MagicHand: A Deep Learning Approach towards Manipulating IoT Devices in Augmented Reality Environment

MagicHand: A Deep Learning Approach towards Manipulating IoT Devices in Augmented Reality Environment

Category: Application & Interaction

ABSTRACT

We present an Augmented Reality (AR) visualization and interaction tool for users to control Internet of Things (IoT) devices with hand gestures. Today, smart IoT devices are becoming increasingly ubiquitous with diverse forms and functions, yet most user controls over them are still limited to mobile devices and web interfaces. Recently, AR has been developed rapidly, and provided immersive solutions to enhance user experience of applications in many fields. Its capability to create immersive interactions allows AR to improve the way smart devices are controlled via more direct visual feedback. In this paper, we create a functional prototype of one such system, enabling seamless interactions with sound and lighting systems through the use of augmented hand-controlled interaction panels. To interpret users' intentions, we implement a standard 2D convolution neural network (CNN) for real-time hand gesture recognition and deploy it within our system. Our prototype is also equipped with a simple but effective object detector which can identify target devices within a proper range by analyzing visual and geometric features. We evaluate the performance of our system qualitatively and quantitatively and demonstrate it on two smart devices.

Index Terms: Augmented Reality—Visual and interactive control—IoT devices—Head mounted display; Deep learning—Image and 3D data processing—Object detection—Hand gesture recognition

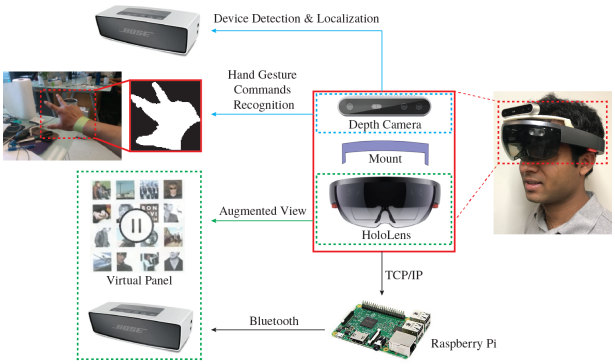


Figure 1: The proposed AR system for controlling IoT devices. The system first detects and localizes available target devices using color and depth information obtained from a depth camera mounted on the HoloLens. Once a smart device (e.g. speaker) is found, a virtual control panel is rendered nearby on the HoloLens screen. The system then detects possible hand gestures in view of the depth camera and infers the user's intentions before performing the corresponding action.

1 INTRODUCTION

The Internet of Things (IoT) market is growing rapidly, particularly in industry, with a McKinsey study projecting that the Industrial IoT (IIoT) may reach $ 11.1 trillion by 2025 [1] with up to 67 billion devices expected across industries [2]. Part of this scale-up includes real-time data acquisition for system control and optimization or real-time predictive applications. Many of these devices offer human interfaces such as manual control panels, mobile devices, computers, voice assistants, or cameras [3]. These interfaces are especially critical to human-in-the-loop systems, which necessitate seamless interaction to acquire inputs [4]. Conventional User Interfaces (UIs) like those found in smartphones simplify control for individual devices, but switching between mobile devices or applications becomes tedious and fails to scale well to a future with abundant, ambient smart devices [5]. The use of Augmented Reality (AR) presents one possible solution to this problem.

AR visualization and interaction tools overlay contextually-appropriate digital controls over real-world visuals. A common digital control is a control panel, for example, a set of on/off buttons with hue, brightness, and saturation sliders to control smart lighting. Such tools afford users with an immersive and intuitive control experience. 3D visualization is also possible in AR, which enables a higher degree of freedom than would be possible with traditional physical panels or the 2D visualization within standard displays. Voice is another emerging interaction, exemplified by the Amazon Echo or Google Home systems. However, voice commands are discrete interactions (e.g. "set room light to blue" or "set brightness to 50%"). Poor granularity, combined with a delay between utterance and action, creates a poor user experience. Instantaneous and continuous adjustments, like those possible with AR color wheels, improve system responsiveness and user satisfaction.

While AR interfaces solve these problems in theory, there are practical considerations. Gesture recognition is a persistent challenge due to background noise, occlusions, and view angle, for example [6–9]. We propose making gesture recognition more accurate and robust with the use of a 2D Convolutional Neural Network (CNN) utilizing both color and depth information. 2D CNNs already improve the accuracy of image-based tasks [10–13], and we can take advantage of them to enhance the user experience of modern Human-Computer Interaction (HCI). As part of this exploration, we examine the gesture recognition model's attention areas to understand its internal workings.

Before rendering a control, the target devices must be identified. Rather than using artificial marking like visually-conspicuous QR codes, we choose to identify IoT devices based on their geometric features. In our system, we introduce a simple but efficient object detection pipeline, which can jointly detect and localize target devices in 3D space.

This paper is organized as follows: In Section 2, we summarize previous works in interactions with IoT devices, AR experience, hand gesture recognition, and object detection. In Section 3, we describe our system in detail. In Section 4, we quantitatively evaluate the performance of the object detection and hand gesture recognition algorithms.

2 RELATED WORK

2.1 IoT Device Interactions

There has been a proliferation of IoT devices without commensurate improvements in ease-of-interaction. The use of dedicated apps for each device scales poorly, and dedicated hardware interfaces are costly to install and maintain.

