Enhancing Internet of Things Experience in Augmented Reality Environments

by

Yongbin Sun

B.S., Shanghai Jiao Tong University (2013)
S.M., Massachusetts Institute of Technology (2016)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Mechanical Engineering and Computation

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2020

Author . . . .
Department of Mechanical Engineering
May 15, 2020

Certified by . . . .
Sanjay Sarma
Professor, Department of Mechanical Engineering
Thesis Supervisor

Accepted by . . . .
Nicolas G. Hadjiconstantinou
Chairman, Department Committee on Graduate Theses

Enhancing Internet of Things Experience in Augmented Reality Environments

by

Yongbin Sun

Submitted to the Department of Mechanical Engineering
on May 15, 2020, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Mechanical Engineering and Computation

Abstract

Seamless perception of objects' physical properties, such as temperature, is a key to improving the way we live and work. Thanks to the rapid development of sensor technology, Internet of Things (IoT) is shaping our world by expanding digital connectivity to real objects. In this way, physical properties of objects can be effectively collected, processed, transmitted and shared. Yet, only being able to sense the surrounding environment is not enough: A user-friendly way to visualize information is also required. Today, Augmented Reality (AR), which overlays digital information onto physical objects, is growing fast, and has been adopted successfully in many fields. This thesis focuses on fusing advantages of various technologies to create a better IoT experience in AR environment.

First, we describe an integrated system to enhance users' IoT experience in AR environments: Users are allowed to directly visualize objects' physical properties and control IoT devices in an immersive manner. This system is able to localize in-view target objects based on their natural appearances without using fiducial markers, such as QR codes. In this way, a more seamless user experience can be achieved.

Second, existing handcrafted computer vision methods can estimate objects' poses only for simple cases (i.e. textured patterns or simple shapes), and usually fail for complex cases. Recently, deep learning has shown promise to handle various tasks in a data-driven approach. In this thesis, 3D deep learning models are explored to estimate objects' pose parameters in a more accurate manner. Hence, better robustness and accuracy can be achieved to support IoT-AR applications.

Third, standard deep learning training pipeline for object pose estimation is supervised, which requires ground truth pose parameters to be known. Manually obtaining such data is time consuming and expensive, making it hard to scale. As the last contribution, methods using synthetic data are studied to automatically train object pose estimation models without human labelling.

Thesis Supervisor: Sanjay Sarma
Title: Professor, Department of Mechanical Engineering

Acknowledgments

I spent 7 years for my graduate study here at MIT. This is a wonderful journey filled with joys, passion, friendship and hard work. There are many people that I would like to acknowledge for their support in this journey.

First and foremost, I would like to thank my advisor, Professor Sanjay Sarma, for his guidance throughout my PhD study. He inspired me with his broad knowledge and sharp intuition, and motivated me with his passion for making the world smarter and better. I am very lucky to work in Auto-ID Lab and have him as my advisor.

I would like to thank my thesis committee members, Professor Maria Yang and Professor Joseph A. Paradiso for their encouragement on my thesis. They gave me many insightful comments and suggestions during my committee meeting discussions.

I also want to thank my fellow labmates for our inspiring discussions and collaborations: Prof. Joshua Siegel, Dr. Rahul Bhattacharyya, Prof. Brian Subirana, Dr. Stephen Ho, Dr. Dylan Erb, Dr. Isaac Ehrenberg, Dr. Jason Ku, Dr. Pranay Jain, Partha Sarathi Bhattacharjee, Pankhuri Sen, Nithin Reddy, Dajiang Suo, Alexandre Urpi, Denise Tellbach, Shane Pratt, Fátima Villa, Prithvi Rajasekaran, Ferran Hueto Puig, Kevin Marty.

My thanks also go to my collaborators at MIT and other institutions: Dr. Ziwei Liu, Yue Wang, Prof. Justin M Solomon, Prof. Michael M Bronstein, Jonathan Dyssel Stets, Wiley Corning, Scott W Greenwald, Aniruddha Das, Xin Liu, Prof. Dahua Lin, Dr. Youzhi Liang, Georgios Pappas, Hongyu Wang.

Finally, I want to thank my mom, wife and friends for their support throughout my entire PhD life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Internet of Things

The Internet of Things (IoT) market is growing rapidly. IoT expands network accessibility to physical objects, and creates a large interconnected system where billions of objects can be sensed and communicated with. The term IoT was initially coined by Kevin Ashton, and Radio-Frequency Identification (RFID) was used at that time to link the Internet and individual physical objects [5]. Since then, the concept of IoT has become popular. In industry, Walmart started to replace traditional barcodes with RFID-based EPC codes for better supply chain management in the early 2000s [6]. In 2009, St. Jude Medical, a medical device company, first adopted IoT in healthcare applications by launching a wireless USB adaptor to collect patients' data and share to physicians [7]. In 2011, Nest company created the world's first IoT thermostat driven by machine learning [8]. In 2014, Amazon released Echo for the first time to enable a variety of smart home applications [9].

In academia, research effort has also been devoted to drive its development. With the convergence of many emerging technologies, such as sensors, embedded systems, machine learning, artificial intelligence, network connectivity and big data analytics, IoT industry has demonstrated its success for various applications. **Health-Care Applications:** IoT systems have been developed to monitor patient's biological factors and hospital environmental data in order to improve medical services [10]. Wearable hardware and low-cost sensors are usually deployed to create intelligent networks to facilitate the monitoring process [11].

Today, smartphones, as a low-power consumption and sensor-integration device, are also used for data collection [12]. **Environmental Applications:** IoT sensor systems are developed to monitor and control environmental factors, such as temperature and humidity. Users can access collected information and control parameters via web interfaces [13]. Beside data collection, sensor energy consumption efficiency, communication workload and system reliability are other key factors for IoT systems in this domain due to the increased scale [14]. In some cases, such as the agricultural production process, statistical analysis is further required to assess detected environmental factors, increase environmental monitoring efficiency, and achieve adaptive control [15, 16]. **Smart City Applications:** This is an important IoT topic, and its applications are directly related to our daily life. For city traffic network, when integrated with vision-based recognition techniques and advanced sensor technologies, IoT systems can help track vehicle and driver information and provide better resource management [17]. In the MIT CloudThink project [18], a concept called "Avacar" was provided to demonstrate how to monitor vehicles' real-time information and parameters by duplicating their physical parameters in the cloud. Other perspectives, such as security and efficiency, have also been discussed in [19]. Similar applications can be found in other scenarios, for example, tracking pets and person in constrained environments [20]. Studies [21] also show that mobile IoT systems can improve life quality of the elderly and disabled people in the house.

IoT has built a road to a connected world by enabling ubiquitous sensing and perception of our surrounding environment, and is now affecting many aspects of the way we live and work. In the smart home context, smart household devices can create an intelligent ecosystem, and every connected component can be easily monitored or controlled via smartphones. In the smart manufacturing context, maintenance operators can quickly check the condition and diagnostics history of machines through web-based applications. These smartphone- and web-based interfaces simplify the interaction between users and devices. However, when viewed from a different perspective, they are also becoming the bottleneck that may limit the efficiency and user experience of current IoT systems. This is especially true for indoor applications. For instance, even though devices are placed very close, the user still has to take out his phones, connect to all the target devices, and open their individual apps to perform some actions. Most of the time, the user also needs to switch back and forth between the smartphone and connected devices to check if they reach the desired status. For such short-range applications, users need more direct visual feedback to enhance interaction experience and efficiency.

1.2 Augmented Reality

Augmented Reality (AR) is an emerging immersive technology that can overlay digital content onto the real world, and helps people visualize and interact with the world better. Since the first functional AR system, Virtual Fixtures, which was developed at the U.S. Air Force's Armstrong Laboratory in 1992 [22], AR has been rapidly evolving over the past years due to the fast development and convergence of computer vision, connectivity, rendering techniques and mobile computing. Compared to standard interfaces, where users have to switch back and forth between control panels and devices, AR visualization tools create a more immersive interface and influence our daily lives as well: Google translate's augmented display [23] to improve productivity, AR GPS navigation app for travel [24], CityViewAR tool for tourism [25], etc. AR applications can be simply deployed on today's smartphones, just like Pokémon Go [26], but the user's view is limited only to the phone screen in this case. More immersive AR experiences can be obtained via Head Mounted Devices (HMD), such as Google Glass and HoloLens. For instance, when wearing the HoloLens, a 3D designer can create 3D models in the space by simply walking around the rendered models and modifying them. Such hands-free devices mix augmented digital content and real world objects in a more seamless manner, and virtual information appears more naturally to users.

1.3 Fusing IoT and AR

Despite AR and IoT having different objectives, they are naturally complementary to each other: IoT senses and shares real-world data, and AR visualizes sensed information by rendering virtual digital content. Fusing both technologies expands the application scope of each other, and an immersive IoT user experience can be created in AR environment. This would be perfectly exemplified by industrial operations in the factory. Today, the helmet worn by operators is mainly for safety, but it could potentially transform industrial workflows if it became “smarter". Work efficiency would be greatly increased if the operator was allowed to directly see all the desired data on the factory floor through the helmet’s safety glass. The data here can be real-time machine status information, such as pressure and temperature of process piping, which is useful for machine inspection; or an overlaid augmented manual/3D guide, which can instruct new operators to easily maintain or repair machine components. Such smart helmets have the potential to reduce the operation complexity and increase the connectivity between people, data, and machines.

To greatly unlock the potential of fusing both AR and IoT technologies, many technical challenges need to be considered, including security, robustness, efficiency and connectivity. In this thesis, we care about how to build a fully functional AR-IoT system and enhance the user experience. Existing IoT applications have demonstrated advanced systems for data sensing and sharing. Similarly, existing AR products have shown robust rendering pipelines for immersive visualization and interaction. But the bridging of these two technologies has not been fully studied, leaving it as a key to unlock the performance of any AR-IoT system. As pointed out in recent work [1, 27], properly rendering associated digital information of the target IoT device is important for such systems, since the digital information reflects physical properties of target objects in the view. Many existing AR systems use fiducial markers, such as QR codes, to identify and localize target objects, but these markers usually introduce unnecessary artifacts and require additional configuration. In this thesis, object pose estimation methods based on the target object’s natural appearance will be explored and added to proposed AR-IoT systems to enhance user experience. Specifically, traditional handcrafted methods and deep learning methods based on both 2D visual and 3D geometric information will be studied to improve the system step by step. Traditional methods, such as 2D visual descriptor SIFT [28] and 3D geometric descriptor FPFH [29], are manually engineered and fit well to small-scale dataset and simple cases, while deep learning methods, such as convolutional neural networks (CNN), automatically learn feature representation from training data and are capable to handle more complex cases. State-of-the-art deep learning methods have outperformed handcrafted methods by a quite large margin on many tasks, and this performance gain will be demonstrated in our proposed system.

1.4 Thesis Outline

The remainder of this thesis is structured as follows. Chapter 2 introduces a prototype framework that enables IoT applications in smart AR environment. Unlike existing AR systems, a marker-free rendering framework is described using handcrafted computer vision methods to enhance the user experience. The proposed system supports both visualization and interaction in a seamless and natural manner. Chapter 3 develops state-of-the-art deep learning methods for 3D data analysis. The proposed deep learning model is used in Chapter 4 to handle more complex 3D shapes that are difficult for traditional handcrafted methods to process. This increases robustness of the proposed AR-IoT approach for challenging industry applications. However, the proposed deep learning method is a supervised method that requires human labeled ground truth data, and obtaining such data is a time consuming and expensive process. Therefore, in Chapter 5, we further explore approaches to train deep learning pose estimation models on easily obtained synthetic data, while maintaining good performance on real testing data.

Chapter 2

Marker-Free AR-IoT Visualization and Interaction Framework

In this chapter, an AR-based IoT visualization and interaction framework is introduced to enhance the user experience with IoT. IoT devices and objects are becoming increasingly ubiquitous with diverse forms and functionalities. IoT has been making data collection easy due to rapid development of modern sensing and network technologies. Users can simply monitor and control wirelessly connected devices via smartphones or web applications. While this allows physical object to be accessed remotely, for many short-range IoT scenarios, direct visual feedback is more desirable to create better interaction experience between users and devices. In this chapter, a fully functional AR-IoT visualization and interaction framework is introduced .

2.1 Background

IoT has emerged as an active and promising field for data collection and analysis in order to support and improve our daily activities. Within the IoT context, devices and objects are becoming “smart" and ubiquitously accessible, enabling people to create a link between

This chapter is based on the author’s two earlier work: X-vision: An augmented vision tool with real-time sensing ability in tagged environments, in 2018 IEEE International Conference on RFID Technology & Application (RFID-TA) ©2018 IEEE, DOI: 10.1109/RFID-TA.2018.8552778; MagicHand: Interact with IoT Devices in Augmented Reality Environment, in 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR) ©2019 IEEE, DOI: 10.1109/VR.2019.8798053.

physical and digital worlds. By collecting and uploading an object's real-time status data, a digital avatar of the object can be created in the cloud and accessed by people from anywhere. Modern smartphones and web applications can easily control wireless connected devices. Yet, switching between multiple standalone applications become tedious and fails to scale well. Moreover, these interfaces usually require physical touchscreens to interact with, and are not suitable for hands-free interaction scenarios. Actually, for many indoor IoT application scenarios, context information and direct visual feedback are more desired to create better user experience. Today, how to efficiently and seamlessly interact with smart devices is still an active research area.

AR provides a convenient way to enhance IoT visualization, and to improve interaction experience by overlaying relevant information and digital content onto a view of the real world. For instance, we can control a smart light bulb by wearing a HoloLens [2], projecting a virtual control panel close to the light bulb, and manipulating it by navigating through the virtual panel. Compared to other hands-free interactive methods, such as voice control, exemplified by Amazon Echo and Google Home systems, which only allow for discrete tasks (i.e. "set room light to blue"), AR visualization tools allow for more granular tasks, like changing color within a color palette or adjusting light brightness value along a brightness bar. Also, voice control usually fails when more than one user interacts with one device at the same time, while AR control allows multiple users to simultaneously interact with the same device. Within an AR environment, it is even possible to include hand gestures to interact with IoT devices without touching them. In fact, a hand gesture is a practically natural way for interaction. For example, HoloLens provide two core gestures, “Air Tap” and “Bloom” for “click” and “go to Start menu” actions, respectively. In this chapter, an image-based hand gesture module will be demonstrated to show the capability of continuous and discrete user operations.

Currently, most existing AR systems use fiducial markers to obtain objects' identity and pose information, and render relevant digital content at correct position. While fiducial markers simplify the process, they add additional artifacts and unnecessary configurations to the system. To create a more seamless experience, objects' natural appearances can be used to obtain this necessary information. Both visual and geometric appearances are useful by providing color and structural cues, respectively. Algorithms have been designed for both cases in research community, but integrating them into an AR system has not been fully demonstrated. In this proposed system, object pose estimation based on their natural appearances is also integrated to improve AR-IoT user experience.

2.2 Related Work

2.2.1 Augmented Reality

AR technology enhances the communication and control between users and surrounding objects. Two main branches exist for AR related research. First, researchers attempt to design core algorithms and technologies for AR. Second, there is research in applying existing technologies to create novel applications.

Core Technology Research. AR systems need to acquire information about the surrounding environment in order to render digital content correctly. Fiducial markers are commonly used in AR systems for detection and pose estimation [30]. They are either attached to target objects to dynamically infer objects' poses [31], or to fixed positions in environments as static landmarks to estimate the camera's pose [32, 33]. Despite there exist many easy-to-use tools and Application Programming Interfaces (APIs), such as ARToolKit [34], ARToolKit+ [35], and ARTag [36], they require additional space to set up. On the other hand, 2D images [37, 38] and 3D shapes [39, 3, 40] can achieve this goal as well, and more details of related work will be given in the next subsection.

Interaction is another important area to support AR. In early AR systems [41, 42], users were only allowed to view digital content without modifying it. Later, researchers explored different ways to interact with AR systems. In [43], a magnetic tracker was used to create AR content, and in [44], tracked pens and tablets were used to modify digital shapes. Other forms of inputs were also explored as in [45]. Recently, hand gestures has been used to support AR systems [46, 47, 48, 49], such as pointing at or grasping virtual objects.

Application-Oriented Research. Research on applications attempts to expand AR applications and improve user experience in different ways, including education [50], tourism

[51], navigation [52], etc., using existing state-of-the-art technologies. In [53], a remote control system for smart homes was proposed. Their proposed system allows users to control rendered augmented objects on a display. [54, 55, 56] show the potential and advantages of integrating AR and IoT. In [57], a systematic study and comparison of technologies and systems was conducted to verify the effectiveness of AR serving as the interface of IoT.

2.2.2 Object Recognition and Pose Estimation

A key goal of this thesis is to design marker-free AR systems by using objects' natural appearances. Computer vision methods suit this purpose well.

2D Object Recognition. Object recognition based on 2D images is a long studied problem in computer vision. In the early phase, shape-based methods were developed to describe an object's appearance using low-level features, like corners [58] and edges [59]. But these methods cannot handle complex shapes and color variances. Later, appearance-based methods became widely used, because they can detect salient regions or points based on local visual features, also called as local descriptors. For these methods, local descriptors of test images are compared against those of reference images for object recognition. Popular methods in this direction include SIFT [60], SURF [38], HOG [61], etc. In recent years, deep learning methods [62, 63, 64] have evolved quickly to recognize visual patterns in a data-driven manner. Although good accuracy achieved by deep learning models, a large and diverse training dataset is required to avoid overfitting, making this technique ill-suited to small scale applications.

3D Object Pose Estimation. The pose of an object includes its position and rotation, with each containing 3 independent variables. Examples of pose estimation using point clouds are commonly found in robotics [65]. This process is usually a coarse matching process: extract local geometric features [66, 67, 68] and match correspondence [69, 29]. 3D descriptors can be derived from 3D shape coordinates [70, 71, 72, 73, 74, 29, 75], manifold space [76, 77, 78, 79] and high dimensional feature space [67, 80]. The estimated pose can be further refined using Iterative Closest Point (ICP) [81], which aims to minimize the difference between two point cloud sets.

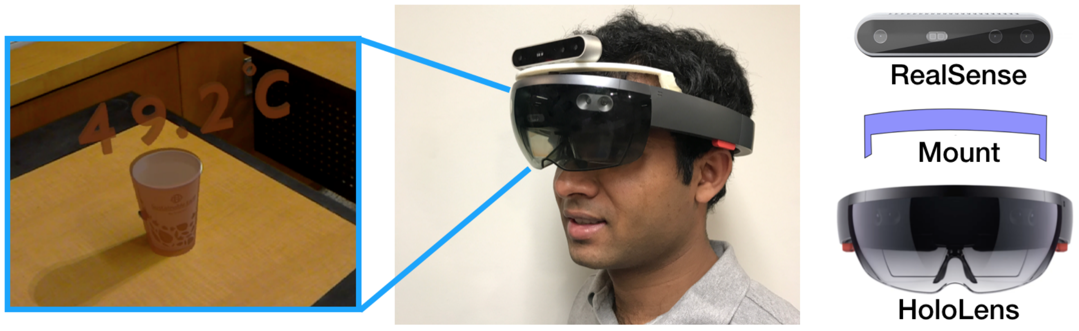


Figure 2-1: A working example of AR-IoT visualization system [1]. Left: Nithin wearing the system sees a cup with overlaid temperature information. Right: System components: an Intel RealSense D415 RGB-D camera is mounted on a HoloLens. The camera captures color and depth image sequences, which are used to identify and localize the in-view target object. Based on this information, the HoloLens projects digital content to the user properly as shown.

2.2.3 Using RFID

RFID is a convenient way to gather information, but poor at inferring location. RFID is used in the proposed system to collect objects' physical status data. Some relevant work is summarized here. RFID, as an identification technique, was primarily developed for supply chain, and has recently been adopted to many other related areas. Many existing research efforts create smart environment using RFID tags. Examples can be found in sensing, control [82] and even gaming [83]. Emerging technologies, such as computer vision and AR, have been fused to explore opportunities beyond ID in RFID and boost industrial values of RFID market. Among them, integration with AR is a promising area. Early study [84] used RFID to interact with physical objects via a smartphone game. Another work [85] used RFID tags to create an AR interface and improve information visualization. In [86], the authors explored the possibility of fusing AR and smart tags for teaching. These studies show an increasing interest in the research community to explore AR applications using RFID tags.

2.3 AR-IoT Visualization System

In standard IoT application cases, the physical information of surrounding objects or devices is collected and displayed to users through smartphones or web pages. Users usually need to check back and forth between their smartphones and nearby devices, which degenerates user experience and work efficiency. Therefore, a more direct user-device interaction method is needed. Motivated by this, we are seeking an approach that allows users to directly “see" associated information together with the target objects. In this way, a more efficient and seamless interaction experience can be achieved. Working towards this goal, we first describe an AR-based IoT visualization system in this section, as shown in Figure 2-1. This system demonstrates IoT user experience enhancement in an AR environment for indoor applications, especially within tagged smart environments.

The proposed system supports both sensing and visualization. It captures color and depth images of the surrounding environment using a depth camera, from which the target object in the view can be identified and localized. Also, the system can collect target objects' physical properties using RFID tags. Information collected from both sources are fused for proper rendering through a head-mounted HoloLens. The complete working flow of the system is shown in Figure 2-2, and details are given in following subsections.

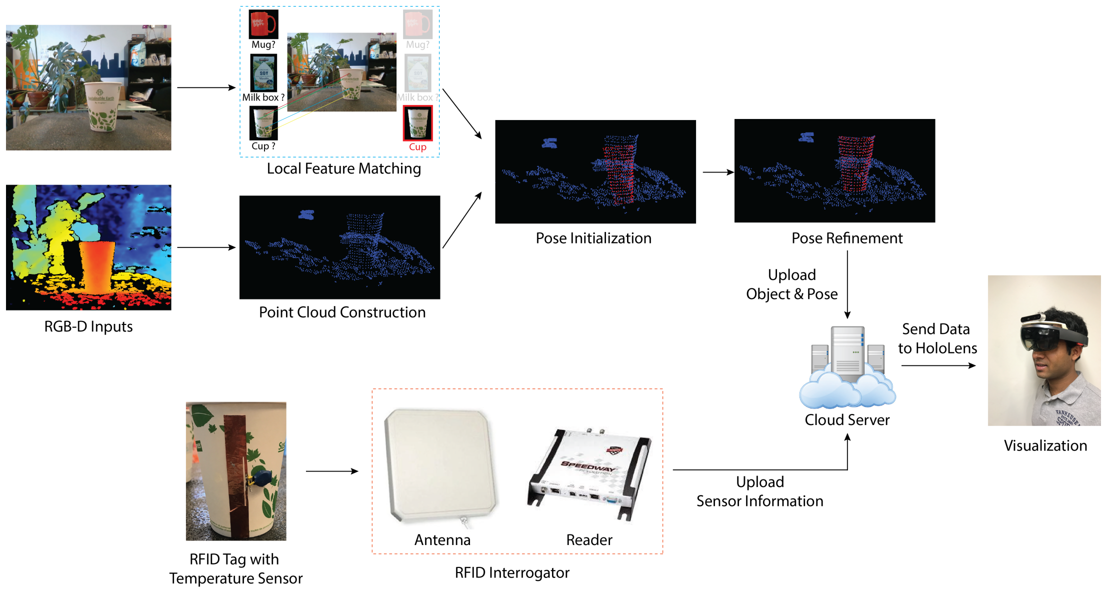


Figure 2-2: The pipeline of the proposed AR-IoT visualization framework [1].

2.3.1 Object Identification and Pose Estimation

A key factor of an AR-IoT system is to render information at correct position accurately. Unlike gaming, where digital content, such as points, has less association with the real world and can be rendered just on the ground or at arbitrary positions in space, the digital content of AR-IoT applications is directly associated with the target object in the view and should be rendered close to it. For example, it would cause confusion if the sensed temperature of a cup is rendered close to some other object. Also, there are usually multiple objects in the database, with each containing its own sensed information, and we only want to render the information associated with the in-view object. To achieve this, two tasks need to be completed: identifying the target in-view object so we know *what* information to render, and localizing the target object so we know *where* to render the information.

In this system, we use a depth camera (Intel RealSense D415) to capture color and depth images, from which we can identify and localize the target object if it exists by running image processing algorithms on a local server. This depth camera is mounted on the HoloLens via a custom mount provided by [87], and faces in the same direction as the HoloLens (Figure 2-1). The HoloLens has built-in cameras for capturing color and depth information separately, but running multiple threads concurrently on the HoloLens slows down its real-time performance. Smooth rendering performance is obtained by splitting computation threads onto different devices. Unfortunately, the depth camera of a HoloLens cannot provide as accurate depth information as the RealSense camera, and this will decrease the localization accuracy if it is used. We next describe how our system achieves in-view target object identification and localization.

Object Identification. We use local feature-based object recognition methods [38, 60] for this purpose, because they are suitable to small-scale databases. Local visual features from both reference objects and scene images are extracted and matched to recognize the target object in the view. The number of matched local feature pairs between the scene image and each reference object is counted. A target object is recognized from the scene if the number of matched feature pairs between it and a reference object exceeds a predefined threshold. In our setup, we assume one scene image only contains at most one target object.

Therefore, if there are multiple reference objects that pass this threshold, the one with the most matched pairs is selected. In this system, SURF [38] is used to compute local features, because compared to other candidate methods, such as SIFT [28], SURF achieves a good trade-off between speed and accuracy.

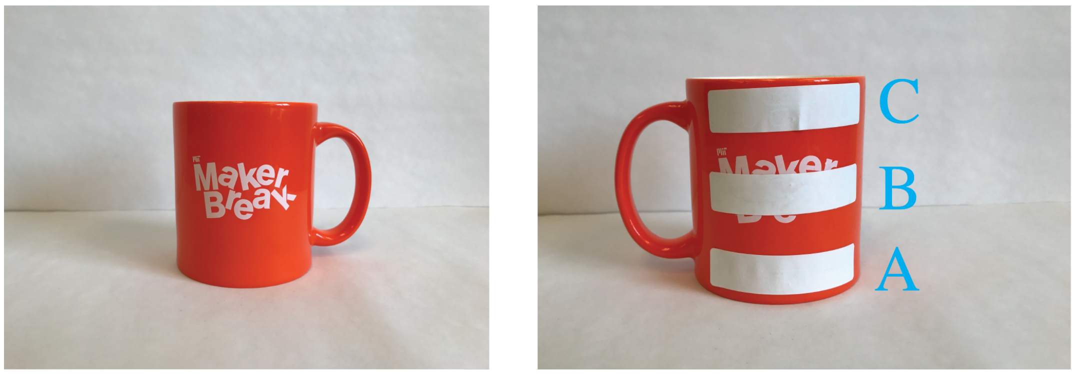
Object Localization. Once the in-view object is identified, its pose parameters need to be estimated in order to render digital content properly. Pose parameters consist of position and rotation parameters. We use the 3D point cloud data, which can be reconstructed from the depth image, for this purpose. The reconstructed point cloud is then aligned with the recognized object's template point cloud data. In our implementation, 3D point clouds of each reference object from multiple viewpoints are kept as its template point clouds. Among many point cloud alignment algorithms, we choose to use the well-known Iterative Closest Point (ICP) algorithm [81] because it achieves good alignment in a quick manner. The performance of ICP heavily relies on pose initialization. Our system finds a good initial pose by moving the reference object's template point cloud to the centroid of 3D points of matched points in the scene. Once the template point cloud is properly initialized, its pose parameters are then refined using ICP. Mathematically, the estimated pose can be represented using a 4×4 matrix, $\mathbf{M}_{pose} = \mathbf{M}_{ini} \cdot \mathbf{M}_{icp}$, where $\mathbf{M}_{ini}$ is the initialization pose transformation matrix, and $\mathbf{M}_{icp}$ is the pose refinement transformation matrix when using ICP. All the transformation matrices share the same the format:

$$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$$

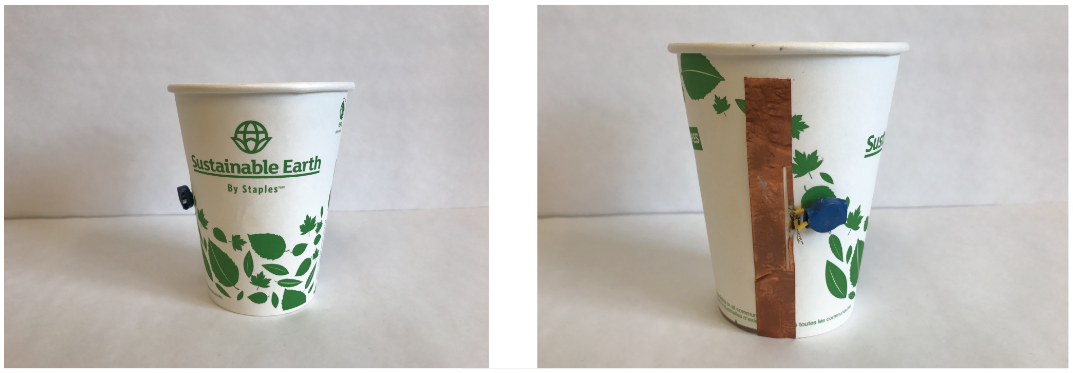
, where $\mathbf{R}$ is a 3×3 matrix for rotation, and $\mathbf{t}$ is a 3×1 vector for translation. More technical details can be found in [88].

