International Journal of Control and Automation
Vol. 11, No. 8 (2018), pp.109-118
http//dx.doi.org/10.14257/ijca.2018.11.8.10

A Study on Motion Recognition through Kinect-based Machine-Learning

Chang-Su Kim[1] and Hoe-Kyung Jung[2*]

[1]Dept. Of Computer Engineering, PaiChai Univ., 155-40 Baejae-ro, Seo-gu, Daejeon, South Korea
[2]Dept. Of Computer Engineering, PaiChai Univ., 155-40 Baejae-ro, Seo-gu, Daejeon, South Korea
[1]ddoja@pcu.ac.kr, [2]hkjung@pcu.ac.kr

Abstract

Motions are a representative, readily accessible visual language. Typical visual languages include predefined sign languages and hand signals for traffic control. However, there are situations where intentions and thoughts should be expressed with motions faster than dialogue.

Understanding of motions requires comprehensive judgment on their lines, time points of occurrence, and appearances from the starting to the ending point.

One of the difficulties is to distinguish whether a motion is an intended one or not. To resolve such difficulty, research has been conducted where different motions are taught to a machine using a learning algorithm to predict certain motions of a user.

Accordingly, this study assumed a situation where data on a user's own custom hand signals of static and dynamic types obtained from Kinect's sensor is stored and machine-learned so that the user who will learn the hand signals can check the data.

Keywords: *Kinect, Machine-learning, Motion Recognition, Tracking*

1. Introduction

Hand signals are a visual language used in various areas. When types of hand signals are searched, the main categories of the results include leisure/sports, traffic, and construction. In particular, a sport, as its area is wide-ranging, is characterized by difficulty with linguistic delivery; hand signals are most actively employed in this area.

User interface technologies are advancing to recognize more natural movements; users do not learn how to use machines but machines identify the existing movements. Understanding of motions requires comprehensive judgment on their lines, time points of occurrence, and appearances from the starting to the ending point. That is, a motion can be identified only when its ending point is reached by tracking it from its starting point. Therefore, it is difficult for a computer to respond to a user's motions on a real-time basis. It is also difficult to distinguish whether a motion is an intended motion or not. To resolve such difficulties, research has been conducted where different motions are taught to a machine using a learning algorithm to predict certain motions of a user.

Recently, various methods to receive data on user's motions have emerged. Among them, the Xbox Kinect invented by Microsoft is in use for various researches owning to its technical merits, such as having its own sensor, not as an accessory for games which is its original function[1-3].

Received (May 16, 2018), Review Result (July 14, 2018), Accepted (July 24, 2018)

* Corresponding Author

Accordingly, this study assumed a situation where data on a user's own custom hand signals of static and dynamic types obtained from Kinect's sensor is stored and machine-learned so that the user who will learn the hand signals can check the data.

2. Related Research

There are different types of methods to recognize motions. Kinect recognizes motions with Hit, Pose, and Gesture. Hit is a method to distinguish whether a specific point is consistent by receiving the coordinates of the target and skeleton on the screen. This method is appropriate for a single input but its application to multiple inputs is difficult. Pose differentiates motions by the distance or angle between joints. It is conceptually easy to create. However, errors are great according to the angles between Kinect and a person, and the number of things to differentiate increases as the number of poses grows, resulting in rapidly increased performance time. Gesture tracks a specific skeleton and differentiates the movements using movement direction and speed. It is more precise than the previously mentioned methods and may be more widely applied. This study attempted to recognize gesture motions with the learning algorithm of Kinect and to decrease reductions in its recognition speed as motions increased[2-5].

2.1. Kinect-based Non-contact Interface

A non-contact method to control three-dimensional (3D) contents tracks and recognizes a person's movements using image information mostly obtained by a camera. Such computer vision-based non-contact technique identifies a user's motion model through the process of initialization (preprocessing), tracking, prediction of a user's motions, and their recognition, and interacts with 3D contents through such motion recognition. The non-contact method is classified into a markerless method that does not utilize a marker and a marker-based method that attaches a specific marker on the part of the body and tracks body movements with visual technology[5].