There have been attempts at visual and non-visual interaction techniques for IoT device interaction, including vision-based wearables [15], hand-held devices with hand- and foot-gesture recognition [16], hands-free interaction with Google Glass [17], gustatory

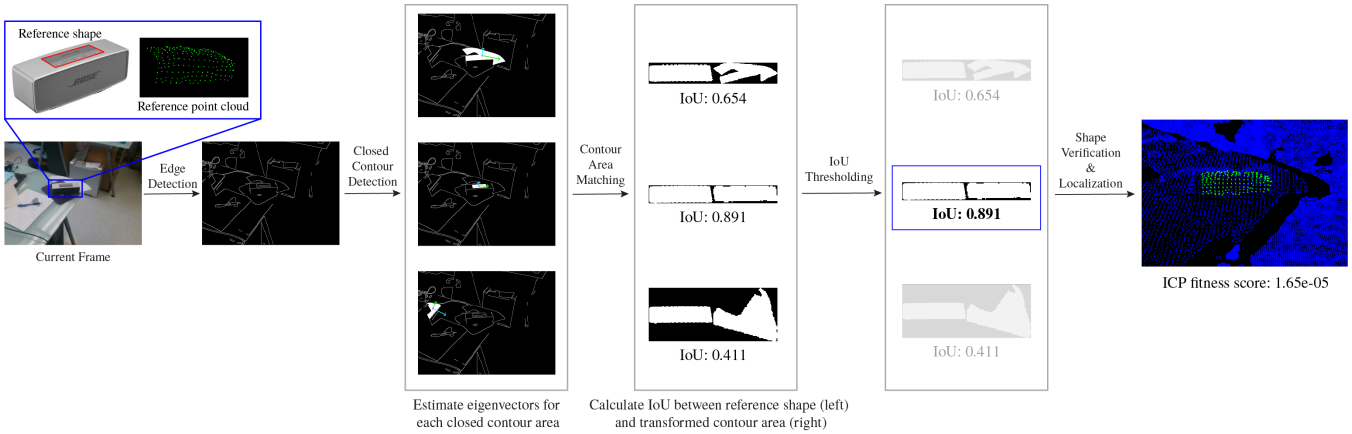
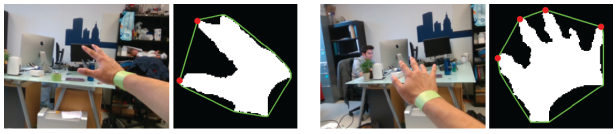


Figure 2: The pipeline for target device detection and localization. Given an input view, our system detects its edges and finds closed contours out of them. We then normalize each closed contour by aligning to the same direction and resizing to the same scale as the reference shape of a registered target device, followed by a shape matching calculation using Intersection-over-Union (IoU). The closed contours with IoU values greater than a predefined threshold are selected, and their corresponding 3D positions are estimated to initialize the reference point cloud for ICP matching. A target device is detected and localized from a closed contour that passes the ICP shape verification process.



(a) 3-finger recognition (b) 5-finger recognition

Figure 3: Failure examples of hand gesture recognition using convex hull method [14]. We show a color image (left) and the derived segmented hand mask (right) for each set. Detected fingertips are indicated with red dots, and the convex hull contours are bounded with green lines.

displays [18], and haptic devices [19], among other approaches.

2.2 AR Enhances User Experience

Theoretical AR research may aim to improve bidirectional interactivity between users and objects in their environment. A key challenge in AR is the design of accurate object detection [20] [21] and 3D pose estimation algorithms [22] [23].

AR application research may explore the application of existing computer vision algorithms to enhance the user's experience and improve immersion in areas including sensing [24], education [25], tourism [26] and navigation [27].

2.3 Hand Gesture Recognition

Hand gestures have been used as inputs to many HCI systems [28–32]. Recognizing hand gestures typically involves two procedures: hand detection and segmentation, and gesture recognition.

Hand Detection and Segmentation. This process segments the hand region from the background in order to simplify gesture recognition. Practically, the problem is complex when examining only color information from a monocular camera. Researchers have approached this problem from different angles, including shape [33], color [34], local [35] and context features [36]. These methods face limitations: the color method requires users to manually adjust skin color thresholds for different conditions (e.g. skin colors and light intensity); The shape- and local feature-based methods may fail when previously unseen gestures are tested. In our system, we resolve this problem by taking advantage of depth information obtained from depth cameras, which allows the hand region to be segmented from the background easily.

Gesture Recognition. Given a hand shape, traditional methods recognize the gesture by identifying the number of fingertips based on convexity defects or curvature [14, 37–39]. These methods require fingertips to form the vertices of a convex hull, and additional parameters, such as the length and intersection angle of a defect, need to be further adjusted to detect valid convexity defects. Even with appropriate settings, failure cases may be observed, as show in Figure 3. Accuracy can be improved by considering eccentricity [40] and elongatedness [41], though generic models that directly describe hand gestures are preferable. Recent deep learning models have achieved high accuracy in many image-related tasks [10–13], and it is feasible to adapt them in modern HCI systems [28, 42]. However, the working mechanism of these systems is unclear. To better understand where a deep neural network model attends in the hand gesture inference process, we adopt the *global average pooling* technique, which was originally proposed by [43] to localize discriminative image regions in image classification tasks.

2.4 Object Detection and Localization

Object Detection. In many AR applications, target devices or objects need to be detected to trigger subsequent actions. Compared to artificial marks used by many AR systems, such as QR codes, detecting objects based on their natural visual and geometric features creates more seamless user experience. Object detection has been studied for a long time in the computer vision community. Existing methods generally fall into two categories: traditional methods [44–47] which mainly impose handcrafted local features, and deep learning methods [48–50] which learn to capture visual patterns in a data-driven manner. Despite the accuracy achieved by deep learning models, a large and diverse training dataset is required to avoid overfitting during model training, making the technique ill-suited to smaller applications. In our system, we use traditional methods to handle a smaller set of target objects. Specifically, we adopt shape-based methods [51] instead of widely used visual feature-based methods [52] due to the textureless surface of many devices.