2.3.2 RFID Sensing

The previous subsection describes what information to render and where it should be rendered depending on what and where the in-view object is, but the information content is not determined. In this subsection, we will introduce how to sense physical properties of



(a) Tag-sensors for water level sensing



(b) Tag-sensors for temperature sensing

Figure 2-3: Tag-sensors for sensing object properties [1]. (a) Water level sensing using paper tags; (b) Temperature sensing using custom tag with EM4325.

target objects so that the information to be rendered is instantiated. For demonstration purpose, we consider an office space equipped with RFID infrastructure in this study. This RFID setup helps us easily collect physical information of objects within the environment. Specifically, we conduct water level and temperature sensing of office water containers (i.e. mug and cup), because in a typical office scenario, accidents such as scalding by hot water can be avoided if we know the water level and temperature. Other sensing options, such as WiFi- or Bluetooth-based sensors, can be used here to collect and synchronize objects' physical information as well. We just found it easier for us to deploy an RFID system, and they are much cheaper for scaling. In the remainder of this subsection, the RFID setup and both sensing cases will be illustrated.

RFID Setup. The space is equipped with an Impinj Speedway Revolution RFID reader, which is connected to multiple circularly polarized Laird Antennas. For the two use cases: Smartrac's paper RFID tags with Monza 5 IC are used to sense water levels based on the antenna's impedance shift, and custom designed tags with the EM 4325 IC are used to sense temperature via the IC's on-board temperature sensor. The Low Level Reader Protocol (LLRP) is implemented using Sllurp (Python library) to enable the communication between the RFID reader and distributed tags.

Water Level Sensing. Water-level sensors are built on the concept of relating the detuning of the tag's antenna in the presence of water in the neighborhood of the tag [89]. In our study, tags are used to sense water-levels on common household objects such as a coffee cup and ceramic mug. When the background dielectric for a tag is air, the backscattered signal strength is at the maximum, so the tag can be easily detected. When the background dielectric for a tag is water, the antenna is significantly detuned due to the change in background dielectric, resulting in an unresponsive tag. This concept helps detect discrete levels of water in the container. Four states: empty, low, mid, and high, are defined and illustrated in Table 2.1. Figure 2-3 (a) shows the level sensor labels implemented on a standard ceramic coffee mug.

Table 2.1: Water Level Indication

Status	**A**	**B**	**C**
Empty	✓	✓	✓
Low	✗	✓	✓
Middle	✗	✗	✓
Full	✗	✗	✗

Temperature Sensing. The temperature sensor is implemented using EM Microelectronics's EM 4325 with on-board temperature as the RFID IC. Figure 2-3 (b) shows such an augmented tag attached on a cup. Temperature from this IC can be measured in both passive and semi-passive modes [90]. In the passive mode, the tag needs to be detectable in the working range of a reader antenna, while in the semi-passive mode, the battery can power the IC. The IC's temperature measurement is triggered by writing random information into a specific word of the user memory bank. The IC updates this word with the measured temperature from the on-board temperature sensor. The updated temperature is detected by reading the word again. The Sllrup library is used to control real-time temperature sensing between -64 and 64 Celsius degrees.

2.3.3 Augmented Visualization

After obtaining all the necessary information: the target object's identity, pose parameters and physical properties, augmented information (i.e. CAD model) can be retrieved and rendered to the user at correct position. Note that the object's 3D pose is estimated based on the depth camera, but the HoloLens rendering system requires the pose in the world coordinate system. So a series of transformations are needed to convert the pose representation. The transformation is expressed as:

$$\mathbf{M}_{pose}^{world} = \mathbf{T}_{HoloLens}^{world} \cdot \mathbf{T}_{dep\_cam}^{HoloLens} \cdot \mathbf{M}_{pose}^{dep\_cam}$$

, where $\mathbf{M}_{pose}^{dep\_cam}$ and $\mathbf{M}_{pose}^{world}$ are the 3D poses in depth camera and world coordinate system, respectively, $\mathbf{T}_{dep\_cam}^{HoloLens}$ maps the pose from the depth camera coordinate system to the HoloLens coordinate system, and $\mathbf{T}_{HoloLens}^{world}$ maps the pose from the HoloLens coordinate system to the world coordinate system.

2.4 Evaluation of Visualization System

2.4.1 Sensing Results Visualization

The system's performance on both water level and temperature sensing is first demonstrated to qualitatively assess the system's performance.

For water level sensing, both single- and multi-object cases are tested, and their results are shown in Figure 2-4 and 2-5, respectively. Template models of recognized objects are rendered in the scene using their estimated pose parameters. To indicate different water levels, the projected color of template models is changed accordingly. From the figure, we can observe that the system accurately aligns 3D models to recognized objects, and the

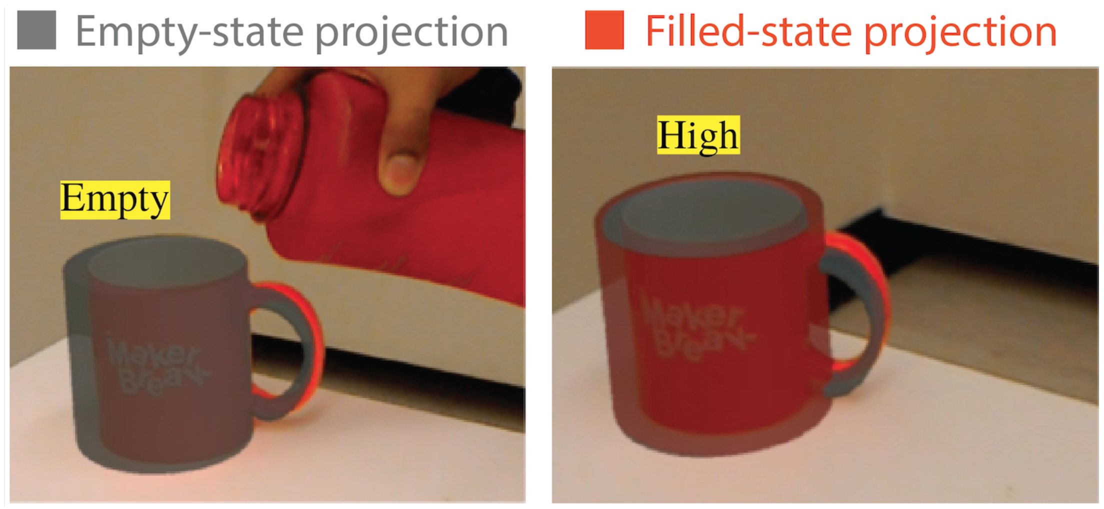


Figure 2-4: Water level sensing results: The HoloLens rendered results before and after water is added into a tagged mug [1].

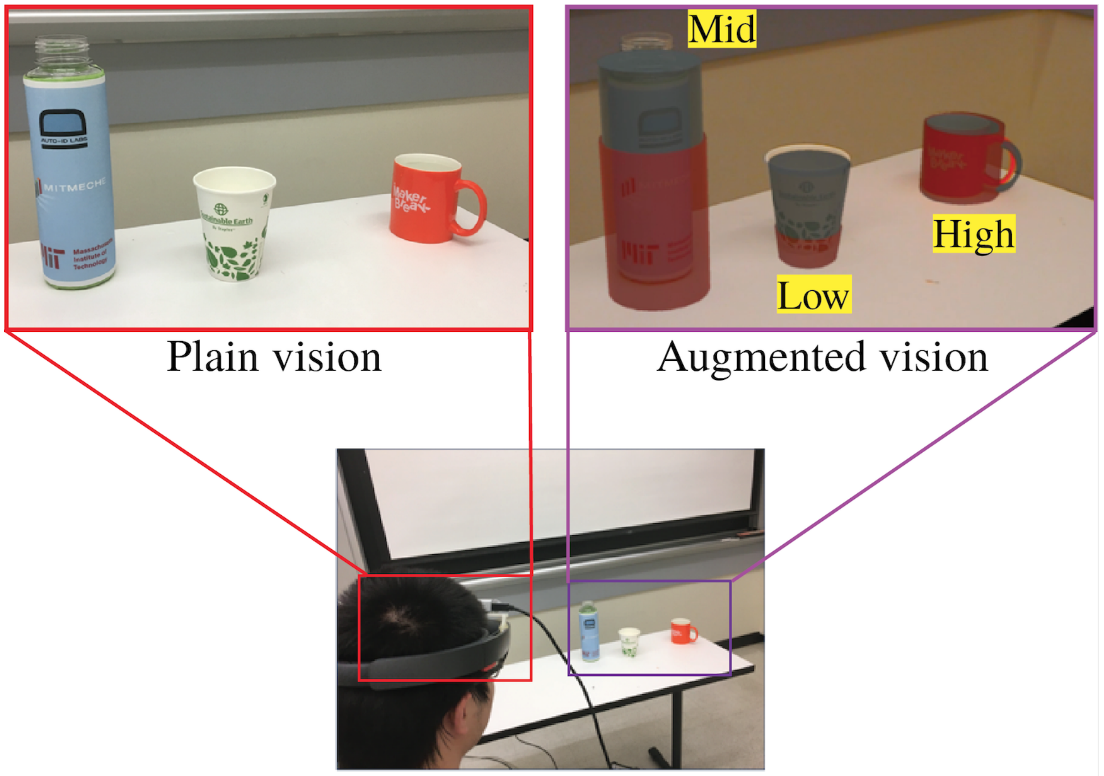


Figure 2-5: Water level sensing results for multiple objects at the same time [1].

detected and actual water levels match.

Examples of temperature sensing are presented in Figure 2-6. We recorded a video clip of the complete process of adding hot water into the cup. The sensed temperature changes

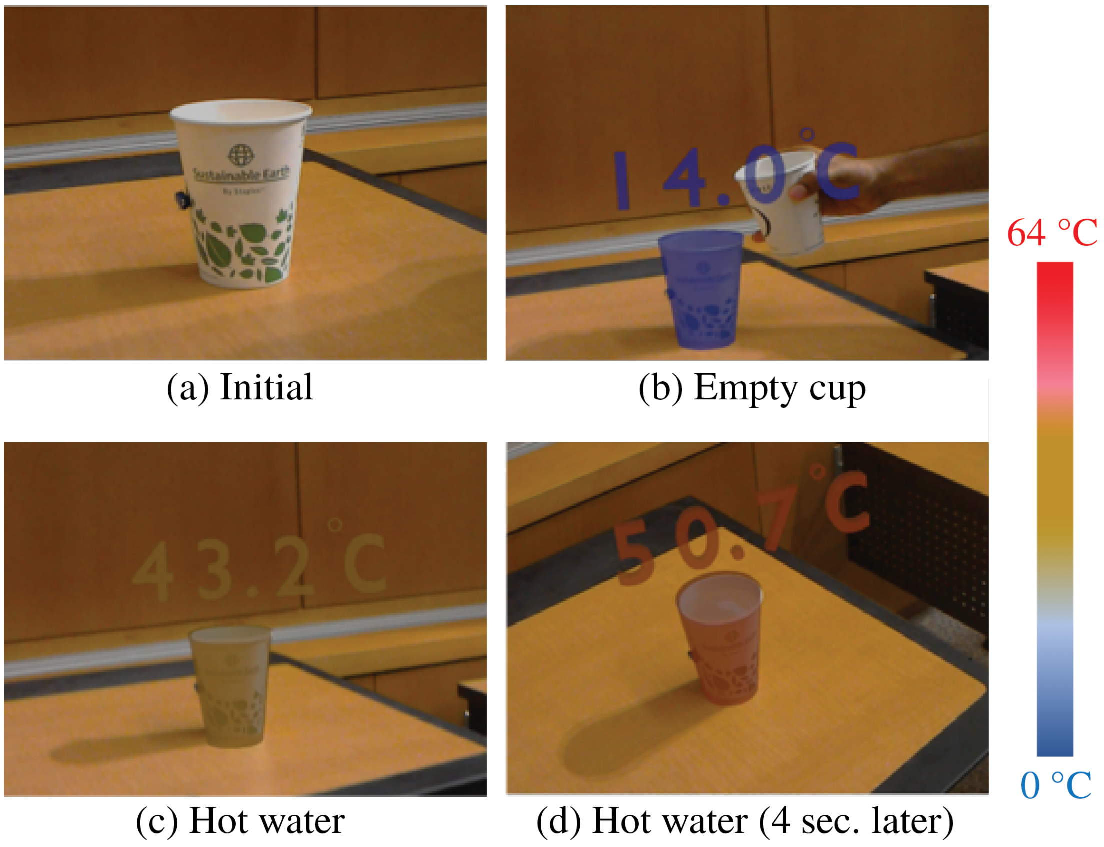


Figure 2-6: A sequence of temperature sensing results after hot water is added [1]. Different temperatures are reflected according to the color code on the right.

from low to high, and this is reflected as the color change for different temperature degrees. The experiment starts with a room temperature for an empty cup. After hot water is added, the system detects a series of temperature changes. We can observe visually appealing results.

2.4.2 Pose Estimation Evaluation

Object pose estimation is important for digital content rendering and user experience, thus the pose estimation pipeline implemented in the system is evaluated. Here, we evaluate the system's recognition accuracy, pose estimation quality and running time. The Fast Point Feature Histograms (FPFH) algorithm [29] is used as the competing method to identify the in-view object and estimate its pose. For pose estimation, the scene point cloud is aligned with each reference object's template point cloud, and the reference object with

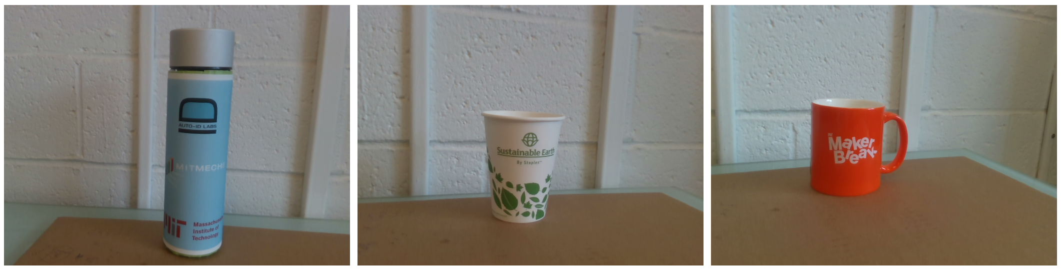


Figure 2-7: Test objects [1].

best alignment is chosen as the identified object in the scene. FPFH supported by local visual features is also implemented as another competing method. This “Local Feature + FPFH" method first computes local visual features to recognize the target object and then estimates its pose using FPFH.

Three different objects, a water bottle, a coffee cup and a mug, are used for testing. Five images from different angles at a range between 0.3-0.5 m are collected for each object (examples are show in Figure 2-7). We choose this range because objects’ texture patterns can be relatively easy to recognize, and their point clouds can be reconstructed with high quality.

Recognition Evaluation. Table 2.2 lists recognition accuracy comparison for different methods. We can observe that the local visual feature enhances the recognition accuracy when comparing “Local Feature + FPFH" and FPFH methods. We argue that the improved accuracy is due to the rotational invariance property of SURF features. Both our method and “Local Feature + FPFH" achieve the same performance.

Table 2.2: Recognition Accuracy [1].

	Bottle	**Cup**	**Mug**	**Avg.**
FPFH	1/5	3/5	3/5	7/15
Local Feature + FPFH	**5/5**	**5/5**	**5/5**	**15/15**
Ours	**5/5**	**5/5**	**5/5**	**15/15**

Pose Estimation Evaluation. We use the average distance error between two point cloud

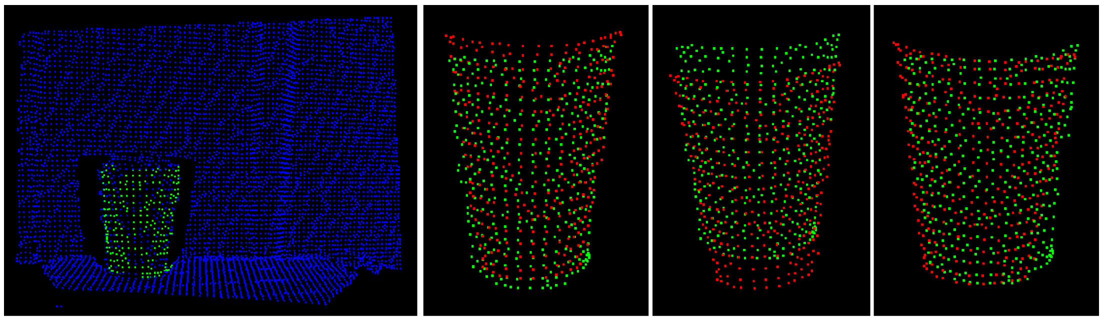


Figure 2-8: Point cloud pose estimation examples of different methods [1]. Blue for scene points, green for reference object points, and red for transformed object points.

sets to evaluate pose estimation accuracy, and this metric is defined as:

$$\mathscr{E} = \frac{1}{n}\sum_{i=1}^{n} \|t_i - p_i\|_2$$

, where t_i is the i^{th} point in the reference point cloud (green points in Figure 2-8), and p_i is the closest point in the transformed object point cloud $\{p\}$ (red points in Figure 2-8) to t_i, such that $p_i = \arg\min_p \|t_i - p\|_2$. Table 2.3 shows the distance error that is averaged across all correctly recognized in-view objects. Local feature-based methods perform better on average, while our method behaves slightly worse than the "Local Feature + FPFH" method. This is due to the fact that ICP can easily get trapped at local minimas. We also visualize pose estimation results for different methods in Figure 2-8.

Speed Evaluation. Running time is another important factor affecting a system's performance. Therefore, the speed of different methods is evaluated, and we report their average running time in Table 2.4. Despite our method shows a slight worse average distance error than "Local Feature + FPFH" method, it runs significantly faster.

Table 2.3: Average Distance Error [1].

	Bottle	**Cup**	**Mug**	**Avg.**
FPFH	0.0069	0.0059	0.0094	0.0075
Local Feature + FPFH	**0.0057**	**0.0055**	0.0087	**0.0066**
Ours	0.0088	0.0074	**0.0070**	0.0077

Table 2.4: Pose Estimation Time (*sec.*) [1].

	Bottle	**Cup**	**Mug**	**Avg.**
FPFH	4.603	4.502	4.590	4.565
Local Feature + FPFH	2.719	0.493	1.329	1.514
Ours	**0.055**	**0.015**	**0.074**	**0.048**

2.4.3 Working Range Evaluation

It is also important to test the proper working range of this proposed system, so users can use it within the range properly. The object recognition accuracy of our system is affected by the camera-object distance. According to the camera manufacturer, the depth camera's suggested minimum working distance is 30 cm. After a certain range, as the distance increases, the quality of the depth data deteriorates; when it is beyond 1 m, it becomes difficult to capture objects' visual details. In a similar manner, RFID communication between tags and the reader is also determined by the distance between them. When the distance is large, the power reaching the tag may not be strong enough to power the IC and backscatter

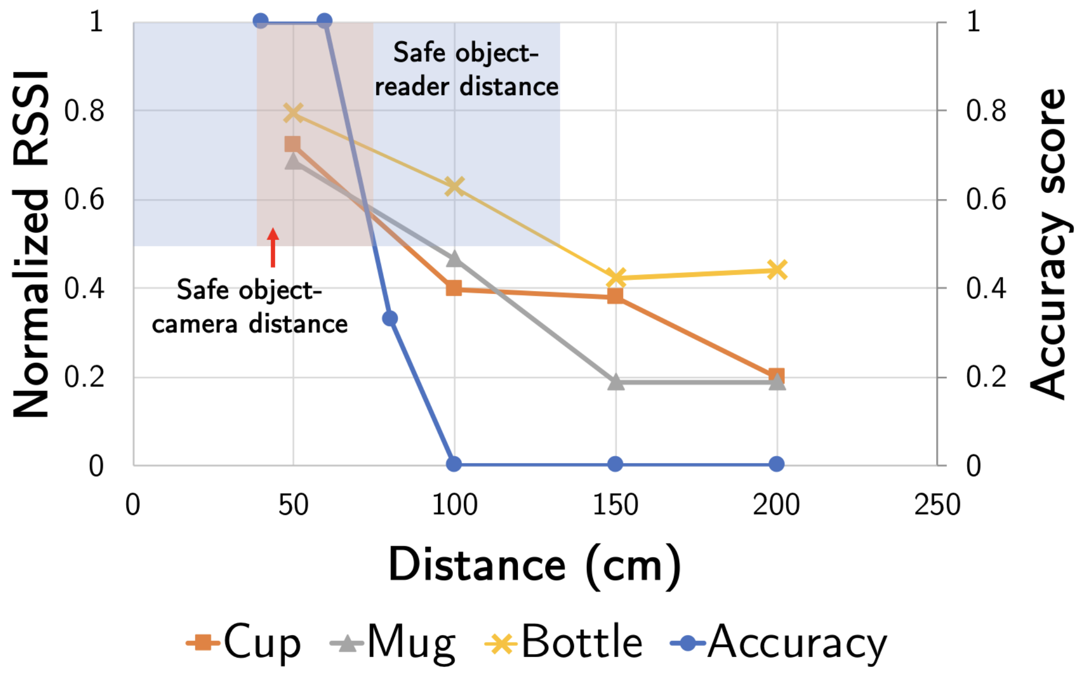


Figure 2-9: Normalized evaluation values for different working distances [1]. Shaded regions show recommended working range.

the signal to the reader. In this evaluation, a scalar score ranging between 0 and 1, named as normalized RSSI, is used to compare different material-range-signal strength. A score close to 1 indicates a good backscattered signal, and a score approaching 0 means signal strength is below the reader sensitivity. Both recognition accuracy and normalized RSSI scores are calculated for different distances. The recorded results are plotted in Figure 2-9. According to our observations, to achieve a reliable sensing and good quality visualization, we set an acceptable score of 0.5-1 for both evaluation metrics. Based on the figure, it is recommended to set 40-75 cm and 100-150 cm as the working range for reliable visual recognition and RF sensing, respectively.

2.5 AR-IoT Interaction System

The previously presented system demonstrates a fully functional framework to enhance IoT user experience in AR environment. In that system, traditional fiducial markers are replaced with a natural appearance-based object recognition and localization module. An AR-based visualization tool is also implemented to let users directly see sensed data. In this section, these advantages are further improved.

The previous system can only handle textured objects, while many devices in today's smart environment are textureless. Therefore, textureless object recognition and localization need to be studied. Further, only visualization is not enough to meet users' needs in many applications, and a seamless interaction between users and smart devices is often required, so that users can control IoT devices as they wish. We are seeking a natural way to interact with the rendered information in the digital world. In the movie Iron Man, the audience is captivated by Tony Stark because he designs his armour by simply manipulating holographic 3D models with his hands. Inspired by this, we believe that it would be impressive if users can manipulate nearby devices that are beyond arm's length by just moving fingers without actually touching them. This can be exemplified by an application scenario that a virtual control panel associated with the in-view target device is rendered to a user, and the user is able to operate on this virtual panel and change the device's physical status via hand gestures. Actually, hand gestures have already been adopted to simplify

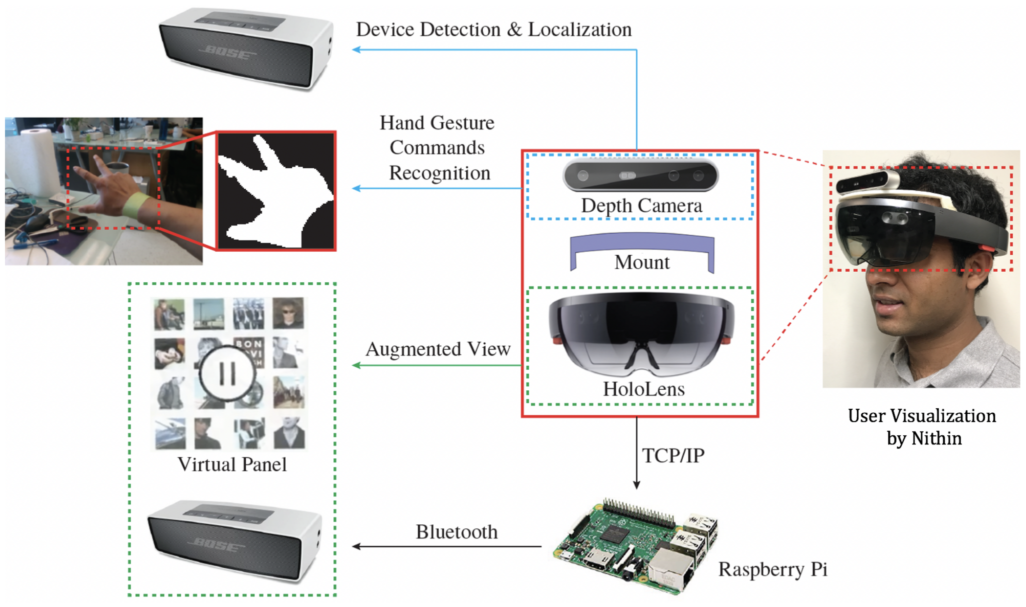

Device Detection & Localization
Hand Gesture Commands Recognition
Depth Camera
Mount
HoloLens
Augmented View
Virtual Panel
TCP/IP
User Visualization by Nithin
Bluetooth
Raspberry Pi

Figure 2-10: The proposed AR system for controlling IoT devices [2]. The system first detects and localizes the target device in the view, and then projects a virtual control panel. The system also detects hand gestures commands to conduct predefined actions.

device control in AR applications: the HoloLens defines two hand gestures for “click" and “go back to main menu" actions. Yet, most modern IoT devices are multi-functional, and complex control is often required. This adds design and implementation challenges for hand gesture control in our AR-IoT system. In this section, we will address all the above mentioned issues.

Before delving into details, Figure 2-10 shows the pipeline of the upgraded system for interacting with a smart speaker. The system first identifies and localizes the target device in the view using a depth camera mounted on the HoloLens as before, and then interprets users' hand gestures in order to control this device. In the rest of this section, details will be given, including detecting and localizing textureless target devices, recognizing hand gesture commands, and controlling IoT devices.

2.5.1 IoT Device Detection and Localization

Figure 2-11 shows the device detection and localization model. This procedure verifies whether a target device exists in the view, and estimates its pose parameters if so.

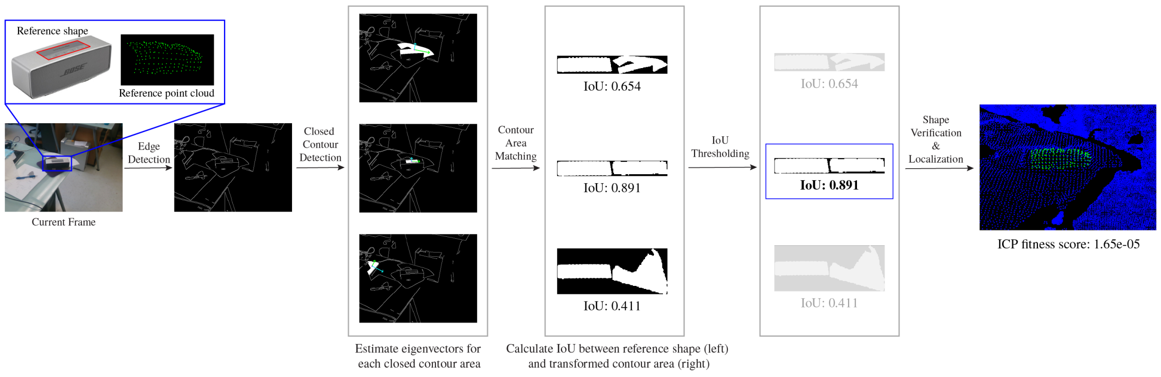

Reference shape
Reference point cloud
Current Frame
Edge Detection
Closed Contour Detection
Contour Area Matching
IoU: 0.654
IoU: 0.891
IoU: 0.411
IoU Thresholding
Shape Verification & Localization
ICP fitness score: 1.65e-05
Estimate eigenvectors for each closed contour area
Calculate IoU between reference shape (left) and transformed contour area (right)

Figure 2-11: The pipeline for target device detection and localization [2]. Given an input view, the model first detects all the edges within it, and finds closed contours out of them. Each closed contour is normalized by rotating to a same direction and resizing to a same scale as the reference shape of each target device, followed by a shape matching step. The closed contours with IoU values greater than a predefined threshold are selected, and their corresponding 3D positions are used to initialize the reference point cloud for ICP matching. A target device is detected and localized from a closed contour that passes ICP shape verification.

In Section 2.3, a texture-based object recognition module is presented. However, many existing household devices are textureless due to the aesthetic design, and texture-based visual feature methods usually fail for such devices. Therefore, in this modified system, another shape-based approach for object recognition is developed. We approach this problem by comparing a 2D reference shape and 3D point cloud of a device with the object found in a captured image. The reference shape is defined as a 2D closed contour that characterizes its 2D geometric features. For example, the top rectangular region of the Bose speaker shown in Figure 2-11. The reference point cloud of an object describes the 3D surface of an object from a selected viewpoint.

Given any captured color image, its edges are first detected using a Canny Edge Detector [59], and then closed contours are found from detected edges. For each closed contour, a similarity score for the reference shape of each target object in the database is calculated. To handle viewpoint and distance variations, each detected closed contour needs to be normalized to have the same orientation and scale as the reference shape. For orientation normalization, 2D eigenvectors of each closed contour are computed using their interior pixel coordinates. Geometrically, the eigenvector associated with the larger (or smaller) eigenvalue indicates the direction of the larger (or smaller) variance of the pixel coordinates. By aligning the eigenvector directions of the closed contour and reference shape, their orientations can be normalized. To normalize scale, the rotated contour is scaled to have the same width as the reference shape while keeping its aspect ratio. If two normalized shapes match, they should show a relatively large overlapped portion. To measure this, we computes a Intersection over Union (IoU) score between two shapes. The IoU is simply calculated by dividing the area of overlap between two shapes by the area of union of those two shapes, so it ranges between 0 and 1. IoU is a common metric to evaluate image segmentation performance [91]. Similar shapes usually give a large IoU value. With a carefully selected threshold, we can filter out false matches.