The marker-based method predetermines the colors, shapes, LED, and infrared light of images as marker attributes and tracks the marker; in this way, it can track a user's motions in a relatively easy and speedy manner.

The method is relatively fast in extracting feature points of the images but it is uncomfortable for users since the user should put the marker on. On the other hand, the markerless method utilizes techniques such as optical flow, background subtraction, and motion history image to detect the direction and speed of a user's movement; it can quickly track the movements on a real-time basis and also can use brightness of light source. However, the markerless method is susceptible to external factors such as the location of camera, light source interference, and shadows[3-6].

In order to complement the disadvantages of the two methods, Kinect tracks information of 3D depth on a person's movements using multiple cameras and an infrared camera. To recognize a person's motions, Kinect uses two cameras: one to receive input of his or her motions on the red, green, and blue (RGB) screen and one to detect depth after laser-pointing. The two pieces of input information on the screen are processed through digital filtering to generate 3D skeleton data from information on a person's motions. The generated 3D skeleton information on a person's motions is reprocessed into linear data through a gesture recognition algorithm. The 3D contents are controlled by the user experience-based non-contact interface. Further, Kinect sensor projects infrared light into the user using a laser, measures intensity of waves reflected from dots, and then gauges the distance using intensity of reflection. Dots with weak intensity of reflection are assumed to come from far away and those with strong intensity of reflection are assumed to come from the user on the front. Thereby Kinect sensor identifies a person's main bone joints and detects body movements[2-7].

Kinect uses an infrared sensor to capture a projected pattern and construct a depth image, which describes the distance between the sensor and each pixel. It fuses a wide variety of new research, including gesture recognition and control[8], natural user interface[9] and 3D surface reconstructions[10-11]. New algorithms are proposed to estimate human postures from the single depth image[12-15]. They provide economical ways for 3D motion acquisition[16].

2.2. Kinect

The Microsoft Kinect is a new product launched by Microsoft in November 2010. Its capability to produce depth and RGB streams at a price much lower than traditional range 2 sensors have made it a sensation in the field of Computer Vision. At the heart of the Kinect lies a time of flight camera that measures the distance of any given point from the sensor using the time taken by near-IR light to reflect from the object. In addition to it, an IR grid is projected across the scene to obtain deformation information of the grid to model surface curvature. We use the OpenKinect driver framework for the Kinect that produces 640 × 480 RGB and depth images at 30 fps[7].

Kinect is equipped with a 1080p camera, an infrared sensor, and a sensor that detects depth. These sensors enable skeleton tracking. The 1080p camera may utilize infrared radiation that identifies shadows and colors at the same time[17].

Skeleton tracking is a technology that automatically recognizes a user's body. Windows Software Development Kit (SDK) available as part of Kinect is used to identify the results of skeleton tracking. The number of joints recognized per person increased from 20 to 25 owing to the launch of SDK V2[3-5].

Kinect is an input device that recognizes a user's face and motions. 48 dots are identified from the user's body to identify the user’s motions. These dots are used to automatically show the map of the whole body reflecting the user's skeleton and body shape [3-4]. Besides, Kinect obtains 30 frames of images per second on depth as well as images on RGB colors on a real-time basis[1]. This study implemented a method to have the Kinect learn how to quickly recognize the effective range of each user and the effective range of hand movements, which vary person to person, using information on head, shoulders, hand joints, color images, and depth[17-20].

3. System Design and Implementation

3.1. System Design

Motions are recognized with Kinect's sensor that receives both hands' x, y, and depth values. An application that can control the sensor delivers the tracked values for parsing so that the motions can be recognized as an array. Figure 1 shows the entire structure of the system.

The parsed array is sent to the server through the socket between the application and the server. The server saves the value in the database, and then machine learning is performed. The weighted value is separately stored.

When a user intends to read the motions, they are captured by a motion capture sensor and delivered to the application, then parsed into an array, which is transmitted to the server. The server identifies the motion through the weight value that was machine-learned in advance.

