Ubii: Physical World Interaction Through Augmented Reality

Sikun Lin, Hao Fei Cheng, Weikai Li, Zhanpeng Huang, Pan Hui, *Senior Member, IEEE*, and Christoph Peylo

Abstract—We describe a new set of interaction techniques that allow users to interact with physical objects through augmented reality (AR). Previously, to operate a smart device, physical touch is generally needed and a graphical interface is normally involved. These become limitations and prevent the user from operating a device out of reach or operating multiple devices at once. Ubii (**Ub**iquitous **i**nterface and **i**nteraction) is an integrated interface system that connects a network of smart devices together, and allows users to interact with the physical objects using hand gestures. The user wears a smart glass which displays the user interface in an augmented reality view. Hand gestures are captured by the smart glass, and upon recognizing the right gesture input, Ubii will communicate with the connected smart devices to complete the designated operations. Ubii supports common inter-device operations such as file transfer, printing, projecting, as well as device pairing. To improve the overall performance of the system, we implement computation offloading to perform the image processing computation. Our user test shows that Ubii is easy to use and more intuitive than traditional user interfaces. Ubii shortens the operation time on various tasks involving operating physical devices. The novel interaction paradigm attains a seamless interaction between the physical and digital worlds.

Index Terms—Augmented reality, wearable computers, user interfaces, human computer interaction, mobile computing

1 Introduction

In the past 30 years, we have relied prion screen-based Graphical User Interface (GUI) to interact with digital world. No matter whether the screen is placed on a desk (e.g., PC screen), held on a hand (e.g., mobile phone screen), or displayed on a wall (e.g., projection screen), it has cultivated a predominantly visual paradigm of human-computer interaction [1]. However, traditional centralized GUI displays all information on a screen and is separated from the physical world. This requires the user to interact with the device physically. The separate-screen GUI also prevents the user from interacting with multiple devices at once or operating any device which the user cannot reach physically.

There have been constant efforts to develop direct interactions between the physical and digital worlds, especially on developing interfaces which allow users to interact with physical objects directly. Architectural partitions [2], physical props [3], mobile devices [4], and human bodies [5] are used to enable tangible interaction at a distance. However, having extra physical objects as input devices obtrude a seamless user interaction. Particularly, conventional user interfaces cannot leverage on the physical affordance [6], [7] of underlying objects. Users have to memorize manipulations like traditional GUI interaction.

As a new interface straddling the real and the digital, AR bridges the gap between physical and virtual worlds [8]. It supports a sense of shared presence and communal social space. AR has been widely used to help users access additional information for a task without getting distracted [9], [10]. Another baseline leverages AR to provide 3D menu interactions in current view of the real world [11], [12]. The work generate egocentric virtual user interfaces in front of the current view, allowing interactions with physical objects without breaking the traditional centralized paradigm.

To develop a natural user interface (NUI) that would not obtrude the user experience and is natural to learn, we present Ubii [13]—a ubiquitous AR-based system which pairs tailored interfaces to the physical affordance of objects. Ubii enables users to directly interact with physical objects at a distance, including objects which would otherwise be unreachable (e.g., objects behind a glass window) [4]. As illustrated in Fig. 1, users are able to manipulate physical objects including computers, projector screens, printers, and architecture partitions to complete tasks such as document copying, printing, sharing, and projection display. Individual AR menu overlays are aligned with physical objects to present their physical affordance. Developed on Google Glass, Ubii enables gesture-based freehand interaction with greater convenience. In this paper we made major improvements to the previous Ubii system and implemented a new SVM-based detection algorithm and computation offloading to achieve better performances. We also conducted a new user evaluation to give a more objective and comprehensive result.

- *S. Lin, H.F. Cheng, W. Li, Z. Huang, and P. Hui are with the HKUST-DT System and Media Laboratory, Hong Kong University of Science and Technology, Hong Kong, China.*
  E-mail: {sklin, hfchengab, weikaili, soaroc, panhui}@ust.hk.
- *C. Peylo is with the Deutsche Telekom AG Laboratories, Ernst-Reuter-Platz 7, Berlin 10587, Germany. E-mail: christoph.peylo@telekom.de.*

Manuscript received 25 Nov. 2015; revised 28 Apr. 2016; accepted 29 Apr. 2016. Date of publication 13 May 2016; date of current version 2 Feb. 2017.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2016.2567378

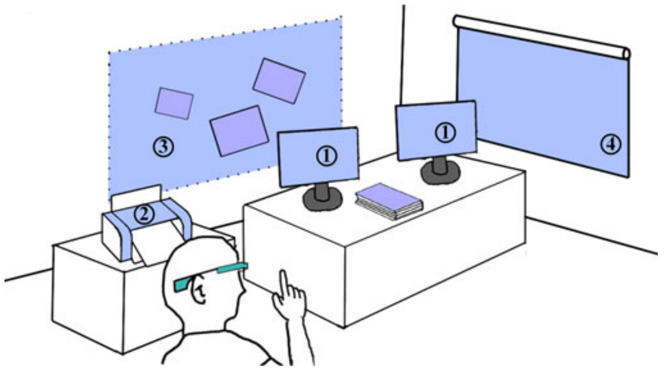


Fig. 1. Ubii enables users to perform in-situ interactions with physical objects using bare hands. Traditional operations such as file sharing, printing, or device pairing can be done just with a pair of smart glass and some defined hand gestures.

2 Related Work

Ubii is built on several technologies including interaction at a distance, freehand interaction, and AR menu.

2.1 Interaction at a Distance

Early systems such as PointRight [14] and Perspective Cursor [15] allow distant pointer motion across independently-driven display surfaces that are not organized in a single plane. The systems require indirect local input and complicated set-up. Other work [16], [17] overcomes the problems using absolute pointing techniques, but requires additional pointing devices. Motion errors are amplified as distances increase.

Mobile devices have been widely used as physical interfaces to provide foundation of the new interaction paradigm [18]. The relative pointing on a mobile device screen is transferred to a distant screen to reduce the interaction distance. The Point & Shoot [19] technique employs visual codes to set up an absolute coordinate system for objects selection with mobile phones. To create a self-contained system without fiducial markers, Boring et al. preformed content arrangement to determine the spatial relationship between the mobile device and display screen [20]. Chang and Li [21] allowed arbitrary content arrangement by matching the captured images with contents on the display screens. The method does not support continuous interaction as content snapshots are required in advance for further analysis.

Some work introduces interactions with physical objects by operating mobile devices. The devices capture the environment, and display the video on screen in real time. Touch Projector [4] enables bi-manual interaction with one hand holding a mobile phone and the other hand interacting with the remote screen through a live video on the mobile phone. Virtual Projection [22] uses mobile phone as a controllable projection to interact with content on stationary displays. More work can be referred to the survey [23]. As the users are holding the mobile devices with their hands, freehand interaction cannot be achieved in a user-friendly way. Additionally, small display screens impedes the precision of fine manipulations.

2.2 Freehand Interaction

Freehand interaction has been explored to deliver natural, intuitive and effective interaction [12]. For a natural user interface, traditional input devices such as keyboard and mouse are not appropriate. Previous work [24], [25] requires fiducial markers or digital gloves to track hand gestures. Other works leverage Microsoft Kinect to detect hand poses and movements for freehand menu selection [26] and object manipulation [27]. However, these methods suffer from drawbacks. Instrumented gloves are encumbered and prone to induce fatigue. Fiducial markers and ambient sensors require delicate set-ups and calibrations.

Image-based methods [28], [29] have been proposed to detect and recognize hand gestures using image processing technology, which are suitable for both closed and public environments [30]. As mobile devices become more powerful, these methods are promising for mobile devices as built-in cameras can be used without resorting to additional devices or sensors.

2.3 AR Menu

Menus are widely used in traditional GUI to facilitate the user experience. However, using menus in the domain of AR is not straightforward, especially when the context is extended to three-dimensional environment. In order to support mid-air operations, non-pull-down menus such as ring menu [31] and tile menu [32] have been designed for manipulation in three-dimensional space.

As traditional input devices have been discarded to reduce obtrusion in AR applications, other devices are developed for menu manipulation. In Studierstube [33], a tablet-and-pen device is used for menu selection. However, users still have to take the tablet and pen with their hands. Some physical props such as fiducial markers [34] and hand fingers are also used to select menu items. To capture diverse motions such as wrist tilt and multiple pinch gestures, digital gloves equipped with motion tracking sensors are used in [12].

2.4 Rethinking

A large part of human interactions are daily operations, which requires minimal visual attention as the functionality of each object comes intuitively [35]. An ideal user interface should minimize the visual attention and obtrusion to users, and only appears when it is needed, otherwise it remains invisible to Interaction should also be directly channeled to the physical objects rather than through a workstation (as traditional GUI).

Ubii is a system that blends into the work place and will not obtrude user activities that can address the issue. Ubii is similar to the work [35] and [36] with a series of improvements. Rather than reprogramming functions of physical objects, we anchor our views on seamless interactions between individual physical objects without obtrusion. Wearable devices such as Google Glass can replace mobile devices to enable freehand interactions. 3D ring menu is used to support mid-air operations and pinch gesture is adopted to meet the required precision of menu manipulation. By putting the technologies together, Ubii attains the user experience of interacting with the physical objects directly, allowing users to manipulate multiple devices through it. The aim of our research is to illustrate a novel approach to move beyond the current GUI paradigm.

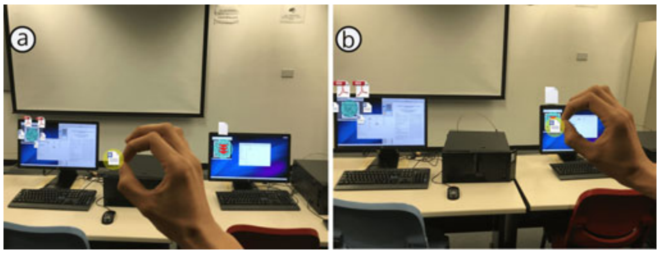


Fig. 2. (a) A user picks a document item from a computer and drags it towards another computer. The selected item is highlighted with a yellow circle around the virtual icon. (b) The user drops the document item onto the capture zone of another computer to finish document copying.

