Cooperative Gesture Recognition
Learning Characteristics of Classifiers and Navigating the User to an Ideal Situation

Hiromasa Yoshimoto and Yuichi Nakamura
Academic Center for Computing and Media Studies, Kyoto University, Sakyo-ku, Kyoto 606–8501, Japan

Keywords: Cooperative Recognition, Gesture Interface, and Usability.

Abstract: This paper introduces a novel scheme of gesture interface that guides the user toward obtaining better performance and usability. The accuracy of gesture recognition is heavily affected by how the user makes postures and moves, as well as environmental conditions such as lighting. The usability of the gesture interface can potentially be improved by notifying the user of when and how better accuracy is obtained. For this purpose, we propose a method for estimating the performance of gesture recognition in its current condition, and a method for suggesting possible ways to improve performance to the user. In performance estimation, accuracy in the current condition is estimated based on supervised learning with a large number of samples and corresponding ground truths. If the estimated accuracy is insufficient, the module searches for better conditions that can be reached with the user's cooperation. If a good improvement is possible, the way to improve is communicated to the user in terms of visual feedback, which shows how to avoid or how to recover from the undesirable condition. In this way, users benefit, i.e., better accuracy and usability, by cooperating with the gesture interface.

1 INTRODUCTION

As the success of Microsoft's Kinect sensor shows, a gesture-based interface is one of practical solution for a natural user interface(Shotton and Sharp, 2011; OpenNI organization, 2010; PrimeSense Inc., 2010). Its fundamental technologies, such as motion tracking and gesture classification, however, still face many difficulties and challenges (Pavlovic et al., 1997; Moeslund et al., 2006; Ren et al., 2013), because of the ambiguities of human's behavior, self-occlusion, and the sensor's measurement noise, etc. Moreover, a gesture-based interface has additional usability requirements. It must not only maintain good recognition performance, but also needs to indicate its internal status, to give easy ways to recover from failures(Nielsen, 2009; Jacko and Sears, 2003).

In this paper, we focus on a framework for dealing with both of the above goals: to improve the performance of gesture recognition, and to improve the usability of the gesture interface. For these two goals, we propose a *cooperative gesture recognition* scheme, which navigates the user into changing their behavior so that both performance and usability are improved at the same time.

For this navigation, we introduce a *condition vector* and a *condition space* to represent how the gesture is easily recognized by the gesture recognition module. The condition vector is defined by the parameters measured by sensors, such as the user's standing position, posture of upper body, and movement speed of the user's hands.

We then use recall and precision as a measure of recognition performance, and calculates the distribution in condition space. Here, we can assume that if the condition space has enough dimensions, i.e., enough parameters, performances around neighboring positions are similar. If the recognition performance is satisfactory for one condition vector, it is expected to be satisfactory in its neighborhood. For covering over the condition space, we assume it can be separated into two portions: a portion in which the recognition performance is satisfactory and another portion with unsatisfactory performance.

With the above assumptions, our cooperative recognition scheme is composed by adding two modules to a common gesture interface scheme: a *performance estimation module* and a *feedback module*. These two modules work as follows. The performance estimation module estimates recall and precision from the current condition vector, and the feedback module advises the user to change his/her behavior to a more desirable condition, where the recall and precision become higher.

We modeled this performance estimation module as functions that map condition vectors into recall and precision. If enough labeled training samples are available, these functions are constructed by supervised machine learning. With these functions, the module can not only estimate the performance of current conditions, but also predict performance for any condition.

The feedback module uses estimate functions to search for more suitable conditions around the current condition, where the predicted performance increases. If a suitable condition is found, then the feedback module synthesizes visual hints to navigate the user into the suitable condition. For example, if the left position is predicted to have more suitable conditions for the gesture recognition, the feedback module draws an arrow-shaped computer graphic (CG) object that induces the user to move a little to the left.

The main contribution of this scheme proposes not a specialized application but a practical and generalized scheme. Our scheme can improve the performance of the existing gesture interface by just adding two modules. No modification of existing gesture classifiers is required.

This paper is structured as follows. In Section 2, an overview of related works is given. The proposed scheme is described in Section 3. In Section 4, we present the experimental evaluation, and we conclude in Section 5.