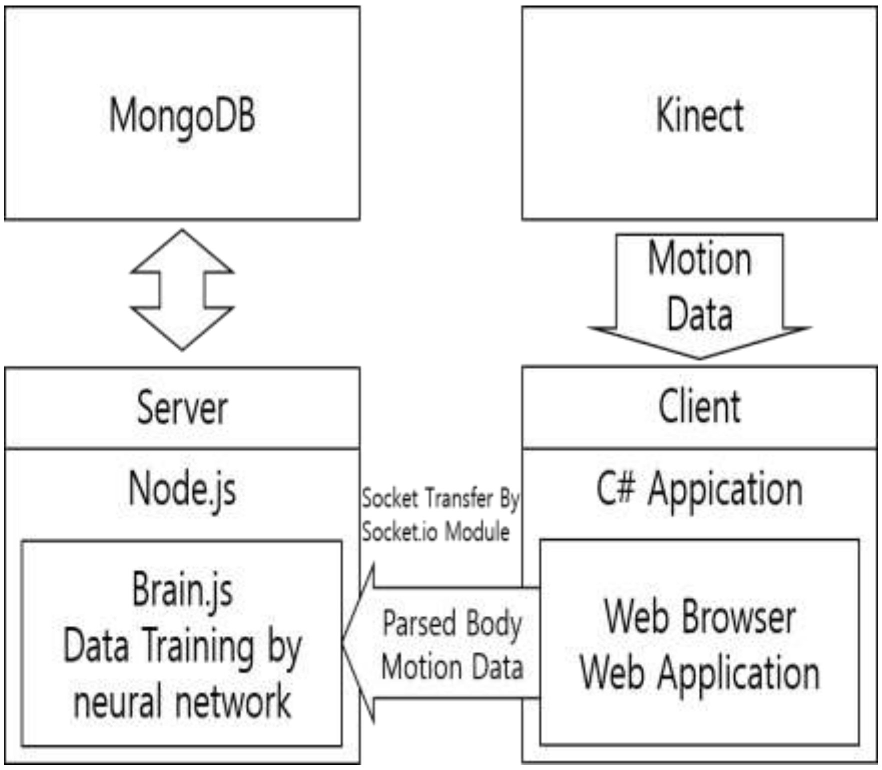

MongoDB
Kinect
Motion Data
Server
Client
Node.js
C# Appication
Socket Transfer By Socket.io Module
Brain.js Data Training by neural network
Parsed Body Motion Data
Web Browser Web Application

Figure 1. The Entire Structure of System

3.2. Development Environment

Figure 2. Implemented System

Kinect was designed to work on Windows 10 operation system. It implements Node.js web server and Brain.js for machine learning in JavaScript through the development tool of C9.io.

In this study, Kinect of Microsoft was employed as a sensor for motion capture and an application was implemented using the Kinect SDK in C# through Visual Studio. Figure 2 displays the implemented system.

3.3. Getting Trajectory

Kinect is run to input the trajectory drawn by a user. The trajectory reflects hand movements obtained by skeleton tracking.

In order to differentiate the boundary of hand signals input by a user, basic motions of Kinect were used including scissors, close, and open. When the user clenches the fist (close), Kinect starts to recognize hand signals and when the shape of hand is changed into open, recognizing stops.

C# application gets the value of the motion sensor coming in from Kinect. After declaring two-dimensional (2D) array in advance, the depth values of hands are input into the locations of x and y values of the hands and the trajectory drawn by the user is stored in the 2D array.

Figure 3, Figure 4 illustrates the screen and algorithms where motions are learned through Kinect.

