# OpenARK Digital Twin: Object Tracking and Applications

HAN CUI*, YUNHAO LIU*, CHUANYU PAN*, ANJALI THAKRAR*, and TIANJIAN XU*

CCS Concepts: • **Human-centered computing → Human computer interaction (HCI)**;

Additional Key Words and Phrases: augmented reality, neural networks, datasets

**ACM Reference Format:**
Han Cui, Yunhao Liu, Chuanyu Pan, Anjali Thakrar, and Tianjian Xu. 2022. OpenARK Digital Twin: Object Tracking and Applications. In . ACM, New York, NY, USA, 8 pages.

## 1 INTRODUCTION

As the metaverse has attracted tremendous industrial and academic focuses, its related technologies, like the virtual reality (VR) and the augmented reality (AR), have also become prosperous areas with growing attention and investments. In particular, the AR Digital Twin is receiving exceptional interest due to its potential for various applications, such as manufacturing and human-computer interactions (HCI) [1]. Digital twin represents real-world objects in AR space and uses a digital camera to track and augment these objects. The fundamental nature of any digital twin algorithm requires the estimation of a object's pose, including position and orientation, with respect to the camera coordinates. Therefore, it is essential to develop precise, robust, and efficient algorithms for pose estimation.

Recently, researchers have devoted many efforts to develop aforementioned pose estimation methods. After deep learning is applied to various vision-related tasks and shows great versatility and flexibility, many studies utilizes deep neural networks and convolution-based neural nets on RGB images with depth information provided by systems like Lidar, stereo cameras, and SLAM [2]. These methods have high precision, demonstrates robustness towards variances like dim illuminations, and can be generally applied to downstream tasks like vision-based robotic grasps [3]. However, the digital twin problem has a different nature compared to grasp experiments and thus leads to more strict requirements for general performance: first, pose estimation for the digital twin problem requires lower computation costs in order to run on image sequences at a near real-time speed. Some state-of-the-art pose estimation algorithms might achieve high precision [4, 5] but are computationally expensive, which makes corresponding AR applications impractical. Also, a digital twin solution requires precisely aligned real-world object pairs, which might be geometrically symmetric. This is a challenging problem as it introduces ambiguity if the model does not learn from objects' textures for extra information. There are also extra requirements for millimeter-level errors and accurate depth maps, all of which contribute to the difficult nature of pose estimation in digital twin applications.

In this project, we therefore investigate the AR application of the digital twin problem, including preparing high-quality RGB-depth dataset preparation, traning state-of-the-art deep learning-based algorithm for stable pose estimation,

*Authors contributed equally to this research.


Manuscript submitted to ACM

and discovering corresponding downstream applications in AR development environments like Unity with appropriate interactions, visualizations, and UI designs. Figure 1 shows our working pipeline to incorporate parts mentioned above.

