Realtime Pointing Gesture Recognition and Applications in Multi-user Interaction

Hoang-An Le[1], Khoi-Nguyen C. Mac[1], Truong-An Pham[1], and Minh-Triet Tran[2]

[1] Advanced Program in Computer Science, University of Science, HCMC, Vietnam
{lhan,mcknguyen,ptan}@apcs.vn
[2] Faculty of Information Technology, University of Science, HCMC, Vietnam
tmtriet@fit.hcmus.edu.vn

Abstract. Pointing is a common gesture of human. Indeed, people tend to involve pointing action in their daily activities using not only bare hand but also with gloves or a kind of pointers like pens, rulers, long sticks, batons, etc. Thus, in this article, the authors propose a new concept of interaction that is centered by the natural gesture of human and a method to detect it under various circumstances. Different from some common approaches which rely on predefined skin color or markers, the proposed method can segment and detect any pointer tip and allow multiple objects to be processed at a time. The method can run with the average accuracy of 91.0%. In case of multiple users using different pointing objects, the accuracy is slightly reduced to 87.9%. The running time is at most 17.14 ms for 9 objects being processed in parallel, and thus can be applied for real time constraints.

Keywords: pointing, gesture, human computer interaction, natural interaction.

1 Introduction

Human-Computer Interaction (HCI) provides natural ways of communication between humans and computers. This means proposed methods should be as similar to the way people communicate as possible. This motivates the development of methods such as speech recognition (to understand human natural language) [9], eye tracking (to understand human's facial expression) [15], or gesture and action recognition (to understand human body languages) [1], etc.

The popularity of pointing action in human's daily communication inspires the authors to propose a new idea of interaction method that based on the concept of human's pointing action. The interaction involves the pointing gesture which has different meanings in different contexts.

This problem is a topic of human gesture recognition whose common approach is to (1) analyze sequences of captured images, (2) detect hand and finger portions, and (3) classify the sequences to some categories that have similar semantic. The classification is based on machine learning method and does not require

A. Selamat et al. (Eds.): ACIIDS 2013, Part I, LNAI 7802, pp. 355–364, 2013.

insight understanding of the gestures' special properties which if being exploited, lead to a simpler but efficient solution. Besides, the detection of hand and finger is based on the assumption of predefined values such as skin color [3, 10, 2] or trained markers [8, 14]. Since people tend to use extra objects, or pointers (police's signal light batons, presenters' sticks, etc.) to point at something that they cannot reach, these methods reduce the naturalness of the interaction (as they require users to wear special markers or perform preliminary configuration) and thus become inefficient [6].

As observed from human-human communication, the pointing actions consists of 3 special characteristics: (1) the sharpness (level of protrusion of pointing tips), (2) the first-appeared point (time instant that a tip is detected), and (3) the farthest point (distance from a pointing tip to the frame edge that it appears.) Exploiting the three characteristics, the authors propose a lightweight method that supports all means of pointing interaction regardless of the kinds the objects should be. Instead of using color and trying to detect the pointing tip in every captured frame, the proposed method based on the special geometry shape of pointing objects, i.e. protrusion, to detect the tips and tracking them through frames series, and thus can be performed in real time.

2 Related Work

There are several approaches in HCI such as developing hardware devices like keyboards, mice, cyber-gloves, magnetic tracking devices, etc., or implementing environment-dependable systems using computer vision such as skin color, fingertips or full-body detection, etc. In the scope of this paper, the authors focus only on hand-finger based interactions which provides natural connection between human and computers [4, 7, 13, 5]. In such systems, there are two kinds of approach: sensing based and vision based. The vision-based approach requires a single or multiple cameras, color or infrared, and color gloves or markers. This approach's accuracy, yet, has some drawbacks since it depends on lighting condition and hard to be maintained for sudden changes of surrounding environment [5]. The sensing-based approach, which gives robust performance because of using electronic devices, is limited in only touch interaction [5].