Object Localization in 3D Space. To create an intuitive user experience in AR, augmented digital information must be placed in proximity to detected target systems. This requires inference of the targets' 3D positions. It is complex to accurately estimate 3D

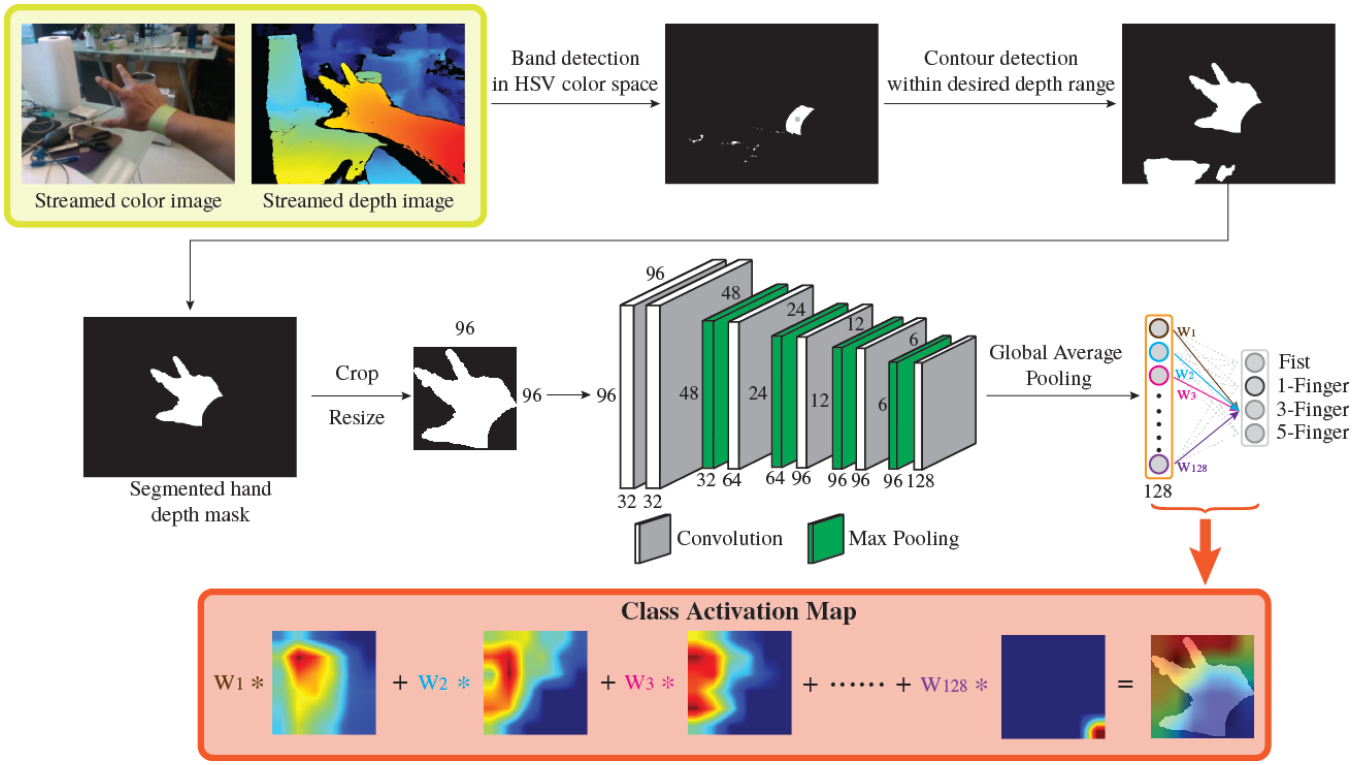


Figure 4: The pipeline for gesture recognition given a pair of streamed color and depth frames. The user wears a colored wristband as an indicator of hand region. The hand shape is detected in the depth image from the contour with sufficiently large area and spatially close to the wristband. The hand region is extracted and fed into a 2D CNN model for gesture recognition. We visualize the resultant activation map of the input gesture patch, which indicates the contribution of different pixels to the classification results (red: high, blue: low).

positions from a single-view image, with the best accuracy being achieved with the use of depth data, e.g. stereo image pairs [53, 54] or depth images [55, 56].

For this research, we choose to use the same depth sensor for hand segmentation and object localization.

Our work builds on these the concepts of gesture recognition, AR, and deep learning, and object detection and localization to create a robust and efficient gesture recognition system for use in IoT device (sound and lighting system) control.

3 System

In this section, we describe the proposed system in detail, including detecting and localizing target devices, recognizing hand gesture commands, and controlling IoT devices via virtual panels.

3.1 System Overview

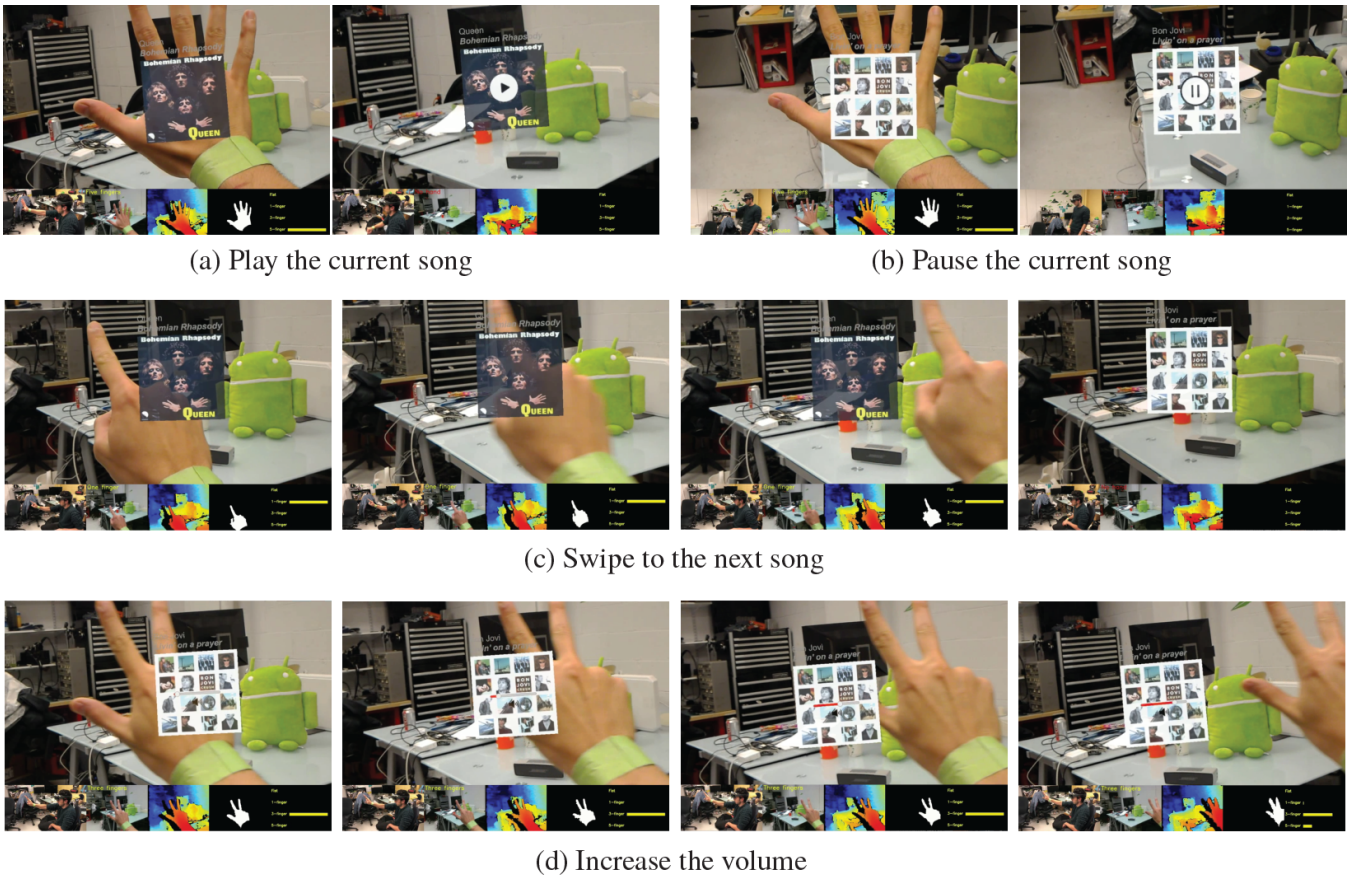
Figure 1 shows the pipeline of the proposed gesture control system for IoT devices. The system first identifies target devices in the view of a depth camera mounted on the HoloLens. Once a target device (e.g. speaker) is detected and localized, the system starts searching for hand shapes and recognizing their gestures. Conditioned on recognized gesture commands, the system performs predefined actions accordingly. Our system uses a Raspberry Pi as a central hub to wirelessly control connected devices.

