Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2016, Article ID 6254827, 14 pages
http://dx.doi.org/10.1155/2016/6254827

Research Article

Performance Analysis of Interaction between Smart Glasses and Smart Objects Using Image-Based Object Identification

Jacek Rumiński,[1] Adam Bujnowski,[1] Tomasz Kocejko,[1] Jerzy Wtorek,[1] Alexey Andrushevich,[2] Martin Biallas,[2] and Rolf Kistler[2]

[1]*Gdansk University of Technology, Narutowicza 11/12, 80-233 Gdansk, Poland*
[2]*Hochschule Luzern, iHomeLab, Technikumstrasse 21, 6048 Horw, Switzerland*

Correspondence should be addressed to Jacek Rumiński; jacek.ruminski@pg.gda.pl

Received 20 November 2015; Revised 5 February 2016; Accepted 16 February 2016

Academic Editor: MoonBae Song

We propose the use of smart glasses to collaborate with smart objects in the Internet of Things environment. Particularly we are focusing on new interaction methods and the analysis of acceptable reaction times in the process of object recognition using smart glasses. We evaluated the proposed method using user studies and experiments with three different smart glasses: Google Glass, Epson Moverio, and the developed eGlasses platform. We conclude that using the proposed method it is possible to recognize objects and process information allowing object detection below the average acceptance response times specified by almost all participants in the user study. Additionally, we showed that eye-tracking can be used for simple interaction between a user and a graphical user interface presented in the near-to-eye display.

1. Introduction

Smart glasses can be a part of Internet of Things (IoT). Particularly for intelligent buildings and homes smart glasses can play a role of the interface between a user and surrounding intelligence. In such environment a user of smart glasses can control smart objects, acquire information from smart objects, or change their configuration. However, this requires implementing a method of objects identification ("Which object am I controlling?"), generation and processing of graphical user interfaces for objects, proper data exchange protocols, and so forth. Smart objects can be recognized and identified using different methods that can be implemented using smart glasses. In this paper we would like to focus on image-based identification of objects: using markers and based on the visual appearance of objects.

When the smart object is identified the related processes can be started that (1) connect the smart glasses to the identified smart object, (2) present the GUI of the smart object, and (3) execute user's actions. We present the related IoT scenario describing image-based object identification, system architecture, objects representation methods (JSON), and high-level communication (JSON-RPC). We also present results of short user studies on acceptable response times of the object identification process and related analysis of computational performance of selected feature detection/extraction methods used for different smart glasses (Google Glass, Epson Moverio, and eGlasses).

Communication with smart objects and related architectures are subjects of many papers [1–4]. Many authors underline the need of integration of different smart objects into one network or grid. In [5] authors proposed the framework that allows users to register their own sensors into a common infrastructure and access the available resources through mobile discovery. Authors in [6] underline the reconfiguration ability of the proposed smart objects that is based on the context received from the Smart Space. Additionally, the important role of gateways and smart object identification is discussed. The role of data processing middleware based on SOA for the IoT is presented in [7]. Authors conclude that the use of middleware is a good foundation for the integration of diverse networks and better interaction among heterogeneous systems in the future, which simplifies

the complexity of the integration process. Similar role of middleware was also analyzed in [8].

Interaction with smart objects starts with the identification of the smart object. Objects are often identified using RFID-tags [9, 10] or using head-mounted sensors [11]. Some authors propose using object identification methods based on graphical features using cameras [12, 13] and smartphones [14]. Different local detectors and descriptors were used for object detection in the context of interaction with smart objects. The Attention Responsive Technology (ART) system proposed in [15] uses eye-tracking and monitors the allocation of visual attention in reference to the local environment. It allows interaction when the users gaze falls on a smart device. Authors in [12] presented another gaze-based system for home automation. They compared "direct" and "mediated" interaction solutions. The authors underlined that often it is difficult to directly control complex devices only by gazing at them. Therefore, PC-based, menu-driven software can be used for such "mediated" interaction. The menu-driven interface can be also controlled by gaze. In [14] authors propose automatic user interface generation on a handheld device using live visual object recognition. Different views (10–15 images) of objects are used in the training phase. The presented system allows differentiation between 8 categories of objects using 128-dimensional SURF descriptor. The processing performance tests run on the tablet computer (1.7 GHz, dual-core processor) showed that it takes approximately 150 ms to process a single frame (up to 7 frames/s).

Objects can be also identified using dedicated graphical markers [16, 17] or active markers [18]. Such methods were also proposed for application of smart glasses in healthcare [19].

In [20] authors proposed a solution for visual attention driven networking with smart glasses (iGaze). Using the eye-tracking the visual attention is captured and the corresponding gaze vector to the visual target is calculated. The user is asked to make a mild head gesture (e.g., head nod) as the postfix of the attention. Together with the head gesture smart glasses emit inaudible acoustic signal to adjacent devices. Nearby devices invoke the phase tracking and device direction determination modules for the estimation of the device vector. The Doppler effect is used.

Dedicated architectures were also proposed for bidirectional interaction between smartphone/smart glasses and smart objects [21].

We can assume the following categories of object suitable for interaction activities:

(i) Passive objects: objects equipped with a wireless interface providing one or two directional transmissions of data (synchronously or asynchronously); for example, turn power on/off and read a value of parameter. It is not possible to provide any parameters to the object or to query the object.

(ii) Active objects: extended passive objects with possibility of querying the object or providing some parameters (e.g., set temperature to X).

Additionally, we assume that many "not smart" objects can be transformed into smart objects when extended with additional electronics. Such objects can be additionally named augmented objects.

In this paper we would like to propose the use of smart glasses to collaborate with smart objects in the IoT environment. Additionally, we would like to focus on the analysis of acceptable reaction times of the object recognition system using smart glasses. We will evaluate this using user studies and experiments with Google Glass, Epson Moverio, and the eGlasses platform developed by us. The eGlasses platform (http://www.eglasses.eu/) is developed to provide an open platform with which developers can change some electronics, print another smart glasses cover using 3D printer, add sensors or electrodes, change display, and so forth. The current prototype uses OMAP 4460 processor, 1024 × 768 transparent display from ELvision Company, 5 MPx cameras, different sensors, and extension slots. The Android 4.1 OS and Linux Ubuntu OS can be used for experiments.

2. Methods

2.1. System Architecture and Proposed Interaction Methods. Theoretically, smart glasses can directly discover smart objects in the local environment (e.g., UPnP multicast in local network). However, this assumes specific implementation constraints for (different) smart glasses (e.g., wireless interface and protocols). Therefore, gateways or bridges are often proposed for IoT environments. Also in this paper we propose that architecture consisted of smart glasses, a bridge, and smart objects. In this IoT scenario, smart objects are connected to a bridge (gateway) in the local area network using different interfaces (e.g., WiFi, ZigBee and Bluetooth, and USB). Each smart object is represented in the form of the JSON object and the actual software library (drivers) that enables the control of the object. The library provides the protocol implementation to interact with the smart object on the low or/and high level. The properties stored in the JSON object represent information, mainly about the name/identifier of the object, the wireless interface address, the transport method (e.g., JSON-RPC over WiFi and ASCII stream over ZigBee), the physical location of the object (name and coordinates), the image icon of the object (encoded using BASE64), object detection method (e.g., VISUAL_APPERANCE_ORB, QRCODE), set of descriptors (if required, e.g., ORB descriptors encoded using BASE64), and a collection of available functions, each represented using JSON notation and JSON-RPC2 description. In the proposed system the application server is used in the bridge to process multicast discovery queries (e.g., UPnP) and to interact with smart glasses using REST-based services. For simplicity, we assume that the smart glasses, using a dedicated service, regularly scan available WiFi networks looking for known ISSD identifiers. In our experiments we used ISSD names starting with "eG" letters. When the router/access point is detected a user is notified. When connected, the smart glasses transmit the multicast query (e.g., UPnP), looking for the IoT bridge. Actually, the URL of the bridge service is retrieved. In the next step, the GET request is sent to the

