Interacting with a Self-portrait Camera Using Motion-based Hand Gestures

Shaowei Chu
University of Tsukuba
Tennoudai 1-1-1, Tsukuba, Ibaraki,
Japan 305-8577
chushaowei@iplab.cs.tsukuba.ac.jp

Jiro Tanaka
University of Tsukuba
Tennoudai 1-1-1, Tsukuba, Ibaraki,
Japan 305-8577
jiro@cs.tsukuba.ac.jp

ABSTRACT
Taking self-portraits with a digital camera is a popular way to present oneself through photography. Traditional techniques for taking self-portraits, such as use of self-timers or face detection, provide only a modest degree of interaction between the user and camera. In this paper, we present an interaction technique that make novel use of image-processing algorithm to recognize hand motion gestures and provides user a natural way to interact with camera for taking self-portraits. User can perform nature gestures to control essential functions of camera and take self-portraits effectively. Three types of gesture (i.e., *waving*, *eight-direction selection*, and *circling* were identified and applied to develop a gesture user interface for controlling a Digital Single-Lens Reflex (DSLR) camera. Two experiments were conducted to evaluate the usability and performance of the gesture interface. The results confirmed that the usability of the gesture interface is superior to a self-timer and the proposed technique achieved about 80% accurate recognition of motion gestures.

Author Keywords
Digital Camera; Gesture User Interface; Motion Gestures; Image Processing; Human Computer Interaction.

ACM Classification Keywords
H.5.2.Information Interfaces and Presentation (e.g. HCI): User Interfaces - *Graphical User Interfaces (GUI)*.

General Terms
Human Factors; Design

INTRODUCTION

Taking self-portraits with a digital camera is a common way of portraying oneself through photography. With the rapid growth of digital cameras, the volume of self-portrait images is growing rapidly [1]. In general, taking self-portraits is becoming popular, particularly among young

APCHI '13, September 24 - 27 2013, Bangalore, India

http://dx.doi.org/10.1145/2525194.2525206

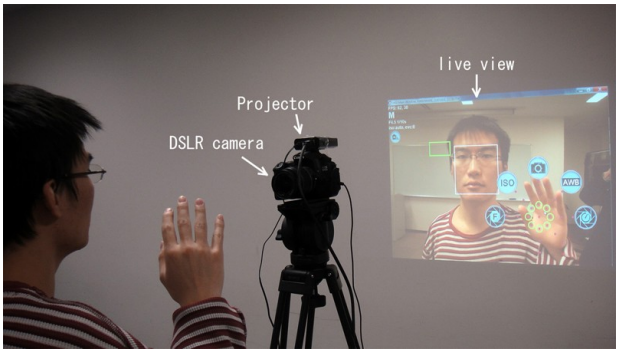

Figure 1. The self-portrait system prototype includes a DSLR camera mounted on a tripod, a projector attached to the camera as a viewfinder, and notebook PC (not shown in figure) to enable processing. The user can use gestures to control the camera functions.

[2]. For example, it is sometimes necessary to create a passport-style photograph, or profile pictures to show off on online social networking services (SNS). Fortunately, advances in digital cameras, including those in certain smart phones and foldable liquid crystal display (LCD) screens, have made self-portraits easier to take [3, 4]. However, users are normally required to touch the camera physically to change the frame, self-timer, and other settings; in some cases users may use a remote control, but this can occupy user' hand and result in unnatural hand positions in photographs [4].

Soon, advancements in camera design will making cameras more interactive, responsive, and accessible to users, with particular emphasis on interactive approaches that smooth the process of taking self-portraits. Image processing algorithms, such as face and smile recognition and motion detection functions are a first step. These vision-based techniques could also be used to develop computational photography [5] and gestural interface for cameras [6]. Particularly, these techniques can provide new ways for users to control the camera remotely, so they can focus on taking a good self-portrait and not on configuring the camera settings.

The previous work [7, 8] originally proposed the concept of a vision-based gesture interface for controlling a self-portrait camera attracted a great deal of interest from researchers and media. This encouraged us to continue

explore this new field and conduct further research. In this work, we attempt to develop rich gesture interface that for interacting with a professional DSLR camera. A prototype system, see **Figure 1**, was implemented. Inspired by Nikon projector camera [9], we used a projector to show the camera live view. User can see the instant preview and perform gestures to make interaction with the camera to take self-portrait photographs.

The prototype system was deployed in an indoor environment, since many good portraits are captured indoors [3, 4]: users can manually configure the lighting conditions and fix colored backgrounds; and because they do not feel self-conscious when posing, they can take as many photographs as they need; additionally a larger size display can be used, which provides a clearer preview and help the user prepare the self-portrait.

The new proposal in this research is that, three types of motion gesture (i.e., *waving*, *eight-direction selection*, and *circling*) were introduced for interacting with the self-portrait camera. By combining the three gestures we can develop rich gesture interface and controlling both digital and analog parameters. The new proposed gesture interface can provide control of various essential functions of the camera: the aperture, shutter speed, ISO, white balance, shutter trigger, etc. This gives a complete solution for gesture-based control of camera interaction.