3.2 IoT Device Detection and Localization

Figure 2 shows our system's pipeline for device detection and localization. This procedure verifies whether a target device exists in the current frame, and if so, localizes it in 3D space.

We achieve this by comparing the reference shape and point cloud of a target device with potential objects found in the current frame. Specifically, the reference shape is a 2D closed contour that characterizes the geometric features of an object. For example, we chose to use the top rectangular region of the speaker as the reference shape, because this region can be easily detected due to high contrast and its aspect ratio is scale-invariant (shown in Figure 2 left). The reference point cloud in our implementation is a set of points describing the 3D surface of an object from a given viewpoint.

Given the current frame, our system first detects its edges using Canny Edge Detector [51], followed by detecting connected closed contours. For each closed contour, we compute its similarity with the reference shape of a target object. Due to the viewpoint and distance change, each detected closed contour needs to be normalized to have the same orientation and scale as the reference shape for comparison. To normalize orientation, we computing the 2D eigenvectors of each closed contour using their interior pixel coordinates. Geometrically, the eigenvector with the larger (or smaller) eigenvalue indicates the direction of the larger (or smaller) variance of the pixel coordinates. By aligning the two eigenvectors of closed contours and the reference shape, we can normalize orientation. To normalize scale, we resize rotated contours to have the same width as the reference shape without changing their aspect ratios. After normalizing orientation and scale, if two shapes match, they should share a relatively large overlapped area. To measure this, we use Intersection over Union (IoU) as the metric, which is widely used to evaluate object detection accuracy [57]. The IoU is simply calculated by dividing the area of overlap between two shapes by the area of union of those two shapes. Similar shapes usually give a higher IoU value. During our implementation, we found that a threshold of 0.8 effectively filters



(a) Play the current song

(b) Pause the current song

(c) Swipe to the next song

(d) Increase the volume

Figure 5: 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, we show 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.

out false matches.