Hand-finger tracking is a noticeable topic of HCI. The goal of tracking algorithms is to predict and estimate the object's position. The approach requires several constraints in order to achieve high performance [17]. For color segmentation-based methods, the performance depends on the color of wearing gloves; for wave-let based methods, the computational cost is expensive and cannot be used for real-time application; for contour based methods, it requires restricted backgrounds to run; for infrared segmentation, the availability of expensive infrared cameras is needed; for correlation-based methods, it is necessary to explicitly setup the stage beforehand; for blob-model based method, the constraint is about the maximum speed of hands' movement [17].

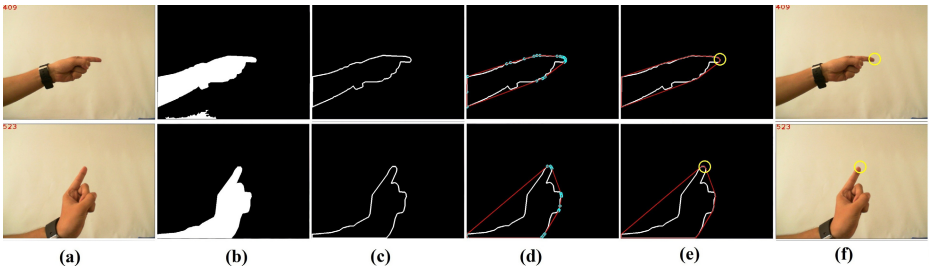

Fig. 1. Pointing tip detection method. (a) The pointing tip; (b) segmentation phase: a foreground mask of the pointing object; (c) pointing object contour; (d) the convex hull of the object contour with points passed through protrusion level k; (e) the farthest point among points in convex hull; (f): the pointing tip is detected.

3 Proposed Methods

Pointing action is popular in human's activities because it helps to identify objects, get audience's focus, or within different contexts, to select things, direct ways, emphasize, etc. To overcome the limitation of other methods that based on assumptions of skin colors, hand shapes or markers, this article detects pointing actions based on the pointing objects' silhouettes, level of protrusion, and distance to image frame's boundary. As observed, the desired points are the farthest points in an image frame because people tend to stretch their arms to reach or point at something (it is unlikely to point at something by just slightly extending the hands.) In addition, the points should satisfy the sharpness constraint as people usually choose pointing tips that stick out of an object and have substantial level of protrusion like a stick, a pen tip, etc.

The proposed method is based on the special geometrical shapes of pointing tips, which are protrusions of objects such as fingertips, pen tips, etc. The method consists of two steps: segmentation (Figure 1b) and identification (Figure 1c, d, e.) The pointing object is extracted from a captured image using background subtraction method called Codebook [11]. The pointing tip is detected from the segmentation based on its level of protrusion [16]. To reduce the cost of computation, not all points in the object's contour are chosen to computed but only the points that reach into the image (computed from the frame margin it had appeared) farther than a predefined threshold (Figure 1c, d, e).

Instead of trying to detect the pointing tip through every frame which may lead to low performance and error-prone, the authors propose to use the Kalman filter [18] to calculate the optimal coordinates of the detected point based on previous results.

3.1 Pointing Objects Segmentation

The authors use background subtraction to extract pointing objects from the other image parts which plays as a stable background. However, as the background is not exactly a constant image but contains two parts: small static

environmental regions and a screen portions which are changed when users trigger an event. On this manner, the background subtraction is carried out not only at the beginning but also right after an event is generated by users to re-train the background model. It should be noticed that the system does not need to continuously update the background but only when the system accepts an event and have the visual content changed.

Capable for both illumination change and moving-background training, the authors propose to use the Codebook algorithm [11]. When re-training the background, some portion of the new background may be occluded by a user's pointing object. To solve this issue, the authors correct the occluded area in the background by applying the homographic transform on the current screen content and mapping the corresponding region into the background.

The background subtraction phase runs through the following steps

- Periodically study a model of the background. During background learning, if a codeword is not accessed for a period of time, it is deleted and replaced by nonstale, i.e. active entries.
- Obtain foreground objects by using the learned model to segment it out of the background.
- Periodically clean out stale codebook entries and update the learned background pixels after a period of time.