We conducted two experiments to determine the feasibility and performance of the proposed technique. In the first experiment we evaluated the usability factors of the gesture interface that for taking self-portraits and compared it with a traditional self-timer technique. In the second experiment we assessed the accuracy and performance of the proposed gesture recognition technique. The experiment results showed that the gesture interface has high user satisfaction and it is superior to the traditional self-timer technique. In addition, the proposed recognition technique achieved about 80% accuracy of detecting motion gestures.

RELATED WORK

The main challenges in this work are: 1) implement a gesture recognition method that robust to lighting and color conditions, 2) develop a gesture interface that provides rich interactions with a professional DSLR camera. So, in this section, we reviewed the gesture recognition techniques and discussed the advantages of our proposed motion-based gesture recognition method.

Commonly used vision-based hand gesture recognition methods can be classified into two groups: model-based methods and motion-based methods [10, 11].

In the model-based approach, the standard procedure of gesture recognition [11, 12] combines several tasks: initialization, tracking, pose estimation, and recognition. Initialization captures prior knowledge of a specific configuration, such as color pattern, to distinguish the shape of the hand, which is then used to constrain tracking and pose estimation. In the final recognition step, actions are distinguished as behaviors performed by the user in one or more frames. Many successful applications have been developed: Wilson [13] developed a background subtraction method to detect a pinching gesture above a tabletop; SixthSense [14] uses a color marker attached to the hand for tracking the fingers, and a pointing technique to recognize actions. Other studies have used color information to distinguish hand models and have applied comparison algorithms to determine the three-dimensional position of the hand stored in the dataset [15, 16, 17]. Other studies have reported static position detection using template matching to recognize hands [18].

The disadvantage of the model-based approach is that it often requires a predefined configuration in initialization, such as a pre-calibrated environment [13], specific color lighting conditions [15, 16], the need to attach markers or wear gloves [14, 17], or restricted hand position during interaction [18]. The computational complexity is also a significant issue in this kind of approach. In our self-portrait camera scenario, color lighting conditions are often dynamic, and wearing markers or gloves can results in unnatural portraits.

Therefore, we used a motion-based recognition approach. In this kind of technique, the recognition procedure combines a motion measurement algorithm to determine image differences between two consecutive frames, and a pattern recognition method to distinguish motion actions. It ignores initialization and position estimation procedures, and therefore allows more freedom in hand motion than model-based approaches. Moreover, it is particularly robust to different color lighting conditions.

Many research prototypes have used this approach, applying a sparse Lucas-Kanade (LK) or dense Horn-Schunck (HS) optical flow measurement [19, 20] to detect motion, and a Support Vector Machine (SVM) or AdaBoost to classify human actions [10, 21, 22, 23]. Due to the considerable computational complexity, some of these studies applied GPU speed-up algorithms to achieve real-time application. However, performance was still poor. Also, use of the machine learning approach to classify gestures restricts the recognition results in digital outputs, reducing the recognition rate and limiting the set of gestures that can be classified.

Instead of measuring motions from a large set of tracking points, we developed a method that involves arranging various layouts of a small set of tracking points on a specific region to detect hand motion. Thus, our approach uses a manually restrained timing algorithm to recognize gestures and track each step of motions in the recognition procedure. The method requires less computational complexity and no pre-training, and enables both digital and analog gesture actions as outputs.

MOTION GESTURE RECOGNITIONE

The innovative point of this study is we focused on three different types of motion gestures, and combined them to develop rich gesture interfaces.

Waving gesture - This gesture involves raising the hand and moving it from side to side. It is commonly used to attract attention at a distance, so it is intuitively useful for mapping to a wake-up interaction function for computer systems [24]. It is also a good clue for the system to recover the hand region in the image. Our interface design is based on the assumption that the user interface is around the user's hand and can be manipulated by small hand motions. Thus, recognition of gestures and recovery of the accurate region of the *waving* hand is important to recognition of the other two types of gesture.

Eight-direction selection gesture - This gesture involves raising the hand and moving it from one position in a specific direction over a given distance. This gesture is similar to that described in a previous study [25]. However, in our work, the directions can be: Left (LT), Up-Left (UL), Up (UP), Up-Right (UR), Right (RT), Down-Right (DR), Down (DW), Down-Left (DL); these eight gesture inputs make it easier to develop a menu selection user interface and offered good accuracy of function mapping [26].

Circling gesture - This gesture involves raising the hand and moving it in circles over a region, in the clockwise or counter-clockwise direction, which was studied in [25]. The *circling* gesture can provide analog data input with direction angle (or moved circles) per frame. This advantage allows a user interface to control linear values.

