

Design and Implementation of Procedural Self-Instructional Contents and Application on Smart Glasses

Hyoseok Yoon*, Seong Beom Kim, Nahyun Kim

Abstract

Instructional contents are used to demonstrate a technical process to teach and walkthrough certain procedures to carry out a task. This type of informational content is widely used for teaching and lectures in form of tutorial videos and training videos. Since there are questions and uncertainties for what could be the killer application for the novel wearables, we propose a self-instruction training application on a smart glass to utilize already-available instruction videos as well as public open data in creative ways. We design and implement a prototype application to help users train by wearing smart glasses specifically designed for two concrete and hand-constrained use cases where the user's hands need to be free to operate. To increase the efficiency and feasibility of the self-instruction training, we contribute to the development of a wearable killer application by integrating a voice-based user interface using speech recognizer, public open data APIs, and timestamp-based procedural content navigation structure into our proof-of-concept application.

Key Words: Self-instruction, Smart glasses, Training, Wearables.

I. INTRODUCTION

Recently, different form factors of wearable computing have been commercialized with the introduction of products such as Google Glass Enterprise Edition 2, Microsoft HoloLens 2, and Facebook Oculus Quest 2. These smart wearable glasses are demonstrating great potential for extended reality (XR) applications covering various configurations of augmented reality (AR), virtual reality (VR), and mixed reality (MR). On these smart wearable glasses, a different set of interaction method is required. Lee and Hui surveyed various possible interaction methods for smart glasses [1]. Among the classified interaction methods, touchless inputs, hands-free interaction, and voice recognition interaction methods are useful in XR training applications where users' hands are preoccupied to carry out tasks. Moreover, we believe that users in XR training can be supplied with rich, high-quality, pre-defined, and external data sources such as YouTube videos and open public data API (Application Programming Interface), beyond consuming simple audio contents [2].

Our contributions in this paper are as follows.

- We propose a general framework for utilizing self-instruction contents hands-free on the smart wearable glasses.
- We introduce a content preparation method using structured procedural timestamps with YouTube videos and open public data API.
- We demonstrate feasibility of our approach by identifying several use case scenarios and implementing a proof-of-concept application on the smart wearable glasses.

II. RELATED WORK

Fiorella and Mayer refer to instruction video as "a video lesson that is intended to help people learn targeted material" [3]. In [3], learning outcomes with instructional video can be improved by segmenting the video (i.e., "breaking the video into parts and allowing students to control the pace of the presentation" [3]) and providing mixed perspective [3]. Similarly, the effectiveness of a video tutorial has been reported [4]. Smart glasses have been used for self-training

Manuscript received December 06, 2021; Revised December 16, 2021; Accepted December 21, 2021. (ID No. JMIS-21M-12-044)

Corresponding Author (*): Hyoseok Yoon, 137 Hanshindaegil, Osan-si, Gyeonggi-do, 18101 Korea, +82-31-379-0645, hyoon@hs.ac.kr.

Division of Computer Engineering, Hanshin University, Osan-si, Gyeonggi-do, Korea, hyoon@hs.ac.kr, sbe3203@hs.ac.kr, rlaskgus03@hs.ac.kr

using Vuzix Blade [5] and between a group of people using Google Glass Enterprise Edition 2 [6].

In comparison to related works, we attempt to design and implement a self-instruction training application that uses rich, high-quality, pre-defined, and external data sources such as YouTube videos and open public data API. Remote collaboration or telementoring [7], [8] scenarios involve human mentors explicitly providing feedback and instructions, instead we integrate external data sources as self-sufficient instruction and training materials.

III. SELF-INSTRUCTION TRAINING SCENARIOS

Our paper is organized as shown in Fig. 1. In this section, we first illustrate two motivation scenarios where a wearable device (i.e., smart glass) can be useful. These two concrete scenarios are used to elicit application requirements in Section IV. Using the elicited requirements, we propose and design a smart glass application based on video contents, public API, and procedural timestamp structure. Then, we show a proof-of-concept (PoC) implementation by developing a series of processors for data source, metadata, speech recognition, and user interface (UI) events.