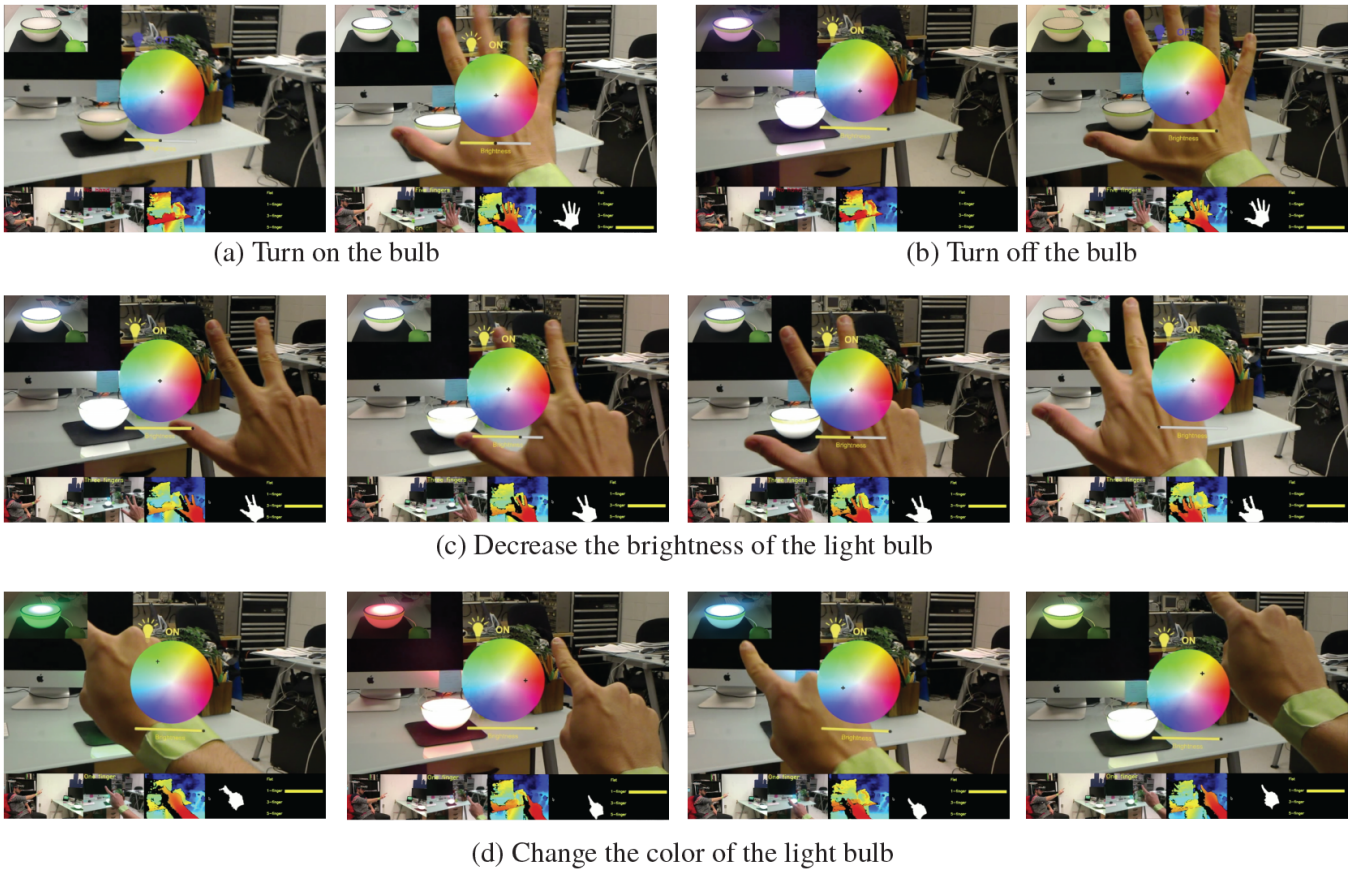
However, when the background contains other objects with contours of similar aspect ratio as the reference shape, false positives may still occur. To reduce the false positive rate, we add additional 3D shape verification using point clouds for all contours that pass IoU threshold in previous step. The point cloud can be obtained from modern depth sensors (e.g. the Intel RealSense used in our system). We adopt Iterative Closest Point (ICP) method [58] to verify the 3D shape matching between detected potential objects and target objects. ICP usually requires a good initial position to converge to the global optimal. In our system, we obtain the initial position using the 3D coordinate of the centroid of the matched closed contour in previous step, which can be computed given available camera intrinsic parameters. The reference point cloud is then translated to the initial position to perform ICP matching, and a fitness score is returned indicating the average inter-point distance between two point sets. We calculate ICP fitness scores for all the retained shape contours, and compare them with a predefined ICP threshold. A detection is reported when the ICP fitness score is less than this threshold. One benefit of this method is that the 3D position of the detected object can be directly obtained from ICP matching results, thus no additional localization is needed.

3.3 Hand Gesture Command Recognition

3.3.1 Hand Gesture Recognition.

The hand gesture recognition process involves two steps: hand region segmentation and gesture recognition. An easy way to detect hand regions is to extract skin pixels by thresholding the value of hue, saturation and value in HSV color space, which better aligns with the way human vision perceives colors than RGB color space. However, this method requires thresholds to be carefully adjusted based on users' skin colors. In our work, we simplify the hand region detection problem by using a colored wristband as an indicator of the hand region in the image. The hand region can be found in the depth image as the contour whose values fall into a depth range starting from the wristband and pointing outwards. Our system uses a depth range of 20cm, covering the length of a regular hand. When multiple contours are detected, the hand contour is selected as the one closest to the wristband (shown in Figure 4 top). One significant advantage of detecting hand region from depth images is that the hand is segmented automatically during the depth image generation process unless the hand touches other objects, which is not a typical case when a user performs gesture commands. Also, hand shapes are not affected by users' skin colors in depth images, eliminating the need for manual threshold adjustment.

To recognize the segmented hand contours, we implement a 2D CNN model to hierarchically extract high-level features for determining gesture types. The CNN model takes as input a cropped hand



(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 6: 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.

shape, and outputs a probability distribution among 4 gestures: fist, 1-finger, 3-finger and 5-finger. When constructing the CNN gesture recognition model, we aim to interpret how the model works during its inference process, since this remains unclear in many current deep learning-based HCI systems. To this end, we adopt the global average pooling (GAP) operation as the last layer to aggregate previously extracted high-level visual information and assign probability to different labels. As claimed in [43], GAP provides an unsupervised way to visualize salient regions for each predicted label. This procedure is demonstrated on Figure 4 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.

3.3.2 Hand Gesture Commands

The hand gesture is recognized within each frame, and the results for a sequence of frames determine a user's commands. In our system, we consider 4 types of user commands: binary control, continuous control, binary status switch and termination, as described below.

Binary Control Command. The binary control command gives two outcomes, for example, playing either the next or previous song when controlling a speaker. Our system decides this command using the starting and ending positions of a constant hand gesture (e.g. 1-finger gesture on Figure 7 top left).

Binary Status Switch Command. Switching between two binary states is an important function for many devices, such as turning on/off and playing/pausing. This is different from binary control command: two consecutive binary control commands can trigger the same action, such as "(swipe left, swipe left)", while two consecutive binary status switch commands trigger different actions, such as "(on, off)". Our system detects the binary status switch command by constantly detecting the same hand gesture (e.g. 5-finger gesture on Figure 7 top right) within a set of continuous frames.

Continuous Control Command. Compared to binary control command, continuous control command can gradually change the status of a device, like the volume of a speaker. It is implemented in a similar way as binary control command, except that the positions of all hand gestures are recorded (e.g. 3-finger gesture on Figure 7 bottom left) instead of just the starting and ending gestures for binary control command.

Termination Command. In our system, the termination command (defined as fist gesture shown on Figure 7 bottom right) does not perform any action, it simply indicates the end of other commands. For example, when a user performs binary control command, the ending position of a hand gesture can be obtained from the frame right before the termination command. Since the termination command does not trigger any action, it also allows a user to move his or her hand freely in the view without unintentional task activation.

3.4 IoT Device Control

Our system works with two devices: a Bose speaker and a Philips Hue Go light bulb, allowing a user to interact with each via gestures.