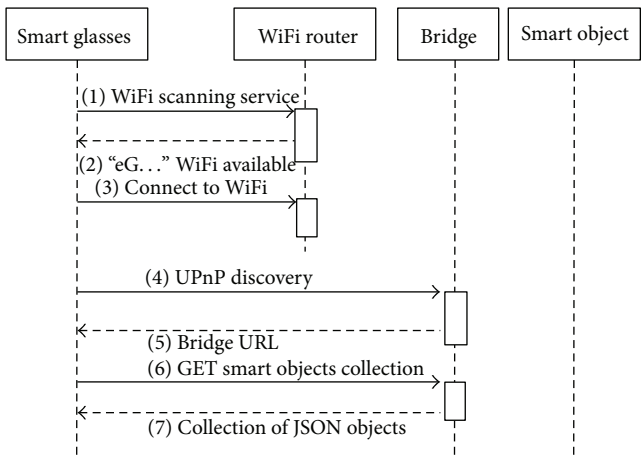


Figure 1: The initial interaction between smart glasses and the IoT bridge.

URL address of the bridge. As a result a collection of JSON objects is transmitted to the smart glasses (or error/exception is raised). Each JSON object represents a smart object. The initial interaction is presented in Figure 1.

A user is notified with the list of available smart objects presented as a list of buttons with the name of the object and its image icon. He or she can choose an object from the list (then the dedicated GUI will be generated) or can switch to the mode of the automatic identification of objects. Then the camera is started detecting either graphical markers (e.g., QR-code) or visual descriptors (see Figure 2).

The related GUI is presented to the user when the object is selected from the list or is detected and identified using the camera. Selected properties from the related JSON object (object name, image icon) are presented. Additionally, the available actions are represented by GUI widgets: a button (with the method name, e.g., POWER_ON) and additional widgets if required (e.g., ImageView for image data, e.g., frames received from the connected camera of the smart object). The action listener is automatically generated for each button. The implementation of the listener is generated based on the transport mechanism described in the JSON object. For example, if the transport mechanism is set to JSON_RPC_OVER_WIFI, the JSON RPC2 request is sent to the bridge (Figure 3). The bridge processes the request for the particular smart object executing related code.

As a result, smart glasses can receive the status of information processing (e.g., success code) and results (data). The received information is presented in the dedicated text view (e.g., Android TextView) widget. The special form of the request is the subscribe request. As a result of this request, the smart objects automatically and systematically (asynchronously) send new data when available (e.g., power consumption values from the smart power e-socket). The described procedure assumed that the remote method (the service of the smart object) does not use any parameters. However, if the method requires the parameter value, then CheckBox or EditText widgets could be generated for the GUI.

Smart glasses do not use typical data entry procedures. Some dedicated text/data entry methods can be used. Possible solutions include the application of the accelerometer [22], smart fabrics [23], or eye-tracker [24]. Other methods and comparison study were presented in [25]. In this study we verified the fundamental interaction procedure using the eye-tracker, which is a part of the eGlasses platform.

2.2. Interaction Using Eye-Tracking. The eye-tracking technology is not new; however the combination of smart glasses with eye-tracking opens new possibilities for human-system interaction. In this study we verified the possible use of eye-tracking algorithms developed earlier [26] for the interaction with graphical user interfaces for the control of smart objects.

In general the eye-tracking module is designed to operate in 1 of 3 main modes as presented in Figure 4. The primary mode is the one that enables controlling the variety of devices and applications of the eGlasses using interaction with the near-to-eye display of the eGlasses. The other two modes allow gaze tracking (scene analysis) and communication by gaze with an external computer/display.

In this study, the first mode is used. The implemented pupil tracking algorithm provides the position of the pupil center while the transformation algorithm relates this data with the graphic content displayed on the near-to-eye display (Figure 5).

The proper use of the eye-tracker requires performing the calibration procedure. The subject looks at a series of target, calibration points, while the eye-tracker records coordinates corresponding to each gaze position. The calibration points are spread at corners of the near-to-eye display. The eGlasses platform is equipped with the display of the native resolution of 1024 × 768, but it can be used with different smaller resolutions, for example, 800 × 600, 800 × 480. To eliminate the possibility that a user can omit some points during the calibration procedure the 640 × 480 calibration board is displayed in the center of the near-to-eye display as shown in Figure 6. The 640 × 480 resolution was chosen as the working resolution of the eye-scanning camera. It provides satisfactory gaze-point detection results with limited performance/power requirements [24]. After the calibration procedure, the transformation matrix is calculated in regard to the relative position of the test board's corners.

The successful calibration allows mapping of the pupil position into the near-to-eye display coordinate system. For example, it gives a user the possibility of controlling the mouse cursor by gaze [26]. The user can interact with GUI by simply looking at it. The idea of interacting with recognized object is to present graphic content (buttons, widgets) around detected objects to enable sending commands using the displayed GUI. Therefore, the interaction with smart objects requires both selection and confirmation of desired commands. In this study relating the selection of graphical component directly to the gaze position (movements) and using fixation within the region of interest (dwell time) for confirmation were decided.

2.3. Implementation. The prototype of the system for interacting with smart objects was implemented mainly using

(a) (b)

FIGURE 2: (a) The smart object can be identified using graphical markers or visual features. (b) The eGlasses platform was tested in the smart home, iHomeLab, Switzerland.

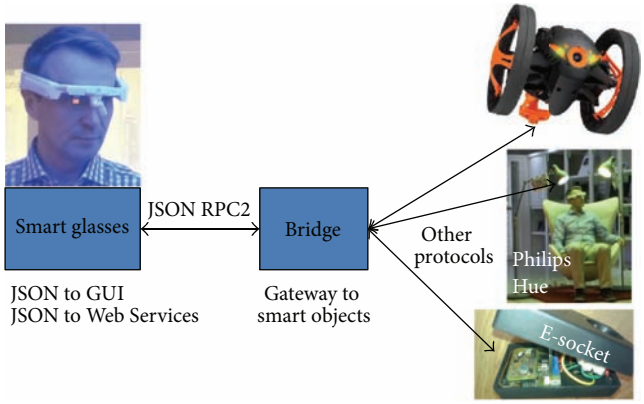


FIGURE 3: The bridge can be a middleware between smart glasses and particular smart objects.

Java programming language. The software for smart glasses was implemented using Android SDK and OpenCV (using both Java and native code by the Java Native Interface, JNI). The services of the bridge were implemented using Jetty Application Server with JSON and JSON-RPC2 libraries. Additionally, low-level software was used to process stream data from ZigBee (using ZigBee for serial conversion). A laptop computer was used as a bridge. Three categories of smart objects were tested: a controllable power socket with a ZigBee interface [21], a Philips Hue system with the propriety bridge (http://www2.meethue.com), and the Parrot Jumping Sumo robot (http://www.parrot.com/usa/products/jumping-sumo/). The bridge directly controls the power socket. For the Philips Hue system the bridge provided mapping of JSON messages between smart glasses and the Philips Hue system. For the Parrot robot the bridge delegates the connection to the robot. Smart glasses execute downloaded activity using Android Intent method. To control the robot we used libraries and codes provided by producers.