3.2 Pointing Tip Detection

The purpose of this section is to identify the hot spot of each pointing object detected from previous phase and the location where users are pointing at. For each pointing object, the tip detection runs through 4 consecutive steps:

- Firsly, because a hot spot can only occur at the boundary an object the authors have the contour of each pointing object extracted.
- The set of points that forms the convex hull of the contour is then computed. Since the pointing tip is usually the part that sticks out of an object, this set is the candidate points for the desired hot spot.
- For each point in the candidate set, the authors compute the level of protrusion and only those that pass a predefined threshold can be present.
- Finally, among the candidate, the hot spot is selected to be the farthest point computed from the frame edge that it first appears.
- In order to improve the accuracy, the authors apply the Kalman filter that uses the previous knowledge of the hot spot to correct the new detected point from noises that may appear.

Contour Extraction. Let $\mathcal{M}$ be the foreground mask obtained from any arbitrary frame after the segmentation phase, $\mathcal{B} \in \mathbb{N}^2$ be a blob or a connected component in $\mathcal{M}$ and $\mathcal{C} = \beta(\mathcal{B})$ be the contour of the blob $\mathcal{B}$,

$$\mathcal{C} = \beta(\mathcal{B}) = \mathcal{B} - (\mathcal{B} \ominus K),$$

where $\mathcal{B} \ominus K$ is the erosion of $\mathcal{B}$ and a 3×3 matrix K, which consists of all points p such that K translated by p, is contained in $\mathcal{B}$, i.e. $\mathcal{B} \ominus K = \{p|(K)_p \subseteq \mathcal{B}\}$. The result of each pointing object in Figure 1b is shown corresponding in Figure 1c.

Convex Hull Computation. As the matter of fact, it is possible to find the pointing tip right after the object contour is extracted. However, since the contour may become zigzag due to noises from background subtraction Figure 2a, b, or complicated Figure 2c because of the shape of the pointing object used, the convex hulls $\mathcal{H}$ of the countour $\mathcal{C}$, $\mathcal{H} \subset \mathcal{C}$ is computed.

Since the pointing tip is a protrusion part on an object's contour, the convex hull computation does not leave out the correct tip (see Figure 1d, e and the third column of Figure 2) but only narrowed down the number of points to be checked. Thus, the step helps to increase the system performance.

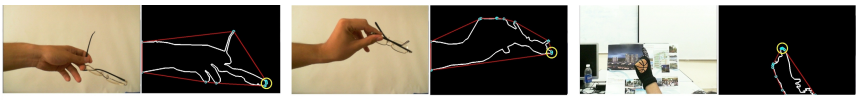

Fig. 2. Convex Hull

Level of Protrusion. Based on the sharpness level k of each point, the candidate set is filtered to those that are sharper than a threshold level k_{thr}. The sharpness level k of a point is computed based on the angle it makes with 2 arbitrary points in the neighborhood:

$$k_i = p_1^- p_2^+ - p_2^- p_1^+$$

For each point $P_i \in \mathcal{H}$ in the convex hull of the contour, let $P_{i-k}, P_{i+k} \in \mathcal{C}$ be any two points on the contour, may or may not be in the convex hull. The authors have $\mathbf{p}^-$ and $\mathbf{p}^+$ are 2 vectors defined by:

$$\mathbf{p}^- = (p_1^-, p_2^-) = \overrightarrow{P_{i-k}P_i}, \text{ and } \mathbf{p}^+ = (p_1^+, p_2^+) = \overrightarrow{P_i P_{i+k}}$$

The threshold level k_{thr} is different for each kind of objects such as the threshold level of a pen-tip is smaller than that of a finger, etc.

Pointing Tip Detection. Let $\mathcal{S}_C$ be the candidate set narrowed down from the convex hull of the contour, i.e. $\mathcal{S}_C = \{P_i|P_i \in \mathcal{H}, k_i > k_{thr}\}$. Since people tend to stretch their arm to reach for the pointed objects, the pointing tips should be the farthest points among all the candidates. By farthest, the authors mean that the distant d of each candidate point is computed from the frame edge that the object first appears. For instance, candidate points on the object in the first row of Figure1 are measured to the left boundary whereas candidate points on the second and third row are measured to the bottom boundary.