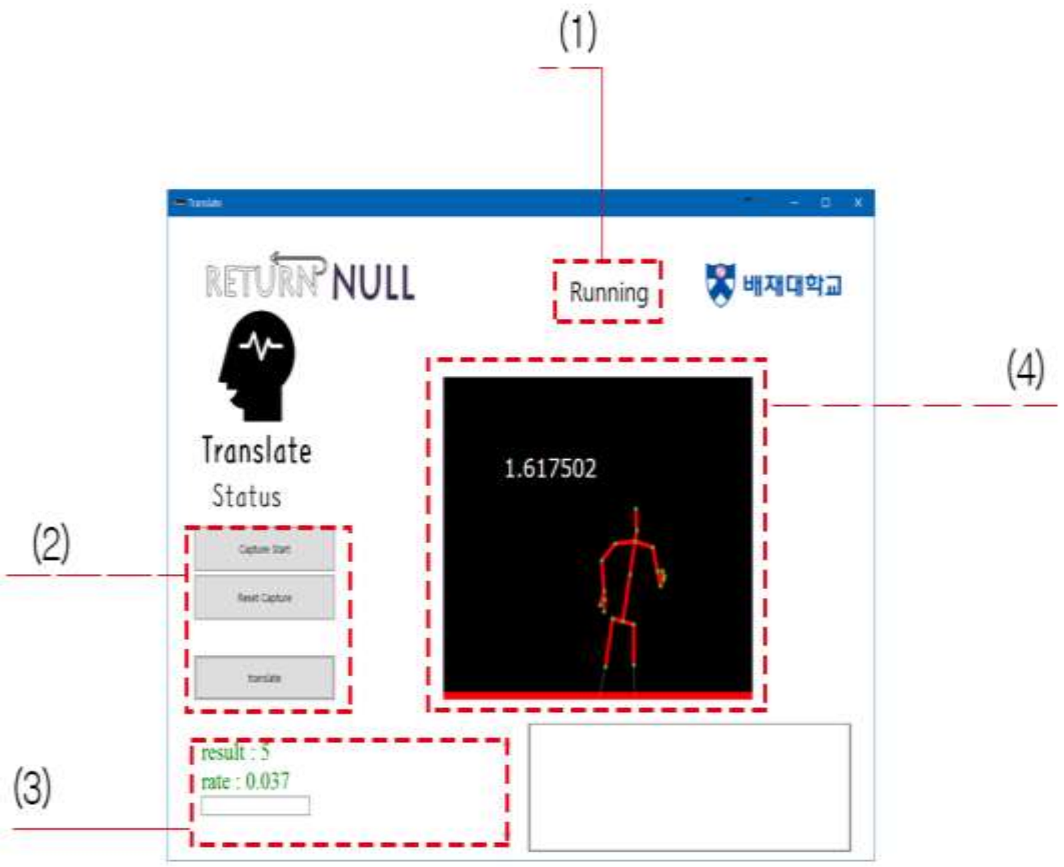


Figure 3. The Screen to Learned Motion through Kinect

```
function lern() {
  var startTime = (new Date()).getTime();
  mnistget.find({}, function(err, data) {
    if (err) console.log(err)
    for (var i = 0; i < data.length; i++) {
      if (data[i].input.length != 28 * 28) {
        data.splice(i, 1);
      }
      else if (data[i].output[''] == 1) {
        data.splice(i, 1);
      }
    }
    training(startTime, data)
  })
}


function training(startTime, data) {
  if (data.length != 0) {
    console.log('start Train... (Length : ' + data.length + ")")
    net.train(data, {
      log: true
    });
    console.log("end Train (Length : " + data.length, parseInt((new Date()).getTime() -
startTime) / 1000 + "초)")
    savefun(net);
  }
  else {
    console.log("no Data for training")
  }
}


function loadlern() {
  fs.readFile(__dirname + '/' + dbName + '.txt', 'utf8', function(err, data) {
    if (err) console.log(err);
    try {
      net.fromJSON(JSON.parse(data));
      console.log("Weights Load Complete")
    }
    catch (e) {
      console.log("Weights Load Err!!!")
    }

  })
```

Figure 4. Motion Learning Algorithms through Kinect

On the translate screen, a view is declared that can capture a person's shape using the Kinect sensor. Movements can be recognized using the capture start button and initialized using the reset capture button.

When a motion is recognized using the reset capture button, a value is output after the trajectory of a motion is compared with machine-learned data using the translate button. (1) indicates the state of Kinect, (2) shows buttons for input actions and extracting results (start, reset, translation), (3) indicates the output of the results, and (4) shows the view function of Kinect.