However, when there exist other contours of similar aspect ratio as the reference shape in the captured image, false positive detection can still occur. To reduce the false positive rate, another 3D point cloud-based shape verification process is added to check all the contours that pass IoU threshold. ICP alignment is used for this process. As stated previously, ICP requires a good initial position to converge to the global optimal. Similar to earlier, we obtain the initial position using the centroid of 3D points corresponding to the matched closed contour points in previous step. The reference object's point cloud is first translated to this initial position, and an ICP matching value, indicating the average inter-point distance between two point sets, is then computed accordingly. We calculate ICP matching values for all the retained shape contours, and compare them with a predefined ICP threshold. A detection is reported when its ICP value is less than this threshold. One benefit of this method over the system in Section 2.3 is that the 3D pose parameters of the detected object can be directly obtained from ICP matching process.

2.5.2 Hand Gesture Command Recognition

Compared to the system in Section 2.3, an augmented interactive interface is included to further enhance user experience for IoT applications. To let users seamlessly interact with surrounding smart devices, a hand gesture-based control module is implemented. In this subsection, hand gesture recognition and hand gesture commands are both described.

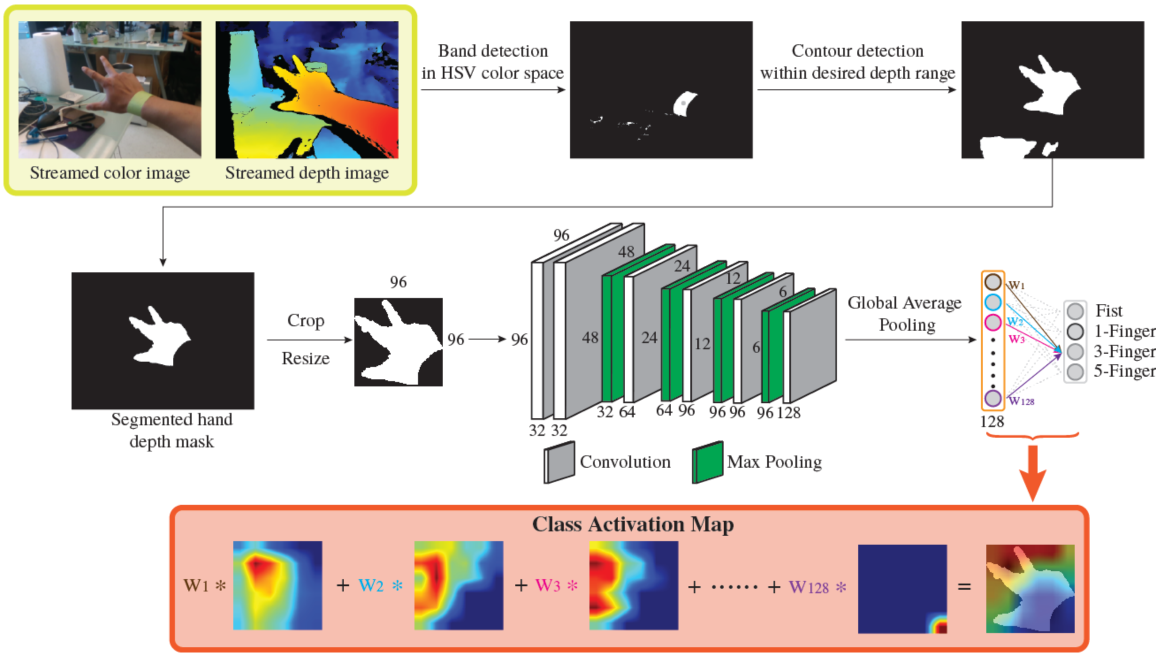


Figure 2-12: The pipeline for gesture recognition given a pair of streamed color and depth frames [2]. The user wears a colored wristband to indicate the hand region. The hand shape is detected and extracted from the depth image, and fed into a 2D CNN model for gesture recognition. We visualize the resultant activation map of the input gesture patch, showing the contribution of different pixels to the classification results (red: high, blue: low).

Hand Gesture Recognition

To recognize a hand gesture, we need to first segment the hand region in the input image and then identifies the hand gesture. An well-known way to detect hand regions is to extract skin pixels by thresholding the value of hue, saturation and value in HSV color space, which actually follows the way that how human perceives color. However, this method requires these thresholds to be carefully adjusted based on users' skin color and light conditions. In this system, we simply detect hand regions by using a colored wristband, which serves as an indicator of the hand region in the image. The hand region can be found in a depth image using the contour whose depth values fall into a depth range starting from the wristband and pointing outwards. Our system uses a depth range of 20 cm, covering the length of a regular hand. When multiple contours are detected, the hand contour is selected as the one closest to the wristband, as shown in Figure 2-12 top. One significant advantage of detecting the hand region from depth images is that the hand can be segmented automatically in the depth image generation process. Also, hand shapes are not affected by users' skin colors in depth images, which eliminates the need for manual threshold adjustment.

In order to recognize the segmented hand contours, a 2D CNN model is designed to hierarchically extract high-level features for determining gesture types. The CNN model takes as input a cropped hand shape, and outputs a probability distribution among 4 gestures: fist, 1-finger, 3-finger and 5-finger. When constructing the CNN gesture recognition model, it is also desired to interpret its internal working mechanism, since this remains unclear in many current deep learning-based Human Computer Interaction systems. To this end, the global average pooling (GAP) operation [92] is used as the last layer to aggregate previously extracted high-level visual information and assign probability to different hand gesture labels. As claimed in [92], GAP provides an unsupervised way to visualize salient regions for each predicted label. This procedure is demonstrated on Figure 2-12 bottom by Class Activation Map (CAM), which is computed as a weighed summation of the feature maps in the last convolution layer using their corresponding learned weights in GAP layer.

Hand Gesture Commands

We determine a user's hand gesture command by recognizing his/her hand gestures from a continuous sequence of captured images. In our system, four types of user commands are defined: binary control, continuous control, binary status switch and termination, as described below.

Binary Control Command. The binary control command is useful to play either the next or previous song when controlling a speaker. We define this command by measuring the shift between the starting and finishing positions of the same recognized hand gesture (e.g. 1-finger gesture on Figure 2-13 top left).

Binary Status Switch Command. Many devices have binary states, such as on/off and playing/pausing, so enabling users to switch between their binary states is important for user-device interaction. We recognize the binary status switch command by detecting the same hand gesture continuously (e.g. 5-finger gesture on Figure 2-13 top right) from a captured image sequence. Note that this binary status switch command is different from the previous binary control command. For example, two consecutive binary control commands

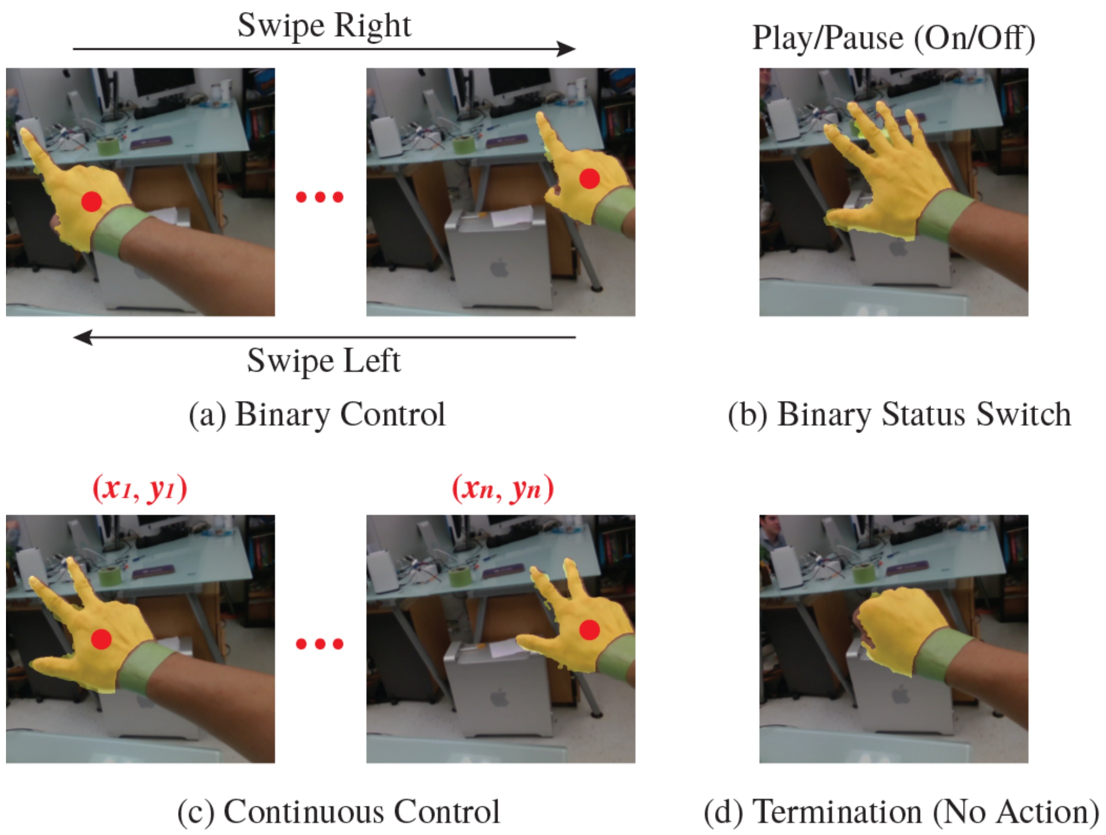
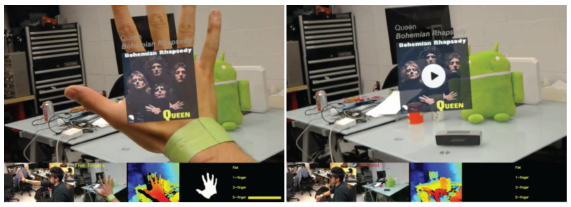


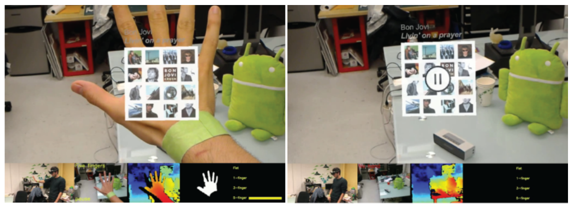
Figure 2-13: Hand gesture commands [2].

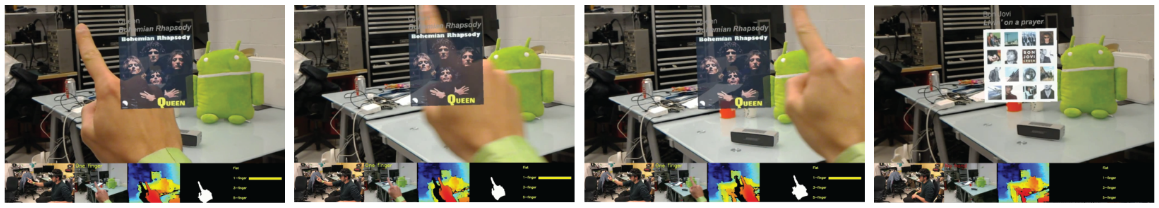
can trigger the same action, such as “swipe left, and then swipe left", while two consecutive binary status switch commands trigger different actions, such as “turn on, and then turn off)".

Continuous Control Command. Different from binary control commands, continuous control commands allow users to gradually modify a device’s status (e.g. the volume of a speaker). We implemented this command similar to binary control command, except that we additionally record all hand gesture positions (e.g. 3-finger gesture on Figure 2-13 bottom left).

Termination Command. In this system, the termination command (defined as fist gesture shown on Figure 2-13 bottom right) performs no action, and simply indicates the end of an action or command. For instance, when a user performs a binary control command, the finishing position of a hand gesture can be obtained from the frame right before the termination command. The termination command also allows a user to move his or her


(a) Play the current song


(b) Pause the current song


(c) Swipe to the next song

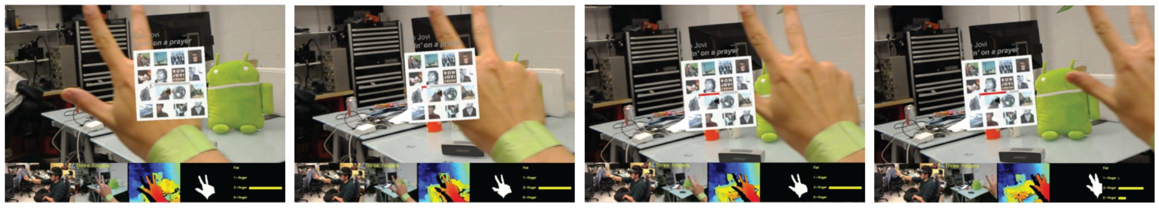
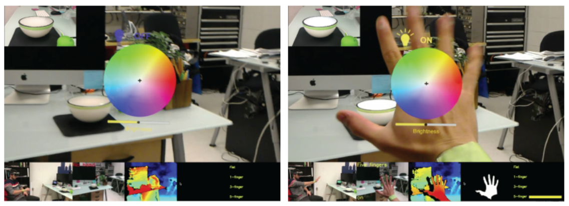

(d) Increase the volume

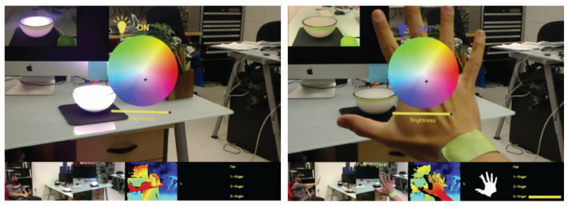
Figure 2-14: Speaker control demonstration. (a) and (b): Play and pause the current song using the 5-finger gesture; (c): Swipe right to switch to the next song using the 1-finger gesture; (d): Adjust the volume using the 3-finger gesture (note how the volume bar (red in the middle of the album) changes according to the hand position). On the bottom of each screenshot, the third-person point of view, color frame, colorized depth frame, segmented hand region before feeding into the CNN hand gesture recognition model, and the output probability of the CNN model are shown.

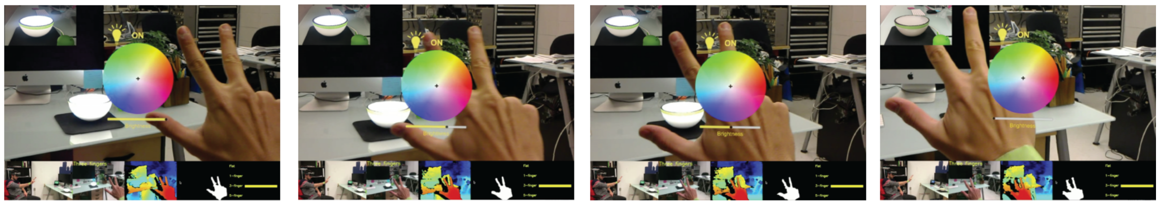
hand freely in the view without triggering unintentional actions.

2.5.3 IoT Device Control

This system is demonstrated on two devices: a Bose speaker and a Philips Hue Go light bulb. The user is allowed to interact with each via hand gestures. A virtual control panel is designed for each device and rendered to the user. The panel is projected close to the detected device and is oriented towards the user. In this way, when the user walks around the device, the panel can still face towards the user. A Raspberry Pi is used to coordinate data communication via TCP/IP. It can receive data from the HoloLens and send the recognized command to detected target devices.


(a) Turn on the bulb


(b) Turn off the bulb


(c) Decrease the brightness of the light bulb

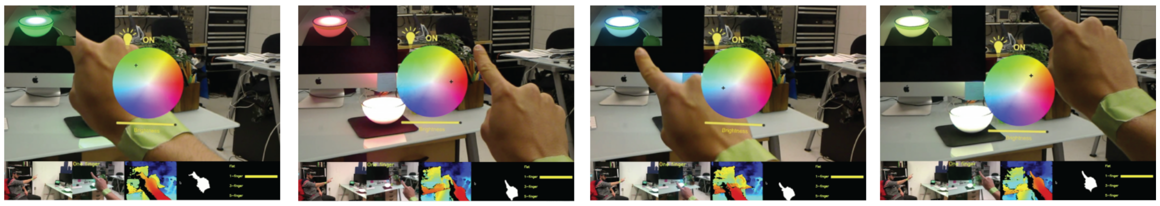

(d) Change the color of the light bulb

Figure 2-15: Light bulb demonstration. (a) and (b): Turn the bulb on and off using the 5-finger gesture; (c): Adjust the brightness using the 3-finger gesture (note how the brightness bar (yellow below the color wheel) changes according to the hand position); (d): Select the color using the 1-finger gesture (the light bulb color changes according to the black cross within the color wheel, which is controlled by the hand gesture). On the top left of each screenshot, we show a zoomed-in view of the light bulb recorded using a separate camera placed nearby for better visualization.

For the speaker, users are allowed to play and pause the current song using the binary status switch command, change songs using the binary control command, and adjust the volume using the continuous control command. For the light bulb, users are allowed to turn the bulb on and off using binary status switch command, and adjust brightness and hue using the continuous control command.

2.6 Evaluation on Interaction System

We made two major changes compared to the previous X-Vision system: device localization to handle textureless objects, and hand gesture-based user-device interaction. In this section, quantitative and qualitative evaluations are conducted to assess these changes individually and also the system's overall performance.

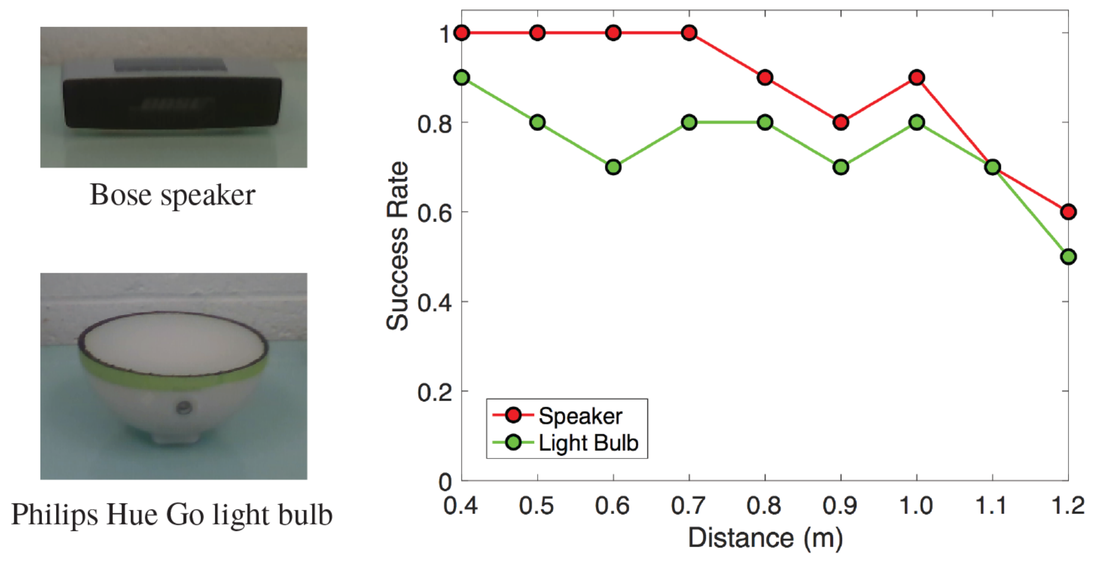


Figure 2-16: The success rate of detection and localization for different camera-device distances [2].

2.6.1 Detection and Localization

The proposed device detection and localization module is first evaluated on the demonstrated speaker and light bulb. Here, we care about this module's performance for different camera-device distances, since it suggests a proper working range for this system. We test its performance for the range from 0.4 m to 1.2 m with an interval of 0.1 m. For each distance, we collected 10 color and depth image pairs containing a target device, and recorded the rate of successful detection and localization. The results are plotted in Figure 2-16. As observed, our method performs relatively well for short ranges, and the success rate decreases as the range increases for both selected devices in a similar pattern. Two possible reasons can explain this: for the visual information, the foreground area of the target device in the color image reduces as the range increases; For the geometric information, the reconstructed scene point clouds become more noisy as the range increases. We can also observe that the success rate of the bulb is lower than that of the speaker, and this

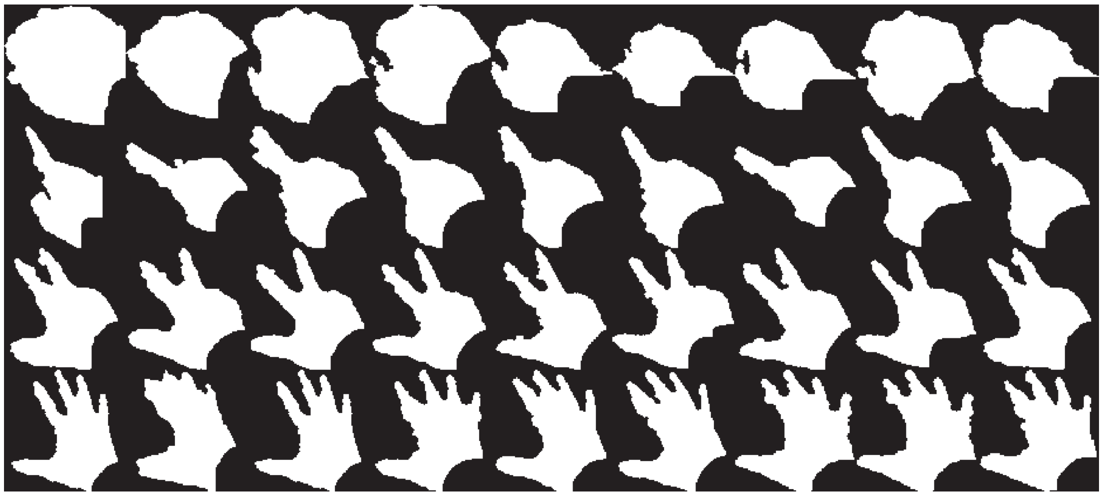


Figure 2-17: Hand shape examples for training CNN gesture recognition model [2]. From top row to bottom row: fist, 1-finger, 3-finger and 5-finger.

might result from the reflective surface of the bulb. In our actual coding implementation, processing each frame takes about 410 ms on average.

2.6.2 Hand Gesture Recognition

In this subsection, we summarize details about training and testing the implemented 2D CNN hand gesture recognition module. As known, deep learning models learn visual patterns in a data-driven manner. For this task, we collected 4,690 hand gesture images covering 4 different hand gestures. To avoid the tedious manual labeling process, a live stream for each gesture type is recorded, and hand gesture samples for each corresponding hand gesture are captured by using the same hand gesture detection operations before the 2D CNN hand gesture classification model. We show some collected hand gesture samples in Figure 2-17. It can be observed that our collected samples have diverse shapes within each hand gesture category. 90% samples of each hand gesture are used for training and the rest 10% are used for testing. We built the 2D CNN hand gesture recognition model using the C++ version of TensorFlow for the sake of speed. The system runs on a MacBook Pro with 2.6 GHz Intel Core i7 prosessor, and the average processing time for one forward inference pass is about 30 ms.

We also implemented a Support Vector Machine (SVM) linear classification model for

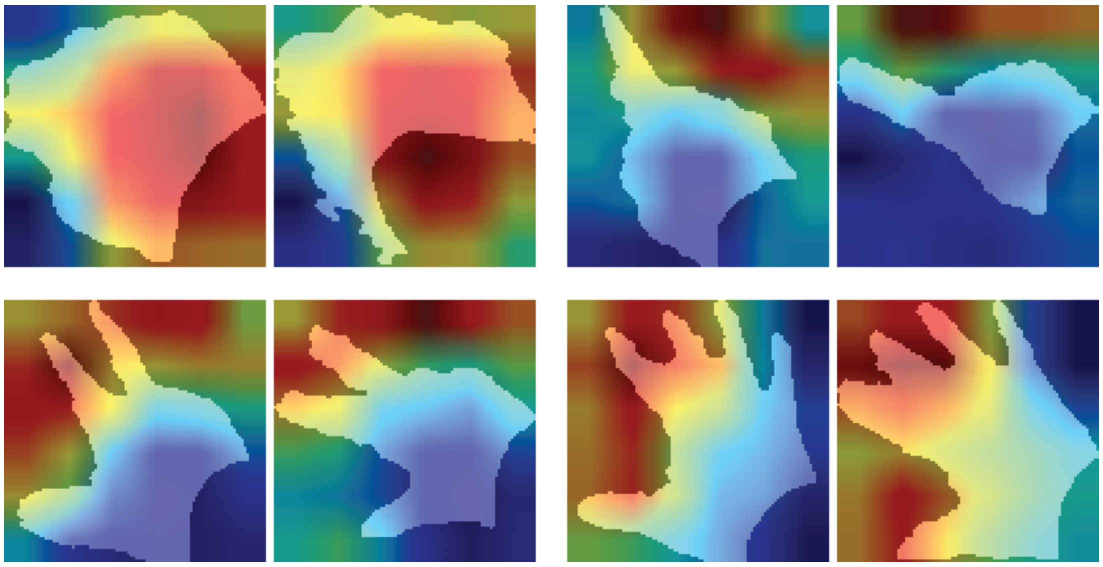


Figure 2-18: The activation maps of the CNN model for different hand gestures.

comparison purpose. This SVM model is trained and tested on the same image sets as our model. Different from our model, which directly takes as input the hand gesture image, the SVM model takes as input a 1D vector, so we downsampled the image and flattened it into a 1D vector before feeding it to the SVM model. For our collected dataset, our method achieves better testing accuracy, as reported in Table 2.5. The activation maps of randomly selected samples are also visualized in Figure 2-18, from which we can observe that different regions for different gestures are attended differently, indicating salient regions for different hand gestures.

Table 2.5: Testing accuracy for SVM and the implemented CNN models [2]. The number of training (left) and testing (right) samples are provided under each gesture.

Gesture	Fist 1024/114	1-finger 1027/115	3-finger 1081/121	5-finger 1087/121	Average
SVM	1.00	0.99	0.87	1.00	0.96
CNN	1.00	1.00	1.00	1.00	1.00

2.6.3 User Study

After modifying the system's components, we want to assess the overall performance of this new system. Therefore, a user study reflecting two aspects, speed and degree of satisfaction, is conducted. We additionally include available commercial applications for comparison. For the speed evaluation, we record the time duration between participants connect the device and finish predefined control tasks. To evaluate the degree of satisfaction, we let users score between 1 to 10 to rate their user experience. 10 users without related background participated in this study. We found that our system takes significantly less time for users to control the speaker. This may result from the fact that most smartphones require users to open Bluetooth and search for surrounding devices, and these steps take a significant amount of time. Another finding is that our system shows higher degree of satisfaction by a relatively large margin in the light bulb control case, and this is possibly due to the direct visual feedback provided by the HoloLens.

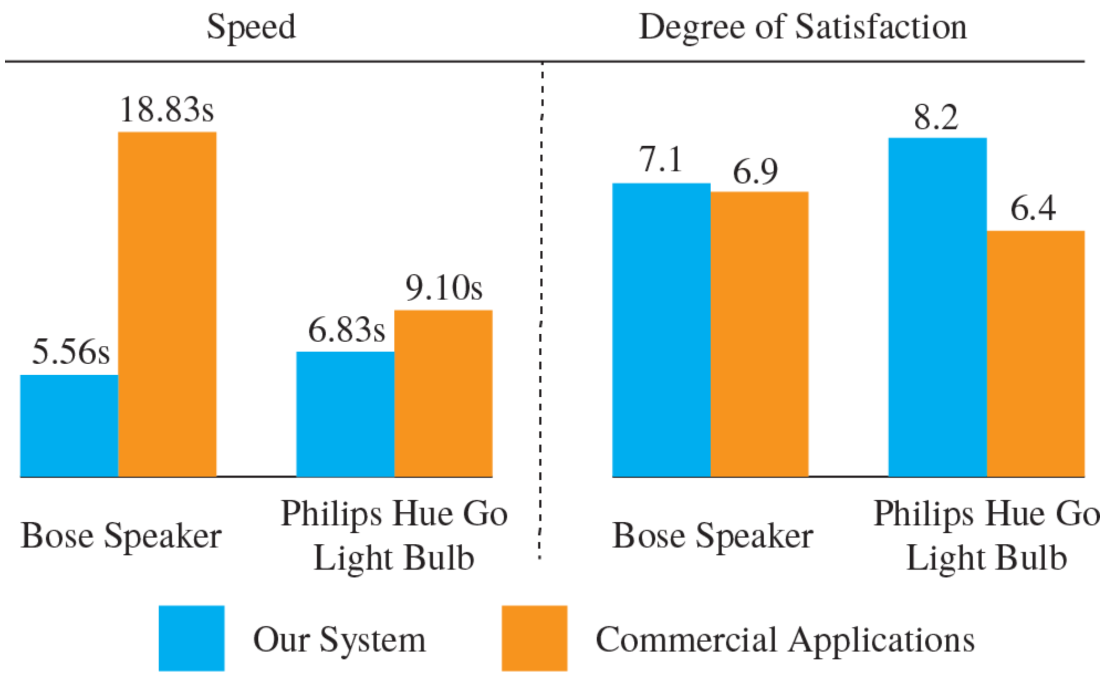


Figure 2-19: User study results on speed and degree of satisfaction for our system and available commercial applications [2].

2.7 Discussion

In this section, the integration of AR and IoT technologies is demonstrated. In the first system, a complete systematic framework is described, and it was tested on simple textured objects to demonstrate better AR-IoT user experience in smart environment. In the second system, a shape-based object recognition module is designed to handle textureless devices, which are common for real application scenarios. Additionally, a hand gesture-based user-device augmented interaction tool is added to further enhance immersive IoT user experience. Extensive experiments and evaluation results prove the effectiveness and efficiency of the AR-IoT system.

Both implemented methods in this section only use conventional handcrafted features. They work fine for demonstrated simple cases, but may fail for complex object shapes. Recent works [93, 94, 4] show that object pose estimation is an important factor for marker-free AR systems, and further research is required to increase their robustness. In the following chapters, deep learning-based approaches will be introduced to analyze geometric properties of 3D shapes and improve object pose estimation accuracy to better support AR-IoT systems.