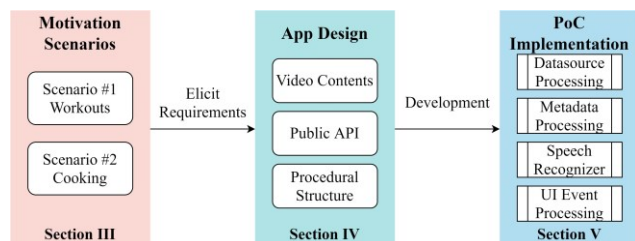


Fig. 1. Presented contents organization and its relationship among the subsequent sections.

We target self-instruction training scenarios for workouts and cooking where the user's hands are already pre-occupied as shown in Fig. 2, Fig. 3, and Fig. 4. Compared to the mobile device platform (i.e., a user watching YouTube video while holding a smartphone), users using our application would perform hands-free interaction while watching instructions on the display of the smart wearable glass.

3.1. Workout Scenario

Our workout scenarios involve two sub-use cases. The first use case is when the user performs a workout outside or without full-length mirrors. In this case, the users do not observe themselves performing actual workout as shown in Fig. 2. The second use case is when performing a full-body exercise or training that the users need to evaluate. In this

case, the user continuously checks, assesses, and corrects their physical movements if needed as shown in Fig. 3.

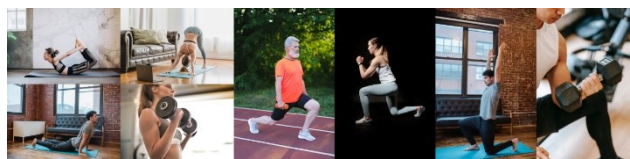


Fig. 2. Self-instruction scenarios for simple and outside workouts.

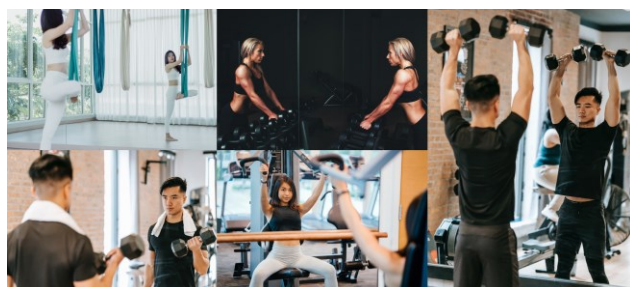


Fig. 3. Self-instruction scenarios for workouts using full-length mirrors.

Similar concepts of smart mirrors have been introduced [9][10] for training and healthcare purposes, but our application can entirely run on the smart wearable glass using a typical full-length mirror. For example, there are non-wearable approaches. FitMirror [11] is a smart mirror system that analyzes users with a Microsoft Kinect attached to the mirror without any other display device. Virtual fitness options have three categories of online fitness classes, personal training, and equipment-based training as identified in [12]. Our target category is personal training to provide “an experience similar to a one-on-one personal training session” [12] unobtrusively.

3.2. Cooking Scenario

Another popular “how-to” video category is cooking. Cooking requires the user to follow instructions on a video. This scenario would be best suited, if the user's hands are free to use cooking utensils and ingredients as illustrated in Fig. 4. In this case, the users can follow instructions displayed on the display of the smart wearable glass while freely operating their hands. A similar approach was proposed by Papadaki et al. where recipe steps were presented sequentially as regular text or symbolic language for children with cognitive impairments [13].



Fig. 4. Self-instruction scenarios for cooking.

IV. SELF-INSTRUCTION TRAINING APPLICATION ON WEARABLES

There are three major components in our proposed self-instruction training application on wearables. The three major components are voice-based user interface (UI), instructional contents as data sources, and self-instruction presentation via Google Glass Enterprise Edition 2 (GG EE2). Fig. 5 represents the proposed general framework for a wearable self-instruction application.



Fig. 5. The proposed general framework for a wearable self-instruction application.

4.1. Voice-based User Interface

Since the user's hands are preoccupied, we used voice-based UI for interacting with the self-instruction training application on wearables. Voice commands are adopted in several commercial smart glasses such as Google Glass and Microsoft HoloLens. Voice-based UI is problematic in noisy environments, so a noise-free environment is preferred. Also, voice-based UI is suitable for personal use rather than for the shared environment. Lastly, to avoid accidental activation, an accurate voice recognizer is required with a small set of pre-defined commands. Voice-based UI is used as an input modality as well as an output modality. For example, text instructions are narrated by text-to-speech (TTS).