The three types of gesture have specific features and are connected logically. First, *waving* gesture recognition not only provides a startup action but also allows recovery of the region of the hand in the frame. The latter outcome is beneficial to narrow gesture recognition to a small region around the hand. Second, the *eight-direction selection* gesture provides eight selection choices. These are appropriate for developing a menu selection interface. Third, the *circling* gesture provides analog output and can use clockwise and counter-clockwise motions to distinguish between positive and negative adjustment. Therefore, it is possible to pop up the user interface by *waving* gesture and selects an option by *eight-direction selection* and then adjust the parameters with *circling gestures*. These operations can be repeated, enabling rich interfaces to control many parameters and functions.

The following sub-sections explain the gesture recognition technique.

Optical-flow motion estimation

The proposed gestures are recognized by using motion estimation based on a standard Lucas-Kanade optical flow tracker [19]. An optical flow tracker is an algorithm that estimates the velocity of movement for a given set of points on a gray scale image, using various images. A typical method for estimating the movement of a point is to calculate derivatives of pixel intensity at each point, and then determine the motions within a window centered at that point in another image. The window is an integral window in which a similarity function is performed to search for an optimal candidate as the estimated moved point. During the process, each point is calculated independently of the others. The Lucas-Kanade method of optical flow tracking has been widely used in various motion tracking and real-time applications. One advantage of optical flow tracking is that it calculates the derivatives of pixel intensity from nearby pixels, and does not rely only on the color information of one pixel; this makes it less sensitive to image noise and brightness [19, 20].

The novel aspect of our proposed technique is that, a set of tracking points are defined at a specific region on an image frame, and the optical flow at these points is calculated repeatedly in sequences of image frames to analyze the gesture motions. This method can greatly reduce the compute complexity and do not require any pre-training. The following sub-sections describe three patterns of layout for recognizing three types of gesture.

Waving Gesture Recognition

To recognize w*aving* gesture, we can apply a matrix layout of tracking points on full-size images (see **Figure 2**) to detect hand motion. The current system uses 15 x 7 = 105 points arranged at a resolution of 360 x 240 pixels. Each point is used to detect motions separately. The waving motion pattern is detected by the motion displacement of a point, determined by optical flow, from a direction angle θ to its semi-opposing direction θ'; length is larger than 1, because a lower value indicates a very slow speed of motion and is therefore excluded. The span angle between θ

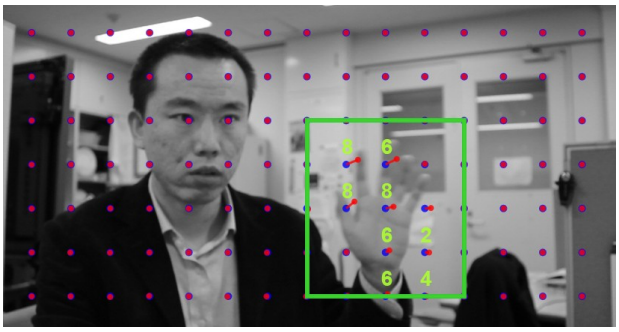

Figure 2. Matrix of tracking points for detecting a *waving* gesture. The detected waving transforms (hand motion side to side) are labeled with numbers, and the rectangle represents the recognized waving hand.

and θ' should be larger than 120°; this indicates a successful transform of wave motion: the point's wave transform value plus 1. If no transform is detected over a specified period (in this case, 500 ms), the transform value is reset to 0. Transform values should be set at 4 or greater

to filter out unnecessary motion, and to identify candidate waving gesture points.

In general, when a *waving* gesture occurs, the waving hand will occupy a region on the image and several neighboring points will detect motion simultaneously. To group the neighboring points and estimate the hand region, we can apply an algorithm to merge the neighboring points into rectangles, and then assign the merged rectangles as the final result of *waving* gesture recognition (see **Figure 2**). This process involves several steps. First, a rectangle is placed around each candidate point in the matrix. The size of the rectangle is the same as the horizontal and vertical distance of two neighboring points. In the second step, the intersected rectangles are merged together to union rectangles. Finally, among the merged rectangles, the one containing the maximum *transforms* value is assigned as the dominant recognized *waving* gesture result.

Our purpose is not only to detect a *waving* gesture but also to recover the hand region in the image, which plays a key role in the two subsequent gesture recognition procedures.

Eight-direction selection gesture recognition

The *eight-direction selection* gesture can be identified after the *waving* motion is identified. In the *eight-direction selection* gesture, the hand moves from the recognized hand region in a specific direction: Left, Up-Left, UP, Up-Right, Right, Down-Right, Down, or Down-Left (**Figure 3**).

To recognize the gesture, 24 tracking points are separated into eight directions with a radial-shape layout (**Figure 3**) to enable detection of hand motions. For each direction, three points are organized as a set and arranged in a radial line outward from the center. The detection of a moving gesture in a specific direction involves several steps. Using the UP direction as an example, each of the three points detects directional motion: *length* as determined by optical flow measurement must be larger than 1, and angle θ must be approximately 270° (±22.5°). If the three points detect this motion within a specified period (in this case, 500 ms), the direction selection gesture is assumed to have been successfully detected. Similar processes are involved in detecting gestures in other directions.