The experimental prototype of the eGlasses platform was used to perform experiments for possible interactions with near-to-eye display using the eye-tracing module (Figure 7). The eye-observing camera was located below the display and was used to track the pupil position in reference to the coordinating system of the display.

3. Experiments

Three categories of experiments were prepared. In the first one we were interested in acceptable response times between the beginning of looking at a controllable object and the point of time when the object should be identified. Therefore, we asked 22 volunteers (avg. age 36.36 ± 9.18; 11 females: avg. age 39.45 ± 7.81 and 11 males: avg. age 33.27 ± 9.74) to look through smart glasses toward three different types of objects: computer screen (working), a fan (not working), and a lamp (switched on). All objects were located at a distance 1-2 m. Each participant signaled two events: the beginning (START) of looking at the object and the expected point of time (STOP) when, according to the participant's opinion, smart glasses should have identified the object. Time period was measured using the manual stopper (by the same operator) between START/STOP commands issued by the participant. The operator response time was additionally calculated in 10 trials and averaged. It was added to the standard deviation of each measurement.

In the second group of experiments we verified the computational performance (frames per second, FPS, versus frame size) of different smart glasses to evaluate potential response time of object detection algorithms. We used Google Glass, Epson Moverio, eGlasses, and the Galaxy Note 2 smartphone as a reference. Three tests were executed for different feature detection/extraction methods (ORB/ORB, FAST/FREAK, FAST/BRIEF): (1) feature detection/extraction time using smart glasses, (2) object detection time (description, matching, and minimal distance

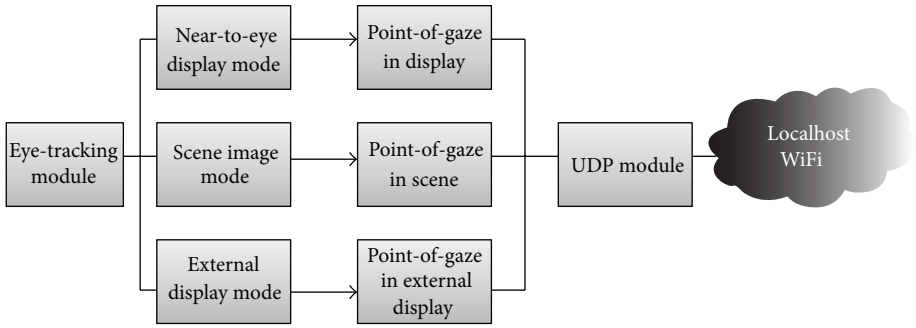


Figure 4: The main operation modes of the eye-tracking module.

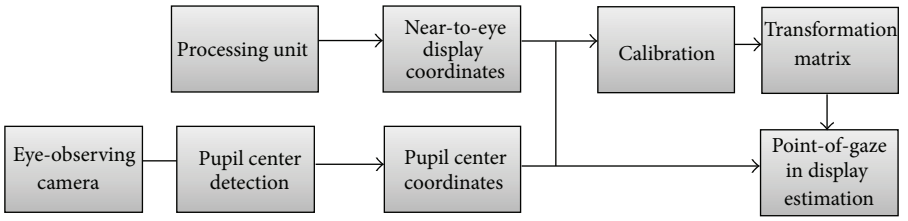
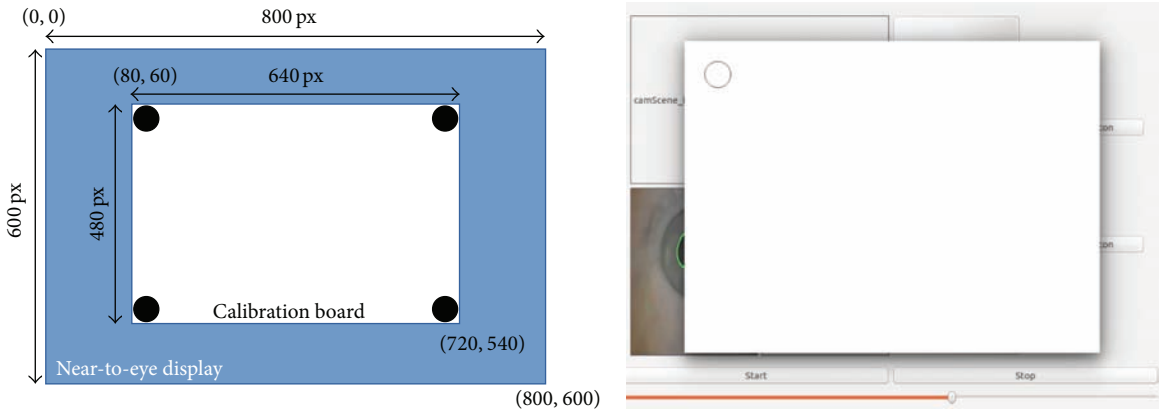
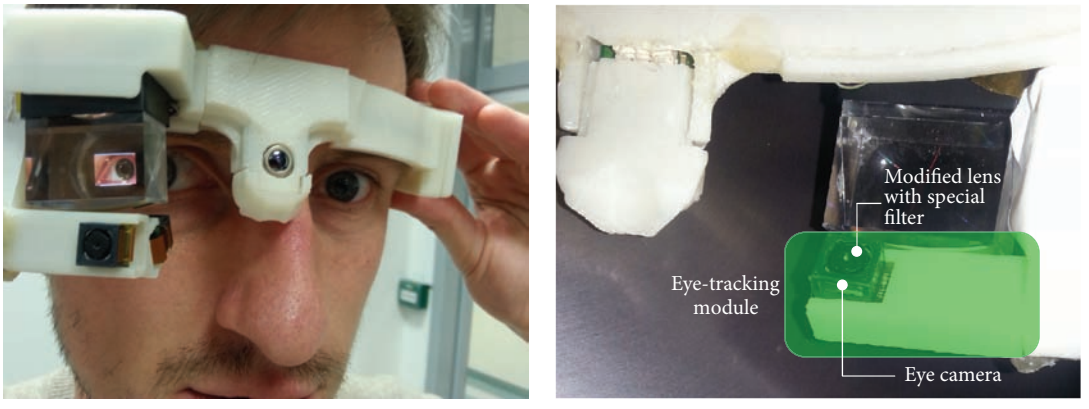


Figure 5: The point-of-gaze estimation in the near-to-eye display mode.



(a) (b)

Figure 6: Example of the calibration board displayed during the calibration procedure. (a) Basic idea, (b) practical use example: the first test point is shown.



(a) (b)

Figure 7: (a) The experimental prototype of the smart glasses with the eye-tracker. (b) The eye-observing camera located below the display.