4.2. Instructional Contents as Data Sources

To provide self-instruction, we integrate two types of data sources. The first type is instructional videos found on YouTube. Instructional videos are used to demonstrate a technical process by showing how to do or use something. They are often used for teaching and lectures in form of tutorial and training videos. The second type is public data provided via open APIs. For example, there are open APIs for retrieving cooking recipes and steps in texts (i.e., Korean food and recipe information from National Institute of Agricultural Sciences).

4.3. Self-Instruction Presentation

Based on the data source type, self-instruction presentations can take two different formats. If YouTube instructional videos are used as the data source, then the

video is played on the smart glass's display while the user can navigate between instruction steps which are pre-marked by a unique timestamp. If public open data is used as the data source, then the retrieved cooking recipes and steps can be presented as a slide presentation with short keywords, pictures, and TTS-generated narratives. A similar approach was employed in the work of Kim et al. [14], where graphical guidance "composed of a sequence of images that show tasks in nursing skills" [14]. In this work, a physical touchpad was used to navigate between instructions constraining the user's hands.

V. PROTOTYPE IMPLEMENTATION

We implemented a prototype of the self-instruction training application as a smartphone application using Samsung Galaxy A31 (SM-A315N) (Fig. 6) and an application on an Android Virtual Device (AVD) emulator using GG EE2 hardware profile (imported hardware profile from https://github.com/googlesamples/glass-enterprise-samples/blob/master/HardwareProfile/glass_ee2_hardware_profile.xml) (Fig. 7), respectively.

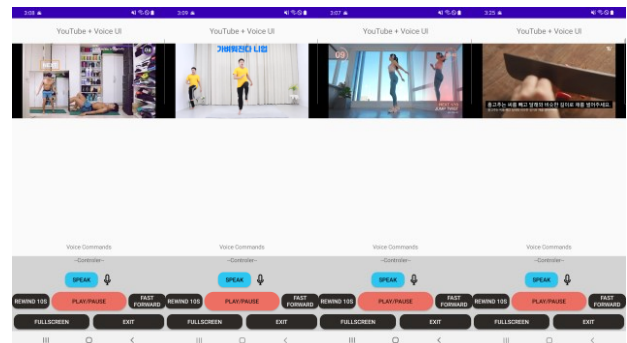


Fig. 6. A prototype application on an Android smartphone.

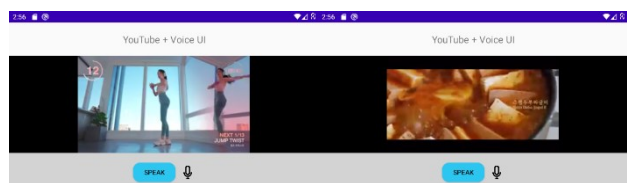


Fig. 7. A prototype application on a GG E2 AVD emulator (640 x 360 resolution display).

Two versions of our application differ in the provided user interface. The smartphone applications also provides several buttons (i.e., touch-based UI) to control the video sources, while the smart glass version provides a simple UI that listens for the user's voice commands. The smart glass version was designed and implemented targeting Google Glass Enterprise Edition 2, but also can be run on other Android-based smart glasses. For the increased compatibility on Android development, we used an AVD

with 640 x 480 resolution and landscape orientation using API level 29 (Android 10.0) as an SDK as shown in Fig. 8.

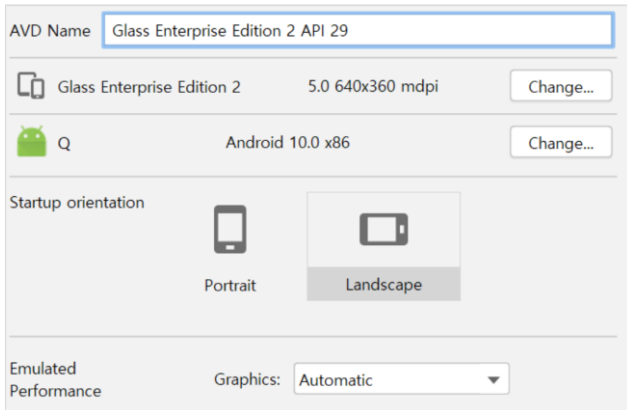


Fig. 8. The used GG E2 AVD emulator configuration.

