The Issues of 3D Hand Gesture and Posture Recognition Using the Kinect

Mohamed-Ikbel Boulabiar, Gilles Coppin, Franck Poirier

The Issues of 3D Hand Gesture and Posture Recognition using the Kinect

Mohamed-Ikbel Boulabiar[1], Gilles Coppin[1], and Franck Poirier[2]

[1] Lab-STICC, Telecom Bretagne, France
{mohamed.boulabiar, gilles.coppin}@telecom-bretagne.eu
[2] Lab-STICC, University of Bretagne-Sud, France
franck.poirier@univ-ubs.fr

Abstract. Besides the emergence of many input devices and sensors, they are still unable to provide good and simple recognition of human postures and gestures. The recognition using simple algorithms implemented on top of these devices (like the Kinect) enlarges use cases for these gestures and postures to newer domains and systems. Our methods cuts the needed computation and allow the integration of other algorithms to run in parallel. We present a system able to track the hand in 3D, log its position and surface information during the time, and recognize hand postures and gestures. We present our solution based on simple geometric algorithms, other tried algorithms, and we discuss some concepts raised from our tests.

Keywords: Gesture, Posture, 3D, Kinect, Interaction, Hand

1 Introduction

During the last years, we have seen a big interest in 3D gesture interaction in the research and the industrial field, many input devices and sensors were and are still being released to translate human movements into computer information. Sadly, many sensors either have complex systems for gesture recognition [11,7], takes a lot of computation power or still lack good recognition algorithms. Some studies show that the mouse is still unbeaten in its current use [2] and this motivates us to figure out new scenarios for gestures and postures systems [4].

3D gestures have also the specificity of not having a clear hardware timing of when a gesture starts and ends. In contrast with multi-touch devices that define the beginning and the end of the gesture by fingers touching the surface and leaving it, in 3D we do not touch physical objects and this is what makes the problem harder.

The increasing number of sensors and devices, and the emergence of new 3D visualization techniques like the 3D stereoscopy [14], pushes us to test a new approach in creating hand gestures and postures recognizers. We target a user commanding a system using a table and the space above. We take the object itself into consideration taking a part in the recognition method in a way different from just using physics simulation libraries [6,17]. We detail in this paper a simple and

real-time solution for recognizing gestures and postures, which can be embedded into other systems. As we take the manipulated object into consideration, the recognition becomes instantaneous and newer concepts start to emerge. We have chosen to mold geometric recognition algorithms towards our needs.

Our contributions are: 1. The fast system for tracking the hands from the 3D raw points data. 2. The use of the same geometric algorithms to detect both gestures. 3. The use of the same algorithms to detect postures by transforming the hand contour into an algorithm input. 4. The experimentation of other methods and ideas in the same context.

In this paper, we start by describing our base system for recognizing, tracking and logging the hand in 3D; then we describe how we have used and extended simple geometric algorithms for 3D gesture analysis and for hand posture recognition. We describe other tested algorithms and finally we discuss the recognition issues and provide some new ideas coming out from our applied study.

2 Kinect-based system for tracking and recognition

2.1 Installation

In our system as shown in fig. 1 we have used the Kinect as a depth sensing camera mounted in the ceiling above the user. We have decided to process the raw data directly to be able to optimize the pipeline and get the maximum speed. The standard Microsoft Xbox Kinect sensor pipes us a raw input of 640x480 3D points cloud at 30Hz frequency. We limit the captured zone to the size of 100x80cm and a 60cm of depth above the table because of the human reachability concerns [10] and table size. The Kinect is connected to an Intel Xeon computer with ubuntu 64bit installed. We have used the open source libfreenect library for kinect access in addition to OpenCV.

2.2 Tracking of the Blobs

To accelerate the hands detection and tracking, we have decided to do all the tracking on 2D surfaces and using well designed one pass per frame algorithms. So while keeping the 3D data of the Kinect on separate data structures, we have flattened the recorded 3d box of points into a single layered image (in comparison to a three layered RGB image) which can proceeded by OpenCV. We have used cvBlob library to label the blobs present on the scene then we apply the same algorithms described in [3] to track the resulted blobs between frames. We still have access to all the 3D information after the flattening operation since the single layer where blobs will be tracked in 2D has been duplicated in memory.