Correction. Because of sudden changes in environmental illumination or occlusions, noises and errors may happen during the process. To bypass the wrong detection, the authors propose using a method to correct the detection result. By observation, within a short period of $10 - 500ms$ users usually have the pointing objects move in a stable direction. Thus, the Kalman Filter is selected to do the correction step. Using the Filter, the authors track the pointing tip from the first time it appears in the viewport, and in every step, the optimal point is computed from both the measurement, i.e. the newly detected result, and the result from previous steps.

4 Experiments and Results

4.1 Dataset Description

The test scenarios contain sets of video clips which are divided into 3 groups A, B, and C:

- Group A includes rigid objects whose shapes do not change much: bare hand ($A1$), finger ($A2$), gloved hand ($A3$), gloved finger ($A4$), and pen ($A5$.)
- Group B includes deformable objects whose shapes change significantly: deformable hand ($B1$), deformable finger ($B2$), glasses ($B3$), blinking semi-transparent stick ($B4$), and slider ($B5$.)
- Group C includes multiple hands ($C1$), multiple pens ($C2$), blinking semi-transparent stick and slider ($C3$), 3 users with different pointing objects ($C4$), and 5 users with different pointing objects ($C5$.)

The test video clips are recorded with $25 frames/ms$; each is 1000-frames long of 3 different resolutions 320×240, 640×480, and 1280×960 pixels. The pointing objects are chosen so that their colors are visually recognizable from the background. In addition, the test cases vary from single object to multiple ones to check the performance and accuracy under different conditions. Besides, the objects' colors and shapes are also picked randomly to guarantee that there is no predefined color or shape used in the test scenarios.

4.2 System Accuracy

The system accuracy, is calculated from the number of frames (out of 1000) giving correct results (Figure 3a.) By experiment, the accuracy over group 1 is 1.4% greater than group 2 and the accuracy over group 2 is 3.9% greater than group 3. It means that, among the 3 groups of test cases, rigid objects are the easiest one to processed while multiple objects used by several users are, on the other hand, the most difficult ones. The difference, however, is not significant as the complexity over the test scenarios increases. Therefore, the system is able to produce high accuracy (with mean of 91.0%) which is stable over several test cases (with standard deviation of 4.4%) and can be applied in real life situations.

4.3 System Performance

System performance is measured using the total running time of the test cases classified by the number of objects: 1, 3, 6, and 9 (Figure 3b). As the frames' resolutions increase, the running times also increase with the mean duration of 3.03ms over the first 2 resolutions, and 12.4ms over the last 2 resolutions. It means that the growth of frames' resolution increases the running time. Besides, the mean running times for the 3 resolutions (over the 4 test cases) are respectively 0.93ms, 3.96ms, and 16.39ms. Therefore the processing time is not affected by the screen's resolution but by the number of objects.

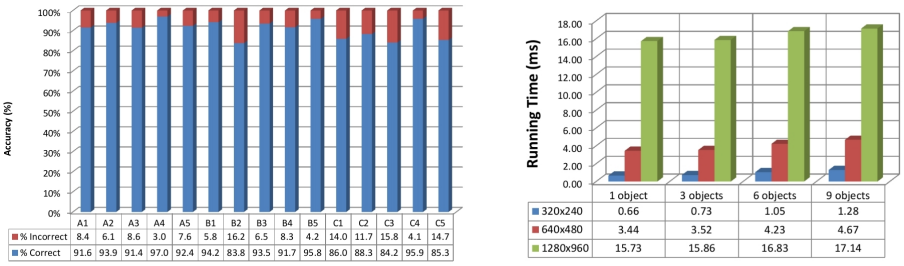

Fig. 3. Accuracy (a) and system performance (b) of different pointing objects

As the number of pointing objects increases, the processing time also increases. However, the two growth levels do not depend on each other because the processing phase for each object contains similar steps (background subtraction and contour extraction,) which are processed at the same time. Therefore, the processing time only increase by an insignificant amount as the number of objects increases.

On the other hand, the running time depends greatly on the frame resolution as the time is proportional to the number of pixels in a frame, i.e. when the resolution is doubled, the running time increases around 4 times. For the test scenario with the highest resolution, it requires 17.14ms at most. Hence, the system still satisfies real-time performance criteria in experiment's worst cases.