More specifically, we developed our applications using Android Studio (Arctic Fox, 2020.3.1 Patch 2) as an IDE with Java programming language. To playback YouTube videos, we used YouTube Android Player API library (<https://developers.google.com/youtube/android/player>). This library requires a developer key (i.e., API key) and video id (i.e., YouTube video id) to play YouTube videos on the Android platform. A code snippet using this library with the provided API key and video id is shown in Fig. 9.

```

listener = new YouTubePlayer.OnInitializedListener() {
    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider,
        YouTubePlayer youTubePlayer, boolean b) {
        player = youTubePlayer;
        youTubePlayer.loadVideo(Video_key);
    }
    @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider,
        YouTubeInitializationResult youTubeInitializationResult) {
        // Video loading fails
    }
};
youTubePlayerView.initialize(API_KEY, listener);

```

Fig. 9. YouTube Android Player API using API key and video id.

5.1. Voice-based User Interface

Since the user's hands are preoccupied, we used voice-based UI for interacting with the self-instruction training application on wearables. We used Android API (`android.speech.SpeechRecognizer`) to implement voice recognition for voice commands. Table 1 shows voice commands for controlling instruction videos. If no procedural timestamps are provided, the rewind and fast forwards skips 10 seconds backward and forward.

Fig. 10 shows a code snippet for processing voice commands to its equivalent video control commands. A user's voice command is turned into String type by using Speech-To-Text (STT) using `SpeechRecognizer` class, (<https://developer.android.com/reference/android/speech/SpeechRecognizer>). The value in this String variable is compared to find the correct command, then

`seekRelativeMillis()` method with to rewind or fast forward video with the provided relative video position using milliseconds as the unit. A similar approach was explored in CuisiNavi system [15] for the cooking guidance system, but they have explored a gesture recognizer instead of a speech recognizer. Their gesture recognition module detected predefined user activities such as cutting, peeling, and mixing with a timing management module [15].

Table 1. Voice commands for controlling instruction videos.

Commands	Korean (Default)	English
Play/Pause	“재생”, “재생해줘”	“Play”, “Pause”
Rewind	“전으로”, “뒤로”	“Rewind”
Fast Forward	“앞으로”, “조금후”	“Fast Forward”
Fullscreen	“전체화면”	“Fullscreen”
Exit	“꺼줘”, “종료”	“Exit”

```

//Rewind
for (String s : stt_result_string) {
    for (String value : before_ten_sec_speak) {
        if (value.equals(sentence) || value.equals(s)) {
            if (player != null)
                player.seekRelativeMillis(-10000);
        }
    }
}

//Fast Forward
for (String s : stt_result_string) {
    for (String value : after_ten_sec_speak) {
        if (value.equals(sentence) || value.equals(s)) {
            if (player != null)
                player.seekRelativeMillis(10000);
        }
    }
}

```

Fig. 10. A code snippet for processing voice commands.

5.2. Instructional Contents




For preparing YouTube videos as a data source, we used YouTube API. We included 3 cooking videos and 3 workout videos as shown in Table 2 and Table 3, respectively. One of the requirements for quality instructional content was concise video length. We wanted the length of the video to be not too short or long since the user has to wear a smart glass and perform certain types of activities (i.e., cooking and workout) that can easily burden users and increase physical fatigue.

From the YouTube site, each video's video IDs are extracted and used in our application to playback the videos with previously mentioned YouTube Android Play API with an API key and a video ID. Video IDs embedded in our application were VoYqcKiqqGE, p9Nn7AqoJpA, AFVw4NLHWx8, 4aI2kfahWP0, PmCBeGCGgpo, and fbYu5yzo4lc which can be also checked and played in a web browser, by adding a prefix of “<https://youtu.be/>” to

each video id.

Table 2 shows 3 instruction videos for Korean cooking recipes for Korean noodles with wild chives, ddangcho kimbap, and spam dubu jjaggul e.