3 System Design

Designing a system as Ubii requires consideration of the device running it, the physical affordance it operates on and how the system recognizes them, the virtual interface presented to the users, and the interaction for users to bridge the virtual interface and the physical affordance.

3.1 Wearable Devices

Ubii is an augmented reality application that requires users to operate with wearable devices. Smart glasses are configured with cameras, processors, network capabilities and can display augmented views to the users. These features are all essential for the Ubii system. As the user is wearing the smart glass, the camera will capture the scene of the surrounding physical environment. The camera recognizes the smart devices in the environment and tracks their locations based on the information on the respective markers. The camera on the smart glass also recognizes the user hand gestures which allows the user to interact with objects freehand. The smart glass also provides users with an augmented reality view, which overlays information and interface neatly over the real world.

3.2 Physical Affordance

All objects present a form of physical affordance, and we only focus on those that have relation with the digital world in our system. For instance, a computer provides an affordance for entering the cyberspace; a printer exhibits an affordance for printing digital documents. Physical surfaces such as table tops and architecture partitions (e.g., walls and ceilings) can also act as a tangible interface. To name an example, a blank wall can be treated as a virtual bulletin board for sharing information among group members. Authorized members are able to view notes and documents on the virtual bulletin board while walk-in visitors only see the blank wall.

Traditionally, one can only leverage on the physical affordance of an object by physically accessing it. Ubii enables users to interact with the affordances of different objects at a distance. The goal of Ubii is to establish a system which users can operate devices and perform daily tasks in an easier and more intuitive way with hand gestures. With Ubii, users can operate devices remotely and can easily operate multiple devices—which is especially preferable for groups of smart devices that are inter-connected. Ubii supports a wide range of pre-defined operations, which users can initiate by dragging objects from one device to another. Ubii features two major functionalities that allow users to operate smart devices with freehand interaction. The applicable interactions between different objects are explained in detail below:

1. *File Transfer.* File transfer is one of the most common device-to-device operation. Ubii enables files to be transferred between devices, which includes computers, smartphones, printers, projector screens and physical surfaces. In the Ubii system, every file is interactive and can be freely dragged around with simple hand gestures. Wearing the augmented reality glasses, users see virtual icons of the available files overlay around the smart devices. Some essential information is also displayed alongside the file. Users have access to the information without any additional efforts. Sending files from one computer to another is also convenient. Users no longer have to rely on USB flash drives and physically accessing the two computers. They only need to have the two computers in his sight. When a user drags and moves the designated file from the source computer to the destination computer with a pinch gesture, the file will be sent to the target automatically. The whole process is intuitive and does not require the user to move between the two computers. The file transfer does not limit to between computers. Ubii is built on the concept of Internet-of-things: different smart devices are connected together in a network to allow seamless interactions between them. More options are available to the users including printing, projecting presentations, sending files to their mobile phones etc. By leveraging natural user interface, there is also a much shallower learning curve and users do not have to remember the native interface of the different devices. The list of possible actions are listed below:

- Drag a file from a computer to another computer. The file will be copied to the destination computer.
- Drag a document from a computer to a printer. The document will be printed out from the printer.
- Drag a document or a presentation from a computer to a projector screen. The file will be opened full screen on the projector screen.
- Drag several documents from a computer to a designated area in the environment (e.g., a whiteboard). The documents will be shared and will be available to other users. Another user can drag the file from the whiteboard to his own computer to make a copy of the file.

Smartphones and computer work interchangeably in the Ubii system. Both of them can be the source and destination of any file transfer.

2. *Device Pairing.* Another major functionality of Ubii is to allow device pairing using hand gestures. Similarly, Ubii aims to bring devices together in a natural and user-friendly manner. In an office environment, many of wireless devices have the capability of being paired with one another to function. Supported peripherals include Bluetooth computer peripherals (keyboard, mouse) and Bluetooth speakers. Pairing these devices traditionally often requires somewhat bothersome operations from the users. With Ubii, users see the virtual representations of the devices and the menu options on the AR glasses. Ubii allows users to pair devices by dragging the overlays of the devices from one to another on the AR interface. The pairing will be done automatically and conveniently in a matter of seconds. When a user wants to pair a Bluetooth speaker to his phone, he can do so by dragging the

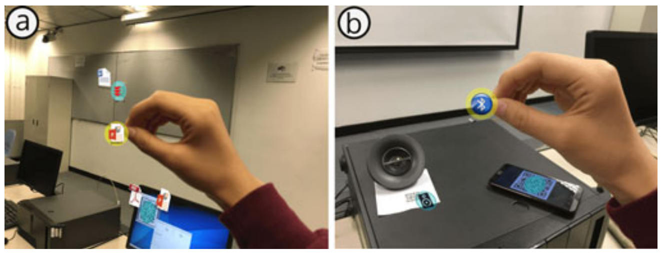


Fig. 3. (a) Dragging a shared file on a physical surface to own computer using Ubii. Virtual document icons are aligned with physical objects. (b) Pairing a bluetooth speaker to a mobile phone using Ubii.

virtual representation of the speak to his phone. When the two devices are paired, they will be shown on the AR interface as paired. If the user wants to pair another speaker with the phone, he simply has to drag the new speaker to his phone virtually, as shown in Fig. 3b. The originally paired speaker will be disconnected and the new one will be paired. The procedure of unpairing two devices is the same as the pairing procedure. Users have to drag the paired devices from one to another again and the devices will be disconnected. Ubii enables devices to be paired in a timely manner and allow multiple devices to be connected together. It can save the users a lot of hassles of operating multiple devices.

3.3 Marker Design

We use physical markers to assign unique tags to physical objects. Each marker contains information such as the type of object, the network location and additional information based on the property of the physical object. The information on the markers determines how the users can interact with the objects. For instance, the marker can define an area in the environment to be a virtual bulletin board. Users can share their files and documents there with other users easily. When another authorized user is in proximity he can see the shared documents as well as the version numbers of the files. When a user has shared a newer version of a document in the network, other users can download a copy the file by dragging the document from the bulletin board to their devices.

Each device is paired with a printed physical marker, using QR code to encode the information. Each marker has the minimum size of 5 cm $\times$ 5 cm, to be recognized by the wearable devices at a usable distance. The camera on the Google Glass captures video at the resolution of 720p, and can recognize small markers at a distance as far as 2 meters. The reliable using distance increases when larger markers are used in the environment (e.g., when the marker occupies part of the projector screen). The marker is normally a printed paper attached to the device. On the mobile phone, it is displayed on the screen by launching the specific Ubii application. The Ubii system identifies the physical markers, and based on the embedded information, overlays the information to the users in an augmented reality interface. The location and angle of the markers also determine the orientation and coordinates of the virtual overlay relative to the user in the interface.

3.4 Menu Design

A menu system in the domain of AR applications is not necessarily to be two-dimensional. It is able to feature depth, rotation, and position in 3D space. We follow the criteria in [37] to design an interactive menu for our system:

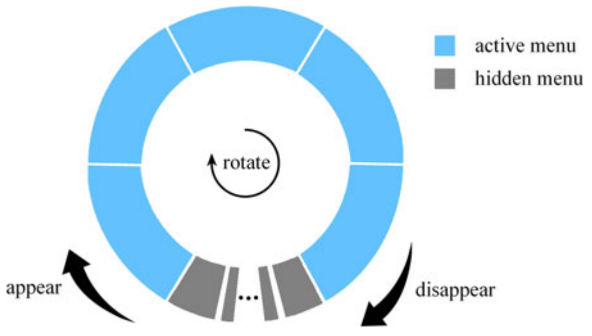


Fig. 4. A rotatable and foldable ring menu.

Placement: Menu placement is classified into four categories [38]. We adopt the object-referenced placement to attach the menus to physical objects, anchoring by visual markers tagged on the objects. It is consistent with the design goal of breaking centralized interface into tailored individual interface associated with physical objects. Menu sizes vary corresponding to detected marker sizes, indicating the relative spatial relation between a user and physical objects.

Orientation: Aligning the menu with users' view makes it easy to read but may occlude the environment [37]; Rather, we align the menu with physical object surface which is tagged with the marker to improve readability, which also achieves a better 3D spatial presence of the menu.

Trigger mechanism: A menu may be visible or hidden based on the active state of the corresponding object. The menu will be visible if the visual marker of the physical object appears in user's current view. A menu item is triggered whenever a user pinches the item with fingers, and a capture zone is activated if a menu item is dropped to its capturing region. We also add a highlighting effect as visual feedback of activation as in Fig. 2, making it easy for users to know which file they have selected. Each activation sends out network sockets to perform the underlying network communication.

Attributed to the capability of short target-seeking time and low error in selection [39], ring menu is adopted to collaborate with hand-based interaction for freehand menu selection in Ubii. Menu items are distributed on an invisible ring around an anchor determined by the visual marker. Traditional ring menu enlarges the ring size or uses hierarchical structure when it is too crowded to display all menu items in a single ring. Neither method is feasible as items increase to squeeze the space. To solve the problem, we enable menu items, namely documents ordered by type, to be rotatable along the ring. As illustrated in Fig. 4, only a few menu items are active and the rest are folded. Active items disappear and hidden items appear as the ring menu rotates clockwise.

