International Journal of Bio-Science and Bio-Technology
Vol.6, No.5 (2014), pp.161-170
http://dx.doi.org/10.14257/ijbsbt.2014.6.5.16

Human Detection for an Interactive Art in a Dark Environment using the GPU Hardware

SangHun Nam[1], YoungEun Kim[2] and Jung-Yoon Kim[3*]

[1]*Center of Human-centered Interaction for Coexistence*
[2,3]*Graduate School of Advanced Imaging Science, Multimedia & Film, Chung-Ang University*
sanghunnam@gmail.com, naankim@gmail.com, kjyoon79@gmail.com

Abstract

Human detection is the initial step of human recognition for an interactive system. A depth sensor is used to detect audiences in a dark environment, such as a media art exhibition or a theater. The depth data are converted to three-dimensional points using camera parameter. The detected human and the background are separated using the position of converted points. As these processes require much computation, in this study, we reduced the amount of computation time for real-time human detection systems through the GPU multi-threaded programming. The head and body are separated using the geometric characteristics of a human shape. The facial components are classified by analyzing projected curves of x- and y-coordination and by applying them to the geometrical characteristics of human face.

Keywords: *Human Detection, Interactive Art, GPU programming*

1. Introduction

Every person has different physical and mental characteristics according to gender, age, and race. Physical characteristics can be classified using video, audio, and medical information. Visual information - including face, skin, fingerprints, and gait - represents physical characteristics. The audio information, such as voice and speech, indicates physical and cultural characteristics. Other special factors, as, for instance, DNA or facial and hand thermograms, can also be used to recognize a person. As physical and psychological characteristics have a statistical similarity of gender, age, ethnicity, and culture, it is possible to predict them for a specific person through visual and audio information. The demographic similarity can be effectively used of human computer interaction, biometrics, and target advertising [1-3].

Image-based and depth-based methods are mainly used to obtain visual information. The former are used for human detection and classification in real time. An image sensor, such as a webcam, has the advantage of a precise analysis producing a high resolution image; however, it is still sensitive to color and illumination. Compared to an image sensor, a depth sensor has a lower resolution, but it can obtain the depth data using an IR camera in the dark environment. A multimodal classification method that simultaneously uses image data and depth data has also been used in previous research [4-5]. Our approach uses Microsoft Kinect as a depth sensor to detect humans at the interactive media art exhibition in the dark environment [6].

*Corresponding Author

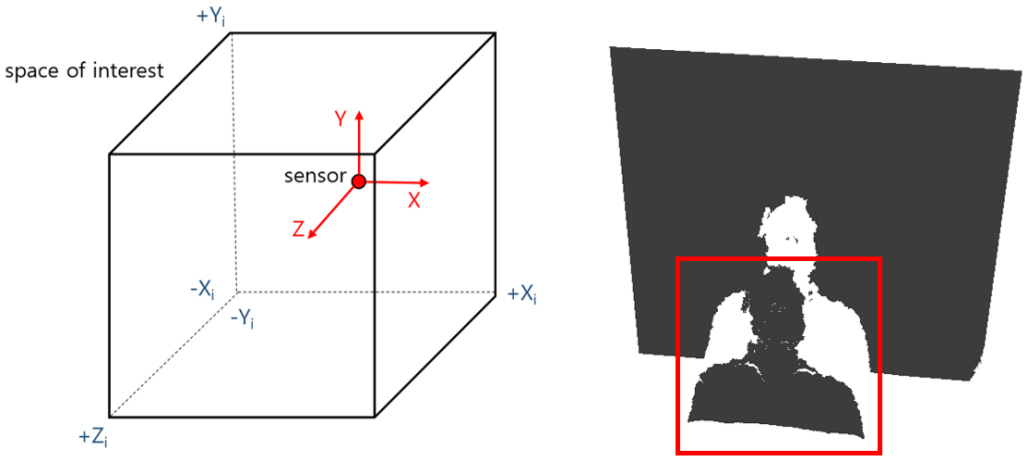
ISSN: 2233-7849 IJBSBT

Previous research used depth sensors for human detection and pose estimation. Human detection technique is the initial step to estimate human pose and make a skeleton. Earlier studies on human detection used stereo cameras. The depth information was estimated using the stereo technique with two image cameras that extracted a human from the background [7]. The depth information from a TOF (Time of Flight) camera can be used to capture human motion without IR markers. The human pose is tracked using a stream of depth images [8]. Recently, Microsoft Kinect has started to be applied for human detection and classification, because it is affordable and allows one to obtain both the depth and the image data.