Table 2. Sample cooking videos.

Instructional Videos	Length (MM:SS)
 https://youtu.be/VoYqcKiqqGE	3:15
 https://youtu.be/p9Nn7AqoJpA	3:24
 https://youtu.be/AFVw4NLHWx8	3:05

These 6 video contents are manually selected considering that the video has clear steps and procedures. For example, cooking recipes are displayed as a separate insert in cooking videos and step numbers are displayed in workout videos.

Table 3 shows workout videos for killer legs workout, workout to lose belly fat, and Tabata home training.

Table 3. Sample workout videos.

Instructional Videos	Length (MM:SS)
 https://youtu.be/4aI2kfahWP0	12:14
 https://youtu.be/PmCBBeGCGgp0	12:10







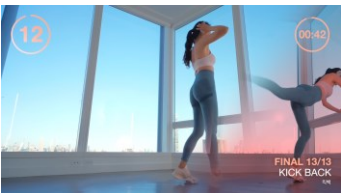
Workout videos typically run for 10+ minutes as the user performs in real-time, while cooking videos only run for 3+ minutes with the user's explicit pauses to follow the instructions. There is one big difference between these two different types of video content. To provide actual workout experience and timing, workout videos are often presented as a one-to-one trainer to trainee style. This characteristic helps even novice users to follow instructions and perform workouts in more proper ways.

Instruction videos with procedural timestamps such as videos illustrated in Table 4 and 5, can be played backwards and forwards to skip to next or previous procedure/steps.

Table 4. Instructional cooking videos with procedural timestamps.

Instructional Videos (https://youtu.be/VoYqcKiqqGE)	Timestamp (MM:SS)
 Introducing ingredients	0:13
 Preparation	0:17
 Cooking step	0:59
 Cooking step	2:04

Table 5. Instructional workout videos with procedural timestamps.

Instructional Videos (https://youtu.be/fbYu5yzo4lc)	Timestamp (MM:SS)
 Cross knee touch	1:42
 Sprinter	4:42
 Knee drive	6:13
 Inchworm	6:57
 Kick back	9:12

To integrate public data via open API, we tested with Korean Food Recipe API (<http://koreanfood.rda.go.kr/kfi/openapi/ckryService>) to retrieve cooking recipes and instructions as shown in Fig. 11. In the XML data, we parsed <ckryDetailList>, <ckryInfo>, and <ordr>. The <ordr> elements represent steps and <ckryInfo> elements represent instructions as shown in Fig. 12. By using a similar mechanism, we can use other public data via open API to request data in XML or JSON and extracting relevant elements.

Alcohol Ingredients Information 우리술 성분 정보 우리술 성분 (일반성분, 유기산, 유리산, 황기성분, 유리아미노산) 정보를 제공하며, 주종코드, 년도, 분석항목으로 조회 메뉴얼 다운로드 API 신청 신청	Food + Ingredients Info. 음식정보 + 재료정보 식단관리(메뉴넷) 음식음식코드, 음식 대분류, 음식 중분류, 음식명, 영양 정보, 재료 정보를 제공하며, 음식명으로 조회 메뉴얼 다운로드 API 신청 신청
Food + Ingredients + Image Info. 음식정보 + 재료정보 + 이미지정보 식단관리(메뉴넷) 음식음식코드, 음식 대분류, 음식 중분류, 음식명, 영양 정보, 조리(조리 순서, 조리 정보) 정보를 제공하며, 음식명, 조리방법으로 조회 메뉴얼 다운로드 API 신청 신청	Food + Ingredients + Recipe Info. 음식정보 + 재료정보 + 조리정보 식단관리(메뉴넷) 음식음식코드, 음식 대분류, 음식 중분류, 음식명, 영양 정보, 재료 정보, 조리정보를 제공하며, 음식명, 조리방법으로 조회 메뉴얼 다운로드 API 신청 신청
Food + Recipe Info. 음식정보 + 조리정보 식단관리(메뉴넷) 음식음식코드, 음식 대분류, 음식 중분류, 음식명, 영양 정보, 조리(조리 순서, 조리 정보) 정보를 제공하며, 음식명, 조리방법으로 조회 메뉴얼 다운로드 API 신청 신청	Food + Ingredients + Recipe + Image Info. 음식정보 + 재료정보 + 조리정보 + 이미지정보 식단관리(메뉴넷) 음식음식코드, 음식 대분류, 음식 중분류, 음식명, 영양 정보, 재료 정보, 조리 정보, 이미지 정보를 제공하며, 음식명, 조리방법으로 조회 메뉴얼 다운로드 API 신청 신청