Capture zone is designed to decrease the precision requirements of interactions. As soon as an item is picked up, all the other valid devices in sight renders a capture zone to prepare for receiving. Any menu item dropped into the range of a capture zone is captured by the receiver to trigger underlying commands. For instance, when a user selects a menu item (e.g., a document) of a computer, all printers, projector screens, and predefined physical surfaces in user's current view are overlaid with their capture zones. If the item is dropped to the capture zone of a printer, the printer will capture it and print out the document. When the item is dropped to the capture zone of a projector screen, the

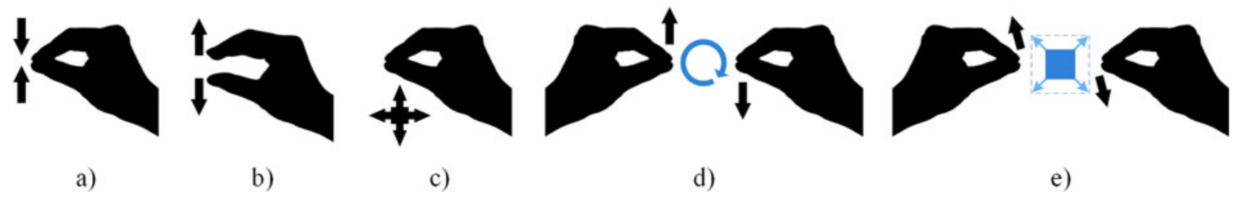


Fig. 5. The supported hand gesture based interactions in Ubii.

computer connected with the projector will receive the document, and then request the projector screen to display it.

To provide users an easier way for the multi-file transfer, we add a virtual buffer zone inside the ring menu. In Fig. 2, the green region inside the menu of the sender computer is the buffer zone. When there are more than one files to be transferred, users can select and drag each file to the buffer zone first, and then drag the buffer zone as a whole to the target destination. Files in the buffer zone will be emptied after being dragged to the destination. Notice that the buffer zone acts just like a capture zone in the destination object, but near the source, thus avoiding files to be dragged through a long distance multiple times.

3.5 Interaction Design

Ubii extends previous video-based interaction to hand gesture interaction without using any digital gloves or fiducial marker tracking. However, pointing gesture is difficult for direct positioning operations (e.g., menu selection) because it cannot achieve the same high precision as physical input devices (e.g., mouse and stylus) in free spaces [12]. Directional gesture is ambiguous and error-prone due to lack of distinctive gesture delimiters [40].

Our essential design principle is to leverage multiple discrete gesture inputs to reduce the precision requirement of continuous hand gestures. The pinch gesture is stable and precise attributed to its discrete and unambiguous states of fingers together and apart [28]. In addition, it appears natural to users as it is evocative of grabbing a physical object [12]. Pinch gesture is suitable for interaction which requires high precision, such as menu picking and dropping. Combined with directional gestures determined by tracked hand position, it also can be used for coarse interaction, such as dragging. Ubii integrates multiple hand gestures to support several types of freehand interaction:

Pick: a pinch gesture (as illustrated in Fig. 5a) is interpreted as a pick operation. The pick operation should be performed on a menu item, otherwise it is invalid.

Drop: an un-pinch gesture (as illustrated in Fig. 5b) acts as dropping the selected item. A pick operation should always finish with a drop operation.

Drag: triggered when a directional gesture is performed with a pick operation (as illustrated in Fig. 5c). The selected item is moved along with directional gesture if only a single drag operation is acted. The drag interaction terminates whenever a drop operation happens.

Rotate: a combined operation of two pinch gestures that move reversely along the Y direction on screen space (as illustrated in Fig. 5d). The interaction is used to rotate a ring menu which is located between the two pinch gestures. Relative distance is mapped as rotation angle. Rotation operation stops whenever either or both pinch gestures end.

Zoom: a combination of two drag operations performed on the same item (as illustrated in Fig. 5e). Relative movement is transferred to scale up and down the item size. As one or both of the drag operations end, the zoom operation terminates.

Pick and drop are instantaneous interactions. All interactions start with the pick and end with the drop operation. The other three start from the pick operation and end at the drop operation. They also enclose self-transitions for continuous interaction.

4 Implementation

Implementing our system requires technologies from fields of object tracking and hand gesture recognition. The physical marker-based method is used for both physical object tracking and camera pose evaluation. Two different approaches, including double-threshold algorithm, and pre-trained SVM with Bag-Of-Words model, are tried out and analysed for pinch gesture detection. Our work explores the synergetic benefits that go beyond the sum of multiple technologies to achieve our design goals.

Fig. 6 shows the system work-flow. The built-in camera on Google Glass takes live videos of the scenes around users, which is streamed for image analysis of object tracking and hand recognition. The system extracts object information of IP address and object type from QR code, which is used for physical affordance recognition and network communication. Camera pose is also evaluated by calculating the homography of the QR code images. Virtual camera is updated according to the real camera pose to render 3D menus from corresponding viewpoint, making virtual menus align with surfaces of physical objects. Menus are rendered as AR overlays to blend with video stream, and then displayed on Google Glass display prism.

Hand detection and pinch recognition are implemented with detailed descriptions in Section 4.2. Users are able to interact with physical objects through virtual menus using pinch gestures, which are interpreted as sockets to perform communication between physical objects through the underlying network, as in Section 4.4.

4.1 Object Tracking

Ubii heavily relies on the built-in camera on Google Glass to understand the domain environment. The object tracking

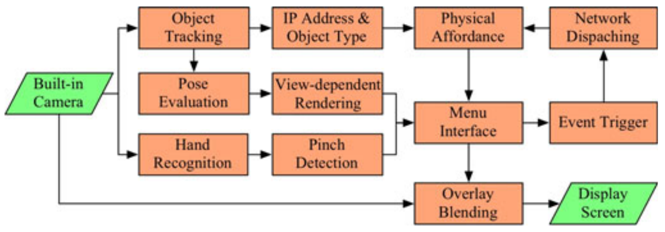


Fig. 6. The system workflow of Ubii.

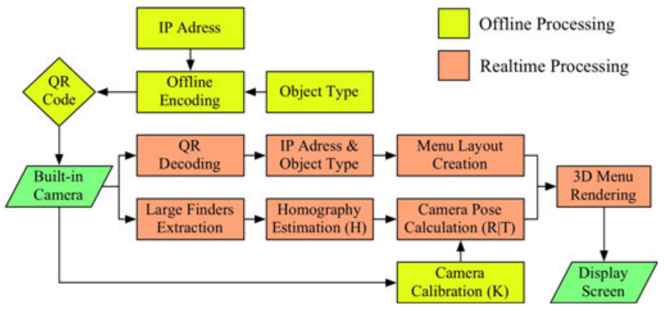


Fig. 7. Object tracking and pose estimation with QR markers.

involves in recognizing objects and evaluating the spatial relationship between objects and users. Rather than markerless methods such as [41], we adopt the marker-based baseline, which is generally more accurate and robust in spite of image distortion and illumination variation [42]. In our system QR codes are used as fiducial markers to encode additional information for understanding object functions.

As illustrated in Fig. 7, object information including IP address and object type are encoded into QR codes in advance using the qrcode tool.[1] For non-wired physical objects such as physical surfaces (e.g., wall and tabletop) and projector screens, IP addresses of their counterparts, namely computers storing documents on physical surfaces and computers connected to projectors for projection display, are encoded.

Menu Orientation: Camera calibration is performed beforehand to calculate the intrinsic matrix K with the classic chess board method [43]. At runtime whenever one or more QR codes appear in the live video stream is captured by embedded camera, the system resorts to the integrated Zbar library[2] to decode object information from QR codes. It is then able to determine the menu item layout according to the extracted information. The large finders in QR codes are also extracted to get the homography matrix H by solving the coordinate vector-valued function, which is then used to evaluate the pose matrix $R|T = K - \mathbf{I} * H$. In practise, we adopt a tricky implementation by using the more efficient and accurate solvePnP method from the library OpenCV4Android.[3] The intrinsic matrix K is mapped to the Normalized Device Coordinates (NDC) in OpenGL ES rendering system for perspective projection, while the pose matrix $R|T$ should be flipped along X axis for consistently locating virtual menus in OpenGL coordinate frame.

QR code tracking: Camera is constantly moving during when a user is using of Ubii. To accurately find the surrounding QR codes, namely identifying the positions of nearby affordances every video frame, tracking of the QR codes is needed. After starting the application, the vertex positions of each newly encountered QR codes will be calculated by the Zbar library, and the respective image is stored in a initially emptied cache folder. Suppose in up to n frames, there are k QR codes encountered and stored for the current environment setting. Then at frame n, a feature matcher with ORB feature detector and extractor would match $(n\%k)\mathrm{th}$ QR code image to current frame scene, finding the position of affordance it represents. Notice that we only perform one matching per frame, due to the inaccuracy in the result and performance lags caused by matching all QR patterns together in every frame. Although each QR pattern is matched per k frames, the result is satisfactory since the environment movement is not significant and frame rate is relatively high benefited from offloading described in Section 4.3.

1. https://pypi.python.org/pypi/qrcode
2. https://github.com/ZBar/ZBar
3. http://docs.opencv.org/

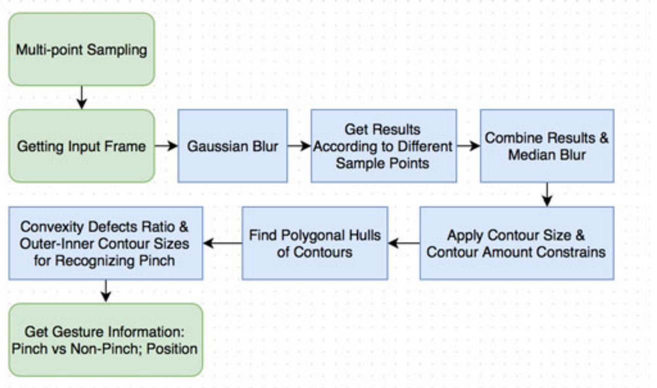

Multi-point Sampling
Getting Input Frame
Gaussian Blur
Get Results According to Different Sample Points
Combine Results & Median Blur
Convexity Defects Ratio & Outer-Inner Contour Sizes for Recognizing Pinch
Find Polygonal Hulls of Contours
Apply Contour Size & Contour Amount Constrains
Get Gesture Information: Pinch vs Non-Pinch; Position

Fig. 8. The whole flow of hand gesture recognition method a—Real-time contour based recognition.

We attach markers with QR codes to every physical objects, which may not be fully consistent with our vision of designing a self-contained system. We view QR markers as a temporary and minor violation of our premise, and believe that advancements in ubiquitous sensors and smart devices will help us to understand environment without additional annotation in the future.