2 RELATED WORK

In the fields of amusement or entertainment, a gesture-based interface is a practical technology. There are many commercial products, such as Microsoft's Kinect sensors(Ren et al., 2013), Nintendo Wii, and Eye cam Sony, etc. However, despite these latest technologies, there are still challenging problems in gesture recognition.

From the viewpoint of computer vision (CV) and pattern recognition (PR), gesture recognition still has challenges in accuracy and the robustness of measurement and recognition. For example, in hand gesture recognition, the difficulties arise from the complex structure of hands and the wide variations in their appearances. Moreover, the ambiguities of the sensor's observation make further difficulties for gesture recognition; e.g., when we use an RGB–D camera as the sensor, the pixel's noise and occlusions on the captured color and depth image cause ambiguities in image analysis. Figure 1 shows actual hand depth images captured by a Kinect sensor; there are various images ranging from "easy" to "difficult" recognition. Although various approaches have been proposed to overcome these difficulties in terms of CV or PR, robust and accurate gesture recognition remains a difficult task(Pavlovic et al., 1997; **?**). In other words, there is no approach that can guarantee 100% accuracy in gesture recognition.

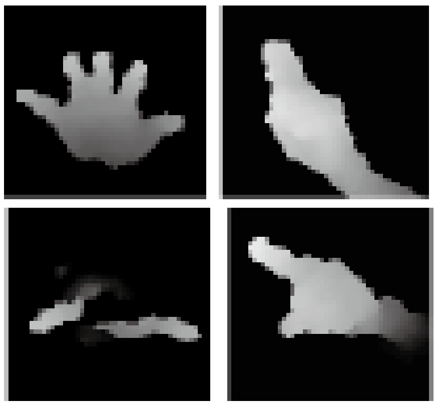


Figure 1: Example of "easy" and "difficult" images for gesture/sign recognition; the difficulties arise from image noise, occlusions, motion blurs, etc.

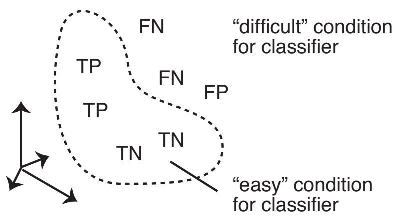
Table 1: Four categories of the classifier's output.

		Ground truth is: X	$\bar{X}$
Output of	X	TP: True positive (Correct)	FP: False positive (Type I error)
classifier is:	$\bar{X}$	FN: False negative (Type II error)	TN: True negative (Correct)

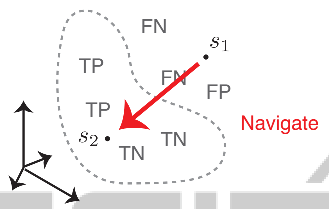
However, from the viewpoint of the usability engineering, the user interface should be designed to be efficient to use and easy to learn. Nielsen's(Nielsen, 2009) guidelines in usability engineering appeal for a gesture-based user interface to provide some functions that help the user to diagnose how accurate the recognized result is, and to rectify the error when gesture recognition fails. These features have already been proposed in many studies and are used in many commercial products. However, the pattern of application design is not yet well discussed and formalized.

3 COOPERATIVE GESTURE RECOGNITION

In general, each algorithm in gesture recognition, such as a feature detector or a gesture classifier, uses


FN
"difficult" condition for classifier
TP
FN
FP
TP
TN
TN
"easy" condition for classifier

(a) Condition space



(b) User navigation

Figure 2: Overview of cooperative gesture recognition: (a) we propose condition space to learn the characteristics function of the classifier. (b) This space is also used to navigate the user to change the user's behavior or the environments where the performance of classification will be improved.

its own assumptions on the human behaviors and environment, e.g., a user is standing in a certain area. Recognition performance is directly related to how far those assumptions are satisfied.

In a case where the assumptions are well satisfied, we can expect good recognition accuracy. However, it is difficult to estimate directly how far the assumptions are satisfied because the process of a feature detector or a classifier usually includes non-linear, high-dimensional computation, and human behavior and the environments that are hard to model.

Instead of dealing with direct estimation, we consider a substitute measure that represents the "appropriateness" of a current gesture and environment as a recognition target. For this purpose, we introduce a *condition vector* , and the space where the vector lies is the *condition space*. Our key idea is that, if we take a sufficient number and variety of dimensions for the condition space, the "appropriateness" can be well approximated and we work similarly to the degree of how far assumptions are satisfied. Components of the condition vector, for example, would be the user's standing position, velocity, or the color of clothes. Moreover, some indices about jitters or the signal-to-noise ratio of the pixels' values on captured images can also be used.

