the user requires while performing the on-site tasks. In our approach, we cannot limit this space in order to achieve an accurate result, as the algorithm requires different perspectives from which photos of the scene are taken. Yet, we can give the user more control over the spatial movement compared to, for example, with a device-based manipulation approach. There, the user might move to the side in order to manipulate an object in the scene indirectly. Hence, the user has to concentrate on two different motions: the movement of the object and the own movement. With our approach, the user just has to concentrate on their own movement. Technically, the device is not even necessary to move in order to find a good angle for a photo, as the camera will

Figure 5: Due to the visualization with semi-transparent cubes, many voxels rendered on top of each other can cause occlusions in the scene. Even though we omit rendering voxels inside the structure, the voxels which lie on the border of the marked area might align impractically (see the yellow-circled area). In such a case, the underlying, real-life structure might not be visible to the user anymore. As a workaround, we introduced a setting which can change the transparency of the rendered voxel cubes.

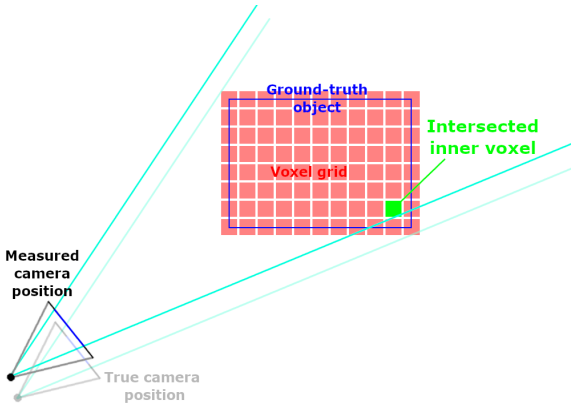

Figure 6: If the camera position is stored falsely (e.g., due to a wrong calculation in the AR software), the entire outline frustum shifts. This might cause the wrong voxels to be cut off, without the user knowing which one of the marked images caused this. A way to report this is to store for each border voxel which outline marking(s) intersected it. The user could then select a voxel, which should not be a border voxel, but an inner voxel, and see which image marking caused the surrounding part to be cut off.

capture the same view as the user has.

Apart from this, our proposed interaction does not require the object of interest to lie on a table or similar surface. Instead, it can highlight areas in any potential constellation. The only requirement for our application is a known point of reference in the scene, such as a marker. Because while the assumption that the object of interest lies on a table might be valid in many scenarios, we cannot expect this to be the case in our use case. Many parts of the machinery the user might want to capture can be integrated in complex industrial systems and be located anywhere in the 3D space. Our approach handles those cases equally well compared to objects on a ground plane and fulfills the corresponding requirement.

Similar use cases to ours have so far resulted in other research focusing strongly on 3D object manipulation techniques [3, 5, 6, 8, 13]. In our use case, this might be realized by prompting the user to move a virtual cuboid to enclose the object of interest. However, this would require the user to manipulate multiple parameters iteratively, which correspond to 3D translation, rotation, and scaling. We regard this approach to be less intuitive, especially for users who are inexperienced in the field of 3D object reconstruction or 3D software applications. However, our method has not been tested with the concerned workers yet, so this theory still requires further investigation and evaluation. But especially with regard to the requirements, which restrict the available input methods to pen-based ones, our approach fits the particular use case at hand very well. Our algorithm never requires more input than received from a single touchpoint, what makes it applicable for the usage with a simple touch-capable pen.

Another advantage of our approach is the easily achieved complexity of the resulting voxel structure, which can consist of a simple shape without holes. This enables it to be used as the sparse grid for explicit volumetric scene synthesis [14]. Furthermore, COLMAP [15] and similar 3D reconstruction pipelines often offer functionalities to shrink down the reconstructed area by adding binary masks for the input images. In our case, this binary mask can be derived from a projection of the resulting voxel structure back onto the view plane of the single photos.

In summary, we believe our prototype to deliver a three-dimensional marking, which, for our use case, is appropriately accurate and easy to create for the user. The pen-based input makes it applicable in the intended scenario and allows a decent amount of precision, while our interaction technique avoids any problems caused by the unavailability of multi-touch input. It successfully calculates and tracks the outlined area and shows it as a semi-transparent mesh as a visual overlay. This gives the user the possibility to see faulty parts of the result, which can then be corrected.

References

[1] J. Cashion and J. J. LaViola. Dynamic adaptation of 3d selection techniques for suitability across diverse scenarios. In *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 165–166. IEEE, 2014.

[2] E. Dubois and A. Hamelin. Worm selector: volume selection in a 3d point cloud through adaptive modelling. *The International Journal of Virtual Reality*, 17(1):1–20, 2017.

[3] E. S. Goh, M. S. Sunar, and A. W. Ismail. 3d object manipulation techniques in handheld mobile augmented reality interface: A review. *IEEE Access*, 7:40581–40601, 2019.

[4] K. Kolev, T. Brox, and D. Cremers. Fast joint estimation of silhouettes and dense 3d geometry from multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):493–505, 2012. doi: 10.1109/TPAMI.2011.150

[5] J. J. LaViola, E. Kruijff, R. P. McMahan, D. A. Bowman, and I. Poupyrev. *3D user interfaces: theory and practice*. Addison-Wesley usability and HCI series. Addison-Wesley, Boston, second edition ed., 2017. OCLC: ocn935986831.

[6] A. Marzo, B. Bossavit, and M. Hachet. Combining multi-touch input and device movement for 3d manipulations in mobile augmented reality environments. In *Proceedings of the 2nd ACM symposium on Spatial user interaction*, pp. 13–16, 2014.

[7] P. Mohr, S. Mori, T. Langlotz, B. H. Thomas, D. Schmalstieg, and D. Kalkofen. Mixed reality light fields for interactive remote assistance. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2020.

[8] A. Mossel, B. Venditti, and H. Kaufmann. 3dtouch and homer-s: intuitive manipulation techniques for one-handed handheld augmented reality. In *Proceedings of the Virtual Reality International Conference: Laval Virtual*, pp. 1–10, 2013.

[9] N. Navab, B. Bascle, M. Appel, and E. Cubillo. Scene augmentation via the fusion of industrial drawings and uncalibrated images with a view to marker-less calibration. In *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, pp. 125–133. IEEE, 1999.

[10] C. Rother, V. Kolmogorov, and A. Blake. "grabcut" - interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004.

[11] É. Rousset, F. Bérard, and M. Ortega. Two-finger 3d rotations for novice users: surjective and integral interactions. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, pp. 217–224, 2014.

[12] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3d model acquisition. *ACM Transactions on Graphics (TOG)*, 21(3):438–446, 2002.

[13] A. Samini and K. L. Palmerius. A study on improving close and distant device movement pose manipulation for hand-held augmented reality. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*, pp. 121–128, 2016.

[14] Sara Fridovich-Keil and Alex Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022.

[15] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] R. Szeliski. *Real-time octree generation from rotating objects*. Digital Equipment Corporation, Cambridge Research Lab, 1990.

[17] M. Veit and A. Capobianco. Go'then'tag: A 3-d point cloud annotation technique. In *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 193–194. IEEE, 2014.