4.2 Hand Gesture Recognition

Two approaches for hand detection and gesture recognition will be discussed in this section. The first one is the real-time contour-based method, mainly adopted by [28]. It is computationally easier, but less robust when the background is complex. The second method pre-trains an SVM classifier with Bag-Of-Words (BOW) model developed in [44], saving the classifier into XML file to support frame-wise pinch and non-pinch classification.

a. Real-time contour based recognition

Pinch gesture provides a simple and reliable way to detect when interactions start and end without additional gesture delimiters. It can be determined by the thumb and index fingers touching together to make a hole as illustrated in the Fig. 9a. In this method, we use OpenCV to filter out some disturbance in the environment as well as those gestures largely different with the pinch, and then improves Wilson's vision-based method [28] to detect one-hand and two-hand pinch gestures.

The whole logic flow is shown in Fig. 8. Since foreground hands and background scene are both changing constantly from the user's view angle, the common background subtractors based on the static background cannot be applied. Getting the idea from [45], we pre-sample the user's hand color in order to eliminate the background. According to Zarit et al [46], HSV performs the best in skin detection. Therefore, each RGB color space frame taken from camera input is changed into HSV color space firstly. Ubii gets initiated with a sampling stage, during which users are asked to put their hands in front of the camera and cover the multiple sampling squares (in our case, five squares as shown in Fig. 10).

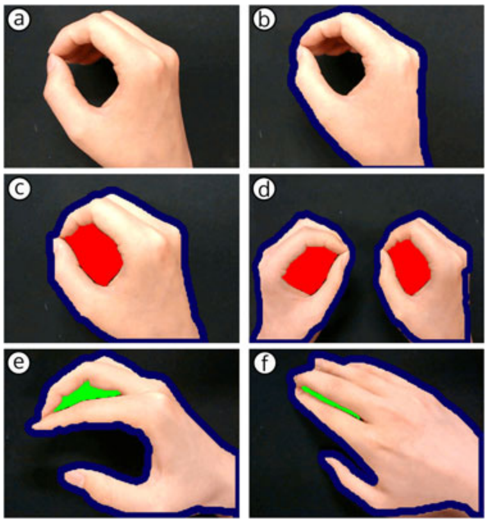


Fig. 9. The pinch gesture detection. a) a typical pinch gesture; b) outer contour of hand is extracted; c) and d): red closed regions are recognized as pinch holes; e) and f): green closed regions are eliminated as non-pinch holes given their sizes are below the lower bound.

Single-color based extraction would give very restricted resulting points instead of the hand shape wanted, so we add a color radius allowing small variation in hand color. Research indicates that removing restriction on Value (V) component will make extortion result illumination invariant, while still keeping high accuracy in skin pixel detection [47]. Thus in our detection approach, we only consider Hue (H) and Saturation (S) components by setting color radius of V to 255.

After sampling for three seconds, Ubii enters the processing stage. We implement Gaussian blur to reduce noise in the input frame. Then based on the five sample colors together with the color radius, we extract corresponding hand-color shapes. After these five shapes are extracted, we perform a bitwise combination on them. The combined shape contains a more accurate hand shape since it covers the colors of most hand positions. We then apply a median blur on the combined shape to filter out obvious irrelevant color blobs.

Next we extract the contours of hand-color sections. Every hand contour has an area greater than a certain threshold since the distance between hand and camera is limited by arm length, in our experiment the lowest area of hand area is around 130 ($pixel^2$) when the arm is straight forward. We first filter out those contours with the area less than 120, getting

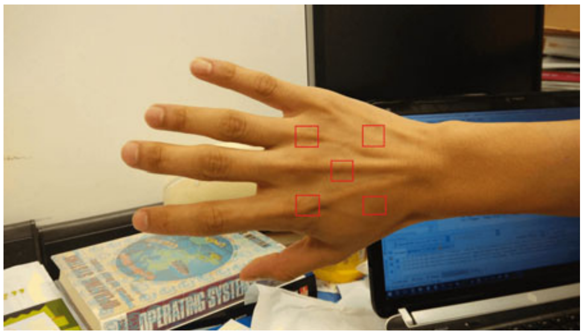


Fig. 10. View of color sampling stage. The average color inside different sampling squares are computed, and selected as different sample colors; In this case, five average colors and color radius are used as references for extraction.

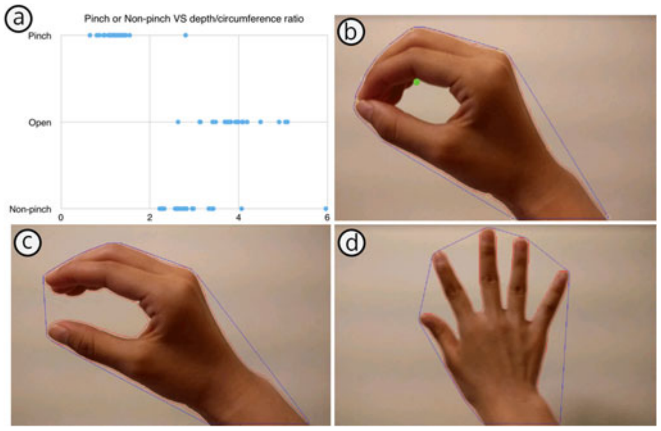


Fig. 11. a) Different ranges of convexity defects/ contour circumference ratio between pinch, open, and non-pinch gestures; b) pinch gesture with position tracker point; c) un-pinch for releasing files; d) non-pinch gesture.

rid of small disturbances in the environment. If still more than two valid hand colored contours remain, we then only keep the largest two hand colored contours since all operations require at most two hands.

After getting contours, we extract the polygonal hulls wrapping them, and calculate the corresponding convexity defects. Pinch gestures have different ratio range of deepest defect length versus contour circumference from ratio ranges of un-pinch and non-pinch gestures as illustrated in Fig. 11. After setting the threshold on the ratio at around 2.1, we can differentiate between pinch-like gestures (ratio lower than 2.2) and other gestures (ratio higher than 2.2).

The problem now is differentiating between pinch gesture and those gestures that produce inner contours as well. Here we follow Wilson's method, extracting both inner and outer contours. The largest one is the outer contour of the hand section as illustrated in Fig. 9b. Any hole in the hand section indicates an inner contour that may be produced by ambient illumination and other non-pinch gestures. Our method adopts contour size to filter out wrong contours. Any inner contour with the size out of bounds is discarded, and the bounds are determined by the pinch acceptance and non-pinch elimination curves as illustrated in Fig. 12. Since non-pinch gestures generally have smaller inner contour sizes, they will be more accurately eliminated if the lower threshold is larger. Conversely, the upper bound is set to let most real pinch gestures to get accepted. We choose a lower bound of 0.06 and an upper bound of 0.08 to achieve over 85 percent accuracy of both pinch gesture detection and non-pinch gesture elimination. In order to avoid background color interruption to the contour sizes, we carry out above steps for choosing two bounds under a pure dark background setting, but the resulting thresholds are applied to the final system with previous background removal process under complex background settings, and the overall accuracy will be discussed in SectionX.

This method works for detecting both one and two hand pinch gestures as shown in Figs. 9c and 9d. For one hand pinch, it works the same for both left hand and right hand. Overall, the method gets inaccurate when the background has a poor illumination or a similar color to the human skin. Besides, differentiation between the pinch and gestures

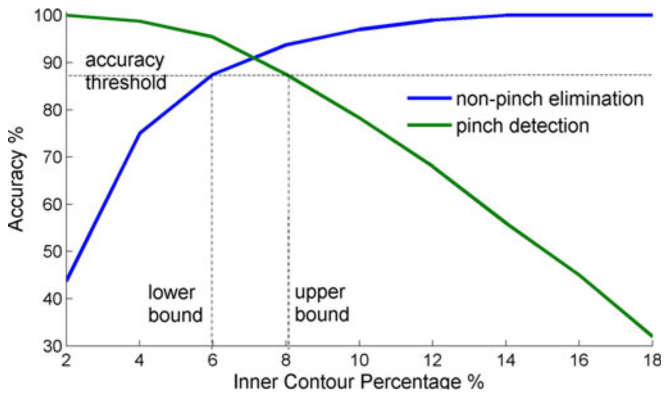


Fig. 12. The inner contour acceptance and elimination under pure background.

with inner holes fails if the inner hole background is occluded by other curled fingers [28] or the hole is not visible to the camera, such as the thumb and index fingers being horizontally coplanar.

b. Pre-trained SVM classifier using Bag-Of-Words model

Apart from above real-time algorithm, a pre-trained SVM classifier with BOW model [44] is tested.

Pre-processing: Since the actual environment where Ubii is used can be rather complex, we need to carefully pre-process the images in order to remove most part of the background. As mentioned earlier, the background removal cannot use simple subtraction of the previous frame onto the current frame. Our method here only considers local frame image segmentation, ignoring the spatial relationship between the current image and the former ones. Lakshmi and Sankaranarayanan [48] gives a foreground extraction method for moving camera: detecting image edges, and then scanning through the pixels per row, filling in those pixels between two edge pixels. This method works well for the continuous backgrounds that do not contribute many edges. However for Ubii in an office setting, there might be many more edges existing in the background, and this algorithm cannot tell which object is the foreground hand. Fatma and Sharma [49] presented two approaches for image segmentation: k-means clustering, and thresholding. Their comparison on Ubii is shown in Fig. 13a. Generally, for our case, thresholding performs much better than k-means clustering with k = 2 (foreground and background), therefore, thresholding using skin color information is used. Sobottka and

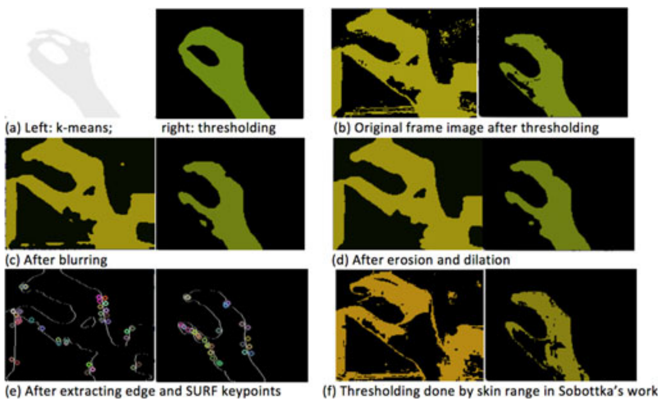

(a) Left: k-means;
right: thresholding
(b) Original frame image after thresholding
(c) After blurring
(d) After erosion and dilation
(e) After extracting edge and SURF keypoints
(f) Thresholding done by skin range in Sobottka's work

Fig. 13. Frame image segmentation process.