Theoretically, as we consider greater numbers and variations of dimensions, the appropriateness is more accurately approximated. However, we have computational problems in handling high-dimensional features, e.g., the curse of dimensionality; therefore, we need to design the instances of condition vectors carefully.

3.1 Predicting Gesture Recognition Performance

Once the condition space is composed, the next problem is to decide what kind of measure is appropriate for "appropriateness". Our suggestion is that the use of recognition accuracy is simple and sufficient.

Let us consider the gesture recognition process as a multi-class classifier of an input gesture. More specifically, we assume an one-versus-the-rest configuration of binary classifiers; for each gesture class C_X, the gesture recognition works as a binary-class classifier F_X that outputs X or $\bar{X}$, indicating whether the input belongs to gesture C_X or not. Next, we categorize the combinations of recognition results and actual classes, as Table 1. For each category, we use the following four labels: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). The terms "positive" and "negatives" refer to the predicted gesture class, and the other terms "true" and "false" refer to the correctness of the recognition result.

Using these labels, the overview of condition space can be shown in Figure 2a. Our framework predicts the performance of F_X at a given query condition vector s as precision $P_X(s)$ and recall $R_X(s)$, which are defined as

$$P_X(s) := \frac{\#TP_X(s)}{\#TP_X(s) + \#FP_X(s)} \quad (1)$$

$$R_X(s) := \frac{\#TP_X(s)}{\#TP_X(s) + \#FN_X(s)} \quad (2)$$

$$P_{\bar{X}}(s) := \frac{\#TN_X(s)}{\#FN_X(s) + \#TN_X(s)} \quad (3)$$

$$R_{\bar{X}}(s) := \frac{\#TN_X(s)}{\#FP_X(s) + \#TN_X(s)}, \quad (4)$$

where # means the number of neighbors, e.g., $\#TP_X(s)$ means the number of samples labeled TP_X around s. Note that these recall and precision are functions of s. In the pattern recognition field, the recall and precision are usually used as indices for average performance over all possible conditions. However, our proposed functions calculate recall and precision around s. Using these functions of s, we can predict and compare the actual performances of the two conditions.

The final problem is how to calculate the functions: $\#TP(s)$, $\#TN(s)$, $\#FP(s)$, and $\#FN(s)$. These functions count the number of samples around any given query point s in condition space. One of possible way is learning by example; we can collect gesture recognition results as training samples where each sample is a set of predicted gesture class, the actual gesture, and its condition vector. Once a sufficient number of training samples are collected, we have several methods to estimate the values of these functions at each position in the condition space. The details of the algorithm that we used in our experiments are described in Section 4.1.

3.2 Navigating the User to an Ideal Condition

It is widely recognized that the usability of an interface will be increased by giving any appropriate feedback to the user within a reasonable time(Nielsen, 2009). Many gesture interfaces already give supplemental information as feedback, such as the confidence or likelihood of the recognition result. This internal information helps the user to diagnose what is going on, and to find the reasons for the errors.

Our cooperative recognition framework proposes the following two additional feedbacks:

1. Feedback to show *"appropriateness" of current condition*, which informs the user whether the current condition is desirable for gesture classifiers.
2. Feedback to show *the method to improve appropriateness*, which is synthesized by a gesture recognition system for the purpose of leading the user towards changing his/her behaviors, or environment so that sufficient gesture recognition accuracy is obtained.

Feedback is achieved by showing the estimated performance for the current condition vector by using our recall and precision function. For example, we often use a kind of a cursor or an avatar to show the results of gesture recognition. In these cases, we can obtain feedback of the appropriateness of the current condition by changing the color or size of the cursor or the avatar.

Figure 2b shows the overview of the second feedback. This feedback is achieved by two tasks: a search task to find a better condition s_2 than the current condition s_1 and an inform task to navigate the user on the way to change the condition. There can be several strategies for these tasks, and their implementation strongly depends on the specific application requirements.