Chapter 3

Deep Learning for 3D Understanding

In the AR-IoT system introduced in Chapter 2, object pose parameters are estimated using conventional 3D geometric methods. While those handcrafted methods work fine for demonstrated simple cases, usually they cannot handle complex and noisy shapes. This is due to the fact that handcrafted methods need to be supported by theory and expertise, but when shapes become complicated and noisy, required knowledge and computation are dramatically increasing, making it extremely difficult to design well-performed algorithms. Hence, feasible and robust object pose estimation approaches to handle difficult cases are desired .

In recent years, deep learning has developed rapidly due to the high performance of modern Graphics Processing Units (GPU), which makes it possible to train a neural network with millions of parameters on a large dataset of millions of samples within a reasonable amount of time. This hardware boost has also driven the development of related software and theoretic algorithms: a number of tools (i.e. TensorFlow and PyTorch) have been developed to make modeling, training and evaluation unprecedentedly easy, and algorithms such as ResNet [95], BatchNorm [96], Dropout [97] have also been designed to accelerate the convergence of model training and improve model performance. Benefiting from them, deep learning models have outperformed conventional methods by a significantly large margin on various types of tasks, such as image classification and segmenta-

This chapter is based on the author's earlier work: Dynamic Graph CNN for Learning on Point Clouds, in ACM Transactions on Graphics, VOL 38, ISS 5 ©2019 ACM. https://doi.org/10.1145/3326362.

tion, voice synthesis, etc. Compared to image, voice and text data, which are in regular format so that standard convolutional operations can be directly applied, deep learning for 3D data analysis has not been well studied due to the irregularity of 3D data. In order to take advantage of current deep learning models, let neural network models automatically learn how to analyze 3D data, and solve relevant tasks, especially object pose estimation, fundamental research on 3D data analysis is required.

In this chapter, deep learning models for 3D data analysis are explored. We specifically focus on 3D point cloud due to its flexibility and popularity in many applications. To demonstrate the effectiveness of the proposed model, classification and segmentation tasks are tackled on public benchmark datasets. Later in the next chapter, this proposed 3D deep learning model is extended to estimate objects' pose parameters and support industrial AR-IoT applications.

3.1 Background

Point clouds are represented as a set of 3D points, and many existing 3D data capturing devices, such as liDAR scanners, output point clouds directly. The rapid development of 3D sensing and processing techniques makes it possible to directly process point clouds for vision applications. This improves efficiency by avoiding the traditional mesh reconstruction process, and extends the application scopes of point clouds to many domains such as indoor navigation [98], self-driving vehicles [99, 100, 101], robotics [102], and shape synthesis and modeling [103, 104]. As the problems to be explored in these areas become more challenging, simple low-level features extracted using non-leaning methods cannot satisfy the requirement. These methods usually designate handcrafted features to capture geometric properties of point clouds [105, 29, 74]. The methods implemented in Chapter 2 belong to this category. On the other side, high-level semantic features extracted using learning-based methods can handle more difficult cases. Recently, the success of deep neural networks for image processing has motivated a data-driven approach to automatically learning features on point clouds during training. 3D deep learning methods have outperformed traditional approaches in many tasks [106]. However, the adaptation of them to point cloud data is far from straightforward. Standard deep neural network operators can only work on regular grid-like input data, while point clouds are irregular in nature: their positions are continuous values, and any permutation of their ordering only changes their relative representation without changing the whole shape representation. An easy way to handle this irregularity problem is to discretize 3D shapes into 3D grids [107, 108]. However, in this manner, quantization artifacts will be introduced and additional unnecessary memory space will also be needed for empty parts in the volume.

The earliest deep learning model that addressed the irregularity problem of point clouds was *PointNet* [109]. It operates directly on each point independently without needing any intermediate regular representation. PointNet handles the order permutation invariance problem by using a symmetric function to accumulate all the point features. But local geometric features are not included in PointNet. To improve the performance, a novel neural network structure is designed in this chapter to capture local geometric information from neighboring point sets. Different from previous work, the proposed model operates on features generated from point pairs. Our approach is also invariant to point ordering by using symmetric functions. In many ways, the proposed model resembles graph neural network architectures, but in our implementation, the graph is reconstructed at each different layer based on their feature representations.

The main goal of 3D point cloud analysis discussed in this thesis is to facilitate object pose estimation and support AR-IoT systems. Yet, in the 3D computer vision area, testing the model on public classification and segmentation benchmark datasets is a common practice to assess the effectiveness of a model. Therefore, the proposed model is first tested on these datasets to validate its 3D data analysis capability.

3.2 Related Work

Recent years have witnessed a breakthrough in deep learning [110, 111]. This motivates the research trend of using deep learning for 3D geometric data processing. Relevant research is summarized in this section. Different from regular structured data, such as images, 3D point clouds do not have grid-like structures, which makes it difficult to directly apply standard convolution operations on them. To resolve this difficulty, view-based [112, 113] and volumetric representations [114, 115] were previously developed to transform original point clouds to regular format for processing. Recently, PointNet [109, 116] has provided a simple but efficient deep learning architecture to operate on point cloud data, graph-structured non-Euclidean data in general. Early graph neural networks can in found in [117], their performance has been improved recently by gated recurrent units [118], neural message passing [119], Laplacian eigenvectors [120], and polynomial or rational filters [121, 122]. Other related works include Geodesic CNN [123] for irregular mesh processing via local parameterization. This work provided a spatial convolution approach, and demonstrated performance boosting over previous spectral approaches. Later, this approach was further improved by local charting techniques, such as anisotropic diffusion [124], Gaussian mixture models [125, 126], and differentiable functional maps [127]. Another class of 3D deep learning approaches embeds input 3D shapes into some other representation domains, where the shift-invariance property is satisfied. Example domains include sphere [128], sparse network lattice [129], torus [130] and spline [131]. 3D analysis is not only limited to solve deterministic tasks, but can also be used for generative tasks under guidelines like variational autoencoders (VAE) [132], and generative adversarial networks (GAN) [133]. Various generative architectures have been proposed in [134, 135, 136]. Slightly different frameworks for 3D mesh generation can be found in [137, 138, 139].

3.3 Point Cloud Neural Network

In this section, a 3D point cloud neural network model is introduced to extract point features for solving inference tasks. The prior art, PointNet, provides a way to process points independently and accumulate their feature via a symmetric function, while in this work, additional local geometric information is taken into consideration. In our model, a local neighborhood graph is constructed to represent point cloud data, and convolutional operations are applied on the edges connecting neighboring point pairs. This edge convolution preserves translation-invariance and non-locality properties. Our model follows the graph CNN structure, but is dynamically reconstructed at each layer according to computed feature vectors, hence the proposed model is named as dynamic graph CNN (DGCNN). In the following, details of model design, discussion and properties will be given.

3.3.1 Edge Convolution

As mentioned earlier, the main contribution of this model is shifting convolution from points to edges connecting neighboring point pairs. In this way, the correlation between points is captured, from which local geometric information is extracted. This edge convolution is illustrated in this subsection.

First, without losing generality, we can represent a point cloud $\mathbf{X}$ of n points with each being an F-dimensional vector: $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq \mathbb{R}^F$. For point clouds in 3D space, where each point is simply represented by its 3D coordinates $\mathbf{x}_i = (x_i, y_i, z_i)$, F equals 3. When additional information, such as color and surface normal, is included, F can be larger accordingly. When the input point cloud is passed to a deep learning model, where each layer operates on the output of its previous layer, the feature dimension F is usually different, and can be much larger than the input dimension.

Here, a graph structure $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ is constructed to represent the local geometric structure of a point cloud, with $\mathscr{V}$ and $\mathscr{E}$ being *vertices* and *edges*, respectively. In the geometric context, we care more about local features hidden in neighboring point sets, thus k-nearest neighbor graphs are first built from $\mathbf{X}$, and then used to form the final graph, $\mathscr{G}$. Our graph structure allows points to point back to themselves, so self-looping is enabled. From every point pair in the graph, we can generate a feature vector encoding the correlation between its two ending points, and this is denoted as an *edge feature*. We can express edge features as $e_{ij} = h_\Theta(\mathbf{x}_i, \mathbf{x}_j)$, where $h_\Theta : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}^{F'}$ is a learnable function Θ to generate a new F'-dimensional feature vector and can be implemented using a neural network.

Once 1D edge features are obtained, we can apply convolutional operations on them. The designed operation in this work applies channel-wise symmetric aggregation on all the convolved edge features associated with each point, thus called *EdgeConv*. After Edge-

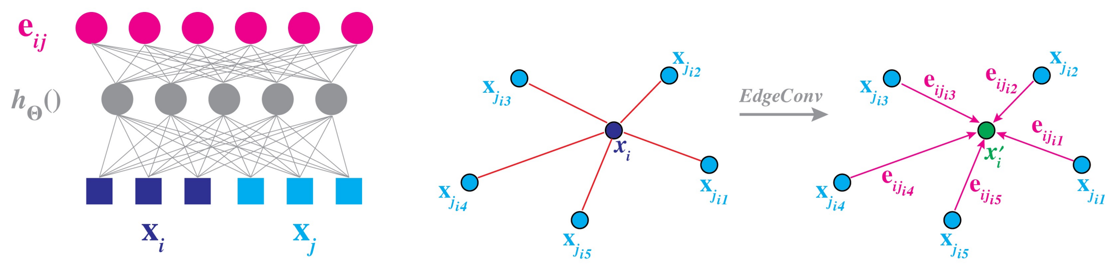


Figure 3-1: An example of EdgeConv [3]. Left: Computing an edge feature, e_{ij}, from a point pair, x_i and x_j. Here, we implement $h_\Theta()$ using a fully connected layer with learnable parameters. Right: Demonstrate the EdgeConv operation. We calculate EdgeConv output by aggregating all the edge features of a central point.

Conv, the output at the i-th point can be given by

$$\mathbf{x}_i' = \mathop{\square}_{j:(i,j)\in\mathscr{E}} h_\Theta(\mathbf{x}_i, \mathbf{x}_j), \tag{3.1}$$

where $\square$ is a symmetric aggregation operator, $\mathbf{x}_i$ is the i-th point and the set $\{\mathbf{x}_j : (i,j) \in \mathscr{E}\}$ indicates the neighboring points around it (see Figure 3-1). For a point cloud with n points, EdgeConv produces another point cloud with the same number of points but represented by feature vectors of different dimensions.

The choice of the edge function and the aggregation operation plays an important role in determining EdgeConv properties. The model architecture demonstrated in PointNet can be formulated as below:

$$h_\Theta(\mathbf{x}_i, \mathbf{x}_j) = h_\Theta(\mathbf{x}_i), \tag{3.2}$$

We can see, this definition only encodes global shape information, and is a special case of the EdgeConv opration. Local geometric structure properties are missing in this instantiation. Therefore, to better exploit local geometric information for 3D data analysis, the edge function adopted in our model is expressed as:

$$h_\Theta(\mathbf{x}_i, \mathbf{x}_j) = \bar{h}_\Theta(\mathbf{x}_i, \mathbf{x}_j - \mathbf{x}_i), \tag{3.3}$$

where $\bar{h}_\Theta()$ is another function to be learned from training data. Note that this is an asymmetric operator. It explicitly integrates global structural information from each patch center,

$\mathbf{x}_i$, and local geometric information from each of its neighboring point pairs, $\mathbf{x}_j - \mathbf{x}_i$. Being specific, the actual edge operator used in our model is expressed as:

$$e_{ij} = \text{ReLU}(\theta(\mathbf{x}_j - \mathbf{x}_i) + \phi(\mathbf{x}_i)), \tag{3.4}$$

and

$$x_i' = \max_{j:(i,j)\in\mathscr{E}} e_{ij}. \tag{3.5}$$

$\theta()$ and $\phi()$ are independent nonlinear functions with learnable parameters. The proposed novel edge convolution operator is used as the building block to construct models for solving 3D classification and segmentation problems. Model architectures are shown in Figure 3-2.

3.3.2 Dynamic Graph

In our implementation, we found that reconstructing the graph by updating neighbors using recomputed feature representations at each layer improves the performance. For the l-th layer, the graph is represented as $\mathscr{G}^{(l)} = (\mathscr{V}^{(l)}, \mathscr{E}^{(l)})$. This differentiates our proposed model with other fixed graph neural networks. In fact, the proposed dynamic graph reconstruction mechanism enlarges the receptive field of the model to the entire point cloud, and guarantees the efficient information diffusion. No human supervision is added during our training process, and the proposed model learns how to rebuild the graph at each layer in an efficient manner on its own. The only constraints added to the model are: a) The metric used to detect point neighbors, which is the Euclidean distance in feature space; b) The number of neighbors selected for each central point.

3.3.3 Properties

The properties of the proposed point cloud operator are described below.

Permutation Invariance: First, we claim the designed edge operator is invariant to point

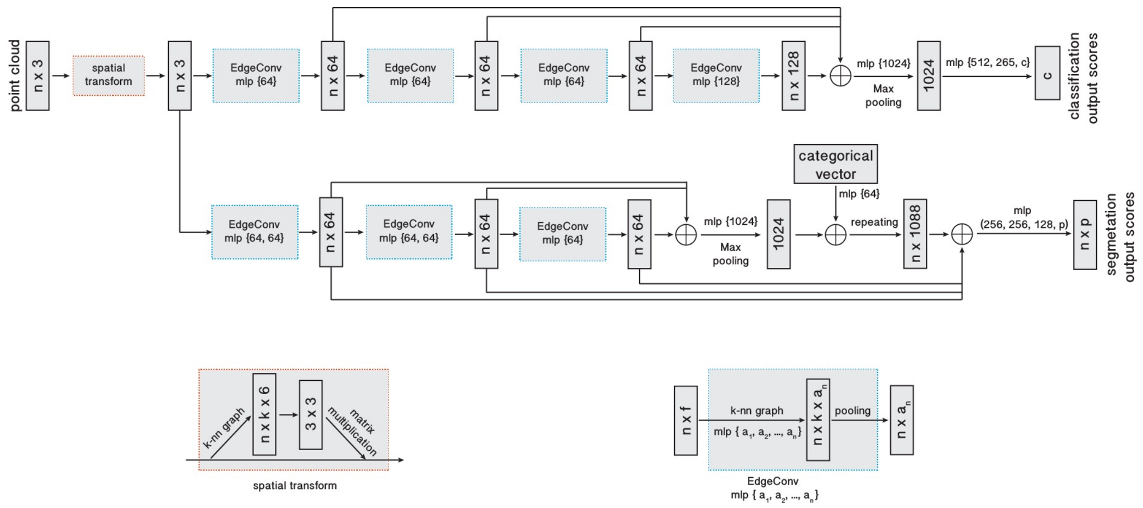


Figure 3-2: The classification model (top) and segmentation model (bottom) [3]. The classification model computes k edge features for each central point at each layer, and accumulates a global feature in the last layer to produce a classification distribution. The segmentation model concatenates the global feature vector with all the individual point features, and outputs per-point distributions. $\oplus$: concatenation.

order permutation. Given the output of a layer

$$x_i' = \max_{j:(i,j)\in\mathscr{E}} h_\Theta(x_i, x_j), \tag{3.6}$$

it is easy to observe that its output x_i' is invariant to the permutation of any input ordering, since the max function is a symmetric function, and its output does not depend on the input order. In fact, other symmetric functions, such as averaging pooling, also follow this property. Similarly, the global max pooling operator used to accumulate individual point features is also invariant to the input ordering permutation because of the same reason.

Translation Invariance. Our edge operator is partially invariant to point translation. This property results from the Equation 3.4 when computing edge features. This can be easily verified from the fact that, if the same translation, T, is added to both x_j and x_i at the same time, their vector difference stays the same. Hence, this part of the edge feature is preserved. In particular, for the translated point cloud we have

$$\begin{aligned} e'_{ijm} &= \theta((\mathbf{x}_j + T) - (\mathbf{x}_i + T)) + \phi(\mathbf{x}_i + T) \\ &= \theta(\mathbf{x}_j - \mathbf{x}_i) + \phi(\mathbf{x}_i + T). \end{aligned}$$

If only $x_j - x_i$ is used to calculate edge features, which is achieved by setting $\phi() = 0$, then fully translation-invariance property can be realized. However, in this way, the global structural information will be missing from the edge feature, which may degenerate the overall performance of the model. Because, by doing so, the model needs to recognize the 3D shape only from an unordered set of point patches, while when both $x_j - x_i$ and x_i are used, local geometric and global structural information are considered together for 3D data analysis.

By taking the advantage of these properties, the proposed model learns not only how to extract low-level local geometric features, but also how to dynamically group points in a point cloud to detect high-level semantic information. Figure 3-3 shows the distance to a randomly selected query point in different feature spaces, exemplifying that the embedding feature vectors in deeper layers carry more semantic information than shallower layers.

3.4 Evaluation

In this section, extensive experiments are conducted to evaluate the proposed DGCNN model on classification and segmentation tasks. Experimental results are also visualized to qualitatively assess the model's performance. Despite these tasks are not directly related to pose estimation for AR-IoT systems, they provide important insight to verify the model's 3D analysis capability, which is vital for all types of 3D tasks.

3.4.1 Classification

Data. The ModelNet40 dataset [108] is used as the classification benchmark dataset to evaluate the proposed DGCNN model. 12,311 3D shapes covering 40 different categories are included in this dataset. Following the setting in [109], we used the same 9,843 shapes for training and 2,468 for testing. In our implementation, we uniformly sample 1,024 points from the surface of each shape, and feed them into the model. Only point coordinates are used. Additional data augmentation techniques, such as randomly scaling and point perturbation, are included to regularize model training.

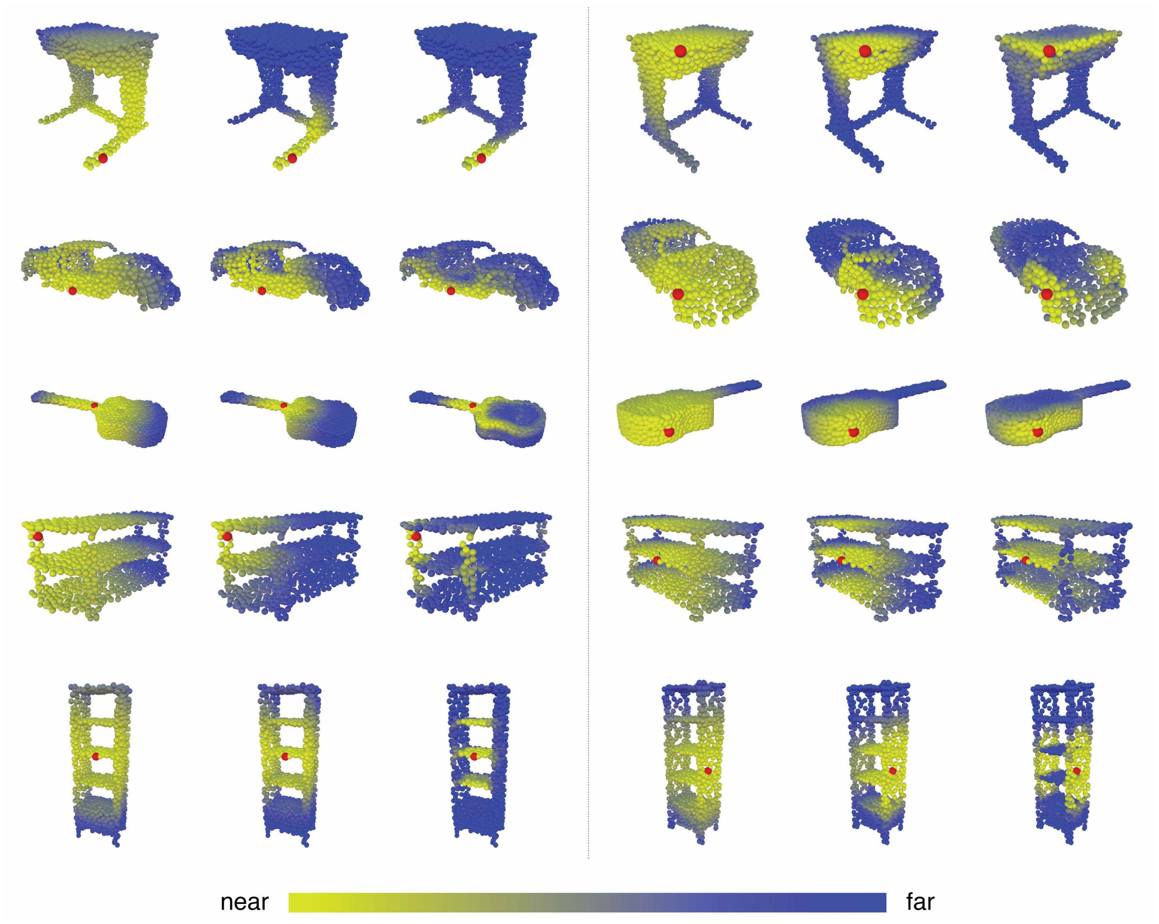


Figure 3-3: Visualize Euclidean distances in different feature spaces for a random query point (red) [3]. For each triplet set, the Euclidean distances are calculated in the original 3D space (left), the feature space after the transformation operation (middle) and the feature space in the last layer (right).

Implementation Details. Figure 3-2 top shows the point cloud classification model architecture. This model uses four consecutive EdgeConv layers to compute latent features, with each of them implemented by three fully-connected layers with 64, 128 and 256 neurons, respectively. We recompute the graph after every EdgeConv layer from nearest neighbor graphs of size 20. The number of neighbors is set according to point density, and will be adjusted accordingly when the number of points changes. Other common neural network implementation techniques, such as concatenation, batchnorm and dropout, are also included to boost model performance. After EdgeConv layers, we add a global max pooling layer to accumulate an 1D global representation of the input point cloud. The obtained global feature is then passed to two fully-connected layers to produce the classification dis-

Table 3.1: Classification results on ModelNet40 [3].

	Mean Class Accuracy	Overall Accuracy
3DShapeNets [108]	77.3	84.7
VoxNet [107]	83.0	85.9
Subvolume [140]	86.0	89.2
VRN (single view) [141]	88.98	-
VRN (multiple views) [141]	91.33	-
ECC [142]	83.2	87.4
PointNet [109]	86.0	89.2
PointNet++ [116]	-	90.7
Kd-net [114]	-	90.6
PointCNN [143]	88.1	92.2
PCNN [144]	-	92.3
Ours (baseline)	88.9	91.7
Ours	**90.2**	**92.9**
Ours (2048 points)	**90.7**	**93.5**

tribution. A dropout with keep probability of 0.5 is used in the last two fully-connected layers. All the intermediate layers use LeakyReLU as the nonlinear activation function. The hyperparameters are chosen through a separate validation set split from the training dataset. Once their values are decided, the whole training dataset is used to train the model from scratch again. For parameter updating, SGD with learning rate 0.1 is used. The momentum for batch normalization is set to 0.9, and training batch size is set to 32.

Results. The classification results on ModelNet40 are reported in Table 3.1, together with many other competing methods. Our baseline model using EdgeConv is also implemented by fixing the graph using spatial neighbors. As can be easily observed, the proposed model, even the baseline model, outperforms many competing methods, and this proves the effectiveness of our model design. 1,024 points are used for all the methods listed in the table, but we also tested our model on point clouds containing 2,048 points sampled in the same way. The increased point density further boosts the performance of our model.

Table 3.2: Complexity, forward processing time, and accuracy of different models [3].

	MODEL SIZE(MB)	TIME(MS)	ACCURACY(%)
POINTNET (BASELINE) [109]	9.4	6.8	87.1
POINTNET [109]	40	16.6	89.2
POINTNET++ [116]	12	163.2	90.7
PCNN [144]	94	117.0	92.3
OURS (BASELINE)	11	19.7	91.7
OURS	21	27.2	92.9

3.4.2 Model Complexity Assessment

Model complexity is an important factor that can affect a system's runtime performance, so it is assessed here. Model configurations of our model and other state-of-the-art models are compared and listed in Table 3.2. The reported accuracy values are classification testing results on ModelNet40. From the table, we can observe that our proposed model presents a better tradeoff of model size, inference time and testing accuracy. Even our baseline model with fixed graph structure can run 7 times faster and achieve 1.0% higher accuracy than PointNet++, which reported the best performance value by the time we finished our work. Our model, with its edge features dynamically updated, outperforms PointNet++ and PCNN by 2.2% and 0.6% respectively.

3.4.3 Robustness Evaluation

The robustness of the proposed DGCNN model is further evaluated against point cloud density changes. In this study, input points are randomly dropped out according to predefined point densities. Figure 3-4 shows testing accuracy changes for different numbers of points retained. The model performs well even when half of points are dropped, but, after that, the model's performance degenerates dramatically.

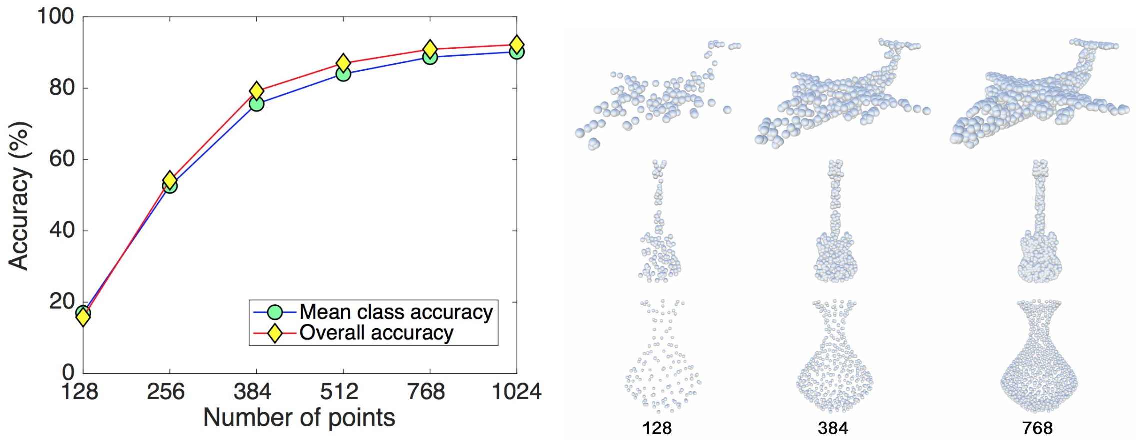


Figure 3-4: Model robustness evaluation against point density [3]. Left: Testing results of our model (trained on 1,024 points) for different numbers of input points. Right: Examples of point clouds with different number of points. The numbers of points are shown at the bottom.

3.4.4 Part Segmentation

Data. The proposed model architectures can be adapted to part segmentation. Under this segmentation setting, we predict a distribution across a few predefined part category labels for each point. The ShapeNet part dataset [145] is used for training and evaluation. This dataset contains 16,881 3D shapes covering 16 object categories, and each point of any shape is annotated by one out of 50 part labels. In the dataset, 2,048 points are sampled from each shape. On average, each shape contains points belonging to no more than six part categories. In the experiment, a train/validation/test split following [106] is adopted.

Model Architecture and Training Details. The network architecture is graphically illustrated at the bottom of Figure 3-2. Three EdgeConv layers are implemented, followed by a global feature extraction layer. This global feature is concatenated to previous individual point feature vectors, in the hope that global and local information can be fused. In the end, three shared fully-connected layers are used to produce a part label distribution for each processed point feature. Batchnorm, dropout, and ReLU nonlinear activation functions are also included in a similar manner as in our classification model. Since more points are included, more GPU memory is required. Therefore, to handle the additional memory requirement, we implement a distributed training framework on two NVIDIA TITAN X GPUs, so that a reasonably large training batch size can be achieved.

Table 3.3: Part segmentation results on ShapeNet part dataset. Metric is mIoU(%) on points [3].

	MEAN	AREO	BAG .	CAP	CAR	CHAIR	EAR PHONE	GUITAR	KNIFE	LAMP	LAPTOP	MOTOR	MUG	PISTOL	ROCKET	SKATE BOARD	TABLE
# SHAPES		2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
PointNet	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
PointNet++	85.1	82.4	79.0	**87.7**	77.3	**90.8**	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
Kd-Net	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
LocalFeatureNet	84.3	**86.1**	73.0	54.9	77.4	88.8	55.0	90.6	86.5	75.2	**96.1**	57.3	91.7	83.1	53.9	72.5	**83.8**
PCNN	85.1	82.4	80.1	85.5	79.5	**90.8**	73.2	91.3	86.0	85.0	95.7	73.2	94.8	83.3	51.0	75.0	81.8
PointCNN	**86.1**	84.1	**86.45**	86.0	**80.8**	90.6	**79.7**	**92.3**	**88.4**	**85.3**	**96.1**	**77.2**	**95.3**	**84.2**	**64.2**	**80.0**	83.0
Ours	85.2	84.0	83.4	86.7	77.8	90.6	74.7	91.2	87.5	82.8	95.7	66.3	94.9	81.1	63.5	74.5	82.6

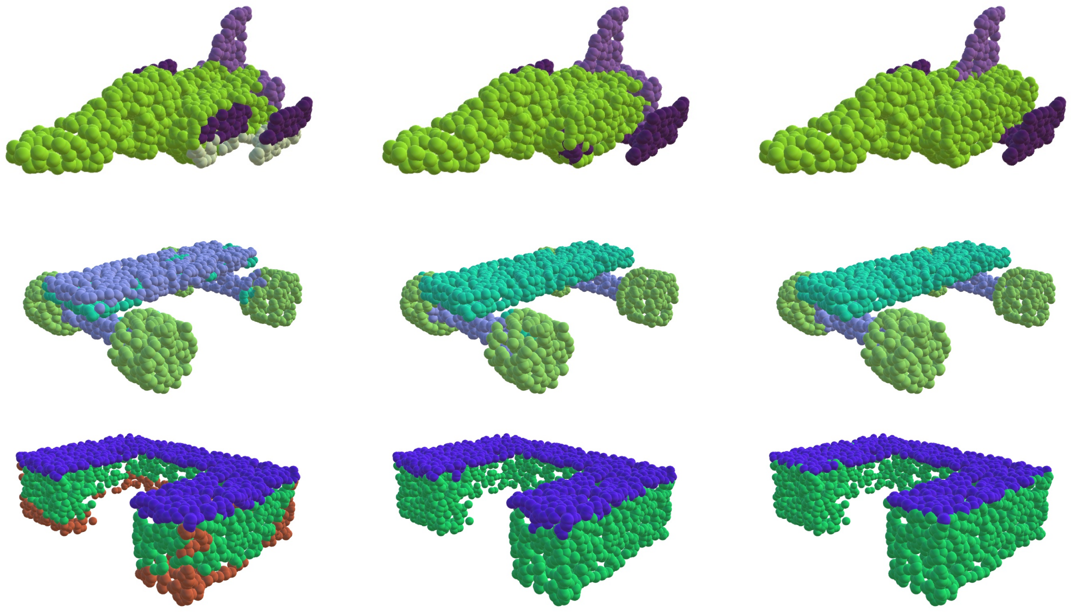


Figure 3-5: Part segmentation results [3]. For each set, from left to right: PointNet, ours and ground truth.