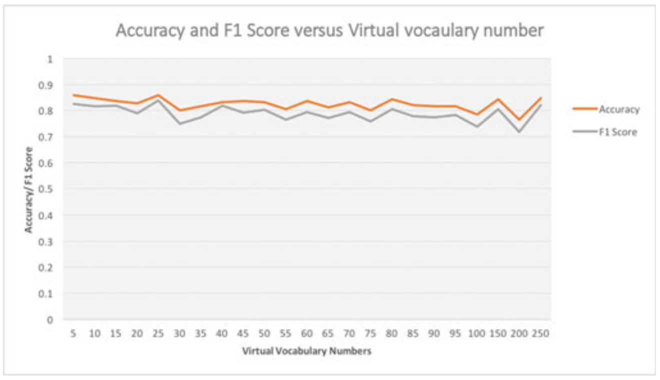


Fig. 14. Accuracy and F1 score of SVM classifier on different BOW size.

Pitas [50] sets the commonly known skin color in HSV range as H = (0, 50) within range (0, 180) and S = (0.23, 0.68) within range (0, 1). The resulting threshold is not very satisfactory, as in Fig. 13f, we reset S to (80, 200) within range (0,255), and the result is shown in Fig. 13b. To eliminate noise, Gaussian blur and one iteration of erosion and dilation are performed, leaving only color blobs that are large enough.

Feature extraction: We extract the edges of color blobs of the previous step at the pixel-color changing points, then detect key points of these edges using SURF keypoint-detector and only keep those points that are near outer contour of the hand-color blob. This step is needed since it will raise the accuracy by 15-20 percent than directly detecting keypoints on color blobs and also lessen SVM training time by a considerable amount. SURF feature descriptors are then extracted based on these remained key points and used to create virtual vocabularies. BOW model collects the relatively large amount of descriptors into defined number of vocabularies, thus making classification feasible through equalizing the number of feature inputs [44].

Training SVM: We train the SVM classifier based on BOW vocabularies. We use six 190$\sim$200 frame videos, three positive ones (pinch) and three negative ones (non-pinch), as training inputs. All the mirror images of processed video frames are also used in the training set in order to support recognition of both hands. Among the videos, two are taken against a pure white color background, two taken in the office, and two more in a cafe where the illumination is poor. Testing set is a 200-frame video taken in the office with about 130 frame pinch and 70 non-pinch. The resulting accuracy and F1 score is shown in Fig. 14. As the performance of different BOW sizes is generally constant, we simply choose 40 as the virtual vocabulary number in our experiment. Pre-trained SVM is saved into an XML file, and then the classifier can be used on real-time pre-processed (steps are the same as former two) frame images. In general, this pre-trained method gives a much better result for hand detection in real settings, and is more resistant to illumination and background disturbance. Detailed comparison is conducted in Section 5.2.

4.3 Computation Offloading

Since Ubii requires advanced algorithms for both object tracking and hand gesture recognition, sufficient computational power is needed to ensure a smooth user experience. Wearable devices have limited computation power and a

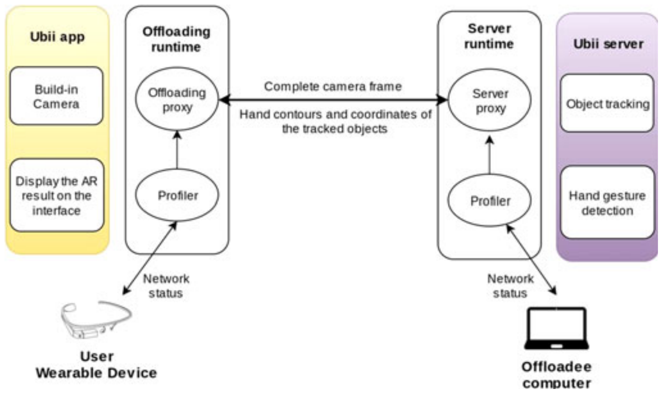


Fig. 15. The computation offloading process.

desktop or a laptop computer has much superior processing power and memory. To optimize the performance of Ubii on the mobile devices, we implement computation offloading to process the most strenuous tasks in the system. In the Ubii system, the wearable device captures the scene and display the augmented view to the user, while the computer acts as the processing unit when offloading is possible. We use a local computer as the processing unit instead of a cloud unit as such a design can eliminate any potential network latencies. By shifting the intense computation tasks to the computers, the load on the wearable devices are reduced. This effectively prolongs the battery life of the devices and the system can be used for an extended period of time.

The wearable device can be paired with a computer in its proximity for offloading purposes. The offloader and offloadee are both connected to the same WiFi network, and communicate through TCP connection in real time. Developed similar to the MAUI architecture [51], the offloading framework is shown in Fig. 15. The offloading system includes two parts - the wearable device and the offloadee computer. The wearable device is governed by a offloading runtime, which consists of a profiler and a proxy. The profiler monitors the network status and battery level on the wearable device. The proxy handles the network communication and data transfer with the offloadee computer. The computer is governed by a server runtime, which consists of a server profiler and server proxy. The server profiler monitors the network connection with the wearable device. The server proxy sends and receives data from the wearable device. When both profilers determine that the network connection is stable, the wearable will send the camera frames to the offloadee computer for processing. If the network bandwidth falls below a threshold determined by the profiler, the computation will be carried out locally by the wearable device. The proxy will resume the offloading once the network returns to a favorable condition.

4.4 Server Network Communication

Acting as an interface that can control multiple devices, Ubii requires smart devices to be connected with one another in order to operate. The devices and computers need to have active Internet connections and ideally to be connected to the same network. For printers and screen projector, they need to be configured to be connected to computers with Internet connections. As Ubii uses physical markers for object tracking, we mark each device with a distinct QR

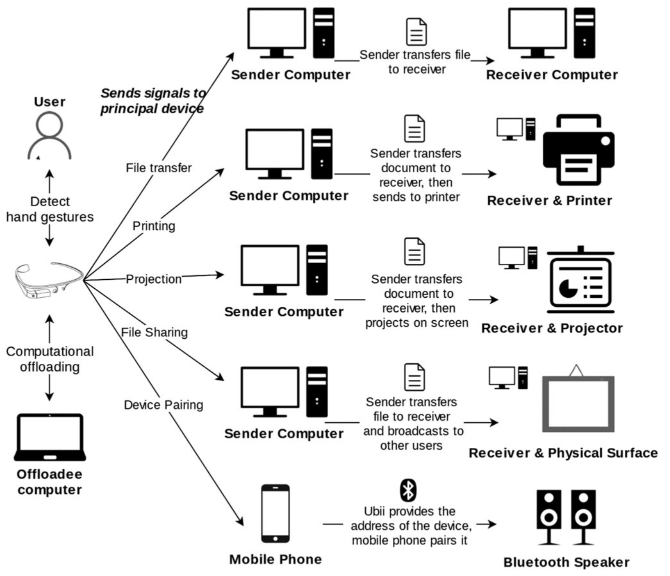

User
Sends signals to principal device
Detect hand gestures
Computational offloading
Offloadee computer
File transfer
Printing
Projection
File Sharing
Device Pairing
Sender Computer
Sender transfers file to receiver
Receiver Computer
Sender Computer
Sender transfers document to receiver, then sends to printer
Receiver & Printer
Sender Computer
Sender transfers document to receiver, then projects on screen
Receiver & Projector
Sender Computer
Sender transfers file to receiver and broadcasts to other users
Receiver & Physical Surface
Mobile Phone
Ubii provides the address of the device, mobile phone pairs it
Bluetooth Speaker

Fig. 16. A diagram showing how the devices communicate with one another in the Ubii system.

code which encodes the IP address of the device. Once Ubii reads the information on the QR Code, it turns the physical object into a fully interactive object in the digital space. When an user is interacting different objects in the environment, Ubii will sent out network requests to other connected devices automatically to complete the operations. The implementation of the system connection for different operations is shown in Fig. 16.

File Transfer: Ubii identifies the QR codes of the computers captured in the device camera. The IP addresses are stored temporarily in the application. When a user initiates a file transfer sequence between two computers with hand gestures, Ubii resolves the two IP addresses of the two computers. Ubii then identifies the file that users would like to transfer according to the hand position, and notifies the sender computer with a network request. The network request contains the IP address of the receiver computer and the requested file. Upon receiving the request, the sender initiates a connection with the receiver end. The sender then proceeds to transfer the file to the receiver computer over the established connection.

Printing: When a user initiates a printing sequence by dragging a document item from a computer towards a printer, Ubii identifies the QR codes assigned to the computer and the printer. The printer is either connected to a computer with Internet connection or to a cloud print service (e.g., Google Cloud Print). A connection is established where the computer acts as sender and the printer as receiver. If the printer is connected to a computer directly, Ubii will directs the sender computer to send a request to the computer connected with the printer. Upon receiving the file, the receiver computer is signaled to print out the document directly. If the printer is connected to the cloud, the sender computer will be signaled to upload the file to the cloud service and print out the document through a designated printer.

Projection: Similar to printing, the projecting device needs to be connected to a computer with Internet connection. When a user drags a document from the sender computer towards the receiver projector screen, Ubii sends out a request to the sender computer that contains information of the receiver computer. Upon receiving the request, the sender computer transfers the target file to receiver computer, with the request of projecting that document. The receiver then opens the document and projects it onto the connected projector screen.