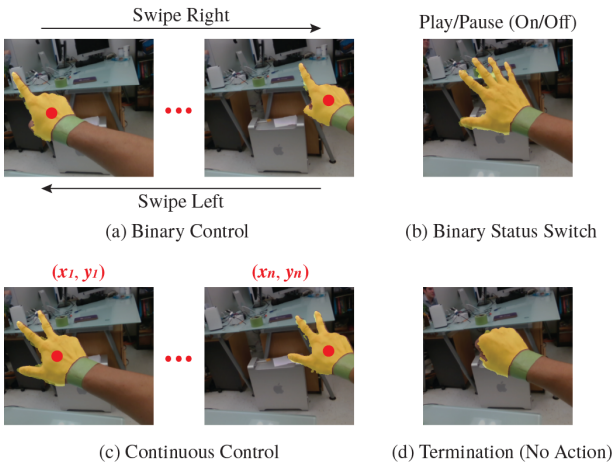


Figure 7: Hand gesture commands.

Each device is associated with its own holographic panel providing visual feedback to the user. The holographic panel is placed close to the detected device and is oriented towards the user in real-time, so that when users walks around the room, the virtual panel can always face towards them. Our system uses a Raspberry Pi as a server hub to receive data from HoloLens and to send the corresponding commands to connected devices. The device control pipeline is described below:

1. When the program starts, HoloLens waits to receive information about the detected device and its location in the real world via TCP/IP from the object recognition subsystem. After receiving successful detection results, HoloLens initializes the corresponding device holographic panel and place it at a location close to the detected device.

2. HoloLens then starts receiving hand gesture commands via TCP/IP from the hand gesture recognition subsystem, and checks if the received command is legitimate to be carried out (e.g. the speaker volume is within a predefined range or color pointer position is inside the color wheel). If so, HoloLens sends commands to the Raspberry Pi via TCP/IP.

3. Raspberry Pi sends received commands to the active device via Bluetooth if connecting to the Bose speaker or TCP/IP if connecting to the Philips bulb.

For the speaker, our system allows users to play and pause the current song via binary status switch command using 5-finger gesture, change songs via binary control command by swiping left or right to previous or next song, respectively, using 1-finger gesture, and adjust the volume via continuous control command using 3-finger gesture (demonstrated in Figure 5). For the light bulb, users are allowed to turn the bulb on and off via binary status switch command using 5-finger gesture, adjust brightness and select color via continuous control command using 3-finger and 1-finger gestures, respectively (demonstrated in Figure 6).

4 Evaluations

In this section, we quantitatively and qualitatively evaluate two system components (device detection and localization, and hand gesture recognition) and the system's overall performance.

4.1 Detection and Localization

We first evaluate the proposed pipeline for device detection and localization on the speaker and light bulb which are used in our application. Here, we are interested in seeing how this pipeline performs for different distances between the device and the user, which suggests a proper working range for our system. We test the range changing from 0.4m to 1.2m with an interval of 0.1m. For each range, we collect 10 color and depth image pairs containing a target device, run our pipeline, and count the ratio of successful detection and localization as the success rate, which is reported for different ranges in Figure 8. As can be observed, the success rate decrease as the range increases, and drops below 0.6 when the range is larger than 1.2m. This results from two aspects: for longer ranges, the occupied region of a target device in the color image reduces, and the depth data also becomes more noisy. The success rate of the bulb is also lower than that of the speaker due to its reflective surface. Despite the success rate of detection and localization drops to 0.6 for the range of 1.2m, our pipleline can still successfully detect and localize the target device from 2 continuous frames on average. Actually, a few more frames may be needed when our system runs depending on the viewpoint, and each frame takes about 410ms to be processed in our C++ implementation. Once the device is successfully detected and localized, this pipeline will terminate.

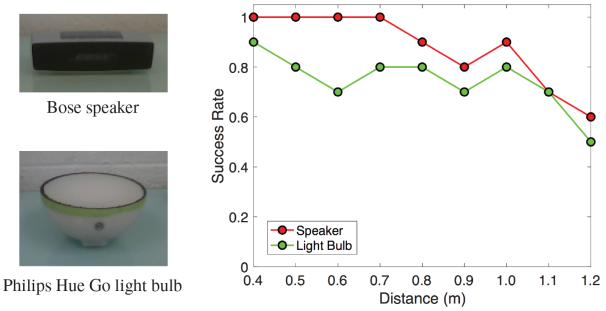


Figure 8: The success rate of detection and localization for different ranges between the depth camera and target devices.

4.2 Hand Gesture Recognition

This subsection summarize details about training and evaluating our 2D CNN hand gesture recognition model. Deep learning models learn visual patterns in a data-driven manner. For our task, we collect a database containing 4,690 hand shapes covering 4 different categories. To avoid the tedious manual labeling process for this supervised learning task, we individually record a live stream for each gesture type, and obtain image samples for that gesture by processing all the frames using our hand gesture recognition subsystem before 2D CNN model. Randomly selected hand gesture examples are shown in Figure 9. Note that the shape and direction of our collected samples vary diversely, including shapes that can hardly be correctly recognized using convex hull-based method (e.g. the palm in the fourth row, second column). We split the samples of each gesture category into training/testing sets following a ratio of 0.9/0.1. We used TensorFlow Python API to construct and train the 2D CNN model. After training, we export the model graph and its trained model parameters into our C++ program to conduct real-time hand gesture inference. Our 2D CNN model runs on a MacBook Pro with 2.6 GHz Intel Core i7 prosessor using CPU, and takes about 30 ms on average to finish one forward inference pass.

To quantitatively evaluate the 2D CNN hand gesture model, we compare it with a Support Vector Machine (SVM) linear classification model. We downsample each hand shape image into a patch

Figure 9: The collected hand shapes for training CNN gesture recognition model. From top row to bottom row: fist, 1-finger, 3-finger and 5-finger.

of size 7×7, concatenate all the resultant pixels in row-major order into an 1D vector, and then feed it into the SVM model. Both models are trained on the same training set. For our collected dataset, the 2D CNN model achieves better testing recognition accuracy than the SVM model, as shown in Table 1. We also visualize the activation maps of different samples in Figure 10, where we can observe that the model focuses on different regions for different gesture types, and those discriminative regions help the model classify hand gestures correctly.