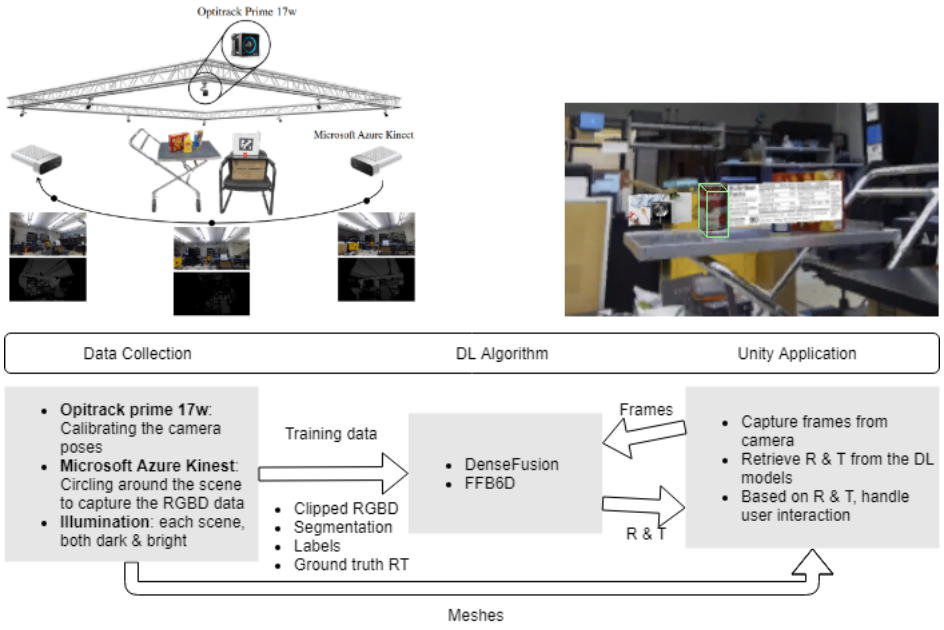


Fig. 1. Overview of the Proposed Project Pipeline

## 2 RELATED WORKS

### 2.1 Pose Estimation Datasets

Pose estimation algorithms are often trained and evaluated on RGB-Depth image data. Existing datasets include LINEMOD, LabelFusion, and Occlusion LineMOD [7–9]. However, these datasets are prone to systematic errors with respect to their depth data sensors and automated pose annotation pipelines. First, researchers often use stereo-depth sensors for data collection. Depth accuracy for stereo-depth sensors is only precise within close range and degrades quickly to centimeter-level for ranges beyond 1.5 meters [10]. Second, most datasets rely on automated pose label generation. For example, LabelFusion takes in a few human labeling inputs at sparse keyframes and relies on machines to compute object pose labels for the remaining frames of the video sequence [8]. Since there is no per-frame refinement being applied, there are visible errors of centimeter-scale in the dataset ground truths. Due to these errors, pose estimation models trained over these datasets often fail to achieve sub-millimeter accuracy.

### 2.2 Pose Estimation Algorithms

Existing pose estimation algorithms are powered by deep neural networks, which typically take in either RGB images or RGB and depth images as input [12, 13, 20]. The greatest success has been achieved when using both RGB and depth image modalities; the introduction of a depth map allows the DNN to reduce ambiguity in scaling, leading to more accurate estimations [3, 14]. The trade-offs in 6DoF pose estimation algorithms lie between efficiency and accuracy.

6DoF estimation has been approached using pixel-wise dense feature embeddings for RGB and depth inputs, which is architecturally simple and thus very efficient [3]. Using bidirectional fusion networks for keypoint matching has been found to be highly accurate but has low efficiency due to the complex architecture of the network [4]. Our work will utilize an existing pose estimation algorithm that evaluates a DenseFusion model using the ADD-S metric for symmetric objects and maximal point-wise alignment error criterion.

### 2.3 AR Interface and Interaction

The interaction interface is one of the primary bottlenecks for immersive computing. There are two main trends in AR interaction design [15]:

- device-assisted interactions with dedicated controlling devices
- tangible user interface(TUI) where [16] where an AR visual interface is coupled with the physical environment

Existing device-assisted interaction tools include gloves, computer mice with six degrees of freedom, joysticks, physically-tracked pens, and PHANToM™. While such devices provide users with direct haptic feedback, they are often bulky, expensive, and sophisticated to learn and use [17].

More recently, there have been various explorations into TUI which allows interaction such as bare-hand interaction and multi-modal interaction. Bare-hand interaction is based on computer vision techniques that detect and track bare features on the user's hand to determine the motion of the hand, and then perform pose estimation to understand its location and orientation within the scene. In this framework, users can press buttons and interact with objects using a single outstretched finger [17]. Another common technique is using a physical object, such as a paddle, that the user can hold and use to interact with virtual objects, in order to introduce a realistic sense of haptic feedback without the overhead of a device [18]. A key limitation of using a single-input modality to design TUI is that a user can only interact with objects that are visible in any given frame [19]. Thus, many of these techniques can be combined to create a powerful multi-modal interaction scheme [18].

Researchers often believe that the TUI is more intuitive than device-assisted interaction in AR applications as it enables people to use skills that they have developed throughout their everyday lives to manipulate physical objects [16]. However, there is no common consensus on TUI design principles that are able to address abstract or complicated manipulations for general purposes.

## 3 IMPLEMENTATION AND EVALUATION

As is mentioned in 1, the group started by collecting enhanced data samples with higher resolution and better annotation accuracy. To evaluate the dataset and meanwhile, to empower downstream applications, state-of-the-art 6DoF estimation machine learning models were trained. We deployed the trained models to support the development of a simulated AR application in Unity where we embedded our proposed 3D human computer interaction specifically designed for digit-twin scenarios.

Fig. 2 provides more details about the cooperation of the machine learning models and the Unity application. In general, the model as the backend digests RGB images and point clouds that are transferred from depth camera outputs. In order to bridge the model's expected inputs and the raw RGB plus point clouds data, the group incorporates an image segmentation algorithm. The image segmentation codes provide the model with a mask to expose objects of interest and hide unnecessary regions. The model returns a 6DoF vector, indicating the objects' rotation and translation (R&T) so that the Unity application can interpret the human being's interaction within the object's vicinity and attach

certain visual effects to the target objects. The following subsections will elaborate on the group's implementation regarding each part.