Results. The IoU is measured on point sets to evaluate and compare the performance of our model and other competing models. We follow the same training pipeline as PointNet, and evaluate our model using the mean IoU values across different parts of a shape and all the shapes of each shape category. Our results are compared against PointNet [109], PointNet++ [116], Kd-Net [114], LocalFeatureNet [146], PCNN [144], and PointCNN [143]. The results are reported in Table 3.3, and some quantitative results are visually compared against PointNet in Figure 3-5.

Segmentation on Partial Point Clouds. In many real situations, point clouds obtained from depth sensors are usually incomplete, so the robustness of DGCNN to partal data is

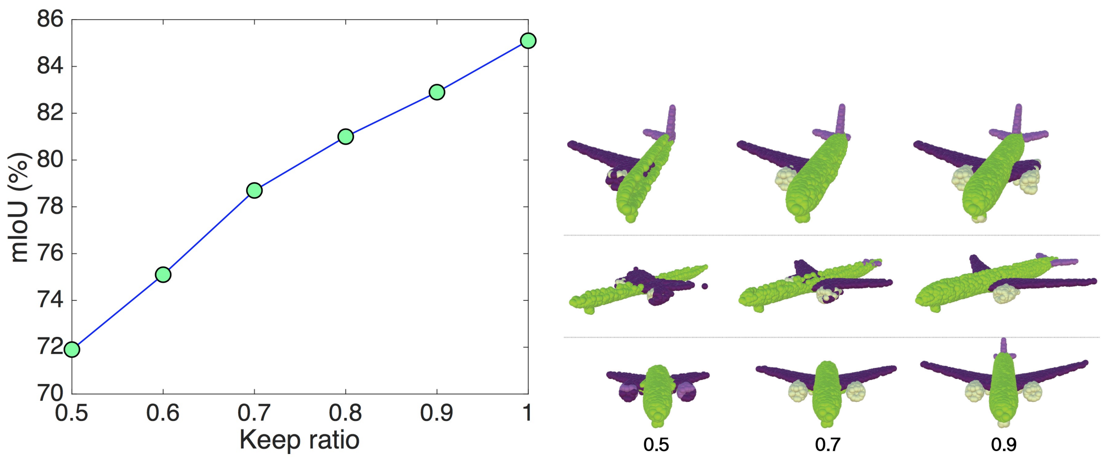


Figure 3-6: Segmentation results on partial point clouds [3]. Left: The mean IoU (%) improves when the ratio of kept points increases. Points are dropped from one of six directions randomly during testing phase. Right: Examples of segmented partial point clouds. Points on each row are dropped from the same side. The keep ratio is shown at the bottom.

evaluated. To generate partial point cloud data, we cut part of the shape from one of six sides by different percentages, and evaluated the model's performance on these partial data. The results are shown in Figure 3-6. On the left, the model's performance (mean IoU) for different ratios of kept points is shown; on the right, generated examples of partial point clouds are visualized.

3.4.5 Indoor Scene Segmentation

The proposed model is also evaluated on real data. In this experiment, the Stanford Large-Scale 3D Indoor Spaces Dataset (S3DIS) [147] is used to evaluate the proposed model for semantic scene segmentation. This dataset includes real 3D scan point clouds for 6 indoor areas including 272 rooms in total, covering 13 semantic types, such as bookcases, chairs and ceilings. For fair comparison, we follow the configuration as in [109] by dividing each room into 1 m $\times$ 1 m $\times$ 1 m cubic blocks, and extracting spatial coordinates, color values and normal vectors to represent point raw features. For training, 4,096 points are sampled within each block; for testing, all points are used. The same 6-fold cross validation scheme [109] is used over 6 areas to report the final evaluation results.

We use the same segmentation model architecture as before, but to produce a probability distribution over semantic labels instead of part labels. The proposed model is compared against PointNet [109], PointNet baseline, the model in [148] and PointCNN [143]. For the model in [148], two different approaches are given: multi-scale block features (denoted as “MS+CU") and grid-blocks (denoted as “G+RCU"). Both of them are augmented with recurrent consolidation units. Evaluation results are reported in Table 3.4, and a visual comparison between PointNet and our DGCNN is shown in Figure 3-7.

Table 3.4: 3D semantic segmentation results on S3DIS [3].

	MEAN IoU	OVERALL ACCURACY
POINTNET (BASELINE) [109]	20.1	53.2
POINTNET [109]	47.6	78.5
MS + CU(2) [148]	47.8	79.2
G + RCU [148]	49.7	81.1
POINTCNN [143]	**65.39**	-
OURS	56.1	**84.1**

3.5 Discussion

This chapter presents a novel deep learning-based framework for 3D point cloud analysis, and its effectiveness on classification and segmentation tasks for both synthetic and real data is assessed. Extensive experiments results validate the importance of intrinsic features extracted from local point sets for 3D understanding. In the next chapter, this point cloud deep learning model will be extended to estimate object pose parameters in order to further enhance IoT user experience in AR environment.

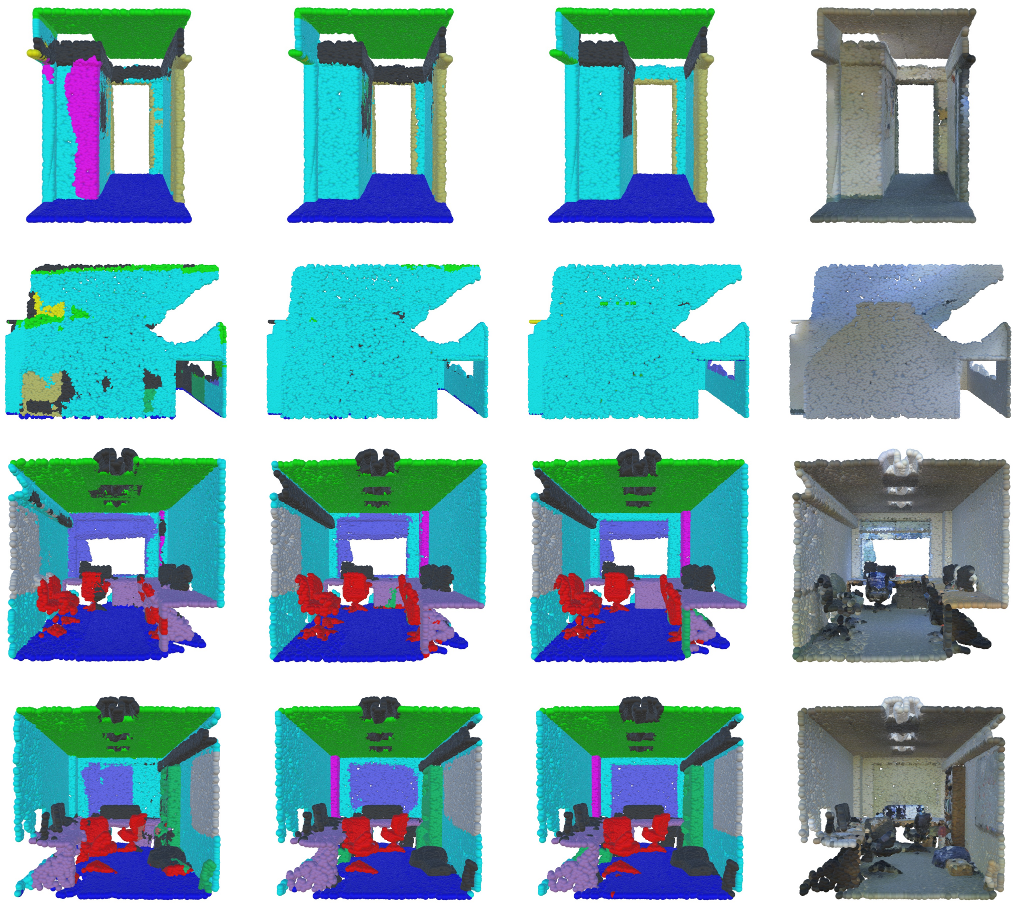


Figure 3-7: Semantic segmentation results [3]. From left to right: PointNet, ours, ground truth and point cloud with original color. Our model produces better segmentation results.

Chapter 4

Deep Learning-Based AR-IoT Interaction System

In Chapter 2, an integrated framework is constructed to demonstrate IoT experience enhancement in AR environment. Yet, within this framework, its object pose estimation modules are implemented only using conventional handcrafted methods, which cannot handle complex and noisy shapes. This motivates the research presented in Chapter 3, where a data driven deep learning approach is developed to analyze complex geometric shapes in an automatic manner. The proposed neural network model will be integrated into the AR-IoT system in this chapter to support robust object pose estimation of challenging 3D shapes. Equipped with such robustness, the application scope of the system can be further extended to an industry level. In this chapter, an AR-IoT system supporting industrial applications, such as mechanical maintenance, and other highly-interactive operations, will be presented.

4.1 Background

IoT makes sensing ubiquitous and data more accessible, and underpins big data analytics. It can easily bypass the border between different application domains, for example, from

This chapter is based on the author's earlier work: Towards Industrial IoT-AR Systems using Deep Learning-Based Object Pose Estimation, in 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC) ©2019 IEEE. DOI: 10.1109/IPCCC47392.2019.8958753.

smart home to industrial IoT, due to its hyperconnectivity. Today, industrial consumers are cost-sensitive, and require on-demand production. Examples include, but are not limited to, seamless machine interaction, data-driven decision making, richer sensing, and real-time monitoring and control.

As demonstrated in previous chapters, AR enables human-device interaction in an immersive manner. Extending this benefit to industrial scenarios opens more opportunities, such as rapid design modifications and fast machine maintenance. Many related works [1, 2] have shown that one major industry value of AR is to seamlessly project real-time surrounding information into the user's perception. This requires an industry-level AR system to accurately capture geometric and semantic information of the surrounding environment. For industrial IoT applications, sensed information is usually related to IoT devices or objects, and the overlapping of digital content from real objects is necessary. Integrating IoT and AR shows great potential for industrial IoT. For instance, it can reduce a machine's downtime by enabling real-time, data-informed condition monitoring, and providing augmented assistance in case of repairs, allowing unskilled employees to learn how to conduct basic maintenance quickly. Currently, machine status monitoring and maintenance assistance have received more industrial attention. Research effort has also been devoted to reducing associated cost and enhancing efficiency in order to build smart factories.

However, AR has not been adopted to the fullest for manufacturing applications. One reason is that such applications usually require accurate object identification and localization to render information at correct positions. Yet, as shown in earlier chapters, many existing smart devices do not have distinguishable visual patterns, and their geometric structures are complex. These factors increase the difficulty of adopting existing AR systems in this area. Therefore, accurate object pose estimation is desired to support industrial AR systems. As mentioned before, many existing AR systems use fiducial markers, such as QR codes, to recognize and localize target objects, but additional configuration is usually needed. Hence, they do not provide a seamless use experience. Many ongoing research projects [149, 150] and the techniques developed in Chapter 2 try to extract handcrafted features for this purpose, but these methods usually suffer from high variability under changing conditions. Recently, advances in deep learning have shown promise for 3D data analysis [109, 116]. To our best knowledge, our achievement in Chapter 3 is one of the first few research of 3D analysis using deep learning. In this chapter, the neural network models proposed in Chapter 3 will be extended to better support AR-IoT in industrial applications. Two industry application cases: machine status monitoring and maintenance assistance will be demonstrated in this chapter.

4.2 Related Work

This chapter focuses on enhancing industry-level machine monitoring and maintenance experience in AR environments by increasing pose estimation accuracy. Only relevant work not covered in previous chapters will be summarized.

4.2.1 AR for Manufacturing Industry

There has been research using AR as a novel solution to assist mechanical maintenance. In [151], the authors demonstrated using available AR tools to improve a factory scheduling system. This idea has been extended in [152], where traditional industrial guidance was improved with low-cost equipment and development platform. In other similar work [153], the authors further included AR-based virtual technical manuals for hydraulic breakers. They demonstrated enhanced training experience and boosted work efficiency. This line of research is important for applications that require complex operations, such as monitoring machines' conditions and replacing their components. The user-machine interaction experience can be further improved if more visual feedback is provided to predict the possible result of an action. In another recent work [154], a system that tracked users' hand trajectories to guarantee desired actions was presented. Their system recognizes the user's actions, and compared them against preconfigured reference actions. Recent years have witnessed the rapid growth of current AR and industrial IoT markets, but challenges still exist. As discussed in previous chapters, accurate object pose estimation is one of them. This is especially true for the manufacturing industry, where machines usually have complex geometric shapes and indistinguishable textures, the working environment is cluttered, and captured information needs to be processed in real-time.

4.2.2 Deep Learning for Pose Estimation

Deep learning, as a data-driven learning approach, has outperformed many traditional methods, and is developing rapidly for object pose estimation as well. Most existing work adopts 2D convolution neural networks for object pose estimation based on captured images [155, 156, 39]. On the other hand, early 3D deep learning models for pose estimation can only work on regular formats, such as volumetric data, so before being fed into the model, 3D point clouds or meshes need to be converted into 3D grids [157, 158]. In Chpater 3, we summarized some early work that can directly operate on irregular 3D point clouds [159], and this framework has been adopted for 3D localization [160]. In this chapter, inspired by multi-source approaches [161, 162, 163], both visual and geometric information will be fused for object pose estimation.

4.3 System

Similar to the framework presented before, this system also include three main modules: data collection, target machine localization, and augmented visualization. The working pipeline is shown in Figure 4-1. The system obtains a machine's real-time information, and renders it to the user correspondingly. Details of this system are described in this section. The visualization module is the same as in previous framework, so its description is neglected.

4.3.1 System Workflow

This system involves multiple technologies, with each being more complex than previous systems, so different parts need to be well coordinated. The required steps to be performed for each module are illustrated in Figure 4-2. These three modules, shown in different colors, work independently before their results are shared.

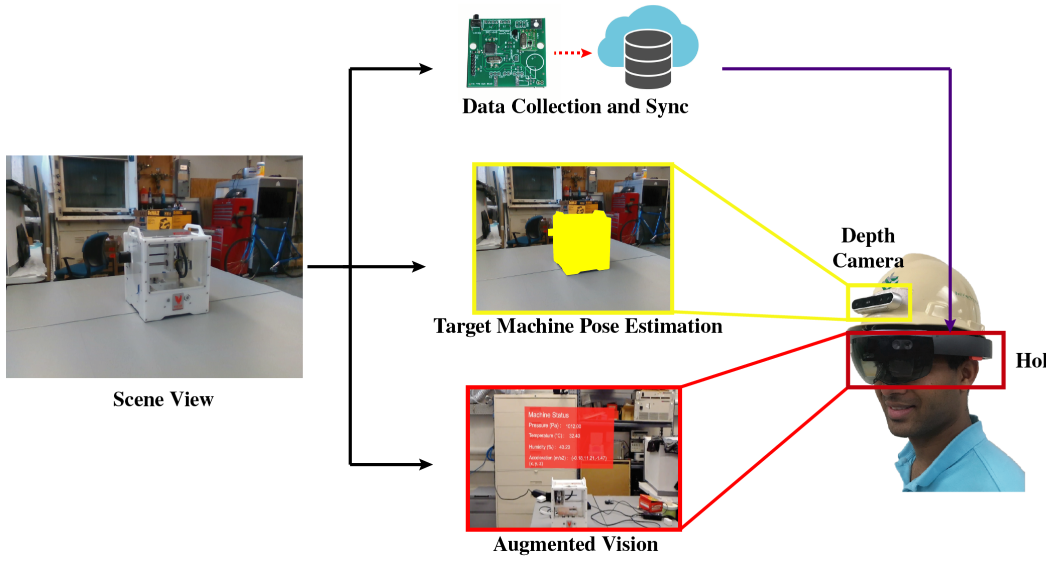


Figure 4-1: The pipeline of the system [4]. It collects the target machine's information in real-time and estimates its pose via a helmet-mounted depth camera. Nithin can see superimposed augmented digital information through the HoloLens.

First, the machine data collection module (pink colored) senses the machine's real-time status data, and uploads it to a central server, so other system modules can access it. This module keeps working while the system runs so that the machine's status can be synchronized in the cloud.

Second, the object pose estimation module (yellow colored) estimates the in-view machine's pose parameters. As before, visual and geometric information from color and depth images are captured for this purpose, but here we use deep learning models to increase the estimation accuracy. We further accelerate the inference speed of deep learning models by taking advantage of modern GPUs deployed on our server. Once the target object's pose parameters are inferred, they, together with preconfigured digital content, are sent to the HoloLens for augmented rendering. After this step, this module terminates, because the object's pose only needs to be estimated once for the HoloLens's rendering system.

Third, the augmented visualization module (blue colored) renders received digital content to users. This module works similarly to that in previous presented systems. Note that, this module will not start working until relevant pose parameters are predicted, and terminates at the same time as the system.

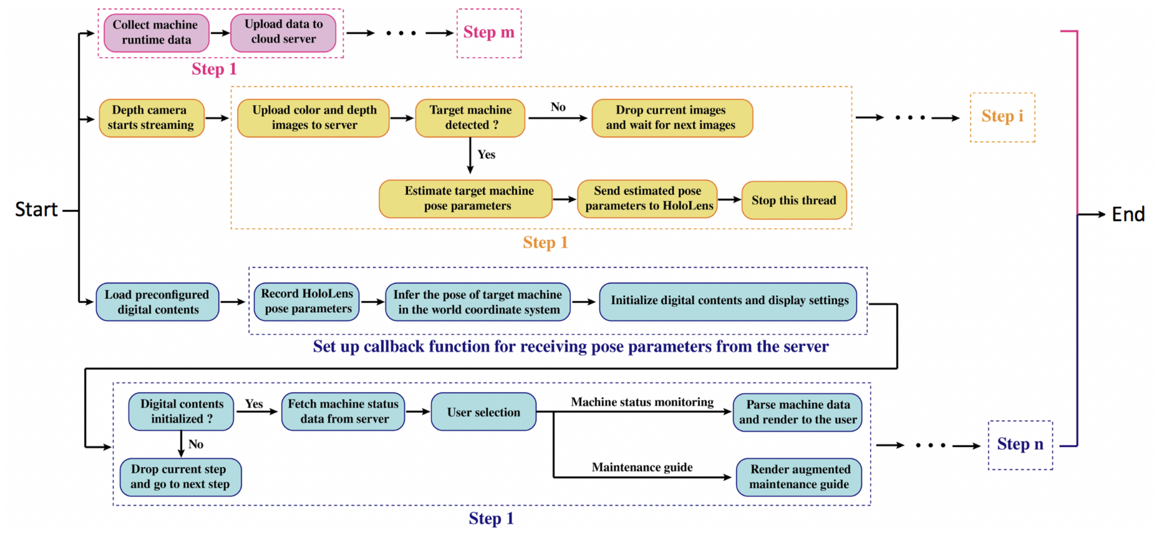

Start
Collect machine runtime data
Upload data to cloud server
Step m
Step 1
Depth camera starts streaming
Upload color and depth images to server
Target machine detected ?
No
Drop current images and wait for next images
Yes
Estimate target machine pose parameters
Send estimated pose parameters to HoloLens
Stop this thread
Step i
Step 1
End
Load preconfigured digital contents
Record HoloLens pose parameters
Infer the pose of target machine in the world coordinate system
Initialize digital contents and display settings
Set up callback function for receiving pose parameters from the server
Digital contents initialized ?
Yes
Fetch machine status data from server
User selection
Machine status monitoring
Parse machine data and render to the user
No
Drop current step and go to next step
Maintenance guide
Render augmented maintenance guide
Step n
Step 1

Figure 4-2: Workflow of the proposed system [4].

4.3.2 Machine Status Synchronization

We demonstrate this upgraded system on a desktop PCB milling machine. This machine is able to cut PCBs at 2,600 mm per minute [164]. To monitor the machine's status, a SensiBLE IoT module is attached to the machine to collect its physical attributes in real-time. This SensiBLE module collects temperature, humidity, pressure and accelerometer data, and send it to our server. A smartphone app is used to control data collection and routing processes of this sensor module. The uploaded data on the server updates its corresponding cloud state simultaneously, so that the real-time machine status can be displayed to users.

4.3.3 Target Machine Localization

Significantly different from the system in Chapter 2, this pose estimation module is learning-based, and includes four different neural networks: segmentation, image feature extraction, point cloud feature extraction, and pose estimation networks. The whole module takes as input a color and depth image pair, and outputs pose parameters of the in-view machine. The pipeline of this pose estimation module is shown in Figure 4-3.

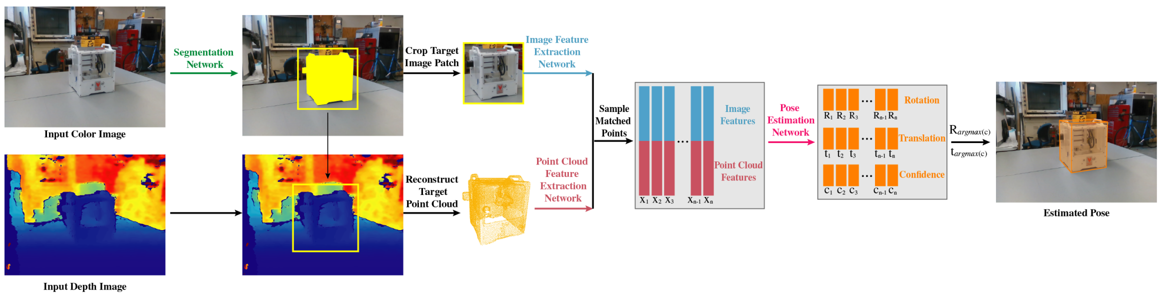


Figure 4-3: Pipeline for target machine pose estimation given an input RGB-D image pair [4].

Segmentation Network

The first task we need to accomplish is to recognize the target machine from the captured image, and crop it. In this way, uninformative background parts can be eliminated to reduce computation cost and avoid potential misleading information. This is achieved by producing a foreground mask indicating the target machine region in the color image. Since color and depth pixels are one-to-one corresponded, from segmented target machine region pixels, we can crop the machine from the depth image as well, and then reconstruct the partial point cloud of the machine in 3D space. We implement a 2D encoder-decoder neural network for this segmentation task. Specifically, this network takes as input a color image including the target milling machine, passes it through a encoding and a decoding 2D CNN network sequentially, and generates a 2D grid of the same spatial dimension as input but with only two channels indicating foreground and background possibilities. Within the network, 2D convolution is used to process feature maps, and pooling/unpooling is used to modify spatial dimensions of feature maps. In Figure 4-4, we show the network architecture.

Image Feature Extraction Network

From the cropped target machine image patch, we next need to extract compacted feature vectors containing helpful visual information for pose estimation. The implemented image feature extraction network takes as input a tight image patch containing the segmented target machine, and produces a compacted 1D vector for each of the pixel within this image patch. We use the ResNet18 [95] network as the backbone structure for this model. The

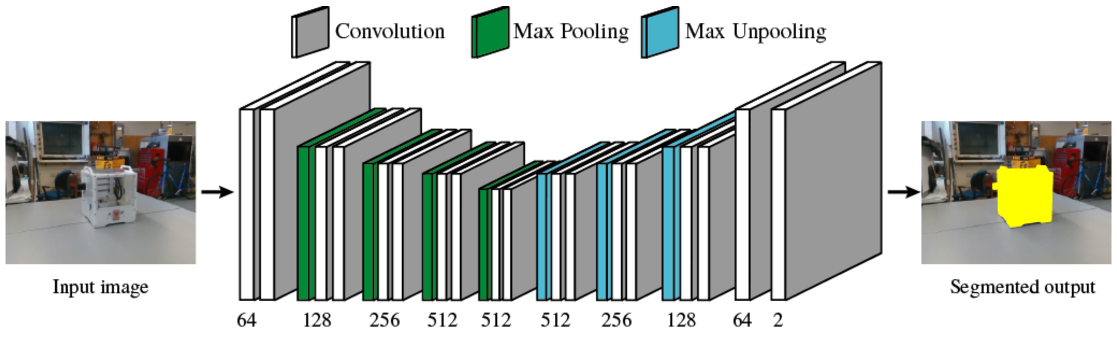


Figure 4-4: The machine segmentation network [4]. The dimension of each corresponding feature map is shown at bottom.

goal here is to transform the original raw color information into a latent feature vector encoding visual appearances for inference computation. Mathematically, this process can be expressed as: for an input image patch, $H \times W \times 3$, this neural network outputs a 2D grid of shape $H \times W \times d_c$, where d_c is the dimension of embedded latent feature vectors.

Point Cloud Feature Extraction Network

Also, we include 3D geometric information to increase object pose estimation accuracy, since the recorded 3D appearance of an object is uniquely determined based on the pose parameters of the machine. For instance, different viewpoints will usually result in different 3D appearances. Despite it is possible to apply the same image feature extraction network on cropped depth images, intrinsic 3D geometric information cannot be fully discovered in this way. On the other hand, 3D point clouds contain this information extrinsically. Therefore, it makes more sense to reconstruct point clouds from the cropped depth image patch and extract geometric feature representations from them for pose estimation. Here, to support point cloud-based feature extraction, the neural network model, DGCNN, proposed in Chapter 3 is adopted. Specifically, we use EdgeConv to compute a feature vector describing local geometric property of a central point.

The adopted procedure is described as below. For each point, $\mathbf{x}_i$, its k nearest neighboring points, $\{\mathbf{x}_{i1}, \mathbf{x}_{i2}, ..., \mathbf{x}_{ij}, ..., \mathbf{x}_{ik}\}$ are found in each feature space, and their feature vectors are acquired via multilayer perceptrons. Essentially, this step converts 3D point coordinates to latent feature embeddings, expressed as $\mathbf{f}_{ij} = MLP(\mathbf{x}_{ij})$. Next, point pairs are

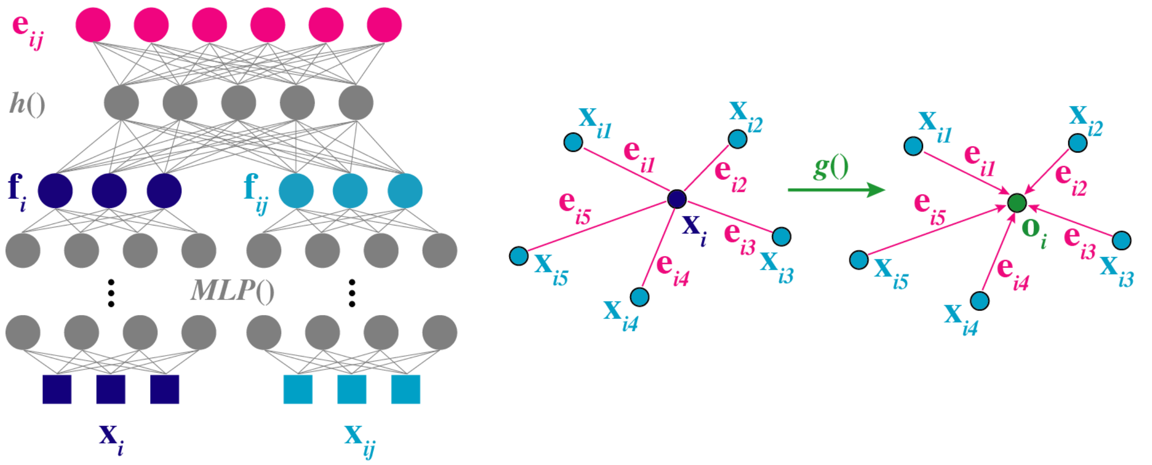


Figure 4-5: Point cloud feature extraction [4]. Left: Compute the pair feature between a central point $\mathbf{x}_i$ and a neighboring $\mathbf{x}_{ij}$. Right: Compute the local geometric feature $\mathbf{o}_i$ for $\mathbf{x}_i$.

generated between $\mathbf{x}_i$ and each of its neighbors $\mathbf{x}_{ij}$ to produce inter-point features. This step is expressed as $\mathbf{e}_{ij} = h(\mathbf{f}_i, \mathbf{f}_{ij})$, with $h()$ and $\mathbf{e}_{ij}$ being a nonlinear operation with learnable parameters and the edge feature, respectively. Finally, all edge features around a central point are accumulated, and expressed as $\mathbf{o}_i = g(\mathbf{e}_{i1}, ..., \mathbf{e}_{ij}, ..., \mathbf{e}_{ik})$, with $g()$ and $\mathbf{o}_i$ being another nonlinear learnable function and the output point feature of $\mathbf{x}_i$, respectively. These operations are illustrated in Figure 4-5.