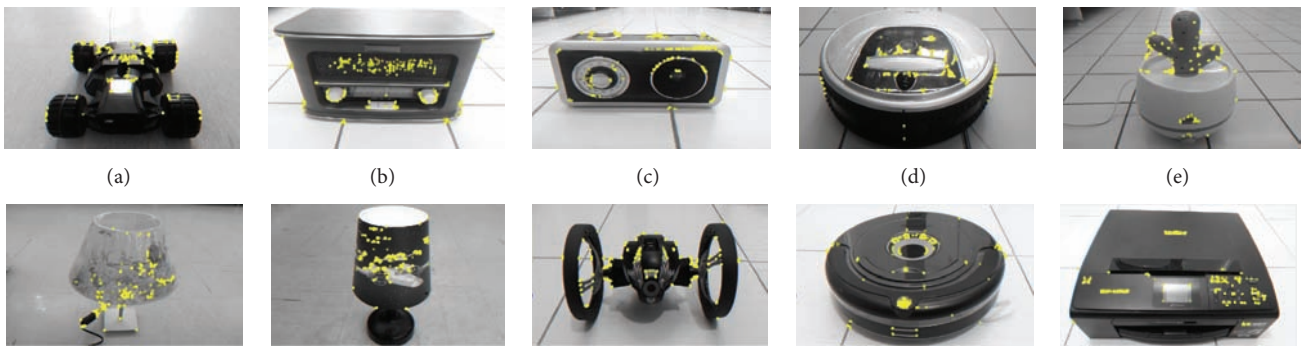


Figure 8: Smart objects images with key points of ORB/ORB descriptors ((a): SO1, SO2, SO3, SO4, and SO5; (f): SO6, SO7, SO8, SO9, and SO0).

classification for K best matches) using smart glasses, and finally (3) efficiency of data compression and transmission for object identification performed on the bridge (e.g., a laptop computer with 1.7 GHz Intel Core i7 processor, 8 GB RAM 1600 Mhz).

Experiments were performed using two sets of images, one consisting of images with the resolution of 1280 × 720 and the second with 800 × 480. Each set contained 40 images (32 resampled images from the reference set described in [27–29]) and 8 images of controllable smart objects (e.g., a lamp, a Parrot Jumping Sumo robot, and a TV decoder connected to the e-socket). Since we were focused on the time analysis the actual content of images in this experiment was not very important (except the complexity of features). Much more important was the number of descriptors used in the performance analysis. After key point detection best key points were sorted and two sets were generated with 100 or 500 best key points. For simplicity the spatial distribution of points was not analyzed. All further matching/detection experiments were performed in reference to sets about 100 descriptors and 500 descriptors generated for chosen key points. The Brute Force-Hamming method was used. The used smart glasses and the smart phone had different Camera Preview resolutions (Google Glass 640 × 360, Epson Moverio 640 × 480, eGlasses 1024 × 768, but reduced to 800 × 480 in experiments, and Samsung Galaxy Note 2 1280 × 720) limiting the size of programmatically captured video frames. Therefore, in one experiment, each device was capturing frames, but for the processing predefined images from the reference sets were used. Since the resolution and descriptor size were known it was better to compare the results.

Additionally, to show the trade-off between runtime versus accuracy of the used features, a simple experiment was designed. Ten actual and potential smart objects were used to perform real experiment: 2 driving robots (SO1, SO8), 2 lamps (SO6, SO7), 2 radio sets (SO2, SO3), 2 duster robots (SO4, SO9), a humidifier (SO5), and a home printer (SO0). In Figure 8 pictures of objects are presented with rendered locations of key points for features detected using the ORB algorithm (features were selected inside manually drawn rectangle during the preprocessing step). Images were acquired from a distance of 1 m (±0.3 m) with the resolution of 800 × 480. Descriptors were calculated using ORB, FREAK, and BRIEF algorithms. The number of descriptors for each object was as follows: (a) for the ORB algorithm: SO1-500, SO2-500, SO3-500, SO4-500, SO5-326, SO6-401, SO7-500, SO8-497, SO9-409, and SO0-500; (b) for the FREAK algorithm: SO1-477, SO2-500, SO3-485, SO4-476, SO5-264, SO6-463, SO7-443, SO8-490, SO9-328, and SO0-498; (c) for the BRIEF algorithm: SO1-467, SO2-493, SO3-500, SO4-475, SO5-283, SO6-488, SO7-465, SO8-428, SO9-361, and SO0-487.

The Android software was modified to measure detection accuracy and detection time. Only for experiments, it was assumed that when the working camera (i.e., continuously capturing frames) is targeted at the object the operator manually starts (using a tap event) the measurement. When the object is detected the current processing is paused, and the related information is presented on the microdisplay: picture of the detected smart object (to check if the object was correctly detected), time period (from the first frame captured after the user had invoked the tap event until detection), and the number of frames (from the first frame captured after the user had invoked the tap event until detection). Smart objects were located on the floor (e.g., robots) or on the table (e.g., lamps, printer). The user was sitting on the swivel chair observing objects from similar distance as used during the generation of the reference set. In this experiment we did not analyze the influence of distance or viewing angle. All previously described feature extraction algorithms were tested. First testes were used for 5 objects in the reference set (SO1–SO5) and then for 10 objects in the reference set. For the ORB algorithm, tests were also executed for 20 (10 SO + 10 other) and 40 (10 SO + 30 other) reference pictures. Additional pictures (other) were taken in a room (books on the shelf, etc.) selecting those with 500 features. The pictures were added to simulate larger reference set for performance tests. In these tests the false positive result was defined as the wrong object detected within 10s. The false negative result was defined as a lack of proper object detection within 10 s. Five detection attempts were used (325 all together) for each method (ORB, FREAK, and BRIEF), for

(a) (b) (c)

FIGURE 9: Different layouts of the GUI used in experiments. The active areas are marked using yellow color: (a) D_1 design, active fields located on the left side of GUI, (b) D_2 design containing elements on the left and right side, and (c) D_3 design containing element only on the right side of the GUI.

(a) (b)

FIGURE 10: (a) An example of the object in typical user settings. (b) An example of the GUI for the power socket control.

each object (SO1–SO0), and for each configuration (O = 5, O = 10, etc.).

The quantitative analysis of compression influence on the quality of features was also performed. Pictures of SO1–SO5 from the reference sets were used to produce compressed version of the original pictures using 4 different compression/quality factors: 90%, 80%, 70%, and 10%. The *JPEGImageWriteParam* class from the Java NIO package was used to specify these compression factors in the specially prepared software. After the generation of the compressed images features were extracted and compared. The threshold for distance measure between descriptors was set to very low value (<3). Then the number of features with distances lower that the threshold was analyzed. Additionally, the sum of distances of the 10 best features (distances sorted in descending order) was calculated.

In the third group of experiments, we verified the use of eye-tracking module for the needs of interaction between a user and the GUI presented on the near-to-eye display.

The GUI for tests was constructed using 16 regularly distributed widgets. Each widget was designed to register and notify a user of its current state. It included the "onGazeOver" state (state 1) that corresponds to the event, when gaze is focused on the current widget and the "onClick" state (state 2). The "onClick" event is fired when the "onGazeOver" state lasted at least T milliseconds (dwell time) for the particular widget. During the experiment three basic GUI layouts were tested (Figure 9). Designs correspond to the possible location of buttons/hot areas displayed around the recognized object.

Our intention was to evaluate these designs in terms of the rate of correct selections of GUI elements and to check the optimal dwell time. To do so, the software randomly highlighted one of the active areas. The user's task was to select and confirm this element by gaze. The new area was highlighted every time the confirmation signal was received whether the user focused on the right element or not. The test was finished after receiving 100 confirmation signals. Three different dwell times were tested (500 ms, 1000 ms, and 1500 ms) for each prepared layout. Each test was repeated five times preceded every time by the calibration procedure. Three parameters were recorded during each event: the selected element's name, the ID, and the result of confirmation (HIT or MISS).