Fig. 11. Open APIs provided by National Institute of Agricultural Science on Korean cooking recipes and food information (<http://koreanfood.rda.go.kr/kfi/openapi/useNewGuidance>).

```

1 <item>
2   <fdCode>D022011</fdCode>
3   <upperFdGruppNm>과자 및 빵류</upperFdGruppNm>
4   <fdGruppNm>과자류</fdGruppNm>
5   <fdNm>기장 강정</fdNm>
6   <foodWgh>675</foodWgh>
7   <foodCnt>5</foodCnt>
8   <ckryNm>기장 강정 조리법</ckryNm>
9   <ckrySumryInfo>강정용 시럽 만들기 (물엿:설탕:물=1컵:1:컵:3큰술) 기장은 튀기고
    명품은 볶아 섞기 강정 재료와 시럽을 넣고 약한 불로 버무리 강정 틀에 부어 잘 펴서
    적당한 크기로 자르기</ckrySumryInfo>
10  <ckryDetailList>
11    <ckryInfo>
12      <ordr>1</ordr>
13      <ckryInfo>강정용 시럽 만들기 (물엿:설탕:물=1컵:1:컵:3큰술)</ckryInfo>
14    </ckryInfo>
15    <ckryInfo>
16      <ordr>2</ordr>
17      <ckryInfo>기장은 튀기고 명품은 볶아 섞기</ckryInfo>
18    </ckryInfo>
19    <ckryInfo>
20      <ordr>3</ordr>
21      <ckryInfo>강정 재료와 시럽을 넣고 약한 불로 버무리 강정 틀에 부어 잘 펴서
    적당한 크기로 자르기</ckryInfo>
22    </ckryInfo>
23  </ckryDetailList>
24 </item>
    
```

Fig. 12. A sample XML data retrieved as cooking recipe and instruction information.

VI. CONCLUSION

In this paper, we designed and implemented a self-instruction training application on wearables that uses rich, high-quality, pre-defined, and external data sources such as YouTube videos and open public data API. By integrating already available external instruction contents, we believe that self-instruction training on wearables can be a feasible and useful application for everyday tasks and casual users. Current version of our prototype needs further work to be ported to the real smart glasses including GG EE2 and Realwear HMT-1. For example, since there are no YouTube App and Google Play Services, currently it is not possible to use YouTube on GG EE2. So, we need to modify the viewer to play stored local instructional videos without YouTube. Also, GG EE 2 uses different APIs for voice commands, so this module should be modified accordingly as well. In future work, we will investigate how to automatically extract and author timestamp and procedural

information from instructional videos. With these added automatic timestamp and extraction modules, instructional videos from several data sources can easily be converted to readily-available content for self-instructional application on smart glasses.

Acknowledgement

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2021R1F1A1063340). Photos used in Fig. 2, Fig. 3 and Fig. 4 are from Pexels with the free license to use (<https://www.pexels.com/license>). Sample instructional videos are from YouTube which can be found on the provided links.