Pose Estimation Network

Once we have both visual features from color image patches and geometric features from reconstructed point clouds, they can be merged to predict the target machine's pose parameters. Usually, the correspondence between each color pixel and reconstructed point is known when the camera configuration parameter is available. Therefore, these extracted features can be fused in a deterministic way. Another observation worth mentioning is that each fused feature vector encodes different information, resulting from different local appearances. We can estimate a set of pose parameters from each fused feature. In fact, we find that even though the point cloud features are fused with image features, the fused features still share the same format as original point features. Therefore, we can use the same MLP layers to process them and produce the desired pose parameters. Here, we also generate a global feature vector by accumulating features across different feature vectors

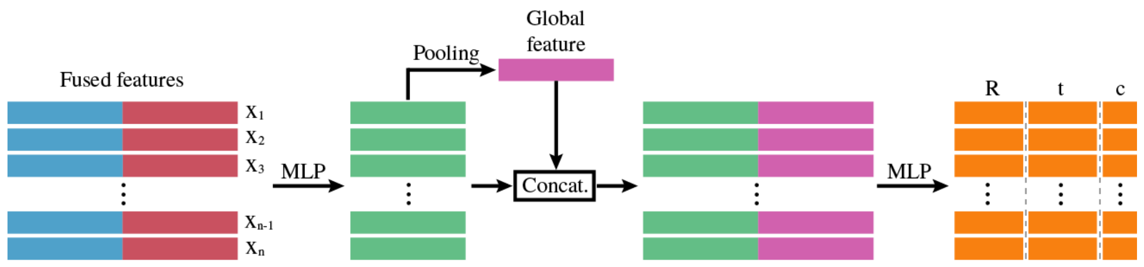


Figure 4-6: Pose estimation network [4].

for each individual feature channel. This global feature is duplicated and concatenated with individual fused feature to cover both local and global information.

Next, a natural question is which set of estimations should be used as the final estimation. To resolve this, for each estimation set, we also predict a confidence value associated with it, in a way that a higher confidence value indicates more accurate estimation. In the end, this network regresses rotation parameters represented by quaternion values, translation parameters represented by spatial coordinates, and a confidence value represented by a normalized scalar number between 0 and 1. During inference, the pose parameter set with the highest value is selected as the final pose prediction. Figure 4-6 graphically demonstrates all the above described operations.

4.4 Evaluation

In this section, the new system is evaluated and demonstrated from two perspectives. First, at the core of this system is a deep learning-based pose estimation module, showing how to use the proposed deep learning model in Chapter 3 to improve AR systems. This module is quantitatively and qualitatively evaluated. Second, the goal of the proposed system is to enhance user experience and working efficacy during manufacturing and relevant processes, so its real-time performance is also showcased.

4.4.1 Pose Estimation Evaluation

Dataset Preparation. The neural network module proposed in this system follows a standard supervised learning pipeline, where ground truth labels are required during training in order to supervise the model to correctly predict pose parameters. For the proposed pose estimation model, the ground truth pose parameters for each input color and depth image pair are required to train the model. In our implementation, we collect 1,125 color and depth image pairs containing the target milling machine and covering various backgrounds from different viewpoints and distances. However, it is not easy to obtain ground truth pose parameters associated with each image pair, and a manual labeling process has to be performed. To obtain these parameters, we built a point cloud-based labeling tool to facilitate the human point cloud alignment process. We first reconstruct a point cloud out of a given color and depth image pair, and then use the tool to select matched points between the reconstructed point cloud and a template point cloud of the milling machine. We select 4 pairs of matched points to estimate the ground truth pose parameters. This manual aligning processing is shown in Figure 4-7. All the manually aligned samples are split for training (90%) and testing (10%).

Competing Methods. We implement another two pose estimation methods for comparison purpose.

Geometric Method. We first implemented a traditional handcrafted method, Fast Point Feature Histograms (FPFH) algorithm [29]. This method is used in our previous systems presented in Chapter 2. Comparing against this method will prove the improvement of the newly proposed deep learning method. We compute FPFH features for both target model and scene point clouds, and then estimate transformation parameters between them according to the Sample Consensus Initial Alignment (SAC-IA) algorithm.

Fiducial Marker Method. QR codes are also used for pose estimation. In our setup, we placed 4 different QR codes uniformly around the target milling machine, and measured

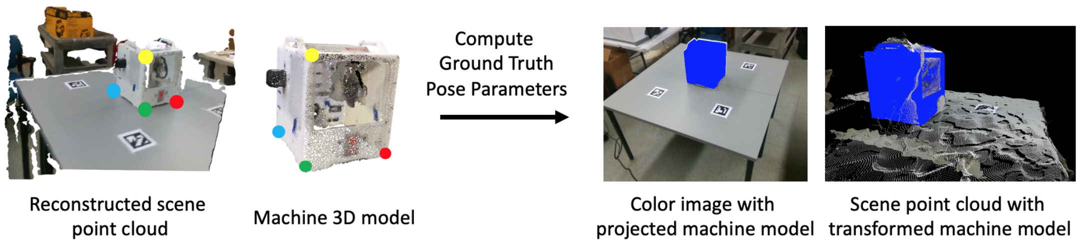

Reconstructed scene point cloud
Machine 3D model
Compute Ground Truth Pose Parameters
Color image with projected machine model
Scene point cloud with transformed machine model

Figure 4-7: Compute ground truth pose parameters from matched point pairs.

their spatial configuration. Once surrounding QR codes are identified and localized, the machine's pose parameters can be easily inferred based on the measured configuration. This method has been used in many existing AR systems, so it should work well. Yet, it is not flexible due to the configuration process.

Implementation Details We adopt the same average distance error introduced in Chapter 2 as the evaluation metric to assess model performance. We implement all the four neural networks using PyTorch Python library [1] on a Nvidia Titan X GPU with 12 GB memory. For competing methods, we run them on an Apple Macbook Pro laptop with an Intel Core i7-6850K 6-Core 3.60-GHz CPU.

Model Performance Evaluation We evaluate the proposed pose estimation module with respect to its speed and accuracy. We test the proposed pose estimation model and both competing methods on the same testing samples. We report the average running time and distance error in Table 4.1. From the table, we can see that our proposed pose estimation

Table 4.1: Model Comparison [4].

	Geometric Method (FPFH + SAC-IA)	**Fiducial Marker** (QR-Code)	**Deep Learning** (Ours)
Run time (*sec.*)	4.732	**0.012**	0.015
Distance error (*m.*)	0.542	0.062	**0.010**

model and QR code method run faster than the traditional geometric method by a relatively large margin. By investigating each component of the geometric method, we find that FPFH algorithm needs to search all the points to detect neighbors for local feature calculation. This takes a large portion of its processing time, and cannot be well optimized on CPU. On the other hand, our proposed model dramatically saves running time by deploying neural networks on GPUs for parallel computing, and achieves similar speed as the simple QR code method. But compared with the QR code method, our model is better on pose estimation accuracy by around 5 cm. This proves the effectiveness of our proposed deep learning model for pose estimation, and provides theoretic support to AR-IoT systems.

We further investigate the module's performance at finer granularity. We count the per-

[1]https://pytorch.org

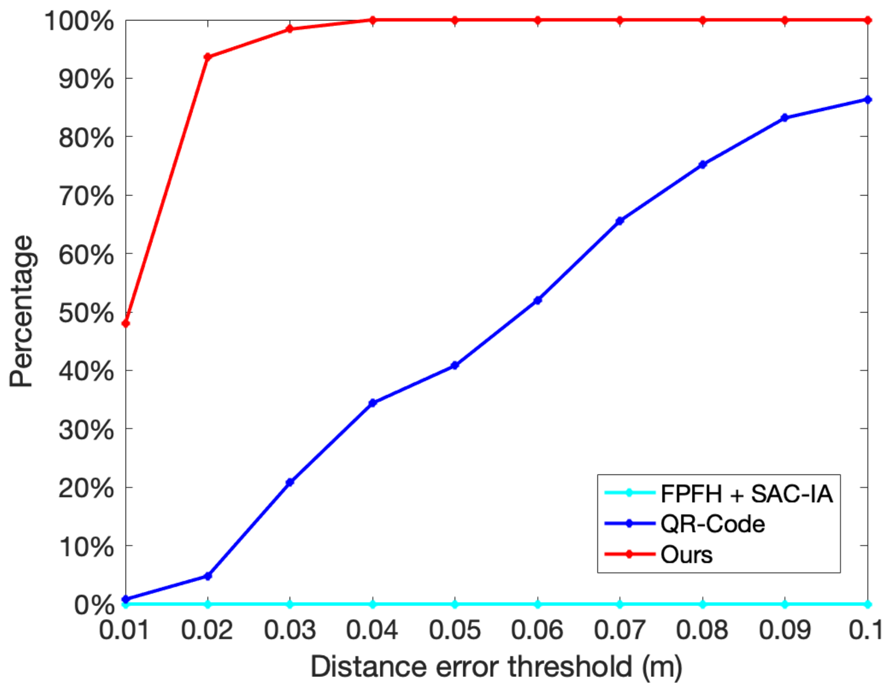


Figure 4-8: The percentage of testing samples with their average distances less than different thresholds [4].

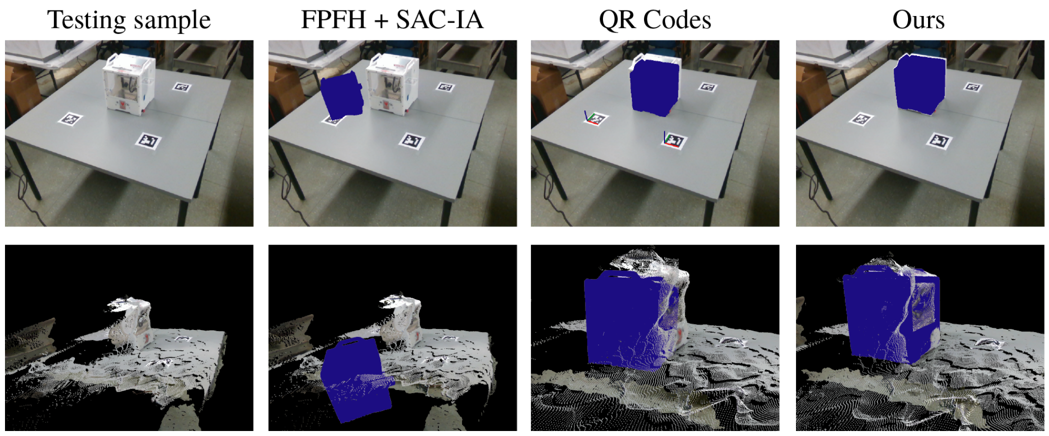
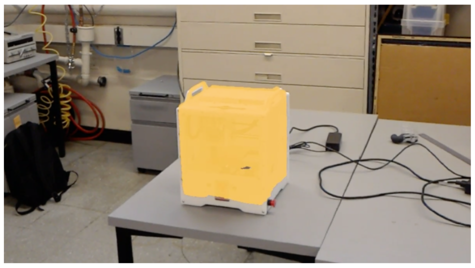
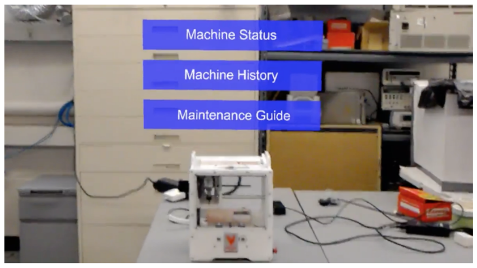


Figure 4-9: Visualize pose estimation results [4]. Top row: Project the transformed machine model onto scene images; Bottom row: Transform the machine model into scene point clouds. Note that QR codes shown in the images are only used for marker-based method.

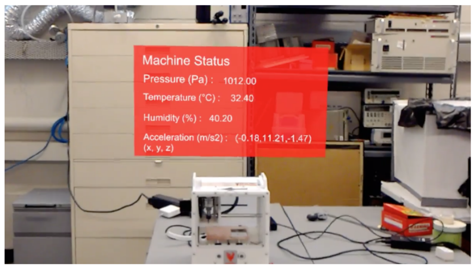
centage of testing samples with their average distance errors smaller than different thresholds. Specifically, we set 10 different error thresholds from 0.01 m to 0.1 m with an interval of 0.01 m. The results for each method are plotted in Figure 4-8. Our proposed method outperforms the other two methods at all the predefined thresholds. The average distance



(a) Superimposed virtual machine

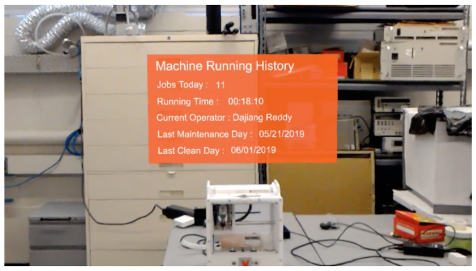


(b) Main user menu



Machine Status
Pressure (Pa) : 1012.00
Temperature (°C) : 32.40
Humidity (%) : 40.20
Acceleration (m/s2) : (-0.18,11.21,-1.47)
(x, y, z)

(c) Machine status panel



Machine Running History
Jobs Today : 11
Running Time : 00:18:10
Current Operator : Dajiang Reddy
Last Maintenance Day : 05/21/2019
Last Clean Day : 06/01/2019

(d) Machine running diagnosis

Figure 4-10: Machine information monitoring and visualization [4].

errors for all testing samples are larger than 0.1 m when the geometric method is used. We also visualize pose estimation results of all the three methods in Figure 4-9 by showing the 2D projection and 3D point cloud of the transformed machine model. These visualization results align well with the reported quantitative results.

4.4.2 System Demonstration

As studied earlier, two use cases are demonstrated: machine status monitoring and augmented maintenance guide.

Machine Status Monitoring. In this case, temperature, pressure, humidity and accelerometer data of the target milling machine are collected in real-time and continuously uploaded to our server. Similar work sharing the same philosophy can be found in eyeDNA [165]. In this demo, all the uploaded data are synchronized to the HoloLens for updating the numbers displayed in a virtual panel rendered to the user. The user can then monitor the machine status data in real-time by simply looking at it. Different application cases are shown in Figure 4-10. Figure 4-10 (a) shows a visual machine overlaid over a real machine in the view of an HoloLens. By showing the virtual machine at correct position, the user can be aware of the machine of interest, and this facilitates other follow-up operations. Figure

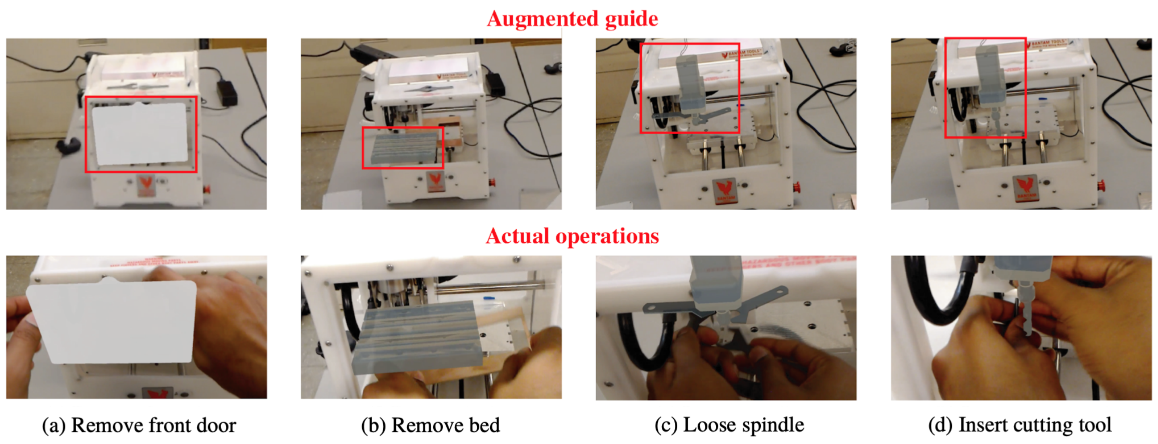


Figure 4-11: Machine information monitoring and visualization [4].

4-10 (b) shows the main operation manual including viewing machine's status, checking its diagnosis history, and performing augmented maintenance. In Figure 4-10 (c) and (d), examples of showing the milling machine's status and diagnosis history are presented.

Augmented Maintenance Guide. We also demonstrate the augmented maintenance assistance using our system. Usually, in common maintenance practice, operators need to check back and forth between a technical manual and the machine to be repaired multiple times. However, if the maintenance guide can be rendered directly onto the machine part to be maintained or repaired, using AR, the operational efficiency and user experience can be greatly improved. Here, our system provides an immersive approach for this case by projecting a sequence of augmented guides through the maintenance process. To demonstrate this, we create an augmented maintenance guide for replacing the cutting tool of a milling machine. Within this guide, CAD models of different machine parts are animated at proper positions to guide operators to perform certain actions. In Figure 4-11, a sequence of real-time recorded augmented maintenance guides are captured to demonstrate this process. We argue that, compared with a physical technical manual, an AR-based maintenance guide can enable the user to learn quickly and intuitively in an efficient manner.

4.5 Discussion

In this chapter, the AR-IoT system presented in Chapter 2 is upgraded with the deep learning model designed in Chapter 3 for robust object pose estimation. This extends the original system from indoor IoT applications to industrial IoT applications. Evaluation results show that the proposed pose estimation module achieves much better accuracy than previous handcrafted traditional methods, while also satisfying runtime requirements. Two real-world applications are demonstrated: machine status monitoring and augmented maintenance guide.

The neural network model implemented in this chapter is a supervised learning approach, which requires ground truth labels to be known during training. Yet, when the number of target machines increases, more manual work is required to obtain ground truth labels. To resolve this scalability issue, an automatic method for pose learning will be discussed in the following chapter.

Chapter 5

Object Pose Estimation without Human Labeling

In the previous chapter, the advantage of deep learning models was demonstrated over other traditional methods for object pose estimation. Yet, this deep learning method requires ground truth labels during training. However, obtaining ground truth labels is a time consuming and expensive process, because it has to be done manually in order to guarantee training accuracy. For the training data collected in the previous chapter, it took ~ 48 hours for 3 people to manually align all 1,125 collected scene point clouds just for one target 3D model. If the database scale becomes large, which is usually the case for an industrial scenario, the time and human effort required for labeling would grow tremendously. This limits the scaling, development and deployment of such method for real applications. Therefore, an automatic training process without much human effort is desired, which forms the main focus of this chapter.

5.1 Background

Object pose estimation has been a long-standing problem due to its applications in various fields, hence a large number of methods have been developed. As shown in Chapter 2, traditional methods extract handcrafted features and match correspondences for pose estimation. These methods usually require recognizable texture patterns for correspondence matching. Recently, learning-based methods have rapidly evolved and outperformed traditional methods. On one hand, following the same pipeline of traditional methods, they can learn how to extract feature vectors and find matched correspondence pairs automatically from training data. On the other hand, they can even learn how to directly estimate objects' pose parameters from input data without going through intermediate procedures, which is almost impossible for traditional methods. These approaches have attracted more research interest due to their compactness. The proposed method in Chapter 4 falls into this category, and this chapter will keep exploring research following this line.

Typically, learning to directly estimate objects' pose parameters requires supervision, meaning ground truth pose parameters need to be known. In this way, a supervised training loss can be calculated and its gradients can be back-propagated through the network to update deep learning model parameters. In order to train a model that performs well on testing samples, training and testing data should follow a similar data distribution. So, if we want to test the model on real data, it would be better to train the model on real data as well, also with similar light conditions, backgrounds and viewpoint ranges. However, ground truth labels are not easy to obtain, especially at a relatively large scale, because human effort is usually required to annotate labels.

To handle this difficulty, generating synthetic data has been proposed recently for image-based object pose estimation. Physical simulation engines and camera models are used to synthesize 2D projections of target objects from arbitrary viewpoints and distances. In this way, labeled samples can be created for free. This approach has been well demonstrated in [166, 167]. In their work, the discrepancy between real and synthetic data needs to be handled well, and this is known as the domain gap problem. Various data augmentation techniques can be used to regularize model training, such as randomly dropping image patches, changing image hues, and adding random noise. However, it is still challenging to minimize the gap between real and synthetic training data, which makes 2D image-based data augmentation still an active research area. Methods like Generative Adversarial Networks (GAN) [168] and the search space-based technique [169] have been proposed to address this problem.

On the other hand, 3D data, especially point cloud data, suffers less from the above mentioned domain gap problem. 3D data encodes geometric information of objects' visible surfaces, and the geometric structure is not affected by light conditions. Also, the noise for 3D data is reflected as the distribution of 3D surface points, and it is easier to simulate compared to 2D visual noise. In this way, the domain difference between real and synthetic 3D data can be better reduced. Further, geometric analysis of 3D objects does not involve color information, so the object pose estimation does not depend on objects' textures. Therefore, in this chapter, the input 2D image is not used, and only synthetic 3D point cloud is taken as input for pose estimation. Different trials and findings will be discussed along our way for improving object pose estimation accuracy.

5.2 Related Work

5.2.1 Image-Based Methods

Color images are the easiest visual information people can obtain using cameras, so many object pose estimation works are based on images. Recent research [166, 167, 39] has demonstrated the value of deep learning models for object pose estimation using both real and synthetic images.

Real images-based methods can be found in [170, 171, 172, 39, 173]. The problem is actually ill-posed, since the scale of projected 2D objects can cause ambiguity for translation parameters, and some work [174, 175] uses additional depth information to avoid this ambiguity. Among them, PoseCNN [39] presents a multi-task approach to jointly segment foreground objects, and estimate the translation and rotation parameters. To avoid the gimbal lock problem [176], the authors of PoseCNN estimated quaternion values first and then converted them to Euler angles. Later, the accuracy was further improved by [177], where instead of estimating one set of pose parameters, many sets of pose parameters are estimated from each foreground pixel. The authors argue that different pixels encode different local information, reflecting their own understanding of local appearance for pose estimation. During the estimation process, a scalar value ranging between 0 to 1 is also estimated for each set of parameters, indicating the confidence for associated estimation. This method achieved state-of-the-art performance on both LineMOD [161] and YCB-Video [39] datasets.

Researchers have also been exploring the opportunity of using modern simulation techniques to generate synthetic data and facilitate the training process. The earliest trial is the template-based matching method as demonstrated in [161, 178], where a synthetic database of 2D projections of an object from different views is created. The 2D projected views serve as templates, and are compared against extracted object patches from testing images in order to retrieve poses associated with templates. Efficient searching strategies and refinement algorithms have been developed to improve both speed and accuracy. Recently, this approach has been adapted for deep learning models as in [167]. Yet, the accuracy is limited due to the discretization of pose values and domain gap. One remedy is to project 3D models onto real background [173] to simulate natural image noise.

5.2.2 Point Cloud-Based Methods

Compared to image-based methods, point clouds suffer less from the domain gap problem. 3D point clouds are not affected by scaling or light condition, and it is relatively easy to simulate the point distribution between real and synthetic shapes. Some research work has used point clouds as a pose refinement processing step for image-based methods [167]. The previously mentioned irregularity problem of point clouds in Chapter 3 hinders the development of point cloud-based deep learning models for pose estimation. Recently, after PointNet [109], this problem has been addressed as in [179, 180, 181, 182], but all these work in the same domain with [179, 180] training and testing models on real shapes, and [181, 182] training and testing on synthetic shapes. Generalizing models from synthetic data to real data has not been well studied for point cloud-based pose estimation. SynthCity [183] is one of very few works that studied this problem by providing large-scale synthetic point cloud data. In the remainder of this chapter, methods of adapting a model trained on synthetic shapes to real shapes will be explored.

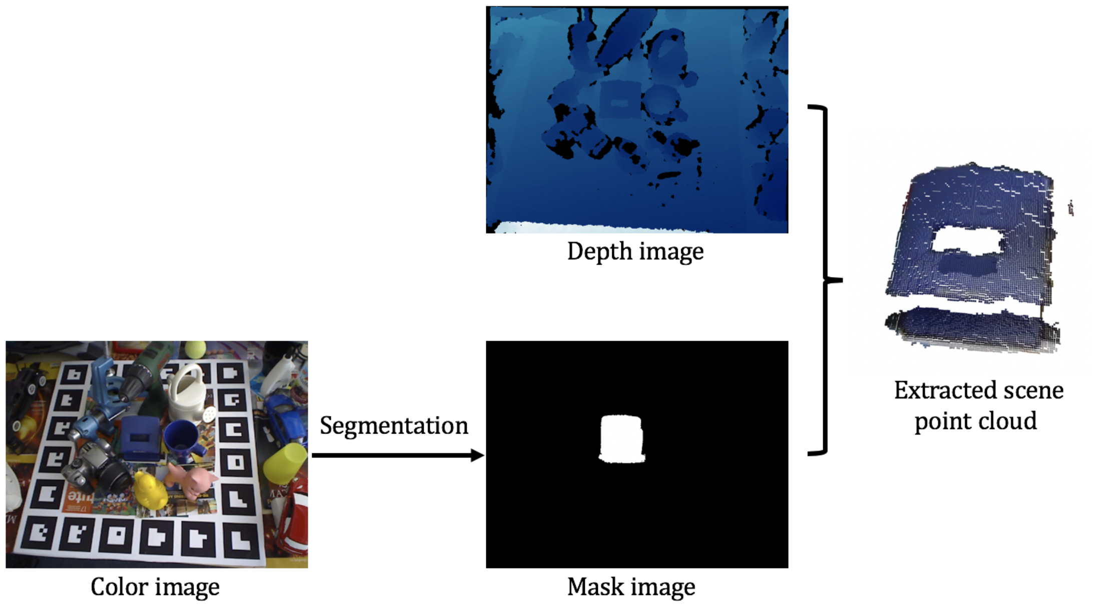


Figure 5-1: Extract the partial point cloud from captured scene images.

5.3 Approaches

In this section, the object pose estimation problem will be formulated, followed by three gradually improved neural network models to minimize the domain gap and improve object pose estimation.

5.3.1 Problem Setup

Input. The inputs to the problem are the captured color image, the depth image and the target 3D model. The depth image contains geometric properties of visible surfaces, as shown in Figure 5-1 top, where different shades reflect different distances of surface points to the camera. We can reconstruct a complete 3D point cloud of the environment from the depth image. Yet, the background 3D data is not helpful for target object pose estimation and can further add unnecessary computation, so it should be removed and only the target object point cloud should be kept. To properly crop the target object from the depth image, a binary mask is obtained from its corresponding color image. This process, as shown in Figure 5-1 bottom, can be easily conducted via image-based segmentation models, such as Faster R-CNN [184] and Mask R-CNN [185].

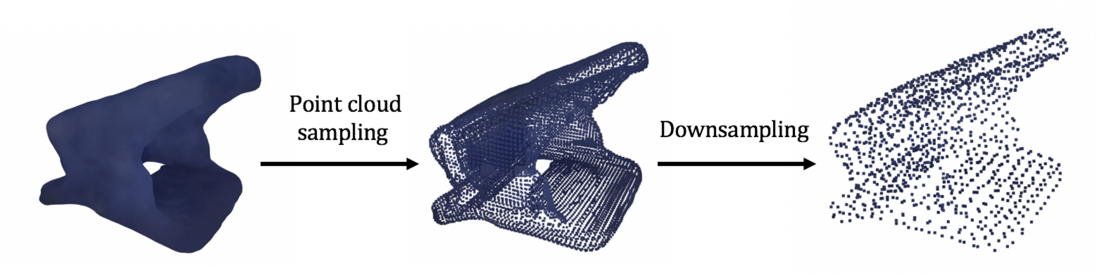


Figure 5-2: Sample point clouds out of meshes.

On the other hand, target 3D models are usually represented by 3D meshes. Although meshes are efficient for rendering, they are not simple to process. Point clouds are uniformly sampled out of mesh surfaces. If the sampled point clouds are dense, the computation cost can still be high. The downsampling is typically performed to adjust point density for a better trade-off between representation power and computation efficiency. This process is demonstrated in Figure 5-2.

Problem Setup. Given a model point cloud of the target object and a partial point cloud cropped from the scene, the goal is to build a pose estimation model, so that after transforming the model point cloud using estimated rotation and translation parameters, it can overlap the partial model point cloud sitting in the scene. This setup is graphically illustrated in Figure 5-3. Note that this problem setup is a little different from that in Chapter

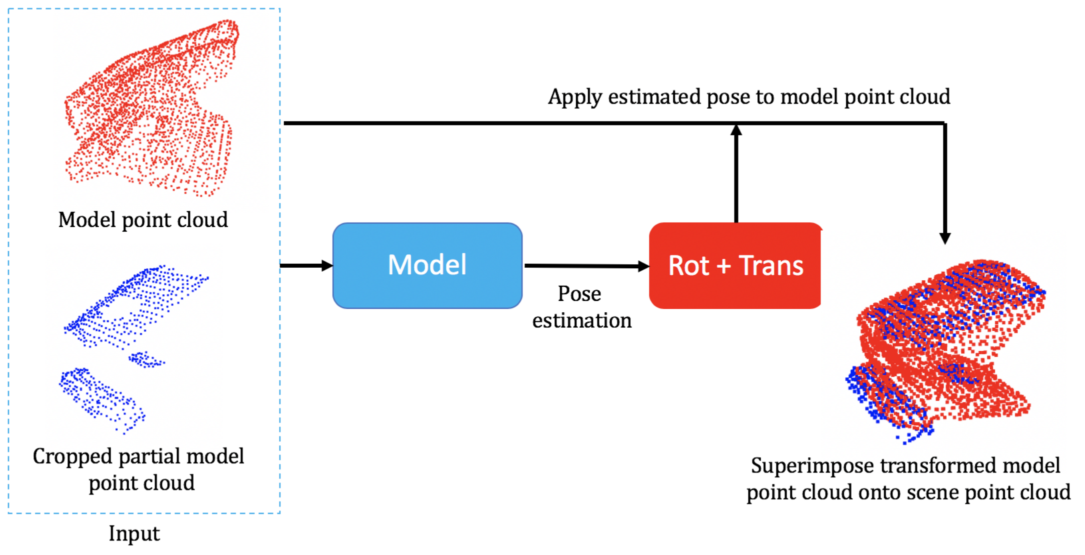

Model point cloud
Cropped partial model point cloud
Input
Apply estimated pose to model point cloud
Model
Pose estimation
Rot + Trans
Superimpose transformed model point cloud onto scene point cloud

Figure 5-3: The problem setup.