4. Results

The implemented prototype of the system was verified using experiments in smart home laboratories: iHomeLab in Switzerland (Figure 2(b)) and the AAL laboratory in Poland.

In Figure 10(a) an example of the object in typical user settings is presented. Another example of the graphical user interface generated for the recognized, augmented object connected to the power socket is presented in Figure 10(b). When the smart object is identified (TV decoder) the GUI is automatically presented. In this case the user can turn on or off the power and observe related power consumption parameters.

In Figure 11 example of image from the robot interface is presented. The control of the robot is performed using the accelerometer sensor of smart glasses: head up: forward; head down: backward; head left: turn left; head right: turn right; head shake: jump. Additionally, the jump control is possible using the magnetometer sensor and the magnetic

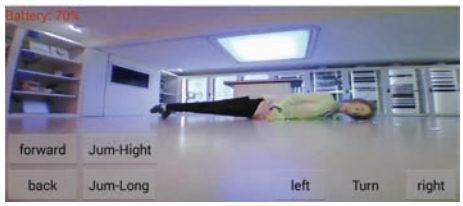


Figure 11: Remote control of the robot using smart glasses: the image is transmitted from the robot to smart glasses. The possible application includes the use of robot for remote monitoring of dangerous events, for example, fall detection.

ring. When the magnetic ring (finger/hand) is reaching the eGlasses (magnetometer) the change in the magnetic field is detected and the related event is fired. The experiments were performed for the monitoring of possible fall of the elderly person.

The control of the Philips Hue lamps was performed based on recognition of identifiers of lamps encoded in QR-codes attached to lamps.

In this paper we are mainly focusing on feasibility of image processing for object detection using smart glasses. Therefore, below, the related results are presented.

4.1. Acceptable Response Time. The results of this experiment were very interesting. The average acceptable response time was 2.62 s ± 1.6 s. The difference between men and women was observed. The result for females was 3.12 s ± 1.91 s, while for males it was 2.11 s ± 0.99 s. Particularly women at middle age (>45) accepted longer response times. Both women and men accepted longer response for computer screen, probably because of the active content. The shortest acceptable response time was 0.87 s (man, 32 y) and the longest one was 6.92 s (woman, 48 y). The average accuracy of time measurement by the operator was calculated as 0.103 s ± 0.026 s (the average response for a sequence of fast 10 START/STOP commands).

4.2. Data Processing by Smart Glasses. First the number of frames per second was calculated for each device without any frame processing tasks. The value was calculated after capturing 110 frames (10 frames skipped and then 100/time between the first and the last frame) in three trials. The results are as follows: Google Glass: 33.11 FPS (640 × 360; 7.28 Mpx/s); Epson Moverio: 25.0 FPS (640 × 480; 7.32 Mpx/s); eGlasses: 29.89 FPS (800 × 480; 10.95 Mpx/s); and Samsung Note 2: 16.6 FPS (1280 × 720; 14.59 Mpx/s).

Frame processing times for key point's detection and feature description are presented in Table 1 (simulated frame size 1280 × 720).

Of course for smaller resolution of frames performance results were better. For example, for the eGlasses, processing of images with resolution of 800 × 480 for FAST/BRIEF descriptor was performed with 15.03 FPS rate (but for ORB only 4.24 FPS). The results mentioned in Table 2 were obtained for the matching of descriptors and actual object detection. In this paper only results for eGlasses (processing using algorithms implemented using JNI) are presented later.

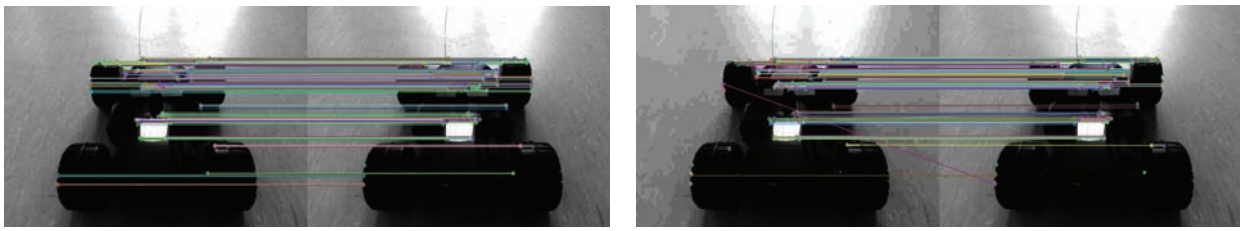
Table 1: Values of FPS for different algorithms and devices (EG: eGlasses; N2: Samsung Note 2; EM: Epson Moverio; GG: Google Glass). The set of key points was reduced to 500. Frame resolution: 1280 × 720.

Device	ORB/ORB	FAST/BRIEF	FAST/FREAK
EG/Java	1.55	8.84	6.24
EG/JNI	1.37	6.17	2.29
EM/Java	1.23	5.99	6.02
EM/JNI	1.66	7.75	2.33
GG/Java	0.30	2.37	2.27
GG/JNI	0.62	3.21	0.94
N2/Java	1.85	8.14	7.79
N2/JNI	3.02	8.26	7.00

For frame resolution of 1280 × 720 results were up to two times worse. Another method for object detection in IoT environment using smart glasses could use bridge/gateway for actual data processing. Therefore we tested the transmission rate for frames acquired by smart glasses. First, transmission speed was estimated sending data packets of different sizes during each *onFrame* event. The results were very similar for all devices: 20.63 Mbps ± 1.74 (the same router at the same distance, signal strength about −62 dBm). For very small data packets the transfer was smaller because of the limitation of frame capture rate for the particular device. In further tests 7 different data sets were processed: uncompressed YUV data (e.g., 1382400B), uncompressed Y data (921600B), GZIP compressed YUV data (737435B), GZIP compressed Y data (572727B), and JPEG compressed data with quality indicator 100%, best (572539B), 90% (148822B), and 80% (97184B). The transmission of uncompressed frame data was slow, allowing only about up to 2 FPS for NV21 YUV data (1.5 ∗ 1280 × 720 bytes) and up to 3 FPS for Y data (1280 × 720 bytes). Results were very similar for all devices as the data transfer rate depends mainly on the quality of the communication link. To obtain better processing of frames the compression must be introduced (or faster networks). However, compression is computationally expensive. For all YUV data the lossless compression (GZIP, Snappy) leads to small reducing of FPS rate. For Y data the FPS was about 4 FPS (about 1 FPS better). Results obtained using JPEG compression were much better especially for higher compression factors (90% and 80%). Of course the compression effect is related to the original content of an image. Therefore we executed tests for images from the prepared test set (known size, etc.). Test images were compressed with the quality factor of 90% (smallest PSNR for test set was 27.67 dB). This allowed obtaining the efficiency of compression and transmission about 7.43 FPS for the smart glasses and 13.18 FPS for the smartphone. Higher compression factors produce even better results (for 80%: 8.88 FPS for the smart glasses and 14.90 FPS for the smartphone). The final efficiency of object detection depends on the computational performance of the bridge/gateway. For the computer used in the study (ORB implemented in Java OpenCV using threads of the application server) the average results for 1280 × 720 test images were as follows: 6.94 ± 0.87 FPS for F = 500/O = 5; 6.69 ± 0.94 FPS for F = 500/O = 10;

Table 2: FPS for object detection using different algorithms, different number of reference objects (O), and different size of features (F) in the representation of each object (res. 800 × 480).