Table 1: Gesture classification accuracy comparison between SVM and the implemented CNN model on testing set. The number of training (left) and testing (right) samples are provided under each gesture.

Gesture **# train/test**	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

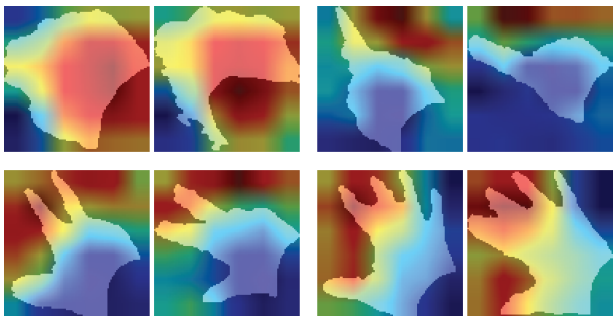


Figure 10: The activation maps of different hand shapes returned by the 2D CNN hand gesture recognition model.

4.3 User Study

To evaluate the performance of our system, we conducted a user study w.r.t. two aspects, speed and degree of satisfaction, between our system and commercially available options such as smartphone apps for the Bose speaker and the Philips Hue Go light bulb. To evaluate the speed, we record the time period from participants opening the application till they connect to the devices and perform their first actions. To evaluate the degree of satisfaction, we let users rate on a scale from 1 to 10 with an interval of 1 about their degrees of satisfaction for both methods. 10 users without computer science background participated and rated the usage experience. We observe that, compared to the light bulb, the operation time for speaker is significantly improved in our system, because most smartphones require users to open Bluetooth and search for the speaker from a list of all the detected devices, which usually takes more than 18 seconds. On the other hand, controlling light bulb using our system obtains a relatively high degree of satisfaction, since users can get more direct visual feedback.

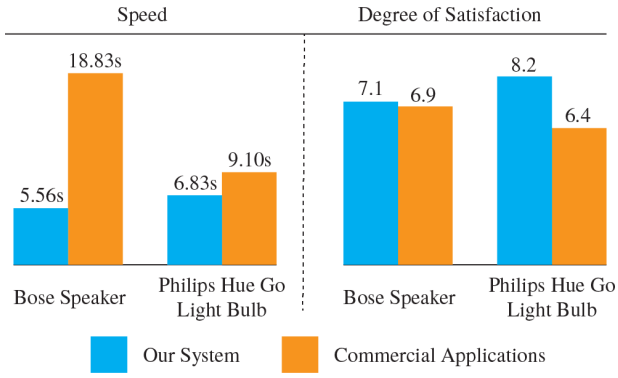


Figure 11: User study results on speed and degree of satisfaction for our system and available commercial applications.

5 Conclusion

The future will bring about an increased number of smart devices in our daily lives. Many of these devices will require human interaction or operate as humans in-the-loop systems. Having to install apps on smartphones and tablets to interact and control these devices is not scalable as device diversity grows. In this paper, we propose MagicHand, an AR-based visualization and interaction tool with deep learning approach, to solve this problem by providing a touchless and robust technique to interface with smart devices. We built an information flow pipeline for the system consisting of a depth camera, Raspberry Pi and HoloLens, allowing the users to interact and control sound and lighting systems with gestures. We built an object detection model, which can jointly detect and localize target devices in the view. We built a 2D CNN model, which is robust enough to classify hand gestures with noisy backgrounds using both the depth and color information from a depth camera. Our quantitative evaluation results show that our system can accurately detect and localize target devices within a proper working range, and achieves high hand gesture recognition accuracy for creating fluent user experience. A user study comparing experiences between our systems and commercially available applications shows increased interaction speed and satisfaction level. This automatic object detection and virtual panel based interaction is scalable to meet the future interface requirements. We plan to extend tool's capabilities to interact with more complex systems such as industrial manipulators and consumer robots.

References

[1] J. Manyika. The internet of things: Mapping the value beyond the hype. *McKinsey Global Institute*, 1(0):144, 2015.

[2] IHS Markit. The internet of things: a movement , not a market. *Critical IoT Insights*, pages 1–9, 2017.

[3] Dominic Gorecky, Mathias Schmitt, Matthias Loskyll, and Detlef Zühlke. Human-machine-interaction in the industry 4.0 era. In *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on*, pages 289–294. Ieee, 2014.

[4] John A Stankovic. Research directions for the internet of things. *IEEE Internet of Things Journal*, 1(1):3–9, 2014.

[5] Roy Want, Bill N Schilit, and Scott Jenson. Enabling the internet of things. *Computer*, 48(1):28–35, 2015.

[6] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[7] Jingen Liu, Jiebo Luo, and Mubarak Shah. Recognizing realistic actions from videos in the wild. In *Computer vision and pattern recognition, 2009. CVPR 2009. IEEE conference on*, pages 1996–2003. IEEE, 2009.

[8] Olivier Duchenne, Ivan Laptev, Josef Sivic, Francis Bach, and Jean Ponce. Automatic annotation of human actions in video. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1491–1498. IEEE, 2009.

[9] K Martin Sagayam and D Jude Hemanth. Hand posture and gesture recognition techniques for virtual reality applications: a survey. *Virtual Reality*, 21(2):91–107, 2017.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017.

[14] Srinivas Ganapathyraju. Hand gesture recognition using convexity hull defects to control an industrial robot. In *Instrumentation Control and Automation (ICA), 2013 3rd International Conference on*, pages 63–67. IEEE, 2013.

[15] Zhihan Lv, Shengzhong Feng, Liangbing Feng, and Haibo Li. Extending touch-less interaction on vision based wearable device. In *Virtual Reality (VR), 2015 iEEE*, pages 231–232. IEEE, 2015.

[16] Zhihan Lu, Muhammad Sikandar Lal Khan, and Shafiq Ur Réhman. Hand and foot gesture interaction for handheld devices. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 621–624. ACM, 2013.