4 in a way that the target model point cloud is missing in the previous pose estimation model. The newly added model point cloud input actually serves as an indicator to inform the model the geometric identity of the target object. Without this input, the estimation model can only handle one object at a time.

Data Synthesis. Modern physics engines, such as Unity 3D [186], provide a convenient way to simulate 2D rendering of 3D objects. In this chapter, Pyrender [1] is used to generate 2D projections of the target 3D model from different distances and viewpoints. These pose parameters are recorded as the ground truth label for each corresponding rendered 2D image. The rendered color image contains visual signals which vary depending on light source configurations, and the rendered depth image contains distance information from which 3D point clouds can be reconstructed. As previously mentioned, the color from natural light sources is hard to simulate, and suffers from the domain gap problem, hence we only consider the geometric structure of point cloud data here. Note that during training, only rendered synthetic data are used, and no real data is accessible; yet, during testing, the trained model will be evaluated on real data. The ultimate goal is to achieve good real data performance while training the model on synthetic data.

5.3.2 Single-Prediction Model

Our first proposed model is very straightforward. It takes as input a synthetic partial point cloud, and outputs one set of rotation and translation parameters, so it is named as single-prediction model. The point cloud processing module, DGCNN, proposed in Chapter 3, is reused to analyze and extract 3D features for each point. After obtaining point features, the same global pooling function as before is used to generate a 1D vector by aggregating feature values across different points, encoding global information of the whole input point cloud. A fully-connected layer is then used to produce pose parameters from the global 1D vector.

During the implementation, we find that the model works well on synthetic training data but performs poorly on real testing data, meaning that the model does not generalize well from synthetic data to real data. This finding is visualized in Figure 5-5. In this figure,

[1] https://pyrender.readthedocs.io

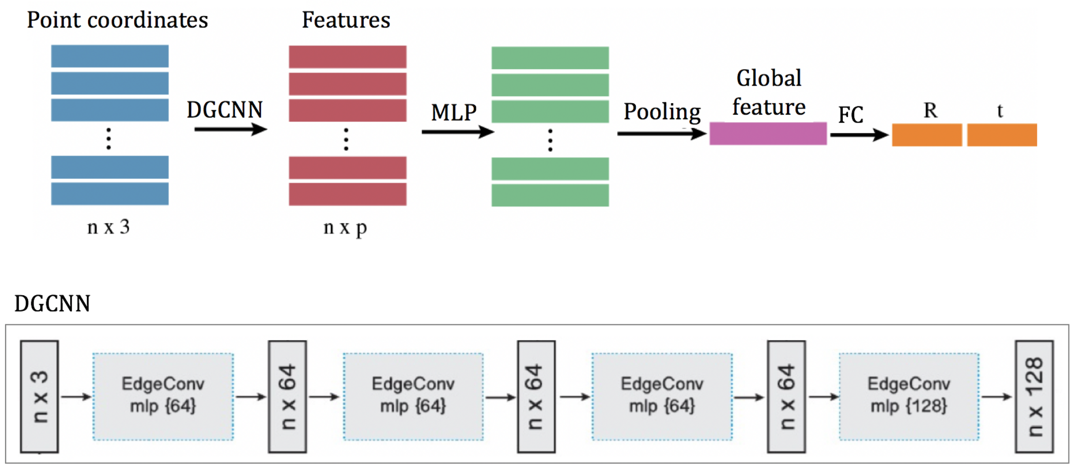


Figure 5-4: The single-prediction model. Top: Model architecture; Bottom: the DGCNN model for point feature extraction.

red point clouds are the transformed point clouds using ground truth pose parameters, and green point clouds are the transformed point clouds using predicted pose parameters. If the pose estimation is accurate enough, we will see a large overlap between two point clouds, as for the training (synthetic) case, while only a small portion of overlap is observed for the testing (real) case.

To further investigate the reason, a real point cloud is placed together with a synthetic point cloud rendered with the same pose parameters. It can be observed that global geometric structure between real and synthetic point clouds are different. Since the proposed

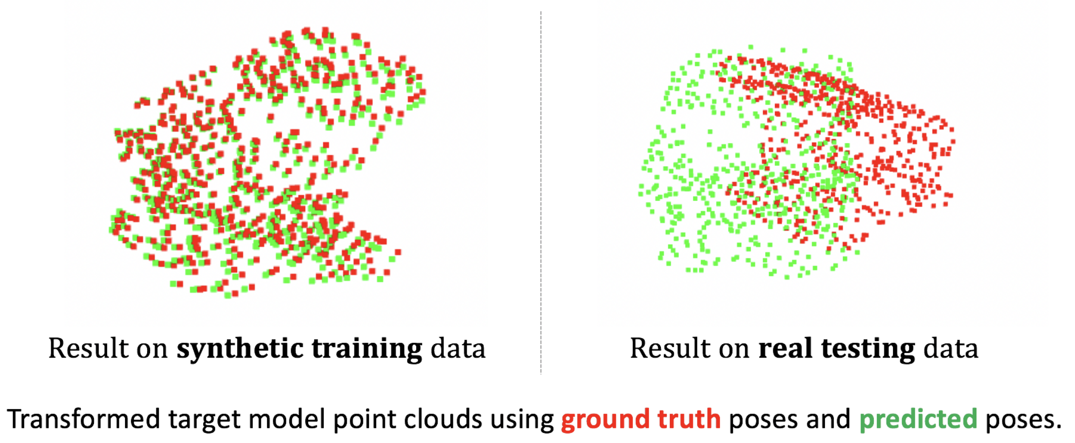


Figure 5-5: Pose estimation results for training (synthetic) and testing (real) samples.

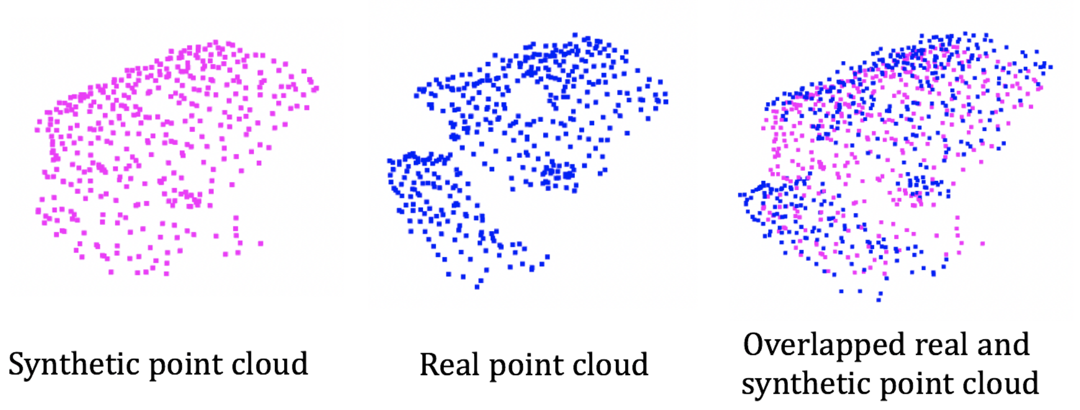


Figure 5-6: Real and synthetic point clouds rendered from the same distance and viewpoint.

pose estimation model is a regression model, different inputs usually result in different pose parameter outputs. This is one key reason for the domain gap problem. However, even though global appearances between real and synthetic data are different, they do share local geometric similarities. These findings suggest that global features are not suitable for object pose estimation when a domain gap problem exists, and local features might be a candidate to minimize this gap and improve estimation accuracy.

5.3.3 Multi-Prediction Model

Inspired by the findings from the single-prediction model, the global feature is removed from the model. From local point features, we can estimate multiple sets of pose parameters, with one set for one point. It is unclear which set to use. To handle this difficulty, we follow a similar framework used in Chapter 4, where a confidence value is estimated together with each set of pose parameters. In this manner, the set with the highest confidence is selected as the final pose prediction. Since multiple sets of pose parameters are estimated from the model, it is thus called an “multi-prediction model". The model architecture is presented in Figure 5-7.

To facilitate model training, we define the loss function as below. First, an average distance between two point clouds, that are transformed by both ground truth and predicted poses, is calculated. The goal is to minimize this distance error, so that a large portion

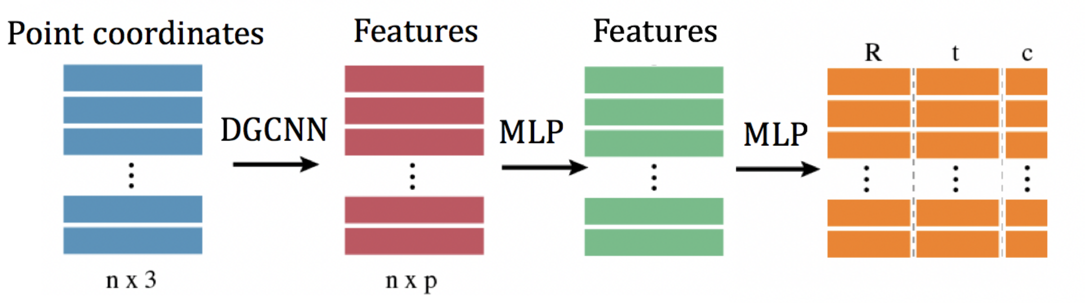


Figure 5-7: The multi-prediction model.

of overlapping can be realized. Mathematically, this loss function for the i^{th} set of pose parameters is expressed as:

$$L_i = \frac{1}{M}\sum_j \|(Rx_j + t) - (\hat{R}_i x_j + \hat{t}_i)\|, \tag{5.1}$$

where R and t are ground truth rotation matrix and translation vector, $\hat{R}_i$ and $\hat{t}_i$ are i^{th} predicted rotation matrix and translation vector, x_j is the j^{th} point of the point cloud, and M is the number of points. Based on this, the final loss function is calculated as:

$$L = \frac{1}{M}\sum_j (L_j c_j - w\log(c_j)), \tag{5.2}$$

where c_j is the estimated confidence for the j^{th} pose parameter set, and w is a hyperparameter for regularization purposes. When minimizing this loss function, each confidence value is pushed towards 1, meanwhile the weighted average distance is minimized.

Qualitatively, the pose estimation accuracy is improved, as shown in Figure 5-8, but the result is still not satisfied. One potential reason is that during training, the loss function, Equation 5.2, is not guided to select the set with the best pose estimation. In fact, no such ground truth label indicating the optimal pose parameter set is available, and it is random, sometimes depends on luck, to find the optimal pose estimation set. However, intuitively, we know that different points should contain different information of the object's pose parameters. For example, visible points with distinguishable local appearance should contain more information. In another word, there must exist a kind of distribution among input points representing their knowledge and confidence for the input object's pose. This

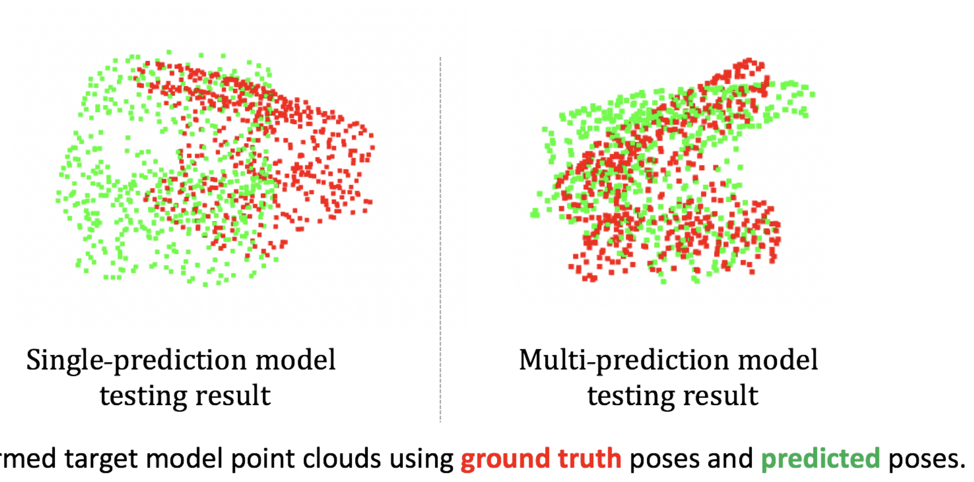

Single-prediction model
testing result
Multi-prediction model
testing result
Transformed target model point clouds using ground truth poses and predicted poses.

Figure 5-8: Testing result comparison between single- and multi-prediction models.

intuition inspires the model design covered in the following subsection.

5.3.4 Knowledge-based Model

We hope to predict a distribution that can reflect the underlining knowledge and confidence for each point, just as shown in Figure 5-9. In this subsection, this intuition is explored and implemented.

The target model point cloud has not been taken as input to the previous two models. In the model to be designed, the target model point cloud is included to provide geometric signals that the model needs to discover from the input partial point cloud. Typically, the numbers of points of the target model point cloud and the partial model point cloud are different, and let us just use $n1$ and $n2$ denote them, respectively. The same DGCNN module

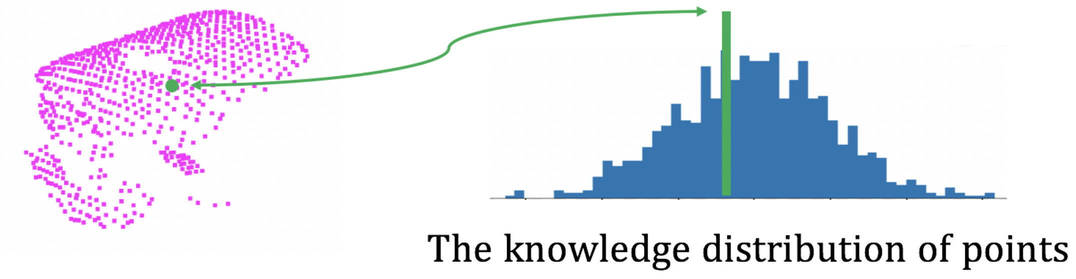


Figure 5-9: A knowledge and confidence distribution of input points.

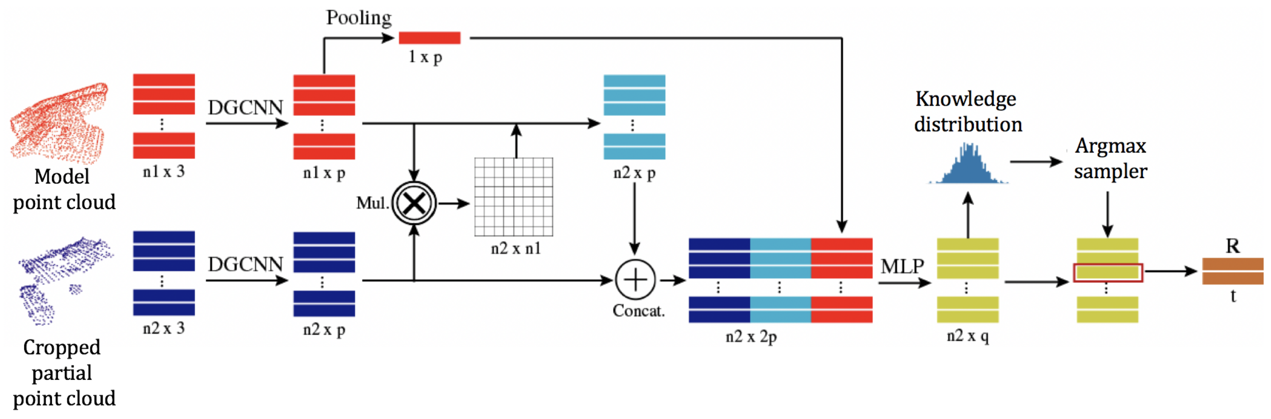


Figure 5-10: The architecture of knowledge-based model.

as previous models can keep being used to extract features for each point, by transforming original spatial point coordinates to high-dimension feature vectors. But note that two independent DGCNN modules are individually used here. From extracted point features, a multiplicative attention module is also implemented. This module basically computes a dot product between each pair of two input point features. A 2D matrix is used to save results, with its (i, j) element storing the dot product result between the i^{th} point feature from the partial point cloud and the j^{th} point feature from the target model point cloud. For each row of this matrix, a Softmax function is applied to normalize their numerical values so that their values sum up to 1. In this way, the values of each row can be considered as a valid probability distribution across the point features of target model point cloud. Usually, these normalized values are called attention weights, and can be used to weighted-accumulate point features from target model point cloud. The attention weights are learned automatically from the data in hope that they can indicate the correspondence or contribution of target model point features to each partial point feature. This weighed accumulated feature provides local geometric reference information, and is calculated for each partial point feature independently.

It is suggested from previous models that a global feature is not helpful for pose estimation due to the appearance difference between synthetic and real point clouds. Yet, the target point cloud is fixed regardlesss whether the partial point cloud is synthetic or real, so it can still be used to provide global geometric information of the target model point clouds.

In this knowledge-based model, point features from different sources are concatenated to fuse information. The complete process is illustrated in Figure 5-10. Note that global features from the target model point cloud is duplicated to match the number of points of the partial point cloud. The fused point features are then passed to a multi-layer perceptron to obtain compact feature representations, and based on that, a knowledge distribution is predicted as promised. It is hoped that this knowledge distribution should reflect the underlining confidence or knowledge about the input partial point cloud's pose. The point feature with the highest value from the distribution is selected to make prediction, using an argmax sampler. During the training process, the model learns how to estimate a reasonable knowledge distribution and select a point feature candidate. Another thing worth noting is that the sampling process is not differentiable, and directly choosing the point feature with the highest confidence does not properly back-propagate gradients. Therefore, the popular Gumbel-Softmax sampler [187] is used as a reparameterization technique to differentiate the training process.

5.4 Evaluation

In this section, the qualitative visualization of the knowledge distribution is presented, followed by another quantitative evaluation. The hole punch model from LineMod dataset [161] is used for all the evaluation.

5.4.1 Knowledge Distribution Visualization

It is hard to quantitatively evaluate the fidelity of the estimated knowledge distribution from the last proposed model, since no such ground truth distribution is available. Yet, it is still possible to visualize the knowledge distribution, and qualitatively verify if the estimated distribution satisfies our intuition.

In this evaluation, the model is trained on synthetic partial point clouds and tested on real partial point clouds to fulfill the predefined problem setup. For each testing partial point cloud, its points are color coded based on their predicted corresponding knowledge

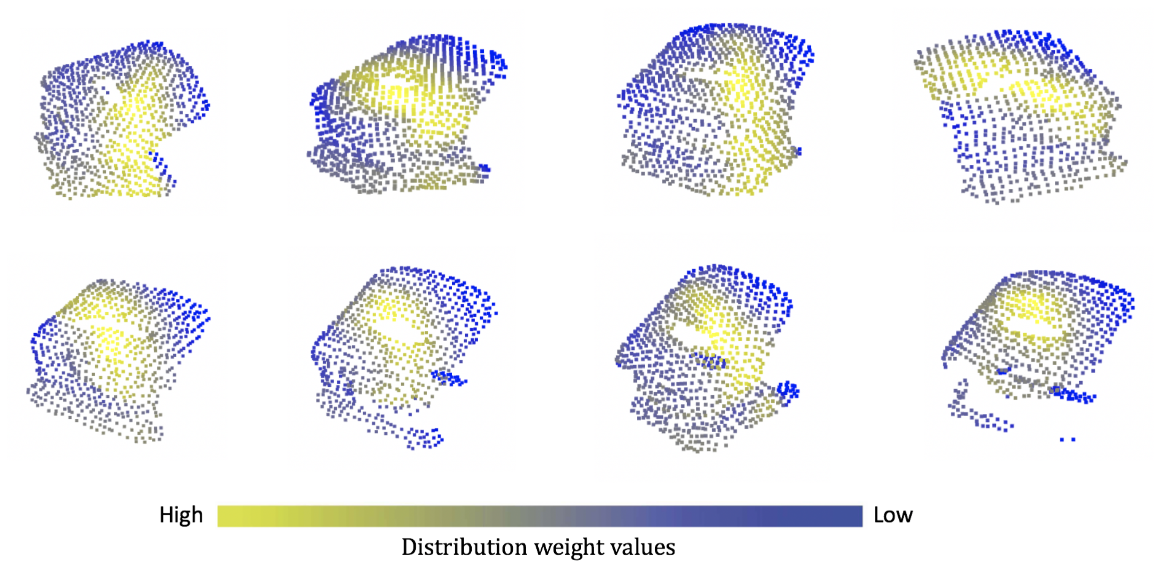


Figure 5-11: Visualize estimated knowledge distribution. For each testing input partial point cloud, its normalized knowledge distribution values are colored according to the color code on the bottom.

distribution values, ranging between 0 and 1. Visualization results are shown in Figure 5-11. From this figure, it can be observed that visible parts with distinguishable local geometric structures are assigned with high knowledge values. For example, despite the hole punch is viewed from different viewpoints, points around the surface hole are assigned with higher knowledge values than other regions. This makes sense because this region is complete, visible and recognizable, thus should contain more information than other incomplete, edge or flat regions.

5.4.2 Pose Estimation Evaluation

The ultimate goal of this chapter is to train a model on synthetic data, and still achieve high accuracy on real testing data. Therefore, the pose parameter estimation accuracy is quantitatively and qualitatively evaluated for all the proposed models. Pose parameters of 1,051 real testing partial point cloud samples are estimated after the model is trained on 186 synthetic partial point cloud samples covering viewpoints different from testing cases. Following the same average distance metric used in Chapter 4, the ground truth and estimated pose parameters are both applied to the same reference model point cloud, and their

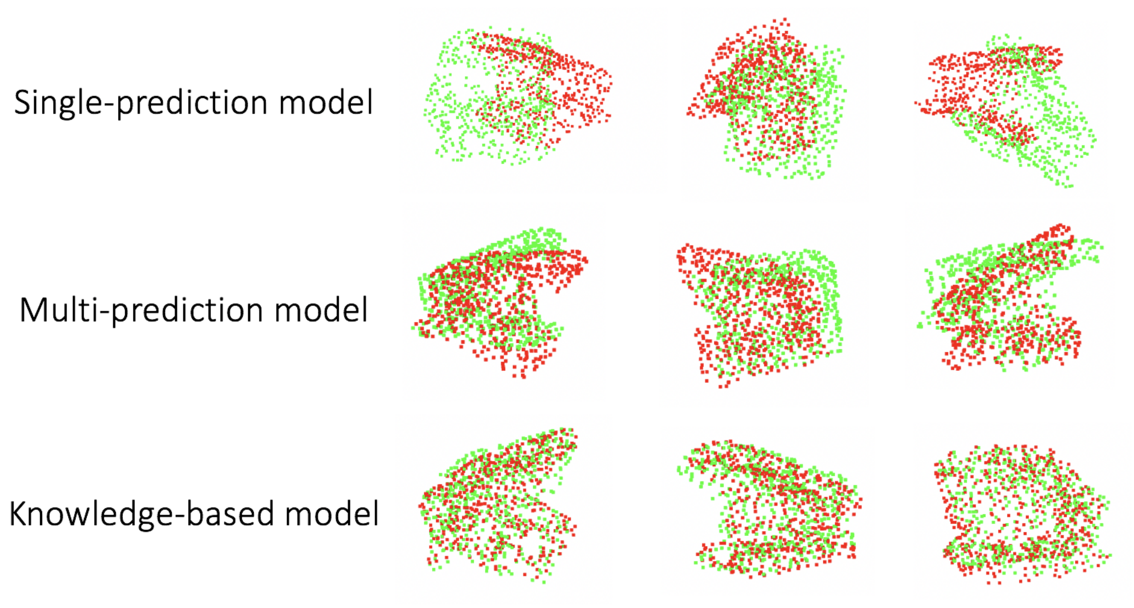


Figure 5-12: Visualize pose estimation results for different models. Similar to Figure 5-8: Red point clouds represent the point clouds transformed by ground truth pose parameters, and green ones are transformed by estimated pose parameters.

resultant average distance is computed. In Figure 5-12, randomly selected testing cases are visually compared. As expected, the knowledge-based model produces satisfactory pose estimation results compared to the other two baseline models. These results are quantitatively verified in Table 5.1.

Table 5.1: The average distance error for different models.

Model	**Single-prediction**	**Multi-prediction**	**Knowledge-based**
Average distance (m)	0.062	0.034	0.021

5.5 Discussion

In this chapter, we explore approaches to train deep learning pose estimation models on synthetic data which can be obtained for free, but still achieve good performance on real testing data. We explore different possible approaches to train object pose estimation models. The main goal is to reduce the domain gap, and three models are developed to show our progress step by step. The first model is simple, and shows that directly estimating object pose parameters from global feature vectors performs poorly, because of the global appearance difference from two different domains. The second model then focuses on local features for pose estimation. The domain gap problem is mitigated, and the performance is improved. Based on all the previous findings, our last model further improves the model performance by including an attention module, target model point cloud as input, and knowledge distribution to explicitly find the best predicted pose parameters. All these models are evaluated.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we introduced our approach towards building fully functional AR-IoT frameworks step by step in order to enhance IoT user experience in AR environment. To conclude, this thesis makes the following contributions:

- Existing IoT technologies are creating a large interconnected system by expanding network accessibility to physical objects. They are now enabling ubiquitous sensing and perception of our surrounding environment, and affecting many aspects of our daily life. People can easily monitor and control any connected device simply via smartphone or web applications. Such interfaces work well for remote scenarios, but become a performance bottleneck for short-range applications due to the lack of direct visual feedback. On the other hand, AR, as an emerging technology that overlays digital content onto the real world, suggests an immersive solution to this problem. So, as the first contribution, we demonstrate how to enhance the IoT user experience in smart AR environments. In the proposed system, users are allowed to directly see the sensed information overlaid on corresponding in-view target objects through head mounted displays. In this way, the digital and physical worlds are overlapped, and users can monitor the real-world data quickly and immersively. To further expand this AR-IoT system's functionality and satisfy users' demands, we also enable the interaction between users and IoT devices in an AR environment. Users can simply manipulate projected virtual control panels associated with in-view IoT devices by using hand gestures. We demonstrate the upgraded system on both light and sound IoT devices.

For AR-IoT applications, it is important to accurately project digital information close to corresponding IoT devices, and any noticeable displacement will result in unsatisfactory user experience and confusion. Many existing AR systems use fiducial markers to estimate pose parameters of in-view objects for proper rendering, but these markers introduce unnecessary artifacts and require additional configuration. Therefore, pose estimation methods based on objects' natural appearances are also studied to support our AR-IoT systems. Specifically, visual and geometric handcrafted methods are implemented to achieve better estimation accuracy. Both textured and textureless objects can be handled by our system.

- Previously implemented handcrafted methods work well for demonstrated simple cases, but fail for complex and noisy object shapes, because the required knowledge and computation dramatically increase for the latter, and it becomes extremely difficult to design well-performed algorithms. Thus, feasible and robust object pose estimation modules are needed to handle difficult 3D shapes for broader applications. Fortunately, deep learning methods provide a data-driven approach to learn how to achieve this goal automatically from training data. So as the second contribution, we study 3D deep learning methods for analyzing geometric data, in hope that a robust neural network model can be trained to extract meaningful 3D features out of complex 3D shapes. In this thesis, we focus on the 3D point cloud data representation due to its flexibility and applicability in many applications. In our proposed model, local information is accumulated hierarchically in a permutation-invariant manner. We evaluate the effectiveness of the proposed model on public classification and segmentation benchmark datasets for both synthetic and real data. Extensive experimental results prove the importance of local geometric features for 3D shape analysis, providing theoretic support for 3D point cloud pose estimation.

- As the third contribution, we integrate the proposed 3D point cloud neural network model into the previous AR-IoT system to support robust object pose estimation for challenging 3D shapes. Four different network models are implemented to segment target machine from the input image, extract both 3D visual and 3D point cloud features, and predict the optimal pose parameters. They are trained on our prepared labeled data, and learn how to handle complex and noisy input 3D shapes automatically. Evaluation results show that the deep learning-based pose estimation module achieves much better accuracy than handcrafted methods, while also satisfying runtime requirement. Equipped with such robustness, the application scope of our system is expanded to industrial level. We demonstrate further enhanced IoT user experience in an AR environment by showcasing augmented mechanical maintenance and other interactive industrial operations.

- The proposed pose estimation neural network model for the upgraded system is a supervised learning method, and requires ground truth labels for training. Usually, labeling training data has to be done manually, which is a time consuming and expensive process. This is handleable for small-scale databases, but as the number of machines increases, the time and human effort required for the labeling process grow tremendously, which limits the application and deployment of the model for real-life scenarios. On the other hand, physical simulation engines and camera models provide a way to synthesize point clouds for any 3D model from arbitrary viewpoints and distances. In this way, labeled training samples can be created for free. Yet, there usually exists a discrepancy between real and synthetic data, which could cause testing performance degeneration. Therefore, as the last contribution, we explore deep learning models that can be trained on synthetic 3D point cloud data but still perform well on real testing data, so that the human labeling process can be avoided. We propose three different deep learning models to approach this problem and mitigate the domain gap problem gradually. The first model shows the inappropriateness of using global feature vectors for pose estimation due to the domain difference; The second model mitigates this problem and improves testing performance by using local feature vectors; The last model achieves visually appealing results by generating a knowledge distribution to find the best predicted pose parameters. We quantitatively and qualitatively evaluate all these models, and the results support the adaption of the third knowledge-based model to applications that involve large-scale datasets.

6.2 Future Work

To end the discussion of this thesis, I present ideas and research directions that might be explored as future work.