However, the design of these two tasks is formulated as follows. For the search task, a possible strategy is as follows. First, we prepare the goal condition candidates, e.g., move to the left by one step, move a hand downward by 20 degrees. Then, we compare these candidate conditions with the current condition and choose the best one. This comparison can be easily achieved by referring the values of estimated recall and precision. For the inform task, we propose the following two visual approaches:

Symbol-based Feedback. This feedback uses symbols, such as an arrow-shaped CG. For example, to encourage the user to move to the left by one step, this feedback displays an arrow-shaped CG that gives the user the distance and direction to move.

Example-based Feedback. This feedback uses a previously captured image or video as a precedent. The image or video can give a concrete explanation of how the previous user improved its condition.

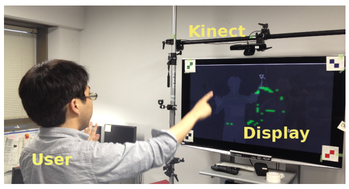


Figure 3: Overview of experimental system.

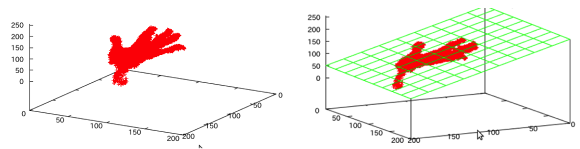
4 EXPERIMENT WITH EXPERIMENTAL GESTURE INTERFACE

We built a prototype gesture system to verify our framework.

Although the characteristics of the gesture classifiers and system usability heavily depend on their application, we think this example is commonly applicable to other gesture interfaces.

Figure 3 shows the overview of the prototype system design that provides a drawing application with the following two functions: drawing a line by using a "pointing hand" gesture, and erasing lines by the "opened hand" gesture. This interface uses the display to show the virtual canvas for drawing, and also to show the silhouette of the user. The user can look at this display as a mirror and can draw something naturally with his/her gestures.

This gesture interface recognizes the user's gesture from a combination of their upper body posture and hand shape. For this recognition process, we use a Kinect sensor to measure the user as color and depth images. For posture recognition, we use the skeleton tracker implemented by OpenNI library(OpenNI or-



(a) Input 3D depth data (b) Result of plane fitting

Figure 4: Example of hand fitting.

ganization, 2010). For hand shape recognition, we implemented the following four–step algorithm:

1. Find the hand regions on color and depth images by using the skeleton information obtained by OpenNI library.
2. Convert the 3D points of the hand into a normalized 2D image. To remove the wrist area, and also to analyze the hand shape, we find a plane that corresponds to the palm of the hand. We use the RANSAC algorithm(Fischler and Bolles, 1981) to find this plane. Figure 4 shows an example of the input 3D points around the hand region, and the obtained hand plane. We also use principal component analysis for the points onto the plane to find the hand orientation and its size. Using these orientation and size parameters, we projected 3D points to the normalized 2D image plane.
3. Count the number of fingertips. This is done by a simple template matching algorithm on the normalized 2D image.
4. Classify the hand shape. This process is achieved by using the number of the fingertips and the distribution of the 3D points.

As mentioned in Section 3, our scheme can help to improve the recognition performance if the condition space contains "easy" and "difficult" regions. The improvement does not depend on the performance of a recognition algorithm. Therefore, a simple algorithm such as our prototype system is sufficient for the evaluation of our proposed idea.

4.1 Estimators for Precision and Recall

We built estimators of precision and recall, as we mentioned in Section 3.1. For simplicity, this study focuses on recognition of hand shape only.

First, we categorized possible gestures into three classes according to the hand's shape: "opened hand" shape as C_A, "pointing hand" shape as C_B, and other shapes as C_Z. Simultaneously, we assumed that target gestures interface uses three binary classifiers: F_A, F_B, and F_Z.

Then, we defined the condition vector with the following parameters: user's forearm direction, 3D position of hands, 3D position of feet, speed of hand

Table 2: Training dataset.