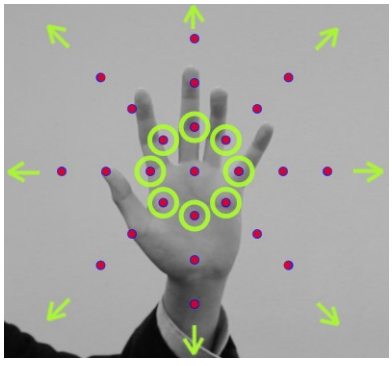

Figure 3. A radial-shaped layout of tracking points for detecting *eight-direction selection* gestures. Each radial line consists of three points from the center to outside. The circles represent motion steps; the circles will move to the next tracking point according to the direction of hand motion.

Circling Gesture Recognition

In the *circling* gesture, the hand moves in circles over the region that has already been identified as the *waving* hand region.

To recognize the gesture, 20 tracking points are arranged in a circular pattern (**Figure 4**). The motion of the set of points is calculated according to two mean values: *direction angle* and *length*. **Figure 5** presents the variation diagrams for these two factors for a clockwise circling motion pattern. The moved circling angle is calculated by accumulating the shifted direction angle (difference in direction angle between two consecutive frames), and by incorporating the timing factor in each frame. If the direction angle in consecutive frames has a clockwise pattern, this indicates a clockwise movement angle. In contrast, a counter-clockwise motion is detected as a counter-clockwise movement angle. If no motion is detected, the moved angle value is reset to 0.

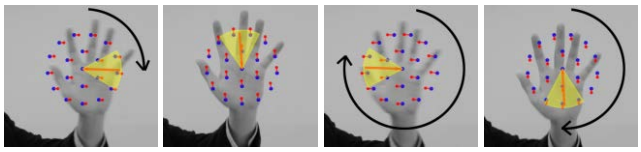

Figure 4. A circular layout of tracking points for detecting *circling* gestures. The four images from left to right show the motion sequences with a clockwise circling pattern, which starts at the top, moves to the right, down, left, and back to the top for one full circling motion.

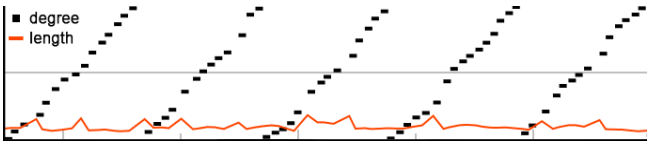

Figure 5. Two analog data parameters per frame time line. The circling angle from 0 - 360° is indicated by black blocks and the length is marked with lines.

This layout of tracking points can also recognize a *waving* gesture in which the hand moves from side to side over the point cloud.

GESTURE USER INTERFACES AND APPLICATION

We designed two user interfaces for controlling a DSLR camera, by incorporating the gestures the system recognizes as well as digital and analog information. One important consideration of interface design is to show appropriate visual feedback to users when they perform gestures, which can improve the user experience [27].

Mode switching interface

The mode switching interface is a pie-like menu that pops up on the screen around the user's hand once a *waving* gesture is detected. The menu can provide a maximum of

eight selection choices: LT, UL, UP, UR, RT, DR, DW, and DL. The user can interact with the system using the *eight-direction selection* gestures.

We mapped five of these items (LT, UL, UP, UR, RT) to five camera functions: shutter trigger, white balance aperture, ISO, and shutter speed (see **Figure 6**, left). When the shutter trigger function (UP item) is selected, a timer appears and counts down from 5 to 1; during this time the camera autofocuses on the user's face and the user prepares his/her pose. Once the timer reaches 1, the camera takes a photograph automatically. When the white balance (UR) function is selected, another set of six function icons pops up. As shown in **Figure 6**, right, the white balance icons from left to right are: auto white balance, daylight, shade, cloudy, tungsten, and fluorescent.

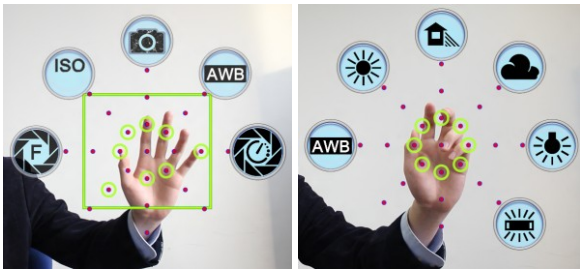

Figure 6. Mode switching interface. Left: Main menu of the interface. The function icons from left to right are: aperture, ISO, shutter trigger, white balance, and shutter speed. Right: The six white balance icons are: auto white balance, daylight, shade, cloudy, tungsten, and fluorescent.

The other three icons represent aperture, ISO, and shutter speed, which are linear pattern values.

Value Adjusting Interface

This interface was designed based on *circling* gestures. The user can perform gestures within the region of points to adjust a parameter value linearly.