2.3 Extraction of the Hand

The blob we obtained form the hand and the arm, but as we only want information about the hand, we extract only that part from the bigger blob of the

Fig. 1. Kinect Install.

full arm and we get the position of the hand center. To extract it, we simplify the blob into a shape just fewer points, we try to mesure and compare distances between points and we select the longer segment. As the segment contains the hand farthest point and the point near the forearm, we select the one which is near the interaction zone center as shown in fig. 2. To select the hand center, we have selected a constant interval from the extreme point towards the other direction. The interval size varies by the hand vertical position.

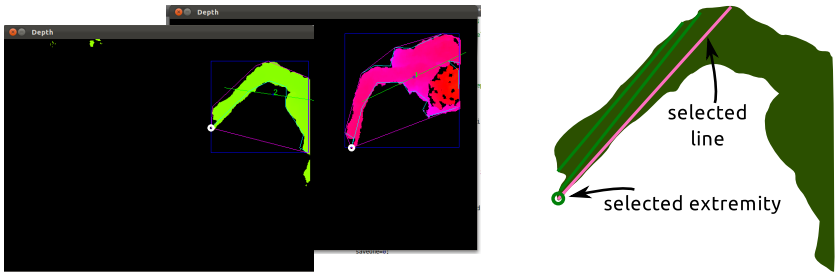

Fig. 2. Hand Extraction.

2.4 Logging of the Hands Information

We are able to process data in near real-time while recording the hand surface shape and the 3D coordinates of hand movements through time for further use in

posture recognition as shown (fig. 3). Data structures for tracking and identifying the hand are separated from 3D raw data, but we use them to select the zone to be recorded. We select a fixed rectangle around hand center and we verify whether it is inside the recording zone. We have used textual files to facilitate debugging and allow direct visualization using gnuplot. One problem faced with 3D sensors is that they can only provide the surface layer of an object, and not the 3D blob in itself. This means that when we speak about recording the hand, we record only the points of its surface which are between the sensor and the real hand. This limits for example recognizing what happens below the hand surface.

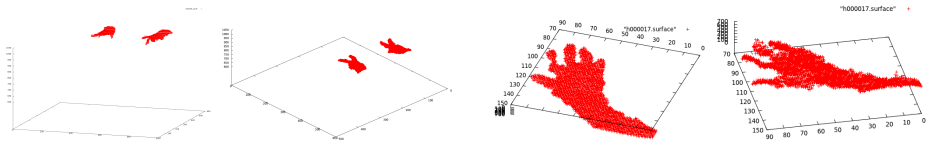

Fig. 3. The recorded scene including hands, and the extracted and recorded hand surface.

2.5 Application and Research Context

In the previous section we have described the 3D input handling part, but the general context where our system as described in fig. 4 is used is the maritime surveillance. The input handling and the maritime systems are connected through a software bus where we can pipe recognition results in one direction and the commands for the mode of recognition tuning in the other direction.

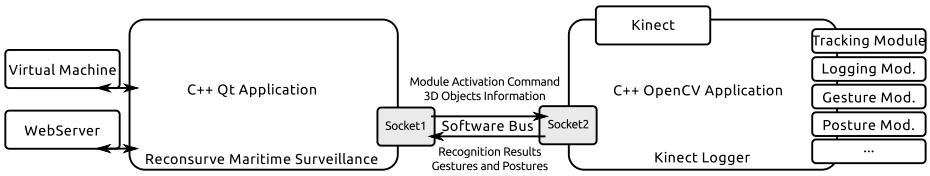

Fig. 4. Maritime suveillance system and Kinect Logger system architecture.

3 Recognition Methods

3.1 Use of Geometric Algorithms

When starting the development of our project, the performance was one of our biggest concerns so we focused on using algorithms that take the shortest computing time and we tuned them to our needs. We have studied the Rubine[13]

and the "1 dollar" families [18,9] of stroke recognizers. We have chosen to use the 1$ recognizer (or its variation "Protractor") for its simplicity and speed. We define geometric algorithms as those which use simple geometry operations and measurements in order to compute a distance value, in contrast to soft computing algorithms.