File Sharing: File sharing on a physical surface is achieved by pairing the physical surface with a computer. The computer will handle the network communication for the purpose. A physical marker encoding the address of the computer needs to be placed on the surface. The process is similar to the file transfer operation. When a file is shared with the physical surface, Ubii will also broadcast the information to other devices in the system so other users can see and interact with the file.

Device Pairing: When the pairing process is initiated, the wearable glass recognizes the marker associated with the Bluetooth device and retrieves the pairing information, which includes the device name and MAC address. Ubii passes the information to the mobile device (or computer) and the mobile device will initiate the pairing with the information. When the user initiates an unpair signal, the mobile device will unpair the Bluetooth device and release it for other uses.

All these above mentioned operations are initiated by Ubii automatically when corresponding hand gestures are detected. Users do not need to interact with the traditional GUI or input devices. By connecting the physical and digital worlds, users can complete tasks with a seamless user experience.

5 Evaluation

5.1 Offloading Evaluation

In this section, we measure the performance differences with and without offloading. For the system without offloading, computations are done on a Google Glass. For the system with offloading, computations are offloaded to a MacBook Pro (2.6 GHz Intel Core i5, 8 GB Memory). Since the profiler on the offloading module only executes when the connection between the wearable device and the offloadee is favorable, both the Google Glass and the computer are connected to WiFi to ensure offloading is used. To measure the performance of the Ubii, we perform the file transfer operation task and measure the average frame rate during the task. We repeat the experiment 10 times for a more reliable result.

Experiment shows that there is a vast increase in the performance when offloading is used. The average frame rate increases from (M = 4.32 fps, SD = 1.61 fps) to (M = 13.54 fps, SD = 1.86 fps). The increase in frame rate could be explained by the fact that the OpenCV algorithms are too strenuous to be computed solely on a wearable device. Computation offloading increases the performance by shifting the bottleneck from processing power to network connection speed. The experiment also shows that as a computer has much higher computing power, offloading is preferable when available.

5.2 Hand Gesture Recognition Evaluation

In this section, we tested the performances of two hand recognition approaches described in Section 4.2 under different scenarios. The experiment is carried out on Google glass with offloading to a MacBook Pro (2.6 GHz Intel Core i5, 8 GB Memory). We use the glass to get input video streams of hands from five users of different genders, nations, and skin colors. The environmental settings include a cafe, a library meeting room, and an outside garden under natural sunlight. All backgrounds are complex, consisting various objects. Table 1 shows their corresponding illumination level.

TABLE 1
Illumination Level of Experiment Environments

	Illuminance[1] (lux)	Average V value in HSV (scale: 0-255)
Cafe	$50 \sim 100$	60 ± 20
Study Room	$300 \sim 500$	160 ± 30
Sunlight	$\sim 10{,}000$	220 ± 30

1. *http://www.engineeringtoolbox.com/light-level-rooms-d-708.html*

Each user performs a set of pinch to un-pinch hand gesture changes, and meanwhile, the glass and hands are turning around (~200 frames per user in one environment). We annotate the frames by hand to see whether they are pinch or un-pinch, and then compare with the recognition results given by two recognition methods. The resulting accuracies are shown in Table reftable: accuracy. We can tell that overall BOW-SVM method performs better in all environmental settings than the improved Wilson's method. This is due to the underlying mechanism of how these two methods work: Wilson's method uses constraints on convexity defects and inner contour sizes to filter out wrong hand contours, but the constraints are handcrafted with fixed thresholds, which could not include all the information about a certain gesture; whereas SVM method classifies the feature points automatically so that it might involve some other undefined attributes to classify the gestures, and can recognize them from more different angles. From the table, we can also see that low illumination (cafe) will significantly reduce the performance of both methods, with more influences on Wilson's method; performance in mid-level illumination (room) and high illumination (sunlight) are rather close. The standard deviations of SVM method are higher than the Wilson's method in all cases, and it will sometimes underperform Wilson's method (user5). We think this is caused by the sampling stage we add to Wilson's method, that it makes recognition process fit the skin color of each user better than the fixed skin color range used in SVM method.

Although SVM method provides better classification results, it also bears a higher computation cost, mainly on the feature extraction part. Handing the vision computation and rendering process to the computer, Google glass itself serves as a thin client, only providing and displaying video frames, and the battery can run Ubii about 40 minutes. On the computer side, the average energy impact, allocated memory, and processing time are shown in Fig. 17. We can tell that the improved Wilson's method has less energy consumption, and needs less than half the memory of the SVM method. Meanwhile, we measure processing times of the sub-processes of each method per frame, dividing Wilson's method into pre-processing stage (hand color blob extraction and contour detection) and recognition stage (invalid contour exclusion and inner contour size thresholding), the SVM method into pre-processing stage (background removal, extracting hand region), SURF feature extraction

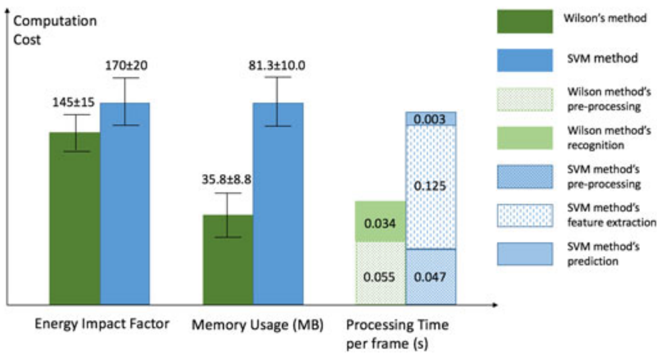


Fig. 17. Comparison of the average computation cost using two hand recognition approaches.

TABLE 2
Accuracy of Two Hand Gesture Recognition Methods in Different Background Settings

(%)	Cafe		Study Room		Sunlight	
	Wilson	SVM	Wilson	SVM	Wilson	SVM
User1	68.4	78.1	81.9	84.3	80.5	86.7
User2	70.3	79.2	82.1	86.6	81.2	85.8
User3	67.9	77.0	80.5	82.7	78.9	83.9
User4	69.8	80.6	83.4	87.5	83.1	89.1
User5	68.8	74.4	79.8	80.1	82.4	82.0
Avg	69.0±1.0	77.9±2.3	81.5±1.4	84.2±3.0	81.2±1.6	85.5±2.7

stage (to create virtual vocabulary), and prediction stage (using pre-trained model to predict). The averaged results showing that the most time-consuming task is to extract feature points for SVM classification, while the pre-processing stages of the two methods take similar times.

Based on the accuracy and computation pay-off of these two methods, we tested a hybrid method of the two: in the environment where the accuracy of Wilson's method drops significantly, we use pre-trained SVM model to predict, otherwise we use Wilson's method to recognize. The detailed implementation is: upon starting Ubii, the system enters a skin color sampling stage, if sampled color has a V value larger than 100 (scale: 0-255) in its HSV color space, we then choose Wilson's method, otherwise we use the SVM model. Learning from the result in Table 2, we now use the sampled skin color in the pre-processing stages of both methods. The finalized hybrid hand recognition approach (with offloading) has the accuracy 80 $\pm$ 4 percent in all environmental backgrounds with average frame rate ranging from 5.8 in a dim environment to 11.3 in a bright environment, and these can satisfy both accuracy and relative smoothness required by the Ubii system.

5.3 User Evaluation

In this section, we tested and evaluated the practicality of using Ubii, and the feedback received from the testing users. 11 participants took part in the evaluation, they are students from the School of Engineering across different departments in the university. Seven participants are male and four participants are female, and their ages range from 19 to 31 (M = 26, SD = 4.24). All participants noted that they have no prior experience of using a Google Glass.

Participants were invited to complete a series of task related to daily computer operations. Participants have to do each set of task twice using the Ubii system, and once using the traditional approach as a comparison. We showed the participants a simple video demonstrating how to operate Ubii. Instruction of the required tasks were also given. Each participant was given a pair of Google Glass.

The experiment took place in our computer lab, in which three of the computers set-up for the participants. We also configured a printer, a screen projector, a Bluetooth speaker and two Android phones for the participants in the laboratory. Each device is attached with a QR code marker, which is encoded with the network information of the device. The Android phones display the QR codes on screen by running a standalone application. USB flash drives are provided for document copying.

The four major tasks that each participant has to complete are as follows:

- Task 1 (Transfer): Select a document on the mobile phone and send it over to a computer. Then display the document on the connected projector.
- Task 2 (Pairing): Pairing a mobile phone with the Bluetooth speaker, then unpair it.
- Task 3 (Printing): Print out a document from a computer
- Task 4 (Batch copying): Select five files from a folder in one computer and copy ehm to a designated computer.

Upon completing the four tasks, each user was required to complete a System Usability Scale evaluation and a questionnaire.

We evaluate the performance of Ubii from following areas:

1. *Task completion time*: For each given task, we measured the time taken for each participant to complete the task. The time taken using Ubii and the normal operation are recorded separately. If one operation method spends less time on a task, it is expected to be a indication that the method is the more efficient one. 2. *Success rate*: We measured the rate that user can complete the task without error in one attempt. An example of a failed attempt is transferring a wrong file from one device to another. Having a higher success rate indicates that the system is a more reliable, and more user friendly one.

3. *System Usability Scale* [52]: We measure the usability of Ubii with the System Usability Scale (SUS). SUS is a simple, subjective and effective way of accessing the usability of a system. The SUS score ranges from 0-100, and a higher score indicates greater usability of the system.

4. *Post-experiment questionnaire*: At the end of the experiment, participants were asked to express their ideas and comments by answering a few questions. The questionnaire includes open-ended questions (e.g., What is your overall experience of using Ubii?) for participants to leave their feedback.