However, the values of aperture, ISO, and shutter speed are not pure linear data, but are sequential. For example, the aperture has a sequence of values: 3.5, 4, 4.5, 5, 5.6, 6.3, 7.1, etc., in which each increase in value represents double the volume of light to the photoreceptor in the camera. The ISO and shutter speed parameters have similar values sequences. Thus, to modify such sequence patterns of values, the user completes a full circling motion, moving the hand 360°, to move to the next step. A complete clockwise circling motion increases the value by one step, while a complete counter-clockwise circling motion decreases the value by one step. Changing the aperture, ISO, and shutter speed involves similar mechanisms. **Figure 7** shows the interface for three cases when adjusting these parameters.

After setting the desired value/s, the user performs a *waving* gesture within the region of points to return to the mode switching interface. The value adjusting interface then disappears, and the mode switching interface pops up. Many of the camera settings can be configured by switching between the two interfaces.

Figure 7. Value adjusting interface. Perform *circling* gesture inside the tracking point circle to adjust the value. The semicircular dial above will rotate according gestures in clockwise/counter-clockwise motions. The parameters adjusted in the image from left to right are aperture, shutter speed, and ISO.

Our demo movies that proved our concept can be found here:
http://www.iplab.cs.tsukuba.ac.jp/~chushaowei/minterface/

IMPLEMENTATION

The current implementation uses the OpenCV library [28], which provides Lucas-Kanade optical flow tracking. We used a multi-core PC, Intel Core i3 2.4 GHz CPU, and dedicated one thread for *waving* gesture detection, and one thread for both the *eight-direction selection* gesture and *circling* gesture recognition. The control signal and data exchange between the DSLR camera and computer were provided by a Canon SDK [29]. The preview and GUI were rendered using Microsoft Direct2D. The entire program was written in C++.

Table 1 summarizes the performance of the algorithm. The *waving* gesture recognition uses a frame image with a resolution of 360 x 240 pixels and 105 tracking points. On average, motion estimation and gesture recognition takes 8.3 ms (120 FPS). The other two processing tasks, *eight-direction selection* gesture recognition (360 x 360 pixels) and *circling* gesture recognition (160 x 160 pixels), were conducted using a single thread and average speeds were 15.7 ms (63 FPS) and 3.1 ms (323 FPS), respectively. The Canon 60D camera provides 30 FPS with a preview video stream resolution of 1056 x 704 pixels, and the gesture recognition processing performance is more than sufficient to support a real-time application.

	Process time (ms)
Waving gesture	8.3
Eight-direction selection	15.7
Circling gesture	3.1

Table 1: Gesture recognition performance.

The apparatus of the prototype system included a Canon 60D DSLR camera, a tripod, a mini projector, and a ThinkPad X201i notebook PC. The camera is connected to

the notebook PC by a USB, to enable processing and to serve as the camera viewfinder. **Figure 1** shows an overview of the system arrangement.

EXPERIMENT 1: EVALUATING THE GESTURE INTERFACE

We conducted an experiment to evaluate the feasibility of using the gesture interface for taking self-portraits. In the experiment, we compared the conventional self-timer technique with the proposed gesture interface. The experiment was designed to measure the efficiency of the shooting procedure, user satisfaction with the resulting portraits, input difficulty (ease of use) of the interface, and user satisfaction regarding the two techniques. The results were based on observation and a questionnaire with scores ranked on a five-point Likert scale (1: strongly disagree … 5: strongly agree) after the experiment.

Apparatus

The apparatus of the proposed gesture-based system was described in the previous section and the arrangement is shown in **Figure 1**.

A similar arrangement was used in the self-timer scenario, but another item was used as a stand-in to allow the camera to autofocus, because the DSLR camera has no autofocus function without the user touching the shutter button. This means that the user must press the shutter button halfway to autofocus, push it down completely to trigger the self-timer, and then run to the front of the camera and pose. Because the user has no opportunity to stand in the correct position and face the camera to allow it to autofocus, we use a method popular in self-portrait photography that positioned a picture board to act as a stand-in while the user was setting the camera functions. The arrangement of the self-timer system is shown in **Figure 8**.

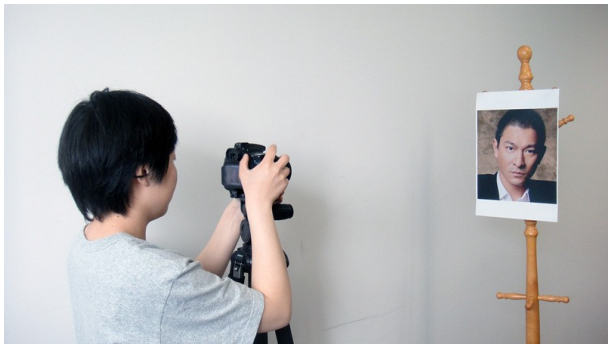

Figure 8. The self-timer scenario in which a picture board was used as a stand-in to allow the camera to autofocus.

Participants

We recruited 11 participants (5 women, 6 men), ranging in age from 23-29 years (mean: 26.1), participated in the experiment.