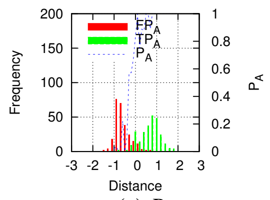
Outputs of classifiers		Number of training samples					
		$\#A$	8033	$\#B$	8086	$\#Z$	8799
$F_A(s)$	A	$\#TP_A$	6717	$\#FP_A$	180	$\#FP_A$	702
	$\bar{A}$	$\#FN_A$	1316	$\#TN_A$	7906	$\#TN_A$	8097
$F_B(s)$	B	$\#FP_B$	200	$\#TP_B$	4812	$\#FP_B$	697
	$\bar{B}$	$\#TN_B$	7833	$\#FN_B$	3274	$\#TN_B$	8102

movement, and depth image quality. For the last parameter, we estimated the image quality from the number of pixels where the distance is not acquired by the Kinect sensor. For other parameters, we use the skeleton information obtained by OpenNI library.

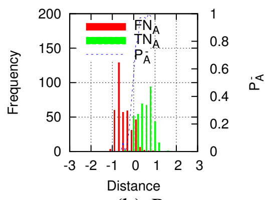
Using this condition vector and corresponding condition space, we captured in a total of 15 min data sequences from four participants. Then, we collected about 24,000 training samples, which consisted of a pair of condition vectors, label of gesture, and outputs of classifiers. Table 2 shows the details of them. Note that to acquire the samples is not an onerous task. For example, to acquire the samples labeled gesture C_A, we just asked the participants to use the target gesture-interface freely while keeping an "opened hand". Then, we recorded a pair of condition vectors of the participants and output of the classifiers at the same time, e.g., when the classifier F_A outputs A, we automatically assign TP_A for its sample. Using this method, we can easily prepare a huge number of training-sets and labels. No tedious tasks, like annotations by human hand, are required.

Then, we built the estimators for precision and recall. To reduce the computational cost for real-time processing, we used a support vector machine (SVM) for approximation. First, suppose you have a classifier F_A. The goal is to decompose the condition space into two areas: where F_A works accurately, and where F_A works inaccurately. When the actual gesture is A, the outputs of F_A are grouped into two categories: TP or FN, where TP means recognized as A correctly, and TN means the gesture is misrecognized as C_B or C_Z. We use the SVM to find the hyperplane that separates these TP and FN categories for C_A when the actual gesture is A. Using this hyperplane, we assume that if the condition vector is more on the TP side, the ratio of $\frac{\#TP}{\#FN+\#TP}$ will increase. In other words, we can substitute the signed distance from the hyper-plane as the recall R_A of F_A. Similarly, we can build a total of four hyper-planes for F_A: TP–FN and TP–FN for actual gesture A, and TN–FP and TN–FN for actual gesture $\bar{A}$. These correspond to four estimators: R_A and $R_{\bar{A}}$ for recall, and P_A and $P_{\bar{A}}$ for precision.

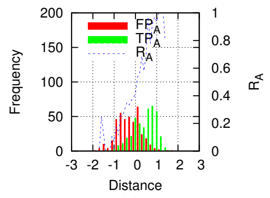
As an evaluation of this approximation, we measured the histograms of TP, TN, FP, and FN in-



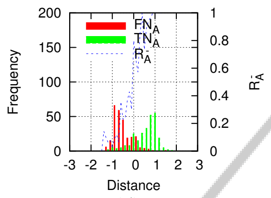
(a) P_A



(b) $P_{\bar{A}}$

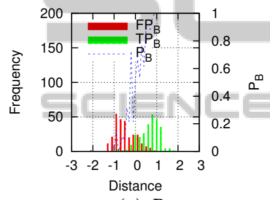


(c) R_A

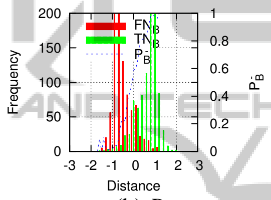


(d) $R_{\bar{A}}$

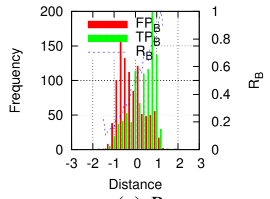
Figure 5: Learning results of gesture A: each graph shows the relation between P or R and the histogram of TP, FP, TN, and FN.