Method	F = 500, O = 5	F = 500, O = 10	F = 500, O = 20	F = 500, O = 40
ORB	3.36	2.84	2.29	2.34
FAST/FREAK	1.81	1.66	1.41	0.91
FAST/BRIEF	9.6	6.73	4.28	2.46
	F = 100, O = 5	F = 100, O = 10	F = 100, O = 20	F = 100, O = 40
ORB	4.23	4.27	4.06	3.89
FAST/FREAK	2.59	2.37	2.29	2.06
FAST/BRIEF	15.25	15.22	14.92	14.28



(a) (b)

Figure 12: Some examples of the matched features (distance < 6) between the compressed picture (left) and the original one (right) for two different compression factors: (a) quality 90% and (b) quality 10%.

5.36 ± 0.30 FPS for F = 500/O = 20; 3.08 ± 0.15 for F = 500/O = 40. The average size of the compressed frame was 229.81 kB ± 85.49. Of course the efficiency for many connected smart glasses would depend on the architecture of the bridge, but it was not evaluated in this work.

The results of the experiment with 10 smart objects (SO1–SO0) are presented in Table 3. In each cell of the table the calculated precision value is presented (for 5 attempts, precision = TP/(TP + FP)) and average detection time is in milliseconds.

In the entire experiment 3 FP and 1 FN results were observed for the ORB algorithm, 5 FP and 1 FN results for the BRIEF algorithm, and 6 FP and 1 FN results for the FREAK algorithm. The average number of frames to detect the object (without FP and FN results) was 1.98 ± 3.19.

In Table 4 some results of the quantitative analysis of influence of compression on the quality of features are presented.

$D_{\min}$ indicates the set of descriptors, whose distances are lower than the threshold value (equal 3). The term "max distance" used in the table means the highest value of the distance in the entire set of matched features between two images.

In general, images compressed with the 90% quality factors have about 50% of all descriptors almost identical (distance < 3) or very similar (~80% with distance < 6). For images compressed with 80% quality factor about 33% of all descriptors were almost identical with those calculated for the original image (distance < 3) or very similar (~71% with distance < 6).

In Figure 12 some examples of the matched features (distance < 6) are graphically presented between the compressed picture (left) and the original one (right) for the SO1 object. Images with two different compression factors are presented: quality 90% (Figure 12(a)) and 10% (Figure 12(b)).

4.3. Interaction Using Eye-Tracking. The possible use of the eye-tracker for the interaction with GUI of the smart glasses was verified using experiments described in Section 3. First test was performed using dwell time set for 1500 ms. The user was asked to calibrate the eye-tracker and then the first designed layout (D_1) was displayed. The active fields were located on the left side. After 100 randomly stimulated events the response results were stored. Next, the user was asked to calibrate again the device and repeat the test. Five repetitions were used for each test layout. The same procedure was repeated for the other two dwell times: 1000 ms and 500 ms. Results are presented in Tables 5, 6, and 7.

The pseudorandom generator was used to highlight widgets (user stimulation) for each test layout. In Figure 13 the distribution of generated events in tested layouts is presented. The figure presents how often each active field was highlighted for the particular layout.

5. Discussion and Conclusions

The average acceptable response time for volunteers was about 2.6 s. However, the performed experiments showed that the acceptable time differs between participants. Discussions with participants after the experiments showed that such acceptance levels might depend on the physical and mental state of a person, hour of a day, and so forth. What was interesting was that many participants underlined that

Table 3: The results of the experiment with 10 smart objects (SO1–SO0).

	SO1	SO2	SO3	SO4	SO5	SO6	SO7	SO8	SO9	SO0
FAST/FREAK O = 5	100% (526 ± 588)	100% (1391 ± 2148)	100% (307 ± 107)	100% (2757 ± 2468)	100% (507 ± 404)					
FAST/FREAK O = 10	80% (1168 ± 857)	100% (704 ± 613)	100% (424 ± 167)	60% (3525 ± 3863)	60% (1189 ± 1729)	100% (464 ± 41)	80% (814 ± 535)	100% (607 ± 494)	100% (655 ± 303)	100% (1328 ± 2201)
FAST/BRIEF O = 5	100% (129 ± 21)	100% (177 ± 51)	100% (197 ± 54)	100% (163 ± 18)	100% (174 ± 21)					
FAST/BRIEF O = 10	100% (182 ± 23)	80% (222 ± 112)	100% (300 ± 138)	100% (224 ± 85)	100% (197 ± 21)	60% (225 ± 34)	60% (182 ± 5)	100% (188 ± 35)	100% (210 ± 78)	100% (207 ± 14)
ORB/ORB O = 5	100% (249 ± 18)	100% (276 ± 26)	100% (361 ± 289)	100% (336 ± 132)	100% (234 ± 16)					
ORB/ORB O = 10	100% (284 ± 27)	100% (307 ± 38)	100% (296 ± 40)	80% (334 ± 162)	100% (301 ± 34)	80% (289 ± 15)	100% (309 ± 39)	100% (275 ± 12)	100% (275 ± 26)	100% (301 ± 31)
ORB/ORB O = 20	100% (385 ± 27)	100% (403 ± 43)	100% (401 ± 48)	100% (1341 ± 1668)	100% (467 ± 182)	100% (364 ± 12)	100% (405 ± 42)	100% (769 ± 853)	100% (588 ± 294)	100% (386 ± 21)
ORB/ORB O = 40	100% (580 ± 19)	100% (946 ± 808)	100% (1426 ± 1183)	80% (1339 ± 1656)	100% (758 ± 242)	100% (625 ± 60)	100% (568 ± 20)	100% (1838 ± 1981)	100% (552 ± 44)	100% (740 ± 244)

Table 4: Results of the quantitative analysis of influence of compression on the quality of features.

	Sum of 10 best distances	Max. distance	Size (D_{min})	File size [kB]
SO1			500	119
SO1_90 versus SO1	0	78	278	62
SO1_80 versus SO1	0	79	224	41
SO1_70 versus SO1	0	78	164	32
SO1_10 versus SO1	23	84	5	10
SO2			500	119
SO2_90 versus SO2	0	103	250	62
SO2_80 versus SO2	0	102	177	42
SO2_70 versus SO2	0	101	119	34
SO2_10 versus SO2	27	86	5	12
SO3			500	119
SO3_90 versus SO3	0	75	209	63
SO3_80 versus SO3	0	75	114	42
SO3_70 versus SO3	0	76	81	33
SO3_10 versus SO3	59	85	0	12
SO4			500	116
SO4_90 versus SO4	0	83	258	69
SO4_80 versus SO4	0	73	199	44
SO4_70 versus SO4	0	77	133	35
SO4_10 versus SO4	23	92	5	12
SO5			326	107
SO5_90 versus SO5	0	82	130	55
SO5_80 versus SO5	3	82	72	35
SO5_70 versus SO5	6	70	34	28
SO5_10 versus SO5	71	89	1	10
SO9 versus SO4	346	91	0	

Table 5: The accuracy of correctly confirmed selections obtained for dwell time = 1500 ms.

Layout/attempt	1	2	3	4	5	Mean
D_1	89%	87%	89%	88%	90%	89%
D_2	91%	90%	91%	91%	92%	91%
D_3	98%	97%	98%	98%	99%	98%

Table 6: The accuracy of correctly confirmed selections obtained for dwell time = 1000 ms.

Layout/attempt	1	2	3	4	5	Mean
D_1	88%	84%	86%	89%	87%	86.8%
D_2	88%	85%	88%	87%	88%	87.2%
D_3	94%	95%	93%	95%	94%	94.2%

they would not expect the fastest response of the system, but response time should be in a preferable range.