Task and procedure

Participants were given a simple introduction to the system. The author demonstrated in person how to take two self-portrait photographs using the two different techniques.

In the self-timer scenario, participants positioned the picture board in the desired position as a stand-in. Then, they went to the camera and pressed the shutter button half-way to trigger the autofocus on the stand-in. Next, the shutter button was depressed completely to trigger the 10-s self-timer countdown. The participants then quickly ran to the stand-in board, removed it, and put themselves in its place. Once the self-timer reached the end of the countdown, the camera released the shutter and took a photograph.

In the gesture interface scenario, participants stood in front of the camera with their upper bodies in view of the camera. They performed a *waving* gesture to wake up the system, and the mode switching interface popped up around the waving hand in the preview. Next they moved their hand in the UP direction to select shutter trigger function, activating a 5-s timer countdown. The camera autofocused on the participant's face and then took a photograph.

Participants were asked to use both techniques. This preliminary test simulated a simple task, triggering the camera shutter, which is the most commonly used procedure when taking portraits.

In the second phase of testing, we evaluated and compared the two techniques in much greater detail. Participants were asked to set many camera parameters (aperture, shutter speed, ISO, and white balance, etc.) using the traditional button-based interface and the proposed gesture interface. Participants were permitted to experiment with the two techniques and to take many self-portrait photographs until they were familiar with and understood the two interaction approaches. This test was designed to assess the ease of use of each interface, and user preference for the two techniques when controlling many camera functions.

After each test, participants were asked to complete a questionnaire.

Results

In the preliminary test, it took about 18 s for participants to complete a self-portrait shot using the self-timer. In contrast, the gesture interface took about 10s. Most participants preferred the gesture interface, and no significant differences appeared in user satisfaction with the resulting portraits between the two techniques. **Figure 9** presents the results.

In the second phase of testing, participants reported slightly less difficulty in camera parameter input and adjustment when using the gesture interface. Participants preferred the gesture interface for controlling the camera functions compared to the button-based interface. **Figure 10** presents the results.

Figure 9. Results of the preliminary test.

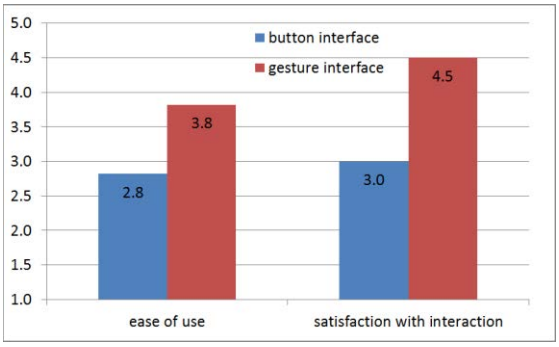

Figure 10. Results of the second test.

EXPERIMENT 2: ACCURACY OF GESTURE RECOGNITION

This experiment was designed to evaluate the results of gesture recognition, and to assess the recognition rate of the three types of gesture.

The apparatus used for the experiment was described in Section 5. A total of 10 subjects (3 women, 7 men), ranging in age from 22-28 years (mean: 25.2) were recruited for this experiment.

Task and procedure

In the *waving* gesture test, participants stood a given distance from the camera with the upper body in view of the camera. They were then asked to perform the *waving* gesture three times. Once a *waving* gesture was detected, visual feedback appeared on the preview screen to notify the participant. During the test, we observed and recorded the sensitivity and speed of gesture recognition.

In the *circling* gesture test, participants were asked to perform the circling gesture for a while to familiarize themselves with the actions. On the screen the circling angle of motion will be showed on top of hand. Then, the participants were asked to perform a full circling gesture, i.e. 360°, for 10 times, and we recorded the errors of the recognition. Because this gesture is analog, we also observed the sensitivity and user experience of the gesture recognition result during the test.

A specific interface was developed for the *eight-direction selection* gesture test (**Figure 11**) with eight icons representing each of the eight directions (LT, UL, UP, UR, RT, DR, DW, DL). Participants were asked to select an icon indicated by a red circle marker. After they selected a direction icon, regardless of whether it was correct, the interface disappeared and then reappeared for the next test trial. Each participant completed 24 test trials (three trials for each direction) presented in a random order.

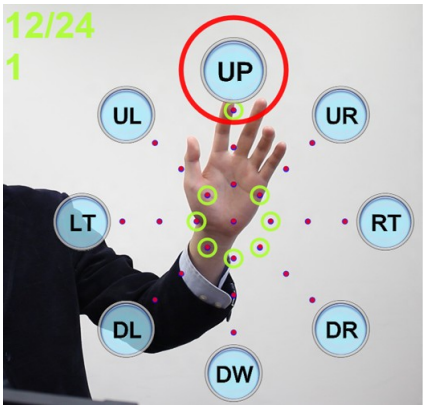

Figure 11. The eight-direction selection gesture experiment. The participant was required to select the red marker, which indicated the destination direction.