4.4 Discussion

Figure 4 shows some experiments of bare hands and fingers with their silhouettes. In these experiments, the hands and fingers, are kept rigid as they are moving. It appears that the detected regions match the one the users pointing to.

As an example of deformable objects, the authors use glasses and change their shapes so that the protrusive and farthest points are changed over time (Figure 5). Without using any tracking methods, the system detects new points with higher level of protrusion and lose focus on the original ones, which should be continued to be detected.

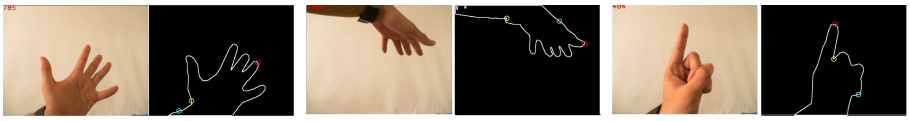

Fig. 4. Hands' silhouettes with detected points (red circle)

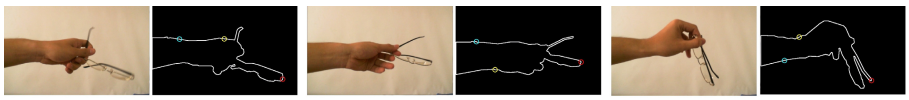

Fig. 5. Deformable glasses' silhouettes with detected points (red circle)

Another case that leads to system inaccuracy is shown in Figure 6 that describes the situation of two objects occluding each other. If the filter is not present, the overlapped point is lost after the occlusion; otherwise, the covered regions are remembered and continued to be tracked.

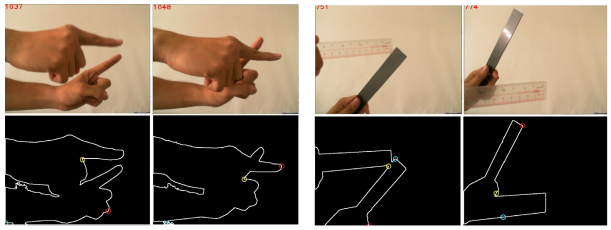

Fig. 6. Occlusion in multiple objects: fingers and rulers

5 Application

The Smart Interactive Map (SIM) [12] transforms normal physical map into a system that can understand users' pointing gestures and offer them information such as the best route, tourist attraction, restaurant, etc (Figure 7.) The system consists of several map stations at different locations in a certain area, each which connects to one processor, for parallel processing. It is designed to serve one user at a time to avoid occlusion caused by multiple users. Instead, two hands of a user are the cause of occlusion, but the frequency of this is low.

Although the system does not serve groups of users, it still shows the parallel processing property of the authors' proposed method. Beside the parallel processing property of the system, understanding pointing gesture of users' fingers or protrusive objects is the most important part of this system. By setting up SIM, the authors show two strong properties of the proposed method that are parallel processing performance and natural way in interaction.

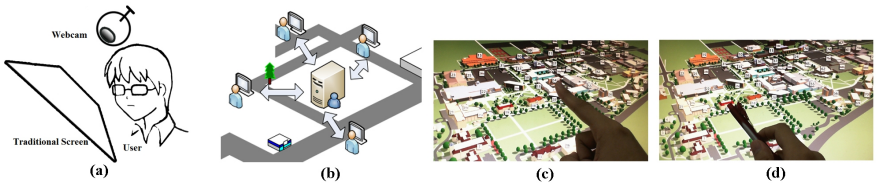

Fig. 7. Smart Interactive Map

6 Conclusion and Future Work

This paper introduces a new kind of interaction method: pointing gesture, to provide a high level of flexibility and naturalness, as it is commonly used. The proposed method can overcome existing obstacles of using predefined colors and shapes. It allows people to use any arbitrary objects to interact with the systems without learning their special features since the method depends on the protrusive regions of the objects.

By experiment, the system takes $17.14ms$ with an accuracy of 91.0% to process 9 objects in parallel. The error is due to lost tracking when the objects become non-rigid or when occlusion happens. As people tend to move pointing objects slowly and linearly in short periods of time, the author propose to use Kalman filter to further improve the method's accuracy.