REFERENCES

- [1] L.-H. Lee and P. Hui, "Interaction methods for smart glasses: a survey," *IEEE Access*, vol. 6, pp. 28712-28732, May 2018.
- [2] H. Yoon and J. Son, "Collocated wearable interaction for audio book application on smartwatch and hearables," *Journal of Multimedia Information System*, vol. 7, no. 2, pp. 107-114, Jun. 2020.
- [3] L. Fiorella and R. E. Mayer, "What works and doesn't work with instructional video," *Computers in Human Behavior*, vol. 89, pp. 465-470, Dec. 2018.
- [4] H. van der Meij, "Reviews in instructional video," *Computers & Education*, vol. 114, pp. 164-174, Nov. 2017.
- [5] S. K. Kim, Y. Lee, H. Yoon, and J. Choi, "Adaptation of extended reality smart glasses for core nursing skill training among undergraduate nursing students: usability and feasibility study," *Journal of Medical Internet Research*, vol. 23, pp. 1438-8871, Mar. 2021.
- [6] H. Yoon, S. K. Kim, Y. Lee, and J. Choi, "Google glass-supported cooperative training for health professionals: a case study based on using remote desktop virtual support," *Journal of Multidisciplinary Healthcare*, vol. 14, pp. 1451-1462, Jun. 2021.
- [7] D. T. Bui, T. Barnett, H. T. Hoang, and W. Chinthammit, "Tele-mentoring using augmented reality technology in healthcare: a systematic review," *Australasian Journal of Education Technology*, vol. 37, no. 4, pp. 68-88, May 2021.
- [8] H. Yoon, "Opportunities and challenges of smartglass-assisted interactive telementoring," *Applied System Innovation*, vol. 4, no. 3, Aug. 2021.
- [9] H. S. Park, G. A. Lee, B.-K. Seo, and M. Billingham, "User experience design for a smart-mirror-based personalized training system," *Multimedia Tools and Applications*, vol. 80, pp. 31159-31181, Dec. 2021.
- [10] R. Miotto, M. Danieleto, J. R. Scelza, B. A. Kidd, and J. T. Dudley, "Reflecting health: smart mirrors for personalized medicine," *NPJ digital medicine*, vol. 1, no. 1, Article number 62, Nov. 2018.
- [11] D. Besserer, J. Bäurle, A. Nikic, F. Honold, F. Schüssel, and M. Weber, "Fitmirror: a smart mirror for positive affect in everyday user morning routines," in *Proceedings of the workshop on multimodal analyses enabling artificial agents in human-machine interaction*, pp. 48-55, Nov. 2016.
- [12] G. T. DeSimone, "Virtual fitness: Choosing a program that is right for you," *ACSM's Health & Fitness Journal*, vol. 24, no. 4, pp. 3-4, Jul./Aug. 2020.
- [13] E. Papadaki, S. Ntoa, I. Adami, and C. Stephanidis, "Let's cook: An augmented reality system towards developing cooking skills for children with cognitive impairments," in *Proceedings of International Conference on Smart Objects and Technologies for Social Good*, pp. 237-247, Nov. 2017.
- [14] S. K. Kim, H. Yoon, C. Shin, J. Choi, and Y. Lee, "Brief paper: Design and implementation of a smart glass application for XR assisted training of core nursing skills," *Journal of Multimedia Information System*, vol. 7, no. 4, pp. 277-280, Dec. 2020.
- [15] N. Idehara and Y. Idehara, "CuisiNavi: Cooking guiding system with gesture recognition," in *Proceedings of the Virtual Reality International Conference*, pp. 1-4, Apr. 2018.

Authors



Hyoseok Yoon is an assistant professor in the Division of Computer Engineering at Hanshin University. He received his B.S. degree in Computer Science from Soongsil University in 2005. He received his M.S. and Ph.D. degrees in Information and Communication (Computer Science and Engineering) from the Gwangju Institute of Science and Technology (GIST), in 2007 and 2012, respectively. He was a researcher at the GIST Culture Technology Institute from 2012 to 2013 and was a research associate at the Korea Advanced Institute of Science and Technology (KAIST), Culture Technology Research Institute in 2014. He was a senior researcher at Korea Electronics Technology Institute (KETI) from 2014 to 2019. His research interests include ubiquitous computing (context-awareness, wearable computing) and Human-Computer Interaction (mobile and wearable UI/UX, MR/AR/VR interaction).



Seong Beom Kim is an undergraduate student in the Division of Computer Engineering at Hanshin University. In October 2020, he joined the HCI Lab at Hanshin University as an undergraduate researcher. His research interests include front-end web development, cloud computing, and Human-Computer Interaction.



Nahyun Kim is an undergraduate student in the Division of Computer Engineering at Hanshin University. In July 2021, she joined the HCI Lab at Hanshin University as an undergraduate researcher. Her research interests include assistive technology, wearable computing, and Human-Computer Interaction.