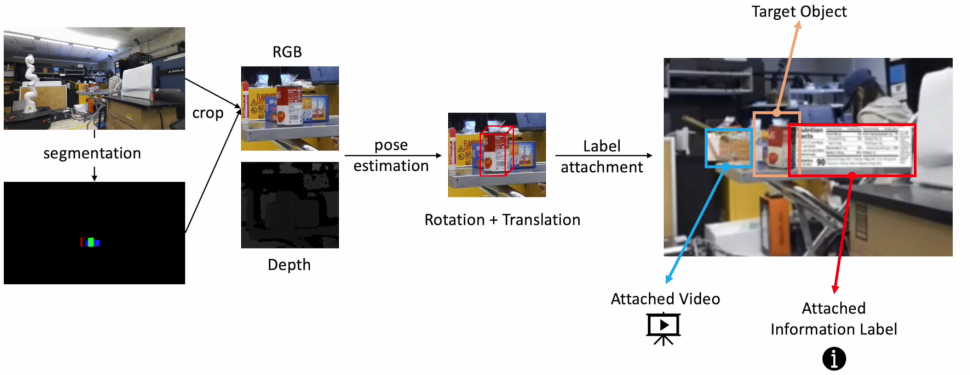


Fig. 2. Cooperation between Unity application and machine learning backends

### 3.1 The dataset collection

An ideal dataset for digital twin scenarios may require data from the following aspects for each sample: a sequence of calibrated and paired high-resolution RGB-D images captured from a different perspective of the same object combination, a ground truth relative transform from the camera to the target objects and segmentation masks for each image in the sequence. The calibration and pairing of the RGB camera and a depth camera might be quite troublesome and can affect the dataset quality significantly. Therefore, the group utilizes the Microsoft Azure Kinect, whose RGB camera and depth sensor are integrated together and calibrated well in advance. This ensures the RGB and depth information we obtain is of the same scale, from almost the same perspective, and with fixed displacement, providing consistency among the samples we collected for the dataset enhancement. The ground truth transform between the camera and the target objects is offered by the Optitrack prime 17w matrix. The variance of perspective is introduced by deploying the Azure Kinect on a trolley and moving it to circle the target objects during the data collection. The segmentation is manually annotated on the image sequences, which fulfills the last requirement for the dataset samples.

Robustness is also among the group's considerations as we aim to develop applications for real-life scenes. The group collected each scene in both bright and dim illumination. Also, for each scene in the dataset, the objects are combined randomly and placed so that mutual occlusion occurs. Some objects in our scenes are geometrically symmetric, such as the chicken soup can, which is of a column shape, or a cookie box. Their textures are included in the dataset as well, offering the machine learning models a feature dimension to resolve the ambiguity due to the symmetry. Fig 3 illustrates the objects we used in our comparison, where (d)(f) and (g)(i) are two pairs of objects that are geometrically identical but have different RGB textures.

### 3.2 Performance of 6DoF models on the dataset

Prior to the collection of this new dataset, the dataset as the benchmark for 6DoF estimation algorithms is the YCB [6]. We build our model based on a prior object pose estimation work, Densefusion [3]. We trained it on the dataset collected in Section 3.1 as the evaluation of the dataset and more importantly, as the backend for our downstream


(a) mac_cheese (b) spam_can (c) mustard (d) pink_expo_marker (e) cheerios_box (f) black_expo_marker (g) tomato_can (h) cheez-it_box (i) clam_can (j) pop-tarts_box

Fig. 3. Illustrations of the objects we used in the dataset.

| | ADD-S | |
|---|---|---|
| **Object** | **DTTD** | **YCB** |
| spam can | 87.9855 | 90.3031 |
| mustard | 84.3483 | 97.0279 |
| black expo marker | 79.2332 | 97.2883 |
| tomato can | 84.9898 | 96.6725 |
| cheez-it box | 86.2357 | 93.6725 |