3.2 Our Use of 1$ Algorithm in Gesture Recognition

In our system, we track the hand center and the pointing finger. We have extended the 1$ algorithm to work on 3D strokes. Works like [5] or 3$ [8] have only used either a different algorithm or a still 2D recognition of 2D strokes performed in the 3D space, the work of Haubner et al. [5] in particular worked mostly on searching the flat space of a gesture. In our 3D adaptation of 1$, we have tested 3D strokes that can not be reduced to a simple plan.

3.3 Our Use of 1$ Algorithm in Posture Recognition

By posture, we define the current configuration of the hand similarly to Baudel et al. [1]. The 1$ algorithm is supposed to be used with mouse, touch screen or pen strokes. We have got the idea to keep using it but for hand posture recognition based on previous work we have made [3]. We have managed to make the hand contour as the input of the algorithm (fig. 5), then we have recorded a set of template, and the slightly modified algorithm have worked and we are now able to detect our set of hand postures, which are just a subset of the American Sign Language [1].

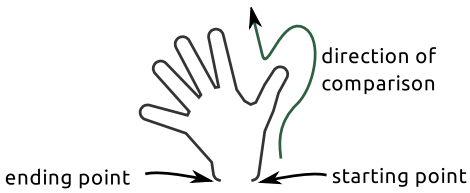

Fig. 5. Using 1$ in posture recognition.

3.4 Pointing 3D Objects on the Table

In our system, as we are able to track the hand center and its extreme, which can be the pointing finger, we have developed a mode where we track these two points in 3D and detect the direction pointed by the finger. As a quick and fast application, we computed the fixed position of the table (Z=constant) to simplify equations and detect where the user finger is pointing on the table shown in fig. 6 below.

[1] `http://en.wikipedia.org/wiki/American_Sign_Language`

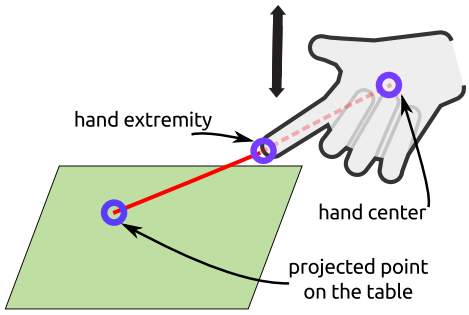

Fig. 6. Pointing on table.

4 Other Tested Algorithms

4.1 Extension of Angle Quantization Method in 3D

The angle quantization geometric algorithm [12] works by coding the stroke into a vector of values. These values calculate the parts being in a specified angular zone. The algorithm allows fast and high detection rate of strokes but fails in differencing between repeated stroke patterns like V, W and WW. These patterns have parts in the same angular zone, so even if they are repeated, the AQ algorithm can't differenciate between them. We have tried extending the AQ to the 3D space [2]

4.2 Application of the ICP Algorithm for Hand Tracking

We have tried using usual point cloud algorithms like Iterative Closest Point (ICP) for aligning the recorded hand on one of the templates and use the angles and positions given by the algorithm to compute the transformation and thus detect the gesture. The prototype code allowed us to get acceptable results but appeared to be very slow. The ICP algorithm was not intended for real-time use and discouraged us from continuing through that research area. We should note that during the tests, we have tried giving the algorithm the fixed part which is the hand back without the fingers. The use of this part makes better rotation recognition. The use of simpler geometric algorithms seems more appropriate.

5 User Experimentation

5.1 Definition of Gestures

In our prototype, and before thinking about how gestures can be natural, we tried recognizing the usual 3D strokes and we have defined 4 arbitrary and simple ones as shown in fig. 7 just to test our recognition algorithm. We have chosen 4 gestures that can not be reduced into a 2D plan by studying their principal components.

[2] `https://github.com/dylandrover/3D-AQ`

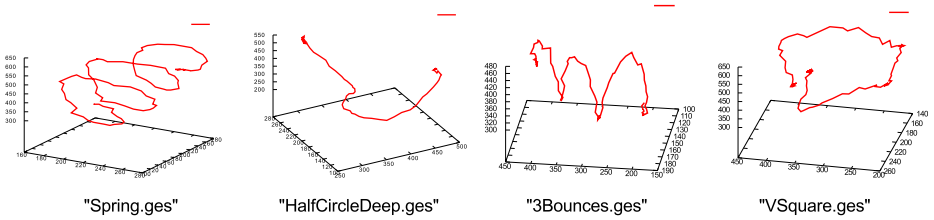

Fig. 7. A set of 4 pure 3D gestures arbitrary selected.