The results of corresponding tasks are as follows:

1. *Task completion time*. Fig. 18 shows the comparison of the task completion time between using Ubii and the traditional method. The experiment shows a trend of participants spending a shorter time when Ubii is used. The difference is more significant when the task involves multiple distant devices (Transfer), which Ubii takes about 20 seconds (M = 19.5s, SD = 5.1s) and the traditional method

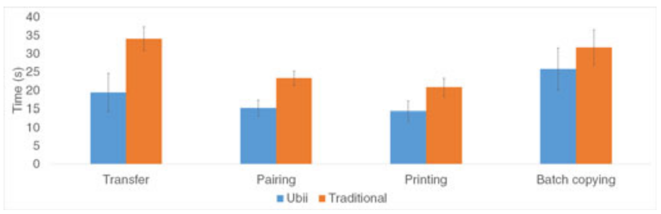


Fig. 18. Comparison of task completion time between using Ubii and the traditional method.

takes more than 30 seconds (M = 34.1s, SD = 3.3s). The improvement in task completion time is less noticeable in the printing task, only between Ubii (M = 14.4s, SD = 2.8s) and traditional (M = 20.9s, SD = 2.5s). This could be due to the fact that a normal print job only involves operating one computer. Ubii excels when the operation involves multiple devices, which having Ubii would save the user time walking from one device to another. In the batch copying task the improvement is also less significant, between Ubii (M = 25.9s, SD = 5.7s) and traditional (M = 31.7s, SD = 4.8s). While Ubii does support batch file operation by dragging files to the buffer zone, it takes time for the participants to get used to the feature and it could be more intuitive for them to use the traditional batch copying.

2. *Failure Rate*. We measured the number of unsuccessful attempts and the failure rate of each task when operating Ubii, as illustrated in Table 3. The unsuccessful attempts are mainly due to the wrong detection of hand gestures, which happens in situations of fast hand movements and having the background color similar to the hand. Transfer and batching copying have the highest unsuccessful attempt, with the failure rate of 14 percent. Transfer involves document transfer twice before projecting it on a screen, while batch copying requires dragging files to a virtual buffer area. The complexity of the task leads to a higher failure rate. Overall the performance is satisfactory since no participant has more than two unsuccessful attempts in a single task.

3. *System Usability Scale*. To use SUS, participants are asked to score the following 10 items with one of five responses that range from Strongly Agree to Strongly disagree: [52]

Q1 I think that I would like to use this system frequently.
Q2 I found the system unnecessarily complex.
Q3 I thought the system was easy to use.
Q4 I think that I would need the support of a technical person to be able to use this system.
Q5 I found the various functions in this system were well integrated.
Q6 I thought there was too much inconsistency in this system.
Q7 I would imagine that most people would learn to use this system very quickly.
Q8 I found the system very cumbersome to use.
Q9 I felt very confident using the system.
Q10 I needed to learn a lot of things before I could get going with this system.

The SUS score of Ubii collected from the 11 participants is (M = 76.4, SD = 14.8). The response of the participants is shown in Fig. 19. Having a mean SUS score between 71.4 and 85.5 indicates that the usability of Ubii is between "Good" and"Excellent" [53]. Particularly the majority of the participants felt that Ubii is easy to use and would like to use Ubii frequently if it is available.

TABLE 3
Failure Rate Using Ubii During Task Completion

	Unsuccessful attempts	Failure rate
Transfer	3	0.14
Pairing	2	0.10
Printing	0	0
Batch copying	3	0.14

4. *Post-experiment comments*. Participants left different comments in the post-experiment questionnaire. Most noted that operating Ubii is a unique and refresh experience and is user-friendly. Many participants showed interest in seeing Ubii in the market in the future. Some specific comments by the participants are as follows:

"Using hand gestures to control computers seems futuristic to me, and controlling a computer remotely looks cool. Just picking a specific file from the ring menu when there are too many files can be frustrating."

"I like the idea and it's really more convenient for the simple tasks in the experiment."

"Interesting experiment but sometimes I just want to sit down and use my computer. Standing still and dragging my hands around can be tiring."

"Overall this is a novel concept. I'm just concerned with whether other users would be able to access all my files without my permission."

The evaluation shows that Ubii is a preferable way to complete the general computer operations. It simulates how a user would use Ubii in a daily office environment. Ubii is a supportive tool that assists users when they need to interact with close by devices at times. As the users would not be using Ubii for a very prolonged time, the efficiency and convenience outweigh the undesirable effects and the fatigue of using Ubii will not be significant.

5.4 Future Work

There are still several ways to improve the accuracy of hand gesture recognition. The first is using a more robust background removal method, as mentioned in [54], common steps for moving camera background removal are: detecting camera motion, stabilizing video by removing camera motion, then doing usual background removal, and finally put camera motion back. Combining embedded sensors in Google glass,

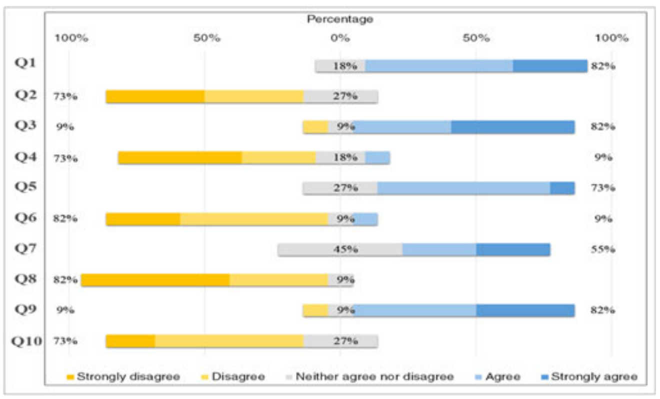


Fig. 19. The response of the participants according to System Usability Scale.

this procedure might work successfully. Second improvement can be done by combining depth cameras, z index is critical for free hand interaction since the depth of foreground hand and background environment are largely different. The importance of depth information can be seen in [55], which presents methods for object detection in RGB-D images, whose result is better than CNN in RGB images. Considering the fast pace of technology development, depth and thermal perception will soon be integrated on wearable devices, and would provide more convenience for vision algorithms.

Moreover, Ubii is aiming to achieve smooth file transfer with less actions, therefore convenience should be placed as the first concern. Using QR code to capture information about devices and affordances in this sense is a shortcoming and worth making effort to overcome. Whereas information about electronic devices can be send out by themselves as signals, how to capture position and other information about affordance like physical surface without attached objects like QR codes should be further investigated.

Furthermore, communication can be extended to more digital devices and physical objects given the QR code problem solved. With the popularizing of AR headset such as Google Glass, transferring files between computers, tablets, mobile devices using bare hand would be a promising trend. This new interaction approach can also applied to more environmental settings in the industry and used to building a larger connected Internet-of-Things.

6 Conclusion

Ubii provides a decentralized interface system for manipulation between the physical and digital worlds. User interface and interaction are precisely designed to enable freehand user experience. The system is evaluated from both aspects of technical and user experience. By interacting with objects at a high-level of physical affordance rather than underlying digital counterparts, the system helps users perform operations in much convenient and natural way. It has been proven to be useful for simple and frequently-performed manipulations with connected digital devices in closed environments.

Acknowledgments

This research has been supported, in part, by General Research Fund 26211515 from the Research Grants Council of Hong Kong, and Innovation and Technology Fund ITS/369/14FP from the Hong Kong Innovation and Technology Commission.

References

[1] B. Ullmer and H. Ishii, "Emerging frameworks for tangible user interfaces," *IBM Syst. J.*, vol. 39, no. 3, pp. 579–601, 2000.

[2] H. Ishii and B. Ullmer, "Tangible bits: Towards seamless interfaces between people, bits and atoms," in *Proc. ACM SIGCHI Conf. Human Factors Comput. Syst.*, 1997, pp. 234–241.

[3] S. J. Henderson and S. Feiner, "Opportunistic controls: Leveraging natural affordances as tangible user interfaces for augmented reality," in *Proc. ACM Symp. Virtual Reality Softw. Technol.*, 2008, pp. 211–218.

[4] S. Boring, D. Baur, A. Butz, S. Gustafson, and P. Baudisch, "Touch projector: Mobile interaction through video," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2010, pp. 2287–2296.

[5] S. G. Gustafson, B. Rabe, and P. M. Baudisch, "Understanding palm-based imaginary interfaces: The role of visual and tactile cues when browsing," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2013, pp. 889–898.

[6] J. Gibson, *Ecological Approach Visual Perception. Resources Ecological Psychology*. Mahwah, NJ, USA: Lawrence Erlbaum Associates, 1986.

[7] D. A. Norman, *Des. Everyday Things*. New York City, NY, USA: Doubleday, 1988.

[8] Z. Huang, P. Hui, C. Peylo, and D. Chatzopoulos, "Mobile augmented reality survey: A bottom-up approach," Hong Kong Univ. Sci. Technol., Tech. Rep. 20130903-182109-4052, 2013.

[9] S. J. Henderson and S. K. Feiner, "Augmented reality in the psychomotor phase of a procedural task," in *Proc. 10th IEEE Int. Symp. Mixed Augmented Reality*, 2011, pp. 191–200.

[10] D. Schmalstieg and D.Wagner, "Experiences with handheld augmented reality," in *Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality*, 2007, pp. 3–18.

[11] A. Marzo and O. Ardaiz, "Collart: A tool for creating 3d photo collages using mobile augmented reality," in *Proc. 21st ACM Int. Conf. Multimedia*, 2013, pp. 585–588.

[12] T. Ni, D. A. Bowman, C. North, and R. P. McMahan, "Design and evaluation of freehand menu selection interfaces using tilt and pinch gestures," *Int. J. Human-Comput. Studies*, vol. 69, no. 9, pp. 551–562, 2011.

[13] Z. Huang, W. Li, and P. Hui, "UBII: Towards seamless interaction between digital and physical worlds," in *Proc. 23rd Annu. ACM Conf. Multimedia*, 2015, pp. 341–350.

[14] B. Johanson, G. Hutchins, T. Winograd, and M. Stone, "Pointright: Experience with flexible input redirection in interactive workspaces," in *Proc. 15th Annu. ACM Symp. User Interface Softw. Technol.*, 2002, pp. 227–234.