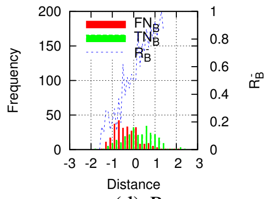
(a) P_B



(b) $P_{\bar{B}}$



(c) R_B



(d) $R_{\bar{B}}$

Figure 6: Learning results of gesture B: each graph shows the relation between P or R and the histogram of TP, FP, TN, and FN.

stances and corresponding estimated recall and precision. Figure 5 shows the result of F_A and Figure 6 shows the result of F_B. In these figures, the x-axis is a signed distance between the SVM's hyperplane and data sample. In the figure 5a, the red bar shows a histogram of $\#FP_A$, and the green bar shows a histogram of $\#TP_A$, while the blue line is the ratio of each pair of red and green bars. This ratio means P_A at the signed distance. As the figure 5a shows, the SVM separates the condition space into FP_A and TP_A regions, and P_A becomes higher as the signed distance becomes bigger. The same relationships are confirmed in the other categories $\bar{A}$, B, and $\bar{B}$, and other classifiers F_B.

Using these relationships, we confirm that we can estimate P and R only from the signed distance of the condition vector. As an additional evaluation result, Figure 7 shows the relationship between the estimated recall and its input depth images. It shows that as the estimated recall decreases, the undesirable factors, such as noise and self-occlusion, appears frequently.

These results show that our approximation approach is practicable and valid.

4.2 Visual Feedback to the User

As feedback for navigating the user to a desirable condition, we implement a symbol-based feedback. First, we prepare feedback candidates; each candidate is a pair of user's actions to improve the condition, and corresponding symbols that can navigate the user to do the action. Figure 8 shows some samples of the symbols. Frame #2822 of Figure 8 shows a feedback that navigates the user to change the orientation of his/her fingertips. Frame #3096 of Figure 8 shows another feedback that navigates the user to move to his/her right side. These arrow-shaped object

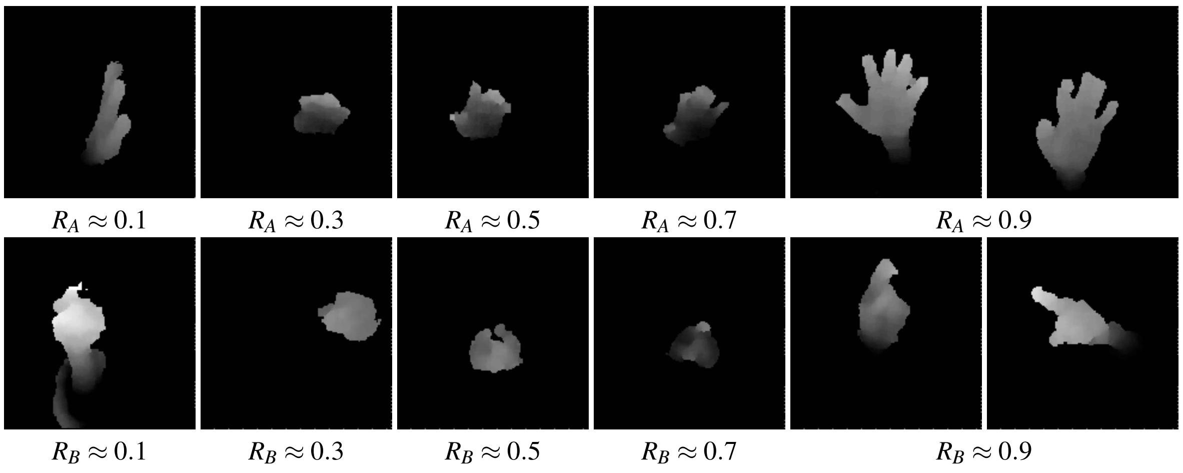


Figure 7: Example of recall estimation: this figure shows captured depth images sorted by R_A (top) and R_B (bottom). As the R_A or R_B increases, the image errors, such as pixel noise and self-occlusions, drastically reduce.