5.2 The Naturality of the Performed Gestures

During preliminary tests and recording of 3D command gestures, we have spotted a problem of memorability, which we think it comes from the background of human activities. Humans are very well used to write and draw on a 2D paper but not in space. Only skilled sculptors can interact with a three dimensional element. What we can do is touching an object, moving it, rotating it, and sometimes compressing it, but not commanding an object or a system with indirect gestures. The natural gesture in reference to a hand and an object should be classified into four basic families: (Touch, Move, Rotate, Scale) in reference to how we manipulate objects in nature [15]. We think that a hand interaction with objects need first to be categorized into one of these four classes, then we look further into sub-properties to achieve a fine-grained classification.

5.3 Benchmarks and Recognition Rates

We have tested the time it takes to compute the gesture after we finish recognition. It takes less than 60ms on our machine, and with our set of gestures. For the hand posture recognition, we have made prior tests in the past using the same posture recognition algorithm and an RGB camera, we were able to reach realtime recognition rates [3]. When using the Kinect, we have flattened the hand capture then extracted its contour and we are able to reach a similar but not yet evaluated recognition rates.

6 Recognition Issues and Ideas

6.1 Gesture Parsing (Start and End)

We have been faced though by the problem of real-time gesture parsing. Knowing when a gesture starts and when it finishes pushed us first into using a foot pad to tell the system when it must start considering the recorded 3D points as part of the gesture, and another foot click to stop recording. The system delivers after start and stop the detected gesture. A priror work [16] used a posture to start an interaction. The work could be improved by inheriting ideas from

[3] `http://youtu.be/AbNKPBCw4EU`

speech recognition system as for detecting the commands between two silences. In our case, the detection of a possible gesture will be performed between two stationary positions while posture detection will be performed in them as shown in fig. 8.

Fig. 8. The gesture parsing by seeking big difference in hand movements.

6.2 Recognition Simplification using Object Position

While searching for methods to easily detect rotation, we have tried first detecting it in the hand itself based only on the point cloud transformation. This appeared a computation hog using the ICP algorithm. Then we have questioned detecting such gesture without the presence of a target object. Having an object interacting with the hand, and willing a rotation, means that the hand-object position will change and the line linking their respective centers will rotate. We can know about the center of an object, and we track the hand, so we know where it is. We are able to detect rotation instantly using this method.

6.3 Spheres of Interaction

The interaction we want to promote is the one with objects because that is where natural interaction goes instead of commands or posture interaction. We have proposed in the previous paragraph that we can simplify algorithms using the hand and object positions. Here, we extend this approach to define interaction zones or spheres of interaction around the object as shown in fig. 9. We dedicate the first sphere around the object, which is bigger than it by two times the hand thickness, to direct manipulation of the object by moving, resizing, rotating and selecting it, and the second layer to accept indirect commands to be applied on it. When the hand is not in these zones, we ignore its posture and movement which is far from the object.

7 Conclusion and Future Work

In this paper, we have shown that we can provide a system capable of tracking hand movements in the space above the table, logging information, and recognizing gestures and postures in real-time. The most relevant feature of our work is that it is able to reach good performances using only very simple algorithms.

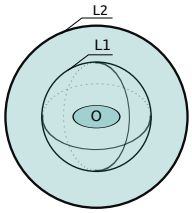

Fig. 9. Spheres or layers of interaction around the object, their size depend on the object and hand ones.

Use them in a new way. Our approach can help other researchers by giving them the tests and examples that worked and those which ended up with some constraints.We think that the simplification of recognition using information about the object along with the hand, and the position of these two is an idea to consider in further studies. Future studies will target making the system more robust and improve its recognition capabilities. We plan also to target more user testing and perform new benchmarks.

8 Acknowledgment

We would like to thank Dylan Drover for the internship he has made in our laboratory through the LEIF Exchange program. He worked on exploring the extension of 2D Angle Quantization algorithm to 3D and on the mentioned ICP tests.