3.4. Array Parsing

The 2D array recognized by Kinect and created by C# application is 1920*1080 in size and it is not appropriate to read actual motions.

Therefore, first, among the input values into the array, the minimum and maximum values corresponding with up, down, left, and right should be selected respectively as an initial index with the input values in the 2D shape.

Then, the size obtained by deducting the minimum value from the maximum value in each length and breadth should be created into a 2D array and brings the existing value of 2D array.

Lastly, the size should be adjusted by removing each line if the size is larger than 28*28 and by adding each line if the size is smaller than 28*28.

3.5. Transmit to the Server

Place the web browser component on C# application and load the webpage implemented to enable socket communication with Node.js. The same content as the title value necessary for array and reading is parsed into JavaScript Object Notation (JSON) format.

After, the data is transmitted through InvokeScript of web browser component, sent to the server through socket communication between Node.ju server and web application that was launched in web browser, and stored by the server in MongoDB through Mongoose module.

3.6. Machine Learning

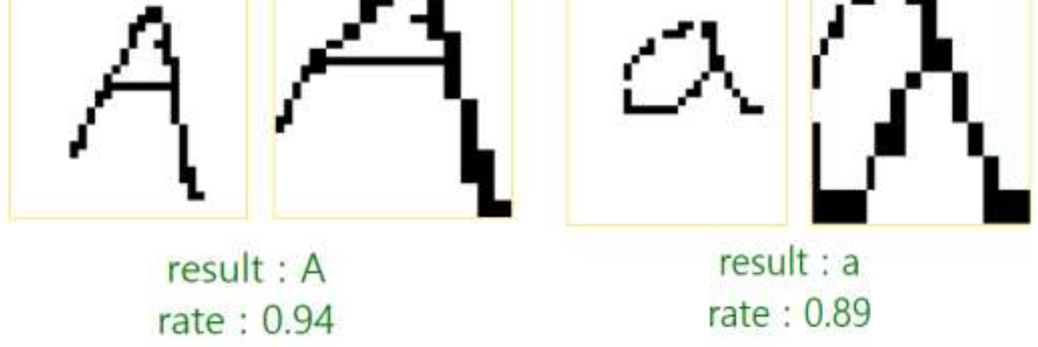
When learning is initiated by Node.js, the value is obtained from MongoDB and machine learning of the value is performed in the artificial neural network implemented through Brain.js module.

The weighted value which is the result of machine-learning, can be stored in JSON format.

3.7. Motion Reading

A user can read a motion by the following process after machine learning is complete; The motion of the user is captured by Kinect and obtained by C# application to create an array. The created array is parsed with the size of 28*28 and then transmitted to the server through InvokeScript of the web browser component. The server verifies the result value from the artificial neural network implemented with Brain.js and sends back to the web browser of C# application.

Figure 5 displays the result value "three" with 99% of match rate when the input array (left) changes into the parsed array (right).