Table 1. Performance of 6DoF estimation model on our DTTD dataset and prior YCB dataset

application exploration. Furthermore, we trained our own segmentation backbone based on UNet[20] to generate objects' segmentation masks. They are then used to crop RGB and depth images, which are the input of the pose estimation network. Table 1 compares the performances of the model on YCB dataset [6] and the enhanced DTTD dataset, the one we collected. The metrics we used is the average distance metrics ADD-S, which is defined as the following:

$$ADD - S = \frac{1}{m} \sum_{x_1 \in M} \left\| (Rx_1 + T) \left( \tilde{R}x_2 + \tilde{T} \right) \right\|$$

where the $R$ and $T$ are the ground truth rotation and translation, and the $\tilde{R}, \tilde{T}$ denotes a model's estimation of the rotation and translation. The $M$ is the point set sampled from the object's 3D model. The formula is designed to resolve the ambiguity when the matching between points clouds involve symmetry. Comparing the ADD-S between the model trained on YCB [6] and that on the enhanced DTTD, it can be concluded our datasets enable a model to produce a more accurate and robust estimation, especially for symmetric cases like the the expo markers.

### 3.3 A proposed UI design

Different from HCI happening in 2D screen space, where users have been accustomed to clicking with left keys or fingertips, scrolling with middle wheels or finger moving or scaling with two fingers, interactions with digit twins in a 3D space has not observe the formation of a standard agreement among hardware manufacturers, software developers, and end users. Comparing the different AR prototypes, manufacturers may define various types of interaction modes, such as what behavior constitutes a 3D click of a virtual digital twin or what gesture triggers a scaling of 3D objects. What can be learned from the development of 2D HCI, the group believes, is the abstraction as the bridge between the hardware support and the software logic. A common example is the "return button" on Android mobile devices. Some have physical buttons, some use virtual buttons, while others ask the user to slide fingers from the screen edges to the center. Regardless of the hardware design, the shared thing among these distinct behaviors is they all trigger the same Android UI event, which leads to all apps reacting in the same manner: returning to the previous view.

| User Behavior | Example UI Event | Mapping in 2D UI |
|---|---|---|
| Come close to target | Highlight notification | Zoom in |
| Physically touch | Pop-up ads, nutrition facts and other details | Click on the screen |
| Gaze | Showing warnings (e.g. “HOT”) or visual attractions | Put target at FOV center |
| Hover a virtual indication | Trigger corresponding pages, like Amazon | Click on the button |

Table 2. The group’s design of UI for a set of abstracted user behavior regardless of hardware implementations.

Therefore, as the enhanced dataset is proven to support future digital twin applications, we would like to explore the 3D HCI and UI design and try to establish a simulated digital twin application based on an abstraction of interactions rather than limited behaviors supported by certain hardware platforms. Table 2 shows the group’s idea. The first column provides an abstraction of behaviors, and the second gives desired UI effects in our application as examples of corresponding behaviors. Take the “gazing” as an example: no matter whether a ”gazing” event is detected via eye-ball tracing or simply by the camera’s angle, the same software application will react. The last column serves as a contrast and mapping between the proposed 3D interaction to their 2D screen-space counterparts. It may as well be helpful if some developers build cross-platform apps that are compatible with both AR headsets and normal smartphones or tablets.

On top of the interaction design, we focus our simulated application on a shopping scenario. Fig. 4 shows the screenshots from the application. When a user walks close to a target object, in this case, a chicken soup can, its silhouette will be marked out as a visual attraction for users. When a user starts to gaze at the object, a circle emerges with an instruction to tap it. A physical tapping will then trigger the nutrition details and commercials. If the user sees through a smartphone’s 2D screen instead of wearing an AR headset, according to Table 2, clicking on the screen area occupied by the target object will trigger the same outcome. Using a hand to reach over the virtual button ”buy me”, whose analog on a 2D screen might be a mouse hover or a finger pressing, will activate the button and launch an Amazon page.

## 4 LIMITATIONS

This platform has a few key limitations in its current implementation. These limitations exist in the application device compatibility, the dataset and data capture, as well as the real-time aspect of the application.

Firstly, the application is currently configured to work in a mobile AR format. The mobile format is inideal because it reduces user mobility; users only have one hard to interact with objects, and must be carrying and pointing their devices at objects in order to interact with them. A more user-friendly application format would be in an extended reality headset device, such as the Oculus Quest or Microsoft Hololens. Headset compatibility would require more complex UI interaction techniques. In particular, in order to build the most user-friendly and lightweight platform possible, it will involve digital twin hand pose and location tracking for interaction mechanisms like user location, real-world object selection, and potentially even eye tracking.