References

1. Baudel, T., and Beaudouin-Lafon, M. Charade: Remote control of objects using free-hand gestures. *Commun. ACM 36*, 7 (July 1993), 28–35.
2. Bérard, F., Ip, J., Benovoy, M., El-Shimy, D., Blum, J. R., and Cooperstock, J. R. Did "minority report" get it wrong? superiority of the mouse over 3d input devices in a 3d placement task. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II* (Berlin, Heidelberg, 2009), INTERACT '09, Springer-Verlag, pp. 400–414.
3. Boulabiar, M.-I., Burger, T., Poirier, F., and Coppin, G. A low-cost natural user interaction based on a camera hand-gestures recognizer. In *Proceedings of the 14th International Conference on Human-computer Interaction: Interaction Techniques and Environments - Volume Part II* (Berlin, Heidelberg, 2011), HCII'11, Springer-Verlag, pp. 214–221.
4. Gustafson, S., Bierwirth, D., and Baudisch, P. Imaginary interfaces: Spatial interaction with empty hands and without visual feedback. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2010), UIST '10, ACM, pp. 3–12.
5. Haubner, N., Schwanecke, U., Dörner, R., Lehmann, S., and Luderschmidt, J. Recognition of dynamic hand gestures with time-of-flight cameras.

In *Proceedings of ITG/GI Workshop on Self-Integrating Systems for Better Living Environments* (2010), vol. 2010, pp. 33–39.

6. Hilliges, O., Izadi, S., Wilson, A., Hodges, S., Garcia-Mendoza, A., and Butz, A. Interactions in the air: adding further depth to interactive tabletops. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (2009), ACM, pp. 139–148.
7. Ionescu, B., Coquin, D., Lambert, P., and Buzuloiu, V. Dynamic hand gesture recognition using the skeleton of the hand. *EURASIP Journal on Applied Signal Processing* (2005).
8. Kratz, S., and Rohs, M. A $3 gesture recognizer:simple gesture recognition for devices equipped with 3D acceleration sensors. *International Conference on Intelligent User Interfaces* (2010), 341–344.
9. Li, Y. Protractor: A fast and accurate gesture recognizer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2010), CHI '10, ACM, pp. 2169–2172.
10. Marquardt, N., Jota, R., Greenberg, S., and Jorge, J. A. The continuous interaction space: Interaction techniques unifying touch and gesture on and above a digital surface. In *Proceedings of the 13th IFIP TC 13 International Conference on Human-computer Interaction - Volume Part III* (Berlin, Heidelberg, 2011), INTERACT'11, Springer-Verlag, pp. 461–476.
11. Oikonomidis, I., Kyriazis, N., and Argyros, A. A. Markerless and efficient 26-dof hand pose recovery. In *Proceedings of the 10th Asian Conference on Computer Vision - Volume Part III* (Berlin, Heidelberg, 2011), ACCV'10, Springer-Verlag, pp. 744–757.
12. Olsen, L., Samavati, F. F., and Sousa, M. C. Fast Stroke Matching by Angle Quantization. *Proceedings of the ImmersCom* (2007).
13. Rubine, D. Specifying gestures by example. *ACM SIGGRAPH Computer Graphics 25*, 4 (July 1991), 329–337.
14. Valkov, D. Interscopic multi-touch environments. In *ACM International Conference on Interactive Tabletops and Surfaces* (New York, NY, USA, 2010), ITS '10, ACM, pp. 339–342.
15. Victor, B. A Brief Rant On The Future Of Interaction Design. `http://worrydream.com/ABriefRantOnTheFutureOfInteractionDesign/`, (2011).
16. Walter, R., Bailly, G., and Müller, J. Strikeapose: Revealing mid-air gestures on public displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2013), CHI '13, ACM, pp. 841–850.
17. Wilson, A., Izadi, S., Hilliges, O., Garcia-Mendoza, A., and Kirk, D. Bringing physics to the surface. In *Proceedings of the 21st annual ACM symposium on User interface software and technology* (2008), ACM, pp. 67–76.
18. Wobbrock, J., Wilson, A., and Li, Y. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology* (2007), ACM, pp. 159–168.