[15] M. A. Nacenta, S. Sallam, B. Champoux, S. Subramanian, and C. Gutwin, "Perspective cursor: Perspective-based interaction for multi-display environments," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2006, pp. 289–298.

[16] A. Wilson and S. Shafer, "Xwand: Ui for intelligent spaces," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2003, pp. 545–552.

[17] B. Myers, R. Bhatnagar, J. Nichols, C. H. Peck, D. Miller, and A. C. Long, "Interacting at a distance: Measuring the performance of laser pointers and other devices," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2002, pp. 33–40.

[18] R. Ballagas, J. Borchers, M. Rohs, and J. Sheridan, "The smart phone: A ubiquitous input device," *IEEE Pervasive Comput.*, vol. 5, no. 1, pp. 70–77, Jan./Mar. 2006.

[19] R. Ballagas, M. Rohs, and J. G. Sheridan, "Sweep and point and shoot: Phonecam-based interactions for large public displays," in *Proc. CHI Extended Abstracts Human Factors Comput. Syst.*, 2005, pp. 1200–1203.

[20] S. Boring, M. Altendorfer, G. Broll, O. Hilliges, and A. Butz, "Shoot and copy: Phonecam-based information transfer from public displays onto mobile phones," in *Proc. 4th Int. Conf. Mobile Technol. Appl. Syst. 1st Int. Symp. Comput. Human Interaction Mobile Technol.*, 2007, pp. 24–31.

[21] T. H. Chang and Y. Li, "Deep shot: A framework for migrating tasks across devices using mobile phone cameras," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2011, pp. 2163–2172.

[22] D. Baur, S. Boring, and S. Feiner, "Virtual projection: Exploring optical projection as a metaphor for multi-device interaction," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2012, pp. 1693–1702.

[23] M. A. Nacenta, C. Gutwin, D. Aliakseyeu, and S. Subramanian, "There and back again: Cross-display object movement in multi-display environments," *J. Human-Comput. Interaction*, vol. 24, no. 1, pp. 170–229, 2009.

[24] W. Hurst and J. Dekker, "Tracking-based interaction for object creation in mobile augmented reality," in *Proc. 21st ACM Int. Conf. Multimedia*, 2013, pp. 93–101.

[25] T. Ni, D. Bowman, and C. North, "Airstroke: Bringing unistroke text entry to freehand gesture interfaces," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2011, pp. 2473–2476.

[26] F. Guimbretiere and C. Nguyen, "Bimanual marking menu for near surface interactions," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2012, pp. 825–828.

[27] P. Song, W. B. Goh, W. Hutama, C.-W. Fu, and X. Liu, "A handle bar metaphor for virtual object manipulation with mid-air interaction," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2012, pp. 1297–1306.

[28] A. D. Wilson, "Robust computer vision-based detection of pinching for one and two-handed gesture input," in *Proc. 19th Annu. ACM Symp. User Interface Softw. Technol.*, 2006, pp. 255–258.

[29] H. Benko, “Beyond flat surface computing: Shallenges of depth-aware and curved interfaces,” in *Proc. 17th ACM Int. Conf. Multimedia*, 2009, pp. 935–944.

[30] G. Ren and E. O. Neill, “3D selection with freehand gesture,” *Comput. Graph.*, vol. 37, no. 3, pp. 101–120, 2013.

[31] J. Liang and M. Green, “A highly interactive 3D modeling system,” *Comput. Graph.*, vol. 18, no. 4, pp. 499–506, 1994.

[32] M. Rahman, S. Gustafson, P. Irani, and S. Subramanian, “Tilt techniques: Investigating the dexterity of wrist-based input,” in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2009, pp. 1943–1952.

[33] Z. Zalavari and M. Gervautz, “The personal interaction panel: A two-handed interface for augmented reality,” *Comput. Graph. Forum*, vol. 16, no. 3, pp. 335–346, 1997.

[34] I. Poupyrev, D. S. Tan, M. Billinghurst, H. Kato, H. Regenbrecht, and N. Tetsutani, “Developing a generic augmented-reality interface,” *IEEE Comput.*, vol. 35, no. 3, pp. 44–50, Mar. 2002.

[35] V. Heun, S. Kasahara, and P. Maes, “Smarter objects: Using ar technology to program physical objects and their interactions,” in *Proc. CHI Extended Abstracts Human Factors Comput. Syst.*, 2013, pp. 2817–2818.

[36] J. Rekimoto, “Pick-and-drop: A direct manipulation technique for multiple computer environments,” in *Proc. 10th Annu. ACM Symp. User Interface Softw. Technol.*, 1997, pp. 31–39.

[37] F. Brudy, “Interactive menus in augmented reality environments,” Media Inform. Univ. Munich, p. 1, 2013.

[38] R. Dachselt and A. Hubner, “Virtual environments: Three-dimensional menus: A survey and taxonomy,” *Comput. Graph. Archive*, vol. 31, no. 1, pp. 53–65, 2007.

[39] J. Callahan, D. Hopkins, M. Weiser, and B. Shneidermann, “An empirical comparison of pie vs. linear menus,” in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 1988, pp. 95–100.

[40] Y. Wang and C. MacKenzie, “The role of contextual haptic and visual constraints on object manipulation in virtual environments,” in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2000, pp. 532–539.

[41] A. Kanezaki, Y. Kuniyoshi, and T. Harada, “Weakly-supervised multi-class object detection using multi-type 3d features,” in *Proc. 21st ACM Int. Conf. Multimedia*, 2013, pp. 605–608.

[42] U. Neumann and S. Y. You, “Natural feature tracking for augmented reality,” *IEEE Trans. Multimedia*, vol. 1, no. 1, pp. 53–64, Mar. 1999.

[43] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

[44] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, “Visual categorization with bags of keypoints,” *Workshop statistical learning comput. vis.*, vol. 1, no. 1-22, pp. 1–2, 2004.

[45] S. Andresen. Hand tracking and recognition with opencv, (2013). [Online]. Available: http://simena86.github.io/blog/2013/08/12/hand-tracking-and-recognition-with-opencv/, Accessed: 2015-08-30.

[46] B. Zarit and B. Super, “Comparison of five color models in skin pixel classification,” *Proc. ICCV Int. Workshop Recognition, Anal Tracking Faces Gestures Real-Time Syst.*, 1999, pp. 58–63.

[47] N. H. Dardas and N. D. Georganas, “Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques,” *IEEE Trans. Instrum. Meas.*, vol. 60, no. 11, pp. 3592–3607, Nov. 2011.

[48] S. Lakshmi and V. Sankaranarayanan, “A robust background removal algorithms,” *Int. J. Netw. Security Appl.*, vol. 5, no. 2, p. 93, 2013.

[49] M. Fatma and J. Sharma, “Leukemia image segmentation using k-means clustering and HSI color image segmentation,” *Int. J. Comput. Appl.*, vol. 94, no. 12, 2014.

[50] K. Sobottka and I. Pitas, “Face localization and facial feature extraction based on shape and color information,” in *Proc. IEEE Int. Conf. Image Process.*, 1996, pp. 483–486.

[51] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, “Maui: Making smartphones last longer with code offload,” in *Proc. 8th Int. Conf. Mobile Syst., Appl., Serv.*, 2010, pp. 49–62.

[52] J. Brooke et al., “Sus-a quick and dirty usability scale,” *Usability Eval. Ind.*, vol. 189, no. 194, pp. 4–7, 1996.

[53] A. Bangor, P. Kortum, and J. Miller, “Determining what individual SUS scores mean: Adding an adjective rating scale,” *J. Usability Studies*, vol. 4, no. 3, pp. 114–123, 2009.

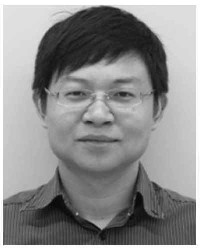
[54] L. W. Kheng, Background removal [Online]. Available: http://www.comp.nus.edu.sg/ cs4243/lecture/removebg.pdf, Accessed: 2015-10-28.

[55] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, “Learning rich features from RGB-D images for object detection and segmentation,” in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 345–360.

Sikun Lin received the BSc degree in both physics and math with first class honors from the Hong Kong University of Science and Technology. She joined the HKUST-DT System and Media Lab as a research assistant, and is currently a MPhil student supervised by Dr. Pan Hui. Her research interests include human-computer interaction, computer vision, and augmented reality.

Hao Fei Cheng received the BEng degree in computer science, and the BBA degree in general business management, both with first class honors from the Hong Kong University of Science and Technology. He joined the HKUST-DT System and Media Lab as a research assistant and is advised by Dr. Pan Hui. His research interests include human-computer interaction, augmented reality, and social networks.

Weikai Li received the bachelor's degree from Beihang University. His is currently a research assistant in HKUST-DT System and Media Lab at Hong Kong University of Science and Technology. His research interests include human-computer interaction and mobile augmented reality.



Zhenpeng Huang received the bachelor and doctorate degrees both from Beihang University. His is currently a postdoctoral in HKUST-DT System and Media Lab at Hong Kong University of Science and Technology. His research interests includes human-computer interaction, mobile augmented reality, virtual reality, and physically-based simulation.

Pan Hui received the PhD degree from the Computer Laboratory, University of Cambridge, and received the MPhil and BEng degrees both from the University of Hong Kong. He is currently a faculty member of the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, where he directs the HKUST-DT System and Media Lab. He also serves as a Distinguished Scientist of Telekom Innovation Laboratories (T-labs), Germany, and an adjunct professor at Aalto University, Finland. He is a senior member of the IEEE, and an associate editor for both the *IEEE Transactions on Mobile Computing* and *IEEE Transactions on Cloud Computing*.

Christoph Peylo studied computer science, computational linguistics, and artificial intelligence at the University of Osnabrück and received the PhD degree from the University of Osnabrück in the field of AI. He has worked at T-Labs since September 2008. He is the vice president of the Telekom Innovation Laboratories in Berlin. He is responsible for the field of cross-domain middleware. He manages activities in sensor networks and backhaul of sensor data by mobile networks.