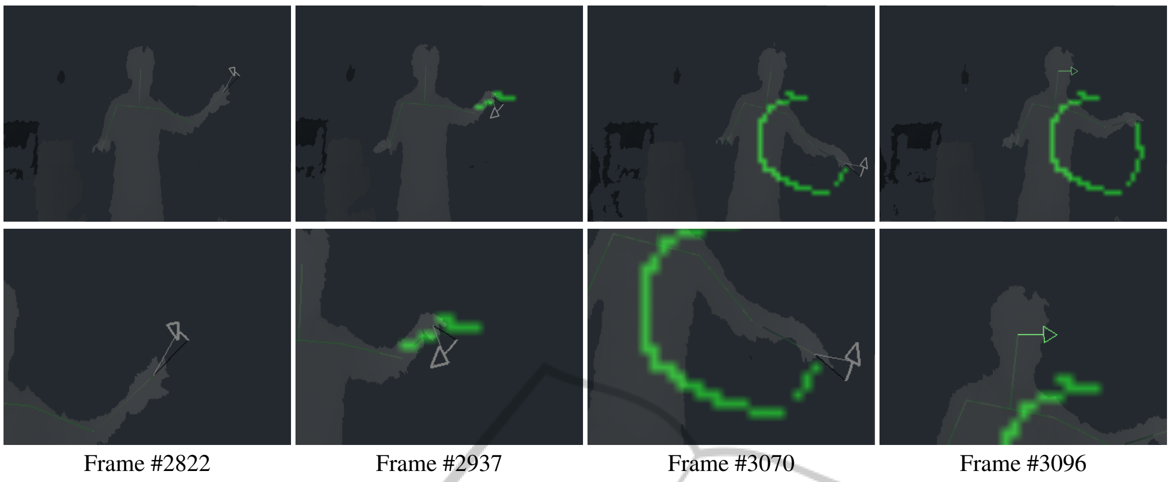
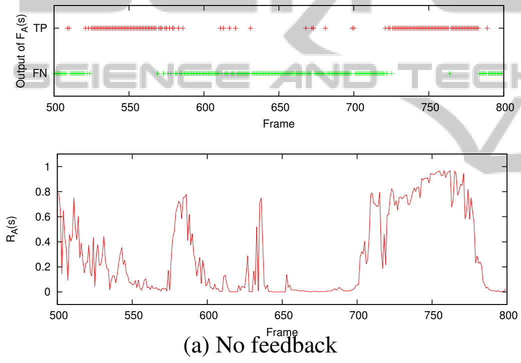
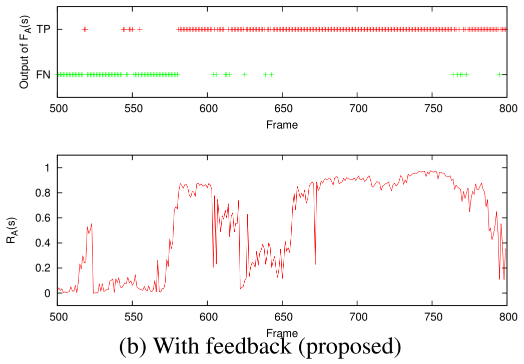


Figure 8: Screen-shots of experiment system: (top) screen-shots; (bottom) zoom-up images round the feedback.

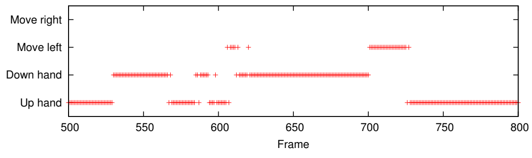
are overlaid on the silhouette of the user in real-time.



(a) No feedback



(b) With feedback (proposed)



(c) Displayed symbols for feedback

Figure 9: Estimated recall and the classified results: (a) without the feedback, the situation sometimes falls into a "bad" condition, where the estimated recall R_A becomes low, and the classifier outputs wrong results. (b) with the feedback, our navigation can retain a "good" condition, where the classifier outputs correct results.

The user can look at this interface as a mirror; seeing himself/herself naturally from this silhouette, and easily understand that the meaning of the arrow-shaped object is the direction and amount of required action.

4.3 Evaluation of Recognition Performance and Usability

In experiments, we focused on the interval during which the participant uses gesture A, and recorded the estimated recall $R_A(s)$ and precision $P_A(s)$, and the actual accuracy of the classifier's output $F_A(s)$.

We prepared four symbol-based feedbacks for the navigation: move a hand upward/downward, and take one step to the left/right. These feedbacks are synthesized as the arrow-shaped objects, as shown in Figure 8. We asked three participants to use our experimental gesture interface and recorded the data sequences for approximately 15 min.