This method can be used as a low cost replacement for current interactive systems. For example, the method can be applied to build (1) interactive map guiding systems in large areas, such as campuses, amusement parks, shopping malls, etc; (2) teaching or presenting systems, where users have to use pointing gestures most of the time; (3) entertainment systems which requires high flexibility while operating.

Acknowledgement. This research is supported by research funding from Advanced Program in Computer Science, University of Science, Vietnam National University - Ho Chi Minh City.

References

1. Aggarwal, J., Ryoo, M.: Human activity analysis: A review. ACM Comput. Surv. 43(3), 16:1–16:43 (2011), http://doi.acm.org/10.1145/1922649.1922653
2. Choi, S.-H., Han, J.-H., Kim, J.-H.: 3D-Position Estimation for Hand Gesture Interface Using a Single Camera. In: Jacko, J.A. (ed.) Human-Computer Interaction, Part II, HCII 2011. LNCS, vol. 6762, pp. 231–237. Springer, Heidelberg (2011)
3. Dawod, A.Y., Abdullah, J., Alam, M.J.: Fingertips detection from color image with complex background. In: The 3rd International Conference on Machine Vision, ICMV 2010, pp. 88–96 (2010)
4. Dietz, P., Leigh, D.: DiamondTouch: a multi-user touch technology. In: Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology, UIST 2001, pp. 219–226. ACM, New York (2001)

5. Do-Lenh, S., Kaplan, F., Sharma, A., Dillenbourg, P.: Multi-finger interactions with papers on augmented tabletops. In: TEI 2009: Proceedings of the 3rd International Conference on Tangible and Embedded Interaction, pp. 267–274. ACM, New York (2009)
6. Fernandes, B., Fernández, J.: Bare hand interaction in tabletop augmented reality. In: SIGGRAPH 2009: Posters, pp. 98:1–98:1. ACM, New York (2009)
7. Han, J.Y.: Low-cost multi-touch sensing through frustrated total internal reflection. In: Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology, UIST 2005, pp. 115–118. ACM, New York (2005)
8. Jones, M.J., Rehg, J.M.: Statistical color models with application to skin detection. International Journal of Computer Vision 46(1), 81–96 (2002)
9. Juang, B.H., Rabiner, L.R.: Automatic speech recognition - a brief history of the technology development. Elsevier Encyclopedia of Language and Linguistics (2005)
10. Kang, S.K., Nam, M.Y., Rhee, P.K.: Color based hand and finger detection technology for user interaction. In: Proceedings of the 2008 International Conference on Convergence and Hybrid Information Technology, pp. 229–236. IEEE Computer Society, Washington, DC (2008)
11. Kim, K., Chalidabhongse, T.H., Harwood, D., Davis, L.: Background modeling and subtraction by codebook construction, vol. 5, pp. 3061–3064 (2004)
12. Le, H.A., Mac, K.N.C., Pham, T.A., Nguyen, V.T., Tran, M.T., Duong, A.D.: SIM - smart interactive map with pointing gestures. In: 4th International Conference on Intelligent Human-Machine Systems and Cybernetics, IHMSC 2012, vol. 2, pp. 344–349 (2012)
13. Letessier, J., Bérard, F.: Visual tracking of bare fingers for interactive surfaces. In: Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology, UIST 2004, pp. 119–122. ACM, New York (2004)
14. Mistry, P., Maes, P., Chang, L.: Wuw - wear ur world: a wearable gestural interface. In: Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems, CHI EA 2009, pp. 4111–4116. ACM, New York (2009)
15. Model, D., Eizenman, M.: User-calibration-free remote gaze estimation system. In: ETRA, pp. 29–36 (2010)
16. Segen, J., Kumar, S.: Shadow gestures: 3D hand pose estimation using a single camera. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, p. 485 (1999)
17. Song, P., Winkler, S., Gilani, S.O., Zhou, Z.: Vision-Based Projected Tabletop Interface for Finger Interactions. In: Lew, M., Sebe, N., Huang, T.S., Bakker, E.M. (eds.) HCI 2007. LNCS, vol. 4796, pp. 49–58. Springer, Heidelberg (2007)
18. Welch, G., Bishop, G.: An introduction to the kalman filter (1995)