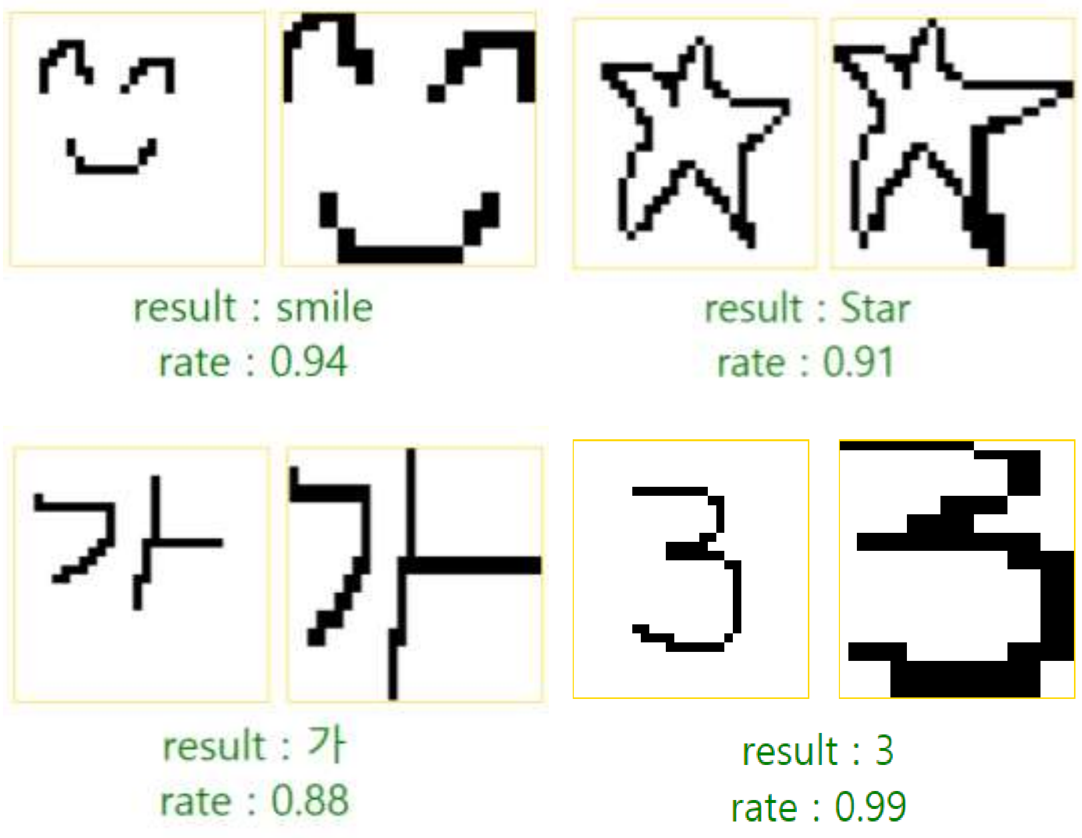


Figure 5. The Result of Motion Reading

4. Conclusion

Motions are a representative, readily accessible visual language. Typical visual languages include predefined sign languages and hand signals for traffic control. However, there are situations where intentions and thoughts should be expressed with motions faster than dialogue.

User interface technologies are advancing to recognize more natural movements; users do not learn how to use machines but machines identify the existing movements. Understanding of motions requires comprehensive judgment on their lines, time points of occurrence, and appearances from the starting to the ending point. That is, a motion can be identified only when its ending point is reached by tracking it from its starting point. Therefore, it is difficult for a computer to respond to a user's motions on a real-time basis. It is also difficult to distinguish whether a motion is an intended motion or not. To resolve such difficulties, research has been conducted where different motions are taught to a machine using a learning algorithm to predict certain motions of a user.

This study used Kinect to capture, machine-learn, and read a person's motions. Through its function of skeleton tracking, Kinect recognized the locations of a user's hands and tracked and stored their trajectory, thereby providing their complete motions.

Although both hands were used in this study, the results were insufficient because of the characteristics of trajectory. For a more practical study, research based on motions using both hands is necessary. There are limitations on the trajectory when using both hands, thus study applying other methods than trajectory is needed in future.

Acknowledgments

This work was supported by the research grant of Pai Chai University in 2018.

References

[1] H. S. Choi, "Kinect-based Motion Recognition Model for the 3D Contents Control", The Journal of the Korea Contents Association, vol. 14, no. 1, **(2014)**, pp. 24-29.

[2] D. P. Hong and W. T. Woo, "Research Trends on Gesture Based User Interface", Telecommunications Review, vol. 18, no. 3, **(2008)**, pp. 403-413.

[3] J. Y. Chang, M. W. Ryu and S. C. Park, "Technology Trends of Range Image Based Gesture Recognition", Electronics and Telecommunications Trends, vol. 29, no. 1, **(2014)**, pp. 11-55.

[4] J. S. Shin, K. R. Ko and S. B. Pan, "Automation of Human Body Model Data Measurement Using Kinect in Motion Capture System", Journal of Advanced Information Technology and Convergence, vol. 12, no. 9, **(2014)**, pp. 173-180.