Dark Work Environment. In the proposed system, we localize the in-view target object based on its visual and geometrical appearances. This information is captured via image sensors under perfect light conditions, so that the object's texture and geometric shapes, both 2D and 3D, can be properly detected. However, in many real-world scenarios, light conditions of the work environment can be low, which causes degenerative image quality. When taking such image data as input, our proposed pose estimation may not work well as demonstrated. So, in order to adapt our AR-IoT system to more real applications, we have to research more on object recognition and localization in low-light conditions. In fact, the Dynamic Vision Sensor (DVS), which was used to reconstruct terrain surfaces when combined with a pulsed line laser [188], sheds light on how to approach this problem. This DVS sensing system shows the potential to estimate both depth information and topographical patterns of objects in the dark, which are helpful for object recognition and localization in the workplace of low light conditions.

Brain Signals for Human-Machine Interaction. In the proposed system, we mainly use hand gestures as the way to interact with target IoT devices in AR environment. Our system defines hand gesture commands, includes an additional hand gestures recognition module to interpret the user's intention, and performs desired actions accordingly. In this system, users' hand gestures are detected by using computer vision methods, so their hands have to be placed within the field of view of the head mounted camera. Also, the recognition performance depends on the light condition as well. These two factors limit the application scope of this hand gesture-based interface. Exploring solutions to this problem leads to another research question: Can the system read the user's intention directly without making any intermediate body movements? Actually, recent work has shown that brain signals can be exploited to interact with computer generated digital content. For example, steady state visually evoked potentials (SSVEP) for different brain frequency responses were used in a Virtual Reality (VR) system to navigate and select menu options [189]. In that work, the author only demonstrated binary decision making processes based on brain signals, while in AR-IoT applications, we need to conduct fine grained and continuous tasks, such as adjusting the brightness of a light bulb. This is a very challenging task, because the brain signal is 1D sequential data and very noisy, making it hard to capture useful information from it. Further research is needed, at both hardware and algorithm levels.

Bibliography

[1] Y. Sun, S. N. R. Kantareddy, R. Bhattacharyya, and S. E. Sarma, "X-vision: An augmented vision tool with real-time sensing ability in tagged environments," in *2018 IEEE International Conference on RFID Technology & Application (RFID-TA)*, pp. 1–6, IEEE, 2018.

[2] Y. Sun, A. Armengol-Urpi, S. N. R. Kantareddy, J. Siegel, and S. Sarma, "Magichand: Interact with iot devices in augmented reality environment," in *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 1738–1743, IEEE, 2019.

[3] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *arXiv preprint arXiv:1801.07829*, 2018.

[4] Y. Sun, S. N. R. Kantareddy, J. Siegel, A. Armengol-Urpi, X. Wu, H. Wang, and S. Sarma, "Towards industrial iot-ar systems using deep learning-based object pose estimation," in *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, pp. 1–8, IEEE, 2019.

[5] T. Mejtoft, "Internet of things and co-creation of value," in *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, pp. 672–677, IEEE, 2011.

[6] Y. Chen, "The walmart rfid initiative," *Mechanical and Aerospace Engineering Course*, vol. 188, pp. p1–19, 2008.

[7] "The history of iot: a comprehensive timeline of major events, infographic."

[8] R. Yang and M. W. Newman, "Learning from a learning thermostat: lessons for intelligent systems for the home," in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pp. 93–102, 2013.

[9] J. Ecko *et al.*, "Amazon echo: Amazon echo user guide," 2015.

[10] X. Fafoutis, L. Clare, N. Grabham, S. Beeby, B. Stark, R. Piechocki, and I. Craddock, "Energy neutral activity monitoring: Wearables powered by smart inductive charging surfaces," in *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–9, IEEE, 2016.

[11] Y. Ding, S. Gang, and J. Hong, "The design of home monitoring system by remote mobile medical," in *2015 7th International Conference on Information Technology in Medicine and Education (ITME)*, pp. 278–281, IEEE, 2015.

[12] F. Jimenez and R. Torres, "Building an iot-aware healthcare monitoring system," in *2015 34th International Conference of the Chilean Computer Science Society (SCCC)*, pp. 1–4, IEEE, 2015.

[13] H. Li, H. Wang, W. Yin, Y. Li, Y. Qian, and F. Hu, "Development of a remote monitoring system for henhouse environment based on iot technology," *Future Internet*, vol. 7, no. 3, pp. 329–341, 2015.

[14] L. Zhang, "An iot system for environmental monitoring and protecting with heterogeneous communication networks," in *2011 6th International ICST Conference on Communications and Networking in China (CHINACOM)*, pp. 1026–1031, IEEE, 2011.

[15] T. Qiu, H. Xiao, and P. Zhou, "Framework and case studies of intelligence monitoring platform in facility agriculture ecosystem," in *2013 Second International Conference on Agro-Geoinformatics (Agro-Geoinformatics)*, pp. 522–525, IEEE, 2013.

[16] S. Fang, L. Da Xu, Y. Zhu, J. Ahati, H. Pei, J. Yan, and Z. Liu, "An integrated system for regional environmental monitoring and management based on internet of things," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1596–1605, 2014.

[17] H. Lingling, L. Haifeng, X. Xu, and L. Jian, "An intelligent vehicle monitoring system based on internet of things," in *2011 Seventh International Conference on Computational Intelligence and Security*, pp. 231–233, IEEE, 2011.

[18] J. E. Siegel, *CloudThink and the Avacar: Embedded design to create virtual vehicles for cloud-based informatics, telematics, and infotainment*. PhD thesis, Massachusetts Institute of Technology, 2013.

[19] J. E. Siegel, S. Kumar, and S. E. Sarma, "The future internet of things: secure, efficient, and model-based," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2386–2398, 2017.

[20] Y.-B. Lin, Y.-W. Lin, C.-Y. Hsiao, and S.-Y. Wang, "Location-based iot applications on campus: The iottalk approach," *Pervasive and mobile computing*, vol. 40, pp. 660–673, 2017.

[21] L. Mainetti, V. Mighali, and L. Patrono, "An android multi-protocol application for heterogeneous building automation systems," in *2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pp. 121–127, IEEE, 2014.

[22] L. B. Rosenberg, "The use of virtual fixtures as perceptual overlays to enhance operator performance in remote environments.," tech. rep., Stanford Univ Ca Center for Design Research, 1992.

[23] O. Good, “Augmented reality language translation system and method,” Apr. 21 2011. US Patent App. 12/907,672.

[24] M. Singh and M. P. Singh, “Augmented reality interfaces,” *IEEE Internet Computing*, vol. 17, no. 6, pp. 66–70, 2013.

[25] G. A. Lee, A. Dünser, S. Kim, and M. Billinghurst, “Cityviewar: A mobile outdoor ar application for city visualization,” in *Mixed and Augmented Reality (ISMAR-AMH), 2012 IEEE International Symposium on*, pp. 57–64, IEEE, 2012.

[26] J. Paavilainen, H. Korhonen, K. Alha, J. Stenros, E. Koskinen, and F. Mayra, “The pokémon go experience: A location-based augmented reality mobile game goes mainstream,” in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 2493–2498, ACM, 2017.

[27] Y. Sun, S. N. R. Kantareddy, J. Siegel, A. Armengol-Urpi, X. Wu, H. Wang, and S. Sarma, “Towards industrial iot-ar systems using deep learning-based object pose estimation,” in *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, pp. 1–8, 2019.

[28] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2, pp. 1150–1157, Ieee, 1999.

[29] R. B. Rusu, N. Blodow, and M. Beetz, “Fast point feature histograms (fpfh) for 3d registration,” in *Proc. ICRA*, 2009.

[30] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, “Speeded up detection of squared fiducial markers,” *Image and Vision Computing*, vol. 76, pp. 38–47, 2018.

[31] V. M. J. Heun, *The reality editor: an open and universal tool for understanding and controlling the physical world*. PhD thesis, Massachusetts Institute of Technology, 2017.

[32] B. Thomas, B. Close, J. Donoghue, J. Squires, P. De Bondi, M. Morris, and W. Piekarski, “Arquake: An outdoor/indoor augmented reality first person application,” in *Digest of Papers. Fourth International Symposium on Wearable Computers*, pp. 139–146, IEEE, 2000.

[33] Z. Rashid, J. Melià-Seguí, R. Pous, and E. Peig, “Using augmented reality and internet of things to improve accessibility of people with motor disabilities in the context of smart cities,” *Future Generation Computer Systems*, vol. 76, pp. 248–261, 2017.

[34] H. Kato and M. Billinghurst, “Marker tracking and hmd calibration for a video-based augmented reality conferencing system,” in *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR’99)*, pp. 85–94, IEEE, 1999.

[35] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, “Recent advances in augmented reality,” *IEEE computer graphics and applications*, vol. 21, no. 6, pp. 34–47, 2001.

[36] M. Fiala, “Designing highly reliable fiducial markers,” *IEEE Transactions on Pattern analysis and machine intelligence*, vol. 32, no. 7, pp. 1317–1324, 2009.

[37] L. Karlinsky, J. Shtok, Y. Tzur, and A. Tzadok, “Fine-grained recognition of thousands of object categories with single-example training,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4113–4122, 2017.

[38] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *European conference on computer vision*, pp. 404–417, Springer, 2006.

[39] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, “Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes,” *arXiv preprint arXiv:1711.00199*, 2017.

[40] J. D. Stets, Y. Sun, W. Corning, and S. W. Greenwald, “Visualization and labeling of point clouds in virtual reality,” in *SIGGRAPH Asia 2017 Posters*, p. 31, ACM, 2017.

[41] M. Bajura, H. Fuchs, and R. Ohbuchi, “Merging virtual objects with the real world: Seeing ultrasound imagery within the patient,” *ACM SIGGRAPH Computer Graphics*, vol. 26, no. 2, pp. 203–210, 1992.

[42] S. Feiner, B. Macintyre, and D. Seligmann, “Knowledge-based augmented reality,” *Communications of the ACM*, vol. 36, no. 7, pp. 53–62, 1993.

[43] K. Kiyokawa, H. Takemura, and N. Yokoya, “A collaboration support technique by integrating a shared virtual reality and a shared augmented reality,” in *IEEE SMC’99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028)*, vol. 6, pp. 48–53, IEEE, 1999.

[44] D. Schmalstieg, A. Fuhrmann, and G. Hesina, “Bridging multiple user interface dimensions with augmented reality,” in *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*, pp. 20–29, IEEE, 2000.

[45] M. Anabuki, H. Kakuta, H. Yamamoto, and H. Tamura, “Welbo: An embodied conversational agent living in mixed reality space,” in *CHI’00 extended abstracts on Human factors in computing systems*, pp. 10–11, 2000.

[46] B. Aruanno, F. Garzotto, and M. C. Rodriguez, “Hololens-based mixed reality experiences for subjects with alzheimer’s disease,” in *Proceedings of the 12th Biannual Conference on Italian SIGCHI Chapter*, pp. 1–9, 2017.

[47] Z. Ren, J. Yuan, J. Meng, and Z. Zhang, “Robust part-based hand gesture recognition using kinect sensor,” *IEEE transactions on multimedia*, vol. 15, no. 5, pp. 1110–1120, 2013.

[48] H. K. Nishihara, S.-P. Hsu, A. Kaehler, and L. Jangaard, “Hand-gesture recognition method,” July 4 2017. US Patent 9,696,808.

[49] W. Lu, Z. Tong, and J. Chu, “Dynamic hand gesture recognition with leap motion controller,” *IEEE Signal Processing Letters*, vol. 23, no. 9, pp. 1188–1192, 2016.

[50] J. Garzón, J. Pavón, and S. Baldiris, “Augmented reality applications for education: Five directions for future research,” in *International Conference on Augmented Reality, Virtual Reality and Computer Graphics*, pp. 402–414, Springer, 2017.

[51] N. Chung, H. Han, and Y. Joun, “Tourists’ intention to visit a destination: The role of augmented reality (ar) application for a heritage site,” *Computers in Human Behavior*, vol. 50, pp. 588–599, 2015.

[52] J. Wang, H. Suenaga, K. Hoshi, L. Yang, E. Kobayashi, I. Sakuma, and H. Liao, “Augmented reality navigation with automatic marker-free image registration using 3-d image overlay for dental surgery,” *IEEE transactions on biomedical engineering*, vol. 61, no. 4, pp. 1295–1304, 2014.

[53] B. Marques, P. Dias, J. Alves, E. Fonseca, and B. S. Santos, “Configuration and use of pervasive augmented reality interfaces in a smart home context: A prototype,” in *International Conference on Human Systems Engineering and Design: Future Trends and Applications*, pp. 96–102, Springer, 2019.

[54] A. Badouch, S.-D. Krit, M. Kabrane, and K. Karimi, “Augmented reality services implemented within smart cities, based on an internet of things infrastructure, concepts and challenges: an overview,” in *Proceedings of the Fourth International Conference on Engineering & MIS 2018*, pp. 1–4, 2018.

[55] D. Jo and G. J. Kim, “Ariot: scalable augmented reality framework for interacting with internet of things appliances everywhere,” *IEEE Transactions on Consumer Electronics*, vol. 62, no. 3, pp. 334–340, 2016.

[56] T. Park, M. Zhang, and Y. Lee, “When mixed reality meets internet of things: Toward the realization of ubiquitous mixed reality,” *GetMobile: Mobile Computing and Communications*, vol. 22, no. 1, pp. 10–14, 2018.

[57] N. Norouzi, G. Bruder, B. Belna, S. Mutter, D. Turgut, and G. Welch, “A systematic review of the convergence of augmented reality, intelligent virtual agents, and the internet of things,” in *Artificial Intelligence in IoT*, pp. 1–24, Springer, 2019.

[58] C. G. Harris, M. Stephens, *et al.*, “A combined corner and edge detector.,” in *Alvey vision conference*, vol. 15, pp. 10–5244, Citeseer, 1988.

[59] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[60] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[61] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 886–893, IEEE, 2005.

[62] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[63] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[64] J. Han, D. Zhang, G. Cheng, N. Liu, and D. Xu, "Advanced deep-learning techniques for salient and category-specific object detection: a survey," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 84–100, 2018.

[65] A. G. Buch, D. Kraft, J.-K. Kamarainen, H. G. Petersen, and N. Krüger, "Pose estimation using local structure-specific shape and appearance context," in *2013 IEEE International Conference on Robotics and Automation*, pp. 2080–2087, IEEE, 2013.

[66] O. Van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or, "A survey on shape correspondence," *Computer Graphics Forum*, vol. 30, no. 6, pp. 1681–1707, 2011.

[67] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan, "3d object recognition in cluttered scenes with local surface features: a survey," *Trans. PAMI*, vol. 36, no. 11, pp. 2270–2287, 2014.

[68] S. Biasotti, A. Cerri, A. Bronstein, and M. Bronstein, "Recent trends, applications, and perspectives in 3d shape similarity assessment," *Computer Graphics Forum*, vol. 35, no. 6, pp. 87–119, 2016.

[69] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. Van Gool, "Hough transform and 3d surf for robust three dimensional classification," in *European Conference on Computer Vision*, pp. 589–602, Springer, 2010.

[70] S. Belongie, J. Malik, and J. Puzicha, "Shape context: A new descriptor for shape matching and object recognition," in *Proc. NIPS*, 2001.

[71] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *Trans. PAMI*, vol. 21, no. 5, pp. 433–449, 1999.

[72] S. Manay, D. Cremers, B.-W. Hong, A. J. Yezzi, and S. Soatto, "Integral invariants for shape matching," *Trans. PAMI*, vol. 28, no. 10, pp. 1602–1618, 2006.

[73] H. Ling and D. W. Jacobs, "Shape classification using the inner-distance," *Trans. PAMI*, vol. 29, no. 2, pp. 286–299, 2007.

[74] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *Proc. IROS*, 2008.

[75] F. Tombari, S. Salti, and L. Di Stefano, “A combined texture-shape descriptor for enhanced 3d feature matching,” in *Proc. ICIP*, 2011.

[76] R. M. Rustamov, “Laplace-beltrami eigenfunctions for deformation invariant shape representation,” in *Proc. SGP*, 2007.

[77] J. Sun, M. Ovsjanikov, and L. Guibas, “A concise and provably informative multi-scale signature based on heat diffusion,” *Computer Graphics Forum*, vol. 28, no. 5, pp. 1383–1392, 2009.

[78] M. Aubry, U. Schlickewei, and D. Cremers, “The wave kernel signature: A quantum mechanical approach to shape analysis,” in *Proc. ICCV Workshops*, 2011.

[79] M. M. Bronstein and I. Kokkinos, “Scale-invariant heat kernel signatures for non-rigid shape recognition,” in *Proc. CVPR*, 2010.

[80] S. A. A. Shah, M. Bennamoun, F. Boussaid, and A. A. El-Sallam, “3d-div: A novel local surface descriptor for feature matching and pairwise range image registration,” in *Proc. ICIP*, 2013.

[81] P. J. Besl and N. D. McKay, “Method for registration of 3-d shapes,” in *Sensor fusion IV: control paradigms and data structures*, vol. 1611, pp. 586–606, International Society for Optics and Photonics, 1992.

[82] M. Ferdik, G. Saxl, and T. Ussmueller, “Battery-less uhf rfid controlled transistor switch for internet of things applications—a feasibility study,” in *Wireless Sensors and Sensor Networks (WiSNet), 2018 IEEE Topical Conference on*, pp. 96–98, IEEE, 2018.

[83] A. Agrawal, G. J. Anderson, M. Shi, and R. Chierichetti, “Tangible play surface using passive rfid sensor array,” in *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, p. D101, ACM, 2018.

[84] O. Rashid, W. Bamford, P. Coulton, R. Edwards, and J. Scheible, “Pac-lan: mixed-reality gaming with rfid-enabled mobile phones,” *Computers in Entertainment (CIE)*, vol. 4, no. 4, p. 4, 2006.

[85] K.-C. Wu, C.-C. Chen, T.-H. Chiu, and I.-J. Chiang, “Transform children’s library into a mixed-reality learning environment: Using smartwatch navigation and information visualization interfaces,” in *Pacific Neighborhood Consortium Annual Conference and Joint Meetings (PNC), 2017*, pp. 1–8, IEEE, 2017.

[86] A. Ayala, G. Guerrero, J. Mateu, L. Casades, and X. Alamán, “Virtual touch flystick and primbox: two case studies of mixed reality for teaching geometry,” in *International Conference on Ubiquitous Computing and Ambient Intelligence*, pp. 309–320, Springer, 2015.

[87] M. Garon, P.-O. Boulet, J.-P. Doironz, L. Beaulieu, and J.-F. Lalonde, "Real-time high resolution 3d data on the hololens," in *Mixed and Augmented Reality (ISMAR-Adjunct), 2016 IEEE International Symposium on*, pp. 189–191, IEEE, 2016.

[88] J. E. Gentle, *Matrix transformations and factorizations*. Springer, 2007.

[89] R. Bhattacharyya, C. Floerkemeier, and S. Sarma, "Rfid tag antenna based sensing: Does your beverage glass need a refill?," in *2010 IEEE International Conference on RFID (IEEE RFID 2010)*, pp. 126–133, IEEE, 2010.

[90] R. Colella, B. Chiffi, N. Rusković, and L. Catarinucci, "Rfid sensing system based on uhf platform-tolerant antenna for harsh industrial environments," in *2019 4th International Conference on Smart and Sustainable Technologies (SpliTech)*, pp. 1–4, IEEE, 2019.

[91] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[92] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2921–2929, 2016.

[93] Y. Su, J. Rambach, N. Minaskan, P. Lesur, A. Pagani, and D. Stricker, "Deep multi-state object pose estimation for augmented reality assembly," in *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pp. 222–227, IEEE, 2019.

[94] D. Jo and G. J. Kim, "Ar enabled iot for a smart and interactive environment: A survey and future directions," *Sensors*, vol. 19, no. 19, p. 4330, 2019.

[95] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[96] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[97] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[98] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. ICRA*, 2017.

[99] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," *arXiv:1711.08488*, 2017.

[100] S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, and R. Urtasun, "Deep parametric continuous convolutional neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[101] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3d object detection," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[102] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3D Point Cloud Based Object Maps for Household Environments," *Robotics and Autonomous Systems Journal*, vol. 56, pp. 927–941, 30 November 2008.

[103] A. Golovinskiy, V. G. Kim, and T. Funkhouser, "Shape-based recognition of 3d point clouds in urban environments," in *Proc. ICCV*, 2009.

[104] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra, "PCPNet: Learning local shape properties from raw point clouds," *Computer Graphics Forum*, vol. 37, no. 2, pp. 75–85, 2018.

[105] M. Lu, Y. Guo, J. Zhang, Y. Ma, and Y. Lei, "Recognizing objects in 3d point clouds with multi-scale local features," *Sensors*, vol. 14, no. 12, pp. 24156–24173, 2014.

[106] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv:1512.03012*, 2015.

[107] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *Proc. IROS*, 2015.

[108] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proc. CVPR*, 2015.

[109] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. CVPR*, 2017.

[110] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[111] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012.

[112] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proc. CVPR*, 2015.

[113] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li, "Dense human body correspondences using convolutional networks," in *Proc. CVPR*, 2016.

[114] R. Klokov and V. Lempitsky, "Escape from cells: Deep kd-networks for the recognition of 3d point cloud models," 2017.

[115] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs," in *Proc. ICCV*, 2017.

[116] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. NIPS*, 2017.

[117] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Tran. Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[118] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," in *Proc. ICLR*, 2016.

[119] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," *arXiv:1704.01212*, 2017.

[120] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[121] F. Monti, M. M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Proc. NIPS*, 2017.

[122] F. Monti, K. Otness, and M. M. Bronstein, "MotifNet: A motif-based graph convolutional network for directed graphs," *arXiv:1802.01572*, 2018.

[123] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on riemannian manifolds," in *Proc. 3dRR*, 2015.

[124] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *Proc. NIPS*, 2016.

[125] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *arXiv:1710.10903*, 2017.

[126] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proc. CVPR*, 2017.

[127] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas, "Functional maps: A flexible representation of maps between shapes," *TOG*, vol. 31, no. 4, p. 30, 2012.

[128] A. Sinha, J. Bai, and K. Ramani, "Deep learning 3d shape surfaces using geometry images," in *Proc. ECCV*, 2016.

[129] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz, "SPLATNet: Sparse lattice networks for point cloud processing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2530–2539, 2018.

[130] H. Maron, M. Galun, N. Aigerman, M. Trope, N. Dym, E. Yumer, V. G. Kim, and Y. Lipman, "Convolutional neural networks on surfaces via seamless toric covers," in *Proc. SIGGRAPH*, 2017.

[131] M. Fey, J. E. Lenssen, F. Weichert, and H. Müller, "SplineCNN: Fast geometric deep learning with continuous B-spline kernels," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[132] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *arXiv:1312.6114*, 2013.

[133] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. NIPS*, 2014.

[134] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image.," in *Proc. CVPR*, 2017.

[135] C.-L. Li, M. Zaheer, Y. Zhang, B. Poczos, and R. Salakhutdinov, "Point cloud gan," *arXiv:1810.05795*, 2018.

[136] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *Proc. CVPR*, 2018.

[137] I. Kostrikov, Z. Jiang, D. Panozzo, D. Zorin, and J. Bruna, "Surface networks," in *Proc. CVPR*, 2017.

[138] O. Litany, A. Bronstein, M. Bronstein, and A. Makadia, "Deformable shape completion with graph convolutional autoencoders," *arXiv:1712.00268*, 2017.

[139] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black, "Generating 3D faces using convolutional mesh autoencoders," *arXiv:1807.10267*, 2018.

[140] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *Proc. CVPR*, 2016.

[141] A. Brock, T. Lim, J. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," in *Proc. NIPS*, 2016.

[142] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. CVPR*, 2017.

[143] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," in *Advances in Neural Information Processing Systems 31* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 820–830, Curran Associates, Inc., 2018.

[144] M. Atzmon, H. Maron, and Y. Lipman, "Point convolutional neural networks by extension operators," *ACM Transaction on Graphics (TOG)*, vol. 37, pp. 71:1–71:12, July 2018.

[145] L. Yi, V. G. Kim, D. Ceylan, I. Shen, M. Yan, H. Su, A. Lu, Q. Huang, A. Sheffer, L. Guibas, *et al.*, "A scalable active framework for region annotation in 3d shape collections," *TOG*, vol. 35, no. 6, p. 210, 2016.

[146] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Neighbors do help: Deeply exploiting local structures of point clouds," *arXiv:1712.06760*, 2017.

[147] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3d semantic parsing of large-scale indoor spaces," in *Proc. CVPR*, 2016.

[148] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, "Exploring spatial context for 3d semantic segmentation of point clouds," in *Proc. CVPR*, 2017.

[149] E. Marchand, H. Uchiyama, and F. Spindler, "Pose estimation for augmented reality: a hands-on survey," *IEEE transactions on visualization and computer graphics*, vol. 22, no. 12, pp. 2633–2651, 2015.

[150] R. Radkowski, "A point cloud-based method for object alignment verification for augmented reality applications," in *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. V01BT02A059–V01BT02A059, American Society of Mechanical Engineers, 2015.

[151] D. Mourtzis, E. Vlachou, V. Zogopoulos, and X. Fotini, "Integrated production and maintenance scheduling through machine monitoring and augmented reality: An industry 4.0 approach," in *IFIP International Conference on Advances in Production Management Systems*, pp. 354–362, Springer, 2017.

[152] E. Tzimas, G.-C. Vosniakos, and E. Matsas, "Machine tool setup instructions in the smart factory using augmented reality: a system construction perspective," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 13, no. 1, pp. 121–136, 2019.

[153] M. Gattullo, G. W. Scurati, M. Fiorentino, A. E. Uva, F. Ferrise, and M. Bordegoni, "Towards augmented reality manuals for industry 4.0: A methodology," *Robotics and Computer-Integrated Manufacturing*, vol. 56, pp. 276–286, 2019.

[154] C. Siew, S. Ong, and A. Nee, "A practical augmented reality-assisted maintenance system framework for adaptive user support," *Robotics and Computer-Integrated Manufacturing*, vol. 59, pp. 115–129, 2019.

[155] S. Tulsiani and J. Malik, "Viewpoints and keypoints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1510–1519, 2015.

[156] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Subcategory-aware convolutional neural networks for object proposals and detection," in *2017 IEEE winter conference on applications of computer vision (WACV)*, pp. 924–933, IEEE, 2017.

[157] S. Song and J. Xiao, "Sliding shapes for 3d object detection in depth images," in *European conference on computer vision*, pp. 634–651, Springer, 2014.

[158] S. Song and J. Xiao, "Deep sliding shapes for amodal 3d object detection in rgb-d images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 808–816, 2016.

[159] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652–660, 2017.

[160] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 918–927, 2018.

[161] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *2011 international conference on computer vision*, pp. 858–865, IEEE, 2011.

[162] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab, "Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation," in *European Conference on Computer Vision*, pp. 205–220, Springer, 2016.

[163] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3d bounding box estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 244–253, 2018.

[164] B. Tools, "Bantam tools™ desktop pcb milling machine specifications," *Othermill*, 2018.

[165] D. Ajilo, *eyeDNA: Tool Condition Monitoring for a desktop CNC milling machine*. PhD thesis, Massachusetts Institute of Technology, 2018.

[166] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1521–1529, 2017.

[167] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 699–715, 2018.

[168] V. Sandfort, K. Yan, P. J. Pickhardt, and R. M. Summers, "Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in ct segmentation tasks," *Scientific reports*, vol. 9, no. 1, pp. 1–9, 2019.

[169] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 113–123, 2019.

[170] O. H. Jafari, S. K. Mustikovela, K. Pertsch, E. Brachmann, and C. Rother, "The best of bothworlds: Learning geometry-based 6d object pose estimation," *arXiv preprint arXiv:1712.01924*, 2017.

[171] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4561–4570, 2019.

[172] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 292–301, 2018.

[173] M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3828–3836, 2017.

[174] J. Sock, S. Hamidreza Kasaei, L. Seabra Lopes, and T.-K. Kim, "Multi-view 6d object pose estimation and camera motion planning using rgbd images," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 2228–2235, 2017.

[175] H. Zhang and Q. Cao, "Combined holistic and local patches for recovering 6d object pose," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 2219–2227, 2017.

[176] D. S. Brezov, C. D. Mladenova, and I. M. Mladenov, "New perspective on the gimbal lock problem," in *AIP Conference Proceedings*, vol. 1570, pp. 367–374, American Institute of Physics, 2013.

[177] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6d object pose estimation by iterative dense fusion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3343–3352, 2019.

[178] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Asian conference on computer vision*, pp. 548–562, Springer, 2012.

[179] F. Hagelskjær and A. G. Buch, "Pointposenet: Accurate object detection and 6 dof pose estimation in point clouds," *arXiv preprint arXiv:1912.09057*, 2019.

[180] L. Ge, Z. Ren, and J. Yuan, "Point-to-point regression pointnet for 3d hand pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 475–491, 2018.

[181] Y. Wang and J. M. Solomon, "Prnet: Self-supervised learning for partial-to-partial registration," in *33rd Conference on Neural Information Processing Systems*.

[182] Y. Wang and J. M. Solomon, "Deep closest point: Learning representations for point cloud registration," in *The International Conference on Computer Vision*, 2019.

[183] D. Griffiths and J. Boehm, "Synthcity: A large scale synthetic point cloud," *arXiv preprint arXiv:1907.04758*, 2019.

[184] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[185] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[186] R. H. Creighton, *Unity 3D game development by example: A Seat-of-your-pants manual for building fun, groovy little games quickly*. Packt Publishing Ltd, 2010.

[187] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[188] C. Brandli, T. Mantel, M. Hutter, M. Höpflinger, R. Berner, R. Siegwart, and T. Delbruck, "Adaptive pulsed laser line extraction for terrain reconstruction using a dynamic vision sensor," *Frontiers in neuroscience*, vol. 7, p. 275, 2014.

[189] A. Armengol-Urpi and S. E. Sarma, "Sublime: a hands-free virtual reality menu navigation system using a high-frequency ssvep-based brain-computer interface," in *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, pp. 1–8, 2018.