[17] Zhihan Lv, Liangbing Feng, Haibo Li, and Shengzhong Feng. Hand-free motion interaction on google glass. In *SIGGRAPH Asia 2014 Mobile Graphics and Interactive Applications*, page 21. ACM, 2014.

[18] Takuji Narumi, Shinya Nishizaka, Takashi Kajinami, Tomohiro Tanikawa, and Michitaka Hirose. Augmented reality flavors: gustatory display based on edible marker and cross-modal interaction. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 93–102. ACM, 2011.

[19] Maurizio Maisto, Claudio Pacchierotti, Francesco Chinello, Gionata Salvietti, Alessandro De Luca, and Domenico Prattichizzo. Evaluation of wearable haptic systems for the fingers in augmented reality applications. *IEEE transactions on haptics*, 10(4):511–522, 2017.

[20] Leonid Karlinsky, Joseph Shtok, Yochay Tzur, and Asaf 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*, pages 4113–4122, 2017.

[21] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[22] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[23] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[24] Yongbin Sun, Sai Nithin R Kantareddy, Rahul Bhattacharyya, and Sanjay E Sarm. X-vision: An augmented vision tool with real-time sensing ability in tagged environments. *arXiv preprint arXiv:1806.00567*, 2018.

[25] Juan Garzón, Juan Pavón, and Silvia Baldiris. Augmented reality applications for education: Five directions for future research. In *International Conference on Augmented Reality, Virtual Reality and Computer Graphics*, pages 402–414. Springer, 2017.

[26] Namho Chung, Heejeong Han, and Youhee Joun. Tourists intention to visit a destination: The role of augmented reality (ar) application for a heritage site. *Computers in Human Behavior*, 50:588–599, 2015.

[27] Junchen Wang, Hideyuki Suenaga, Kazuto Hoshi, Liangjing Yang, Etsuko Kobayashi, Ichiro Sakuma, and Hongen Liao. Augmented reality navigation with automatic marker-free image registration using 3-d image overlay for dental surgery. *IEEE transactions on biomedical engineering*, 61(4):1295–1304, 2014.

[28] Pei Xu. A real-time hand gesture recognition and human-computer interaction system. *arXiv preprint arXiv:1704.07296*, 2017.

[29] David J Rios-Soria, Satu E Schaeffer, and Sara E Garza-Villarreal. Hand-gesture recognition using computer-vision techniques. 2013.

[30] Hui Liang, Junsong Yuan, Daniel Thalmann, and Nadia Magnenat Thalmann. Ar in hand: Egocentric palm pose tracking and gesture recognition for augmented reality applications. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 743–744. ACM, 2015.

[31] Siddharth S Rautaray and Anupam Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, 43(1):1–54, 2015.

[32] Zhihan Lv, Alaa Halawani, Shengzhong Feng, Shafiq Ur Réhman, and Haibo Li. Touch-less interactive augmented reality game on vision-based wearable device. *Personal and Ubiquitous Computing*, 19(3-4):551–567, 2015.

[33] Eng-Jon Ong and Richard Bowden. A boosted classifier tree for hand shape detection. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 889–894. IEEE, 2004.

[34] S Kolkur, D Kalbande, P Shimpi, C Bapat, and J Jatakia. Human skin detection using rgb, hsv and ycbcr color models. *arXiv preprint arXiv:1708.02694*, 2017.

[35] Qing Chen, Nicolas D Georganas, Emil M Petriu, et al. Real-time vision-based hand gesture recognition using haar-like features. In *Instrumentation and Measurement Technology Conference Proceedings*, pages 1–6. Citeseer, 2007.

[36] Patrick Buehler, Mark Everingham, Daniel P Huttenlocher, and Andrew Zisserman. Long term arm and hand tracking for continuous sign language tv broadcasts. In *Proceedings of the 19th British Machine Vision Conference*, pages 1105–1114. BMVA Press, 2008.

[37] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

[38] Ronald L Graham and F Frances Yao. Finding the convex hull of a simple polygon. *Journal of Algorithms*, 4(4):324–331, 1983.

[39] Yi Li. Hand gesture recognition using kinect. In *Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on*, pages 196–199. IEEE, 2012.

[40] Vishal Bhame, R Sreemathy, and Hrushikesh Dhumal. Vision based hand gesture recognition using eccentric approach for human computer interaction. In *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*, pages 949–953. IEEE, 2014.

[41] Md Mohiminul Islam, Sarah Siddiqua, and Jawata Afnan. Real time hand gesture recognition using different algorithms based on american sign language. In *Imaging, Vision & Pattern Recognition (icIVPR), 2017 IEEE International Conference on*, pages 1–6. IEEE, 2017.

[42] Oyebade K Oyedotun and Adnan Khashman. Deep learning in vision-based static hand gesture recognition. *Neural Computing and Applications*, 28(12):3941–3951, 2017.

[43] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.

[44] Pauline C Ng and Steven Henikoff. Sift: Predicting amino acid changes that affect protein function. *Nucleic acids research*, 31(13):3812–3814, 2003.

[45] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011*

IEEE international conference on, pages 2564–2571. IEEE, 2011.

[46] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[47] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[48] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[49] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

[50] Junwei Han, Dingwen Zhang, Gong Cheng, Nian Liu, and Dong Xu. Advanced deep-learning techniques for salient and category-specific object detection: a survey. *IEEE Signal Processing Magazine*, 35(1):84–100, 2018.

[51] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[52] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[53] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[54] Hae-Gon Jeon, Jaesik Park, Gyeongmin Choe, Jinsun Park, Yunsu Bok, Yu-Wing Tai, and In So Kweon. Accurate depth map estimation from a lenslet light field camera. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1547–1555, 2015.

[55] Teng Wang, Leping Bu, and Zhongyi Huang. A new method for obstacle detection based on kinect depth image. In *Chinese Automation Congress (CAC), 2015*, pages 537–541. IEEE, 2015.

[56] Mostafa Karbasi, Zeeshan Bhatti, Parham Nooralishahi, Asadullah Shah, and Seyed Mohammad Reza Mazloomnezhad. Real-time hands detection in depth image by using distance with kinect camera. *International Journal of Internet of Things*, 4(1A):1–6, 2015.

[57] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[58] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992.