This model is also currently trained on data that has been collected by an Azure Kinect Sensor, which is a significantly richer form of depth data than that which comes from an iPhone’s LiDAR sensor. Specifically, iPhone LiDAR data is unable to accurately capture the depth of an object to the extent that is necessary for fine-tuned 6DoF estimation techniques, which limits the ability to use such data for precise ML tracking and estimation techniques. As such, the application implementation is currently based on a prerecorded video that the model has been trained on. In a real-world use case, the user would be capturing and processing video data in real-time. Thus, the image capture, model inference,

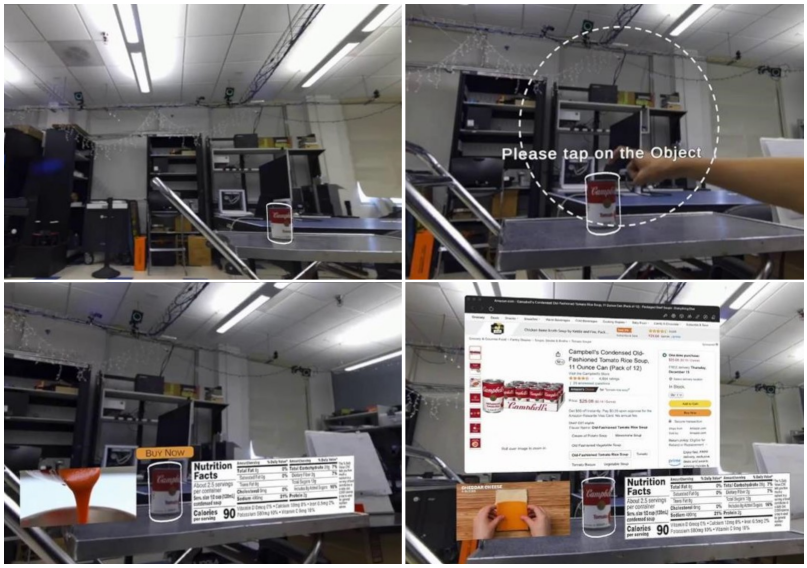


Fig. 4. Screenshots of the Unity Application: **Upper-left**: close to the object, it is circled but with no extra visual clues, attracting customers to concentrate; **upper-right**: gazing the object triggers an indication; **lower-left**: a physical touch pops up ads and nutrition facts; **lower-right**: hovering the "Buy" opens an Amazon page.

and AR object overlay processing would need to happen both on-device and on the order of milliseconds. This is not yet feasible given current hardware constraints and code architecture, and would require further optimizations on both laterals.

This application is also limited by the data. The model is trained on the Digital-Twin Tracking Dataset, which is manually captured in-house. This limits the amount of data available, as well as the type of data capture that is possible. Further, this limits the number and types of objects that can be captured – as of right now, the data contains a set of normal grocery objects with regular shapes, which might not be sufficient to cover every real-life objects.

Finally, the current model is trained on individual objects, rather than performing multi-object detection. As such, every object has its own trained model, which largely reduces modularity in the application's ultimate use cases.

## 5 CONCLUSION AND FUTURE WORK

This is a compelling implementation of an application can be extended for use in a variety of contexts. In the current implementation, the primary users of this project will be both retailers and consumers. It is envisioned to work in conjunction with a retailer to display information about a given product and other products in the store. It can be used in a variety of contexts and industries, encompassing everything from a user walking through a retail store to a user interacting with everyday objects. At a grocery store, it may display nutritional information and a cooking video of a can of soup, or comparable pricing at other retailers. It may also be used to display intelligent shopping suggestions, pricing, and advertisements for similar products. In other contexts, it may show instructions for how to use a coffee machine, or allow display instructions for building Ikea furniture, for example. As such, this application would work in conjunction with advertisers and retailers to create an immersive and compelling experience for users.

In the future, this project can be extended by making the application and UI interactions compatible with a headset. In order to make it more modular, the 6DoF pose estimation models may also be extended such that there is one model trained on all objects, rather than one model for each object. Finally, this project can be further optimized such that the inference would work for a real-time application.

The technological and social implications of this application are endless. This technology can generally be applied as an auxilary feature to any everyday task to allow humans to operate more effectively with the information and space around them. Digital twin technology, object recognition, and 6DoF estimation will be the backbone of futuristic technologies that will allow humans to interact with everyday objects and other humans in a fully immersive fashion.