Results

In the *waving* gesture test, the gesture was usually detected when the participant waved his/her hand four times from side to side within 2.5 meters from camera. It could detect the waving hands in the image from size 62 x 73 pixels to 930 x 511 pixels. A single waving gesture takes about 2 s to perform. The gesture was not be detected if the participant waved at a slow speed (i.e., the four side to side movements lasted longer than 2 s).

During the *circling* gesture test, we found that the recognition was very sensitive to hand motions, the visual feedback of the interface can accurate report the circling angle to user. However, participants feel difficult to set the value of their expected in the range of 360°. The full circling gesture test showed 138° error on average. But participants felt no difficult to set values using a full circling motion to adjust one step of value change.

We collected the results of 240 test trials (30 trials for each direction) from the *eight-direction selection* gesture test. **Table 2** lists the mean accuracy results for recognition of the eight directions; the left-most column indicates gesture inputs, and the corresponding row shows the recognition result.

As shown in **Table 2**, the mean accuracy of the eight directions was 81%, and the accuracy for each direction exceeded 70%. The upper directions, LT, UL, UP, UR, RT, had better accuracy, while DW had the best result at 97%. A few cases of failure were observed; these occurred for several reasons. First, the user's forearm sometimes conflicted with the hand motions. Second, rapid hand

movement sometimes produced an incorrect result or no result. In cases with no result, the user often tried to move his/her hand back to the center of the interface and perform the gesture again, but overshot the center and moved in the opposite direction, causing an incorrect result. Because incorrect actions can occur, the system is designed to allow the user to perform a *waving* gesture to cancel the selection and then perform the selection gesture again.

	LT	UL	UP	UR	RT	DR	DW	DL
LT	0.80	0.03	0.03		0.03			0.10
UL		0.83	0.13				0.03	
UP		0.07	0.83				0.10	
UR	0.10		0.03	0.80		0.07		
RT	0.03		0.10	0.03	0.80	0.03		
DR		0.10	0.10	0.03	0.03	0.70	0.03	
DW			0.03				0.97	
DL			0.07	0.07		0.03	0.07	0.77

Table 2. Accuracy of eight-direction selection gesture recognition.

DISCUSSION

Participants were positive about the idea of a vision-based gesture interface for controlling a self-portrait camera. Most of the participants agreed it can become the next generation of interactive technology of the camera. However, they were frustrated by the imperfect gesture recognition rate.

In the future, we will test different motion estimation algorithm and implement better recognition algorithm. Some suggestions referred that to use sound feedback to the user's selection, which may improve the user experience. With regard to the experiments, we are planning to add an additional setting with a remote control to get a detail comparative result of the gesture interface in the future.

The proposed *waving* gesture recognition may be less accurate for estimating hand size, but it is more accurate at estimating hand position. During the experiments, participants did not report any significant deviation in the recognized hand position. Future research will improve the accuracy by adding more tracking points with a dense matrix layout, but this must be balanced with the associated decrease in performance.

In the *eight-direction selection* gesture recognition and mode switching interface tests, participants said they preferred using gestures in the upper semicircle directions (LT, UL, UP, UR, RT), which were more convenient to reach than the lower directions (DL, DW, and DR). Therefore, the interface should be designed with the most frequently used functions arranged on the upper semicircle.

In the *circling* gesture and interface tests, which simulated the analog input, users found it difficult to stop the motion at a particular value. Thus, we designed the interface to use a full circling gesture (360°) to increase or decrease the value, to make it easy to stop at a particular value.

The present study focused on interactions and ignored the results of the photographs. A recent paper [30] discussed a technique for selecting still candid portraits from video sequences. In future studies, we plan to develop a system to support video recording and continuous shooting, which would provide possibilities for a wider range of satisfying portraits.

CONCLUSION

In this study, we developed a prototype self-portrait camera system that allows users to take self-portrait photographs efficiently using gestures in an indoor environment. Our experiments confirmed that the system performed much better than those using conventional techniques such as self-timers.

We developed a motion-based gesture recognition technique that uses optical flow tracking in real-time manner. Three different types of gesture were recognized for use in the system: *waving*, *eight-direction selection*, and *circling*. These gestures provide not only digital but also analog inputs, which enabled us to develop a rich gesture interface that allows users to control various camera functions such as aperture, shutter speed, ISO, white balance, and shutter trigger.