In the experiments with object detection algorithms standard OpenCV implementations of key points detection and feature extraction/description algorithms were used.

Table 7: The accuracy of correctly confirmed selections obtained for dwell time = 500 ms.

Layout/attempt	1	2	3	4	5	Mean
D_1	39%	43%	56%	49%	43%	46%
D_2	41%	40%	41%	56%	44%	44.4%
D_3	40%	44%	55%	56%	53%	49.6%

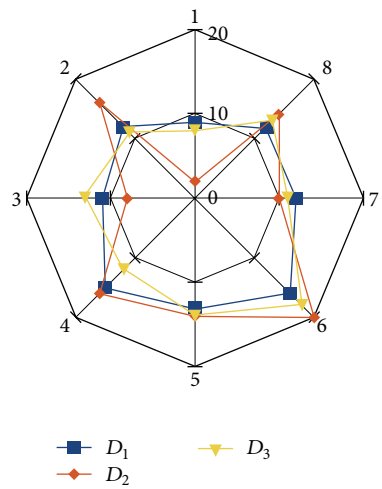


Figure 13: The distribution of randomly generated events in the tested D_1–D_3 layouts. Numbers from 1 to 8: the ID of the widget, for which the stimuli were randomly generated; total number of events: 100.

Sometimes these implementations differ from the implementations provided by authors of algorithms (e.g., therefore we did not use the BRISK algorithm).

The interaction with GUIs of smart objects presented on the smart glasses display was performed using a mouse wirelessly connected to the eGlasses. Additionally, the eye-tracker was tested.

In this study we were not primarily focused on the evaluation of the precision and recall of particular object detection algorithms. This can be found elsewhere (e.g., [28, 30]). However, we did simple experiments to verify the methodology for the optimistic assumptions: short distance to objects (up to 1.3 m), one observation angle, and similar lighting conditions. We tested the system for ten smart objects. The results showed good precision for the tested methods. The analyzed processing times of the methods were acceptable (response time < 2.6 s). The worst results were achieved for the FREAK algorithm (6 false positives, 1 false negative, and longest processing times). Excluding FP and FN cases objects were detected in first 5 frames. Only for some cases it required more frames to detect the object (even 21 frames). For all 325 trials only 3 FN cases were detected. The false negative result means that the object was not detected with 10 s. Such cases (FN or many frames to process) were sometimes related to the camera auto settings (auto zoom, auto exposure). In this experiment we used preprocessed single images of smart objects to calculate descriptors and build the reference set. However, relatively good results of

the processing times suggest that more images for each smart object could be used, especially when the number of smart objects is limited. In conclusion, when smart glasses are used for the detection and control of small number of smart objects it is possible to process frames with the rate allowing object detection below the average acceptance response times specified by almost all participants of the initial study.

In general, the results related to the detection accuracy would not be so good if we take into account different angles of observation, different distances, changing illumination, camera properties, and so forth. Those are very well-known limitations of object detection methods based on visual appearance of objects. However, this requires larger study and it will be a subject of another analysis we will perform.

For more objects to be detected (e.g., in a large building) a proper bridge/gateway can provide the service of object detection. The very important goal of this paper was to analyze the performance of two setups for object detection: by smart glasses and using external service (a gateway). Standalone smart glasses (i.e., not glasses connected by a cable to a raspberry pi or other external computer) are technologically limited because of acceptable thermal emission of the electronics, limited power capacity of batteries, and so forth. These technological features are design limitations in the choice of processors. If more processing is required (e.g., more smart objects) object detection procedure should be performed mainly using external services. However, the question is how to efficiently provide data for such external analysis. One possibility is to process each frame to detect descriptors and transfer those descriptors to the gateway for further object detection steps (e.g., feature matching). Another possibility is to transfer each frame to the gateway. In this paper we performed many experiments to analyze this problem analyzing different algorithms, different image representation methods (lossless and lossy compression), different color models (e.g., use only Y component), and so forth. It was shown that lossy compression with good quality factors (e.g., 90%) preserves about half of the features practically unchanged (distance < 3; about 80% with distance < 6) and enables fast object detection.

There are also other practical aspects important for the future use of such interaction methods. For example, if in the field of view of the camera there is more than one smart object then only one object (one nearest neighbor) will be detected. Future solutions can offer a menu with all detected objects in the current view, and the user could choose the proper one. Another possibility is to use the eye-tracker to select the detected object (e.g., select this object for which gaze is focused inside the rectangle containing the object). Similar, gaze-based procedure can be used if more than one similar object is present in the camera view (e.g., two identical lamps).

The gaze-based interface for the interaction with the near-to-eye display was tested for possible use in simple gaze-based actions. The design of the interface with the camera and the display built into the frame of smart glasses potentially excludes the negative influence of head movements. That is because the user's head and utilized hardware remain in constant geometrical relation. In this study we assumed that the smart glasses are well fixed on the head and there are no rapid movements of the head that could lead to the change of the smart glasses position in reference to the user's head. The main challenge was the accurate pupil detection. It is essential for the reliable transformation from the pupil position to accurate and precise "cursor" position on the near-to-eye display. The used eye-tracking algorithm was previously described in [26] and the study dedicated to the accuracy analysis in reference to near-to-eye display was presented in [24]. Here we analyzed the possible use of events that are related to eye movements for the interaction with a simple GUI of the smart object. Each tested layout contained eight active GUI components that cover approximately half of the available display size. Such an approach simulates the situation when a user observes the smart object and can interact with it by sending commands controlling GUI widgets/buttons displayed around or next to the object icon (or preview). Different confirmation actions are possible to model using the eye-tracking module. Some examples include "eye blink" or "gaze-fixation for T milliseconds." In this study we verified the second approach. The achieved accuracy was very high for all tested layouts for relatively long dwell times: 1.5 s and 1 s. For shorter values of dwell times results were poor. It might be explained that the very short dwell time and randomly highlighted widgets to "visually press" lead to some confusion. The user does not have time to overtake fast changes in the GUI. Additionally, short dwell times can also lead to involuntary selections (events). The proper dwell time parameter could be specified for the particular user as his/her preference and therefore it was not necessary to perform studies with many users. Such a study can be interesting for many other reasons (e.g., acceptance of the technology) and will be performed in future studies. The simultaneous generation of different events using gaze tracking and gaze-fixation can have some limits, especially for rapid actions. Therefore other sources for "confirmation events" can be potentially used, including head movements (accelerometer) and change in head muscle signals (electrodes in the inner part of smart glasses). Potentially such methods allow increasing the privacy of simple data input in comparison to audio commands or the use of traditional keyboards (that can be observed by bystanders).

In [31] authors identified two distinct styles of smart object sensing: object-centric style and human-centric style. Smart objects belonging to the object-centric type are deployed in the real world and can detect changes in their physical status or/and changes in the surrounding environment. Smart objects from the second category serve as personal companions. Smart glasses or smart watches belong to the second category of smart objects [32]. However, as shown in this paper, smart glasses can cooperate with smart objects located in the user's neighborhood. Using discovery services and the shared protocol smart glasses can potentially connect to different networks or objects, especially when the user changes his or her location. This gives many interesting opportunities in different fields of applications for smart homes, intelligent buildings, smart cities, and so forth.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work has been partially supported by NCBiR, FWF, SNSF, ANR, and FNR in the framework of the ERA-NET CHIST-ERA II, European project *eGlasses—The Interactive Eyeglasses for Mobile, Perceptual Computing*, and by Statutory Funds of Electronics, Telecommunications and Informatics Faculty, Gdansk University of Technology.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: a survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials Journal*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] Internet Architecture Board, "Architectural considerations in smart object networking," RFC 7452, IAB, 2015, https://www.iab.org/2015/03/20/rfc-7452-architectural-considerations-in-smart-object-networking/.