[5] K. Khoshelhan and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications", Sensor, vol. 12, no. 2, **(2012)**, pp. 1437-1454.

[6] H. Park, J. Choi, J. Park and K. Moon, "A Study on Hand Region Detection for Kinect-Based Hand Shape Recognition", Journal of Broadcasting Engineering, vol. 18, no. 3, **(2013)**, pp. 393-400.

[7] A. Kar, "Skeletal tracking using microsoft Kinect", Methodology, 1.1 **(2010)**, pp. 1-11.

[8] Z. Ren, J. Meng, J. Yuan and Z. Zhang, "Robust hand gesture recognition with kinect sensor", in Proceedings of the 19th ACM international conference on Multimedia, ser. MM '11. New York, NY, USA: ACM, **(2011)**, pp. 759-760.

[9] S. Kean, J. Hall and P. Perry, "Meet the Kinect: An Introduction to Programming Natural User Interfaces", 1st ed. Berkely, CA, USA: Apress, **(2011)**.

[10] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison and A. Fitzgibbon, "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera", Proceedings of the 24th annual ACM symposium on User interface software and technology, ser. UIST '11. New York, NY, USA: ACM, **(2011)**, pp. 559-568.

[11] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking", Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium, **(2011)** October, pp. 127-136.

[12] M. Sun, P. Kohli and J. Shotton, "Conditional regression forests for human pose estimation", Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference, **(2012)**, pp. 3394-3401.

[13] R. Girshick, J. Shotton, P. Kohli, A. Criminisi and A. Fitzgibbon, "Efficient regression of general-activity human poses from depth images", Computer Vision (ICCV), 2011 IEEE International Conference, **(2011)**, pp. 415-422.

[14] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman and A. Blake, "Efficient human pose estimation from single depth images", IEEE Transactions on Pattern Analysis and Machine Intelligence, **(2012)**.

[15] A. Baak, M. Muller, G. Bharaj, H.-P. Seidel and C. Theobalt, "A data-driven approach for real-time full body pose reconstruction from a depth camera", Proceedings of the 2011 International Conference on Computer Vision, ser. ICCV '11. Washington, DC, USA: IEEE Computer Society, **(2011)**, pp. 1092-1099.

[16] H. P. Shum, E. S. Ho, Y. Jiang and S. Takagi, "Real-time posture reconstruction for Microsoft Kinect", IEEE Trans. Cybernetics, vol. 43, no. 5, **(2013)**, pp. 1357-1369.

[17] Wikipedia Onlind Documentation [Internet]. Available:http://en.wikipedia.org/wike/Kinect.

[18] B. Su, W. Huang Wu and S. Min, "Human action recognition method based on hierarchical framework via Kinect skeleton data", Proceedings of the 2017 International Conference on Machine Learning and Cybernetics, Ningbo, China, **(2017)** July 9-12.

[19] I. Oikonomidis, N. Kyriazis and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using Kinect", BmVC, vol. 1, no. 2, **(2011)**, pp. 1-11.

[20] J. Sun, Y. Wang, J. Li, W. Wan, D. Cheng and H. Zhang, "View-invariant gait recognition based on kinect skeleton feature", Multimedia Tools and Applications, **(2018)**, pp. 1-27.

Authors

Chang-Su Kim, he received his B.S., M.S., and Ph.D. degrees from the Department of Computer Engineering of Paichai University, Korea, in 1996, 1998, and 2002, respectively. From 2005 to 2012, he worked for the Department of Internet at Chungwoon University as a professor. Since 2013, he has worked in the Department of Computer Engineering at Paichai University, where he now works as a professor. His current research interests include multimedia document architecture modeling, IoT, data mining and the semantic web.

Hoe-Kyung Jung, he received his B.S degree in 1987 and Ph. D. degree in 1993 from the Department of Computer Engineering of Kwangwoon University, Korea. From 1994 to 1995, he worked for ETRI as a researcher. Since 1997, he has worked in the Department of Computer Engineering at Paichai University, where he now works as a professor. His current research interests include multimedia document architecture modeling, information processing, information retrieval, and databases.