REFERENCES

1. Huang, L., Xia, T., Wan, J., Zhang, Y., and Lin, S. Personalized portraits ranking. *In Proceedings of the 19th ACM international conference on Multimedia*, ACM Press (2011), 1277–1280.
2. Okabe, D., Ito, M., Chipchase, J., and Shimizu, A. The social uses of purikura: photographing, modding, archiving, and sharing. *In Pervasive Image Capture and Sharing Workshop, Ubiquitous Computing Conference* (2006), 2–5.
3. 4 tips for taking gorgeous self-portrait and outfit photos. http://www.shrimpsaladcircus.com/2012/01/self-portrait-outfit-photography-guide.html.
4. Taking a great self portrait with your camera. http://www.squidoo.com/self-portrait-tips.
5. Adams, A., Talvala, E.-V., Park, S. H., Jacobs, D. E., Ajdin, B., Gelfand, N., Dolson, J., Vaquero, D., Baek, J., Tico, M., Lensch, H. P. A., Matusik, W., Pulli, K., Horowitz, M., and Levoy, M. The frankencamera: an experimental platform for computational photography. *ACM Transactions on Graphics* 29, 4 (2010), 29:1–29:12.
6. Gomez, S. R. Interacting with live preview frames: in-picture cues for a digital camera interface. *In Adjunct proceedings of the 23nd annual ACM symposium on User interface software and technology* (2010), 419–420.
7. Chu, S., and Tanaka, J. Hand gesture for taking self portrait. *In Proceedings of the 14th international*

conference on Human-computer interaction: interaction techniques and environments - Volume Part II (2011), 238–247.

8. Chu, S., and Tanaka, J. Head nod and shake gesture interface for a self-portrait camera. I*n ACHI 2012, The Fifth International Conference on Advances in Computer-Human Interactions* (2012), 112–117.
9. Nikon coolpix s1200pj camera. http://www.nikonusa.com/en/Learn-And-Explore/Article/g022fmeg/Built-in-Projector.html.
10. Bayazit, M., Couture-beil, A., and Mori, G. Real-time motion-based gesture recognition using the GPU. *In IAPR Conference on Machine Vision Applications (MVA)* (2009), 9–12.
11. Wachs, J. P., Kolsch, M., Stern, H., and Edan, Y. Vision-based hand-gesture applications. *Communications of the ACM* 54, 2 (2011), 60–71.
12. Moeslund, T. B., Hilton, A., and Kruger, V. A survey of advances in vision-based human motion capture and analysis. *Journal of Computer Vision and Image Understanding* 104, 2 (2006), 90–126.
13. Wilson, A. D. Robust computer vision-based detection of pinching for one and two-handed gesture input. *In Proceedings of the 19th annual ACM symposium on User interface software and technology* (2006), 255–258.
14. Mistry, P., and Maes, P. Sixthsense: a wearable gestural interface. *In ACM SIGGRAPH ASIA 2009 Sketches* (2009), 11:1–11:1.
15. Lee, T., and Hollerer, T. Handy AR: Markerless inspection of augmented reality objects using fingertip tracking. *In Wearable Computers, 2007 11th IEEE International Symposium on* (2007), 1–8.
16. Wang, R., Paris, S., and Popovic, J. 6D hands: markerless hand-tracking for computer aided design. *In Proceedings of the 24th annual ACM symposium on User interface software and technology* (2011), 549–558.
17. Wang, R. Y., and Popovic, J. Real-time hand-tracking with a color glove. *ACM Transactions on Graphics* 28, 3 (2012), 63:1–63:8.
18. Stenger, B., Woodley, T., and Cipolla, R. A vision-based remote control. *In Studies in Computational Intelligence*, vol. 285 (2010), 233–262.
19. Baker, S., and Matthews, I. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision* 56, 3 (2004), 221–255.
20. Bruhn, A., Weickert, J., and Schnorr, C. Lucas/kanade meets horn/schunck: combining local and global optic flow methods. *International Journal of Computer Vision* 61, 3 (2005), 211–231.
21. Chen, M., Mummert, L., Pillai, P., Hauptmann, A., and Sukthankar, R. Controlling your TV with gestures. *In MIR 2010: 11th ACM SIGMM International Conference on Multimedia Information Retrieval* (2010), 405–408.
22. Fathi, A., and Mori, G. Action recognition by learning mid-level motion features. *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)* (2008), 1–8.
23. Takahashi, M., Fujii, M., Naemura, M., and Satoh, S. Human gesture recognition using 3.5-dimensional trajectory features for hands-free user interface. *In Proceedings of the first ACM international workshop on Analysis and retrieval of tracked events and motion in imagery streams* (2010), 3–8.
24. Hardy, J., Rukzio, E., and Davies, N. Real world responses to interactive gesture based public displays. *In Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia (2011)*, 33–39.
25. Kim, J., Mastnik, S., and Andre, E. EMG-based hand gesture recognition for realtime biosignal interfacing. *In Proceedings of the 13th international conference on Intelligent user interfaces* (2008), 30–39.
26. Atia, A., and Tanaka, J. Interaction with tilting gestures in ubiquitous environments. *International Journal of UbiComp* 1, 3 (2010), 1–13.
27. Eisenstein, J., and Mackay, W. E. Interacting with communication appliances: an evaluation of two computer vision-based selection techniques. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2006), 1111–1114.
28. Open source computer vision library (OpenCV). http://opencv.org/.
29. Canon digital camera software developers kit. http://usa.canon.com/cusa/consumer/standard_display/sdk_homepage.
30. Fiss, J., Agarwala, A., and Curless, B. Candid portrait selection from video. *ACM Transactions on Graphics* 30, 6 (2011), 128:1–128:8.