## REFERENCES


[1] Liu, M., Fang, S., Dong, H., & Xu, C. (2021). Review of digital twin about concepts, technologies, and Industrial Applications. Journal of Manufacturing Systems, 58, 346–361. https://doi.org/10.1016/j.jmsy.2020.06.017

[2] Campos, C., Elvira, R., Rodriguez, J. J., M. Montiel, J. M., & D. Tardos, J. (2021). Orb-slam3: An accurate open-source library for visual, visual–inertial, and Multimap Slam. IEEE Transactions on Robotics, 37(6), 1874–1890. https://doi.org/10.1109/tro.2021.3075644

[3] Wang, C., Xu, D., Zhu, Y., Martin-Martin, R., Lu, C., Fei-Fei, L., & Savarese, S. (2019). DenseFusion: 6D object pose estimation by iterative dense fusion. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/cvpr.2019.00346

[4] He, Y., Huang, H., Fan, H., Chen, Q., & Sun, J. (2021). FFB6D: A full flow bidirectional fusion network for 6D pose estimation. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/cvpr46437.2021.00302

[5] He, Y., Sun, W., Huang, H., Liu, J., Fan, H., & Sun, J. (2020). PVN3D: A deep point-wise 3D keypoints voting network for 6DOF pose estimation. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/cvpr42600.2020.01165

[6] Calli, B., Singh, A., Walsman, A., Srinivasa, S., Abbeel, P., & Dollar, A. M. (2015). The YCB object and model set: Towards common benchmarks for manipulation research. 2015 International Conference on Advanced Robotics (ICAR). https://doi.org/10.1109/icar.2015.7251504

[7] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In Kyoung Mu Lee, Yasuyuki Matsushita, James M. Rehg, and Zhanyi Hu, editors, Computer Vision – ACCV 2012, pages 548–562, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[8] Pat Marion, Peter R Florence, Lucas Manuelli, and Russ Tedrake. Label fusion: A pipeline for generating ground truth labels for real rgbd data of cluttered scenes. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 3325–3242. IEEE, 2018.

[9] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. 2018.

[10] Hussein Haggag, Mohammed Hossny, D. Filippidis, Douglas C. Creighton, Saeid Nahavandi, and Vinod Puri. Measuring depth accuracy in rgbd cameras. 2013, 7th International Conference on Signal Processing and Communication Systems (ICSPCS), pages 1–7, 2013.

[11] Peng, S., Liu, Y., Huang, Q., Zhou, X., & Bao, H. (2019). Pvnet: Pixel-wise voting network for 6dof pose estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 4561-4570).

[12] Xiang, Y., Schmidt, T., Narayanan, V., & Fox, D. (2017). Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. arXiv preprint arXiv:1711.00199.

[13] Zakharov, S., Shugurov, I., & Ilic, S. (2019). Dpod: 6d pose object detector and refiner. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1941-1950).

[14] Mo, N., Gan, W., Yokoya, N., & Chen, S. (2022). ES6D: A Computation Efficient and Symmetry-Aware 6D Pose Regression Framework. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 6718-6727).

[15] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier and B. MacIntyre, "Recent advances in augmented reality," in IEEE Computer Graphics and Applications, vol. 21, no. 6, pp. 34-47, Nov.-Dec. 2001, doi: 10.1109/38.963459.

[16] Mark Billinghurst, Raphael Grasset, and Julian Looser. 2005. Designing augmented reality interfaces. SIGGRAPH Comput. Graph. 39, 1 (February 2005), 17–22, doi: 10.1145/1057792.1057803

[17] Y. Shen , S. K. Ong & A. Y. C. Nee (2011) Vision-Based Hand Interaction in Augmented Reality Environment, Intl. Journal of Human–Computer Interaction, 27:6, 523-544, DOI: 10.1080/10447318.2011.555297

[18] Nizam, S. M., Abidin, R. Z., Hashim, N. C., Lam, M. C., Arshad, H., & Majid, N. A. A. (2018). A review of multimodal interaction technique in augmented reality environment. Int. J. Adv. Sci. Eng. Inf. Technol, 8(4-2), 1460.

[19] Billinghurst, M., Kato, H., & Myojin, S. (2009, July). Advanced interaction techniques for augmented reality applications. In International conference on virtual and mixed reality (pp. 13-22). Springer, Berlin, Heidelberg.

[20] Huang, H., Lin, L., Tong, R., Hu, H., Zhang, Q., Iwamoto, Y., ... Wu, J. (2020, May). Unet 3+: A full-scale connected unet for medical image segmentation. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1055-1059). IEEE.