First, we compared the influence of our feedback on these values. Figure 9a shows the recorded sequence in the case without feedback. This is 300 frames (10 seconds) sequence. The above red-graph shows the time sequence of $R_A(s)$. As this graph shows, the value of $R_A(s)$ changes dynamically. This means that the condition vector s is drastically varying between ideal and undesirable conditions. As a result, the recognition result of F_A also varies between accurate (labeled as TP) and inaccurate (labeled as FN). In contrast, with our visual feedback, the performance was improved. Figure 9b shows the case with feedback, and Figure 9c shows the displayed symbols for the feedback. The graph shows that the estimated recall $R_A(s)$ with feedback is higher than without feedback, and the recognition result $F_A(s)$ retains an accurate (TP) output.

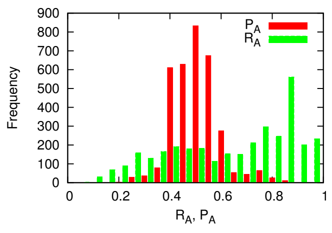


Figure 10: Histogram of R_A and P_A, without user navigation.

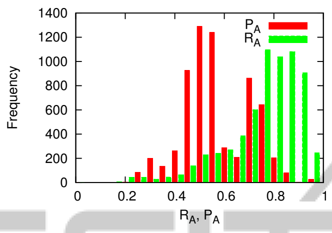


Figure 11: Histogram of R_A and P_A, with user navigation.

Table 3: Performance of F_A.

	Precision	Recall
No feedback	0.59	0.70
With feedback (proposed)	0.61	0.88

We compared the histograms of R_A and P_A during the experiments. Figures 10 and 11 show the cases without and with feedback, respectively. Comparing those two histograms, these results show that our feedback scheme can improve situation in terms of $R_A(s)$.

We also evaluated how our approach can improve the actual performance. Table 3 shows the actual recall and precision through the experiments. In this experiment, we use visual feedback to navigate the participant mainly for increasing the estimated recall. As a result, the recall is improved from 0.70 to 0.88, while precision remains unchanged. This means that our proposed scheme can predict the ideal situation in terms of recall and navigate the user to the ideal situation.

From these results, we demonstrate that our cooperative recognition scheme is practicable and efficient in increasing the accuracy of gesture recognition.

4.4 Discussion

The above experiments support our assumption mentioned in Section 3: the system performance in the neighbor of a ”condition” is similar to the system performance of the ”condition”. By the symbol-based feedback method, we get better performance when getting further in the direction of the ”easy” condition.

As far as the above assumption holds, our scheme can be applied to a wide variety of recognition systems, because we do not need to change the inside of a system. Our scheme also gives the users the know-how for using a system, i.e., how to use a system with good performance, and it also saves the user’s time of trial and error for learning how to use the system. We believe that this scheme can greatly increase the usability of the gesture interface.

5 CONCLUSION

This paper proposed a cooperative gesture recognition scheme for a gesture-based user interface, which navigates the user towards making gesture recognition more accurate. Our experiments showed its practicability and effectiveness.

We plan to apply and evaluate our gesture recognition scheme with different types of gesture interfaces in the future.

ACKNOWLEDGMENTS

This work was supported in part by the JST-CREST project “Creation of Human-Harmonized Information Technology for Convivial Society.”

REFERENCES

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395.

Jacko, J. A. and Sears, A., editors (2003). *The Human-computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

Moeslund, T. B. and Granum, E. (2001). A survey of computer vision–based human motion capture. *Computer Vision and Image Understanding: CVIU*, 81(3):231–268.

Moeslund, T. B., Hilton, A., and Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Comput. Vis. Image Underst.*, 104(2):90–126.

Nielsen, J. (2009). Ten usability heuristics. http://www.nngroup.com/articles/ten-usability-heuristics/.

OpenNI organization (2010). *OpenNI User Guide*. OpenNI organization.

Pavlovic, V. I., Sharma, R., and Huang, T. S. (1997). Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:677–695.

PrimeSense Inc. (2010). *Prime Sensor^{TM}NITE 1.3 Algorithms notes*. PrimeSense Inc.

Ren, Z., Yuan, J., Meng, J., and Zhang, Z. (2013). Robust part-based hand gesture recognition using kinect sensor. *Multimedia, IEEE Transactions on*, 15(5):1110–1120.

Shotton, J. and Sharp, T. (2011). Real-time human pose recognition in parts from single depth images. *IEEE Conference on Computer Vision and Pattern Recognition (2011)*, 2(3):1297–1304.