[3] L. Mainetti, V. Mighali, and L. Patrono, "A software architecture enabling the web of things," *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 445–454, 2015.

[4] T. Perumal, A. R. Ramli, and C. Y. Leong, "Interoperability framework for smart home systems," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 4, pp. 1607–1611, 2011.

[5] A. J. Jara, P. Lopez, D. Fernandez, J. F. Castillo, M. A. Zamora, and A. F. Skarmeta, "Mobile digcovery: discovering and interacting with the world through the internet of things," *Personal and Ubiquitous Computing*, vol. 18, no. 2, pp. 323–338, 2014.

[6] S. Bartolini, B. Milosevic, A. D'Elia, E. Farella, L. Benini, and T. S. Cinotti, "Reconfigurable natural interaction in smart environments: approach and prototype implementation," *Personal and Ubiquitous Computing*, vol. 16, no. 7, pp. 943–956, 2012.

[7] F. Wang, L. Hu, J. Zhou, and K. Zhao, "A data processing middleware based on SOA for the internet of things," *Journal of Sensors*, vol. 2015, Article ID 827045, 8 pages, 2015.

[8] D. Villa, C. Martín, F. J. Villanueva, F. Moya, and J. C. López, "A dynamically reconfigurable architecture for smart grids," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 2, pp. 411–419, 2011.

[9] Y. Jie, J. Y. Pei, L. Jun, G. Yun, and X. Wei, "Smart home system based on IOT technologies," in *Proceedings of the 5th International Conference on Computational and Information Sciences (ICCIS '13)*, pp. 1789–1791, Shiyang, China, June 2013.

[10] S. Dey, J. K. Saha, and N. C. Karmakar, "Smart sensing: chipless RFID solutions for the internet of everything," *IEEE Microwave Magazine*, vol. 16, no. 10, pp. 26–39, 2015.

[11] P. Simoens, J.-F. Van Wijmeersch, E. De Coninck, T. Ingelbinck, T. Vervust, and T. Verbelen, "Vision: smart home control with head-mounted sensors for vision and brain activity," in *Proceedings of the 5th International Workshop on Mobile Cloud Computing and Services (MCS '14)*, pp. 29–33, ACM, New York, NY, USA, June 2014.

[12] D. Bonino, E. Castellina, F. Corno et al., "A blueprint for integrated eye-controlled environments," *Universal Access in the Information Society*, vol. 8, no. 4, pp. 311–321, 2009.

[13] J. Ruminski and K. Czuszynski, "Application of smart glasses for fast and automatic color correction in health care," in *Proceedings of the 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC '15)*, pp. 4950–4953, Milan, Italy, August 2015.

[14] S. Mayer, M. Schalch, M. George, and G. Sörös, "Device recognition for intuitive interaction with the web of things," in *Proceedings of the ACM Conference on Ubiquitous Computing (UbiComp '13)*, pp. 239–242, Zürich, Switzerland, September 2013.

[15] F. Shi, A. G. Gale, and K. J. Purdy, "Eye-centric ICT control," in *Contemporary Ergonomics 2006*, pp. 215–218, Taylor & Francis, London, UK, 2006.

[16] K. Czuszynski and J. Ruminski, "Interaction with medical data using QR-codes," in *Proceedings of thev 7th International Conference on Human System Interactions (HSI '14)*, pp. 182–187, Costa da Caparica, Portugal, June 2014.

[17] J. Ruminski, M. Smiatacz, A. Bujnowski, A. Andrushevich, M. Biallas, and R. Kistler, "Interactions with recognized patients using smart glasses," in *Proceedings of the 8th International Conference on Human System Interactions (HSI '15)*, pp. 187–194, June 2015.

[18] A. Bujnowski, M. Benoit, M. Kaczmarek, P. Przystup, J. Ruminski, and I. Pecci, "Active and dynamic graphical code for object identification in healthcare," *Journal of Medical Imaging and Health Informatics*, vol. 5, no. 8, pp. 1631–1639, 2015.

[19] C. Borchers, Google Glass moves into the hospital at Beth Israel, the Boston Globe, 2014, http://www.bostonglobe.com.

[20] L. Zhang, X.-Y. Li, W. Huang et al., "It starts with iGaze: visual attention driven networking with smart glasses," in *20th ACM Annual International Conference on Mobile Computing and Networking (MobiCom '14)*, pp. 91–102, Maui, Hawaii, USA, September 2014.

[21] J. Ruminski, A. Bujnowski, J. Wtorek, A. Andrushevich, M. Biallas, and R. Kistler, "Interactions with recognized objects," in *Proceedings of the 7th International Conference on Human System Interactions (HSI '14)*, pp. 101–105, Costa da Caparica, Portugal, June 2014.

[22] R. McCall, B. Martin, A. Popleteev, N. Louveton, and T. Engel, "Text entry on smart glasses," in *Proceedings of the 8th International Conference on Human System Interactions (HSI '15)*, pp. 195–200, IEEE, Warsaw, Poland, June 2015.

[23] M. Stoppa and A. Chiolerio, "Wearable electronics and smart textiles: a critical review," *Sensors*, vol. 14, no. 7, pp. 11957–11992, 2014.

[24] T. Kocejko, J. Ruminski, J. Wtorek, and B. Martin, "Eye tracking within near-to-eye display," in *Proceedings of the 8th International Conference on Human System Interactions (HSI '15)*, pp. 166–172, IEEE, Warsaw, Poland, June 2015.

[25] Y.-C. Tung, C.-Y. Hsu, H.-Y. Wang et al., "User-defined game input for smart glasses in public space," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*, pp. 3327–3336, ACM, Seoul, South Korea, April 2015.

[26] T. Kocejko, A. Bujnowski, and J. Wtorek, "Eye mouse for disabled," in *Proceedings of the Conference on Human System Interaction (HSI '08)*, pp. 199–202, Krakow, Poland, May 2008.

[27] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV '11)*, pp. 2564–2571, IEEE, Barcelona, Spain, November 2011.

[28] O. Miksik and K. Mikolajczyk, "Evaluation of local detectors and descriptors for fast feature matching," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR '12)*, pp. 2681–2684, IEEE, Tsukuba, Japan, November 2012.

[29] X. Xu, L. Tian, J. Feng, and J. Zhou, "OSRI: a rotationally invariant binary descriptor," *IEEE Transactions on Image Processing*, vol. 23, no. 7, pp. 2983–2995, 2014.

[30] Y.-D. Kim, J.-T. Park, I.-Y. Moon, and C.-H. Oh, "Performance analysis of ORB image matching based on android," *International Journal of Software Engineering and its Applications*, vol. 8, no. 3, pp. 11–20, 2014.

[31] B. Guo, D. Zhang, Z. Yu, Y. Liang, Z. Wang, and X. Zhou, "From the internet of things to embedded intelligence," *World Wide Web Journal*, vol. 16, no. 4, pp. 399–420, 2013.

[32] S. Cirani and M. Picone, "Wearable computing for the Internet of things," *IT Professional*, vol. 17, no. 5, pp. 35–41, 2015.