2. Human Detection in Interactive Art

In recent years, there have been many interactive art performances that use the movement of the audience as user interaction. A Kinect sensor consists of an image sensor, a depth sensor, and the software to obtain image, depth, and skeleton from the hardware. Many algorithms with image provide the high-recognition results using a low-cost webcam with full HD (1920∗1080) resolution. As the influence of the various lights decreases the recognition of human detection, the depth sensor has the advantage of obtaining the depth data with a low resolution. Thus, the Kinect is useful for interactive media art performances in a dark environment.

Interactive artworks set the space to observe the shape and movement of the audience. As shown in Figure 1(a), we define the space of interest as the interaction space with audiences. The space of interest is set as a reference point where the sensor is installed. The depth data of Kinect is the distance map obtained from the depth sensor. In order to generate three-dimensional human data, the distance map should be converted into three-dimensional positions using the camera parameter. The following depth camera parameter was used: $F_x = 528.32, F_y = 527.03, C_x = 320.10, C_y = 257.57$ for converting [9]. Calculation time can be reduced by using the GPU multi-thread programming, as there are 3,072,000 loops to convert the distance map into the position.



(a) Space of interest (b) Points in space of interest

Figure 1. The Point Generation of Space of Interest

The converting process uploads the depth memory array from CPU memory to GPU memory. The 'ComputePointCloud' function (see Table 1), u and v is the depth map index calculated with a multi-threaded process. The positions are calculated with the depth (z) and four camera parameters. After the process, the position memory arrays are downloaded from the GPU memory to the CPU memory. On converting to the position of points, the z value of points outside the space of interest is set to 0 (see Figure 1 for separating the human and the background using the depth information).

Table 1. Converting Depth Map to Position of Vertex in CUDA

```
ComputePointCloud(vector<unsigned short> depth, vector<float>
position, float Fx, float Fy, float Cx, float Cy)
{
  int u = threadIdx.x + (blockIdx.x * blockDim.x);
  int v = threadIdx.y + (blockIdx.y * blockDim.y);
  float z = depth (v)[u]

  float vx = z * (u - Cx) * (1.0/Fx);
  float vy = z * (v - Cy) * (1.0/Fy);
  float vz = z;

  position (v)[3*u + 0] = -vx;
  position (v)[3*u + 1] = -vy;
  position (v)[3*u + 2] = vz;
}
```

The position of a converted vertex can be used only for estimating the area of a human shape. In order to visualize the shape of a human, a mesh is generated using vertices. The depth map is composed in the order of the index v(i, j). As shown in Figure 2, a mesh consists of three adjacent vertices. Two triangle meshes are generated by four vertices (see Equations (1) and (2)).

$$M(i) = \{v(i, j), v(i, j+1), v(i+1, j)\} \quad (1)$$

$$M(i+1) = \{v(i+1, j), v(i, j+1), v(i+1, j+1)\} \quad (2)$$

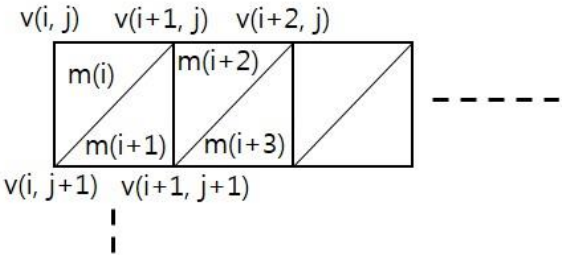


Figure 2. Mesh Generation

The mesh generation using all vertices (640*480) can be calculated quickly using the GPU, but the result visualizes the mixture of the human and the background. For their separation, the space of interest and the distance with the neighborhood vertex must be considered. The human is separated from the background by filtering 'distance' (see

Table 2). The 'distance' is the value Z_i of the space of interest in Figure 1(a). The 'threshold' of the distance between neighborhood vertices should be applied for removing meshes connecting separated surfaces. For these processes, the use of only GPU is not efficient. The mesh selection is calculated using the GPU programming (see Table 2). Vertex array are uploaded to the GPU memory. The 'CreateMesh' function examines the position of adjacent vertex and checks 'mesh' to create. The array 'mesh' is defined to Boolean type to reduce the download time from the GPU memory to the CPU memory. Mesh array is generated with the vertex array index and 'mesh' information.

```
CreateMesh(vector<unsigned short> vertex, vector<bool> map, bool
           filter, float distance, float threshold)
{
  int u = threadIdx.x + (blockIdx.x * blockDim.x);
  int v = threadIdx.y + (blockIdx.y * blockDim.y);

  float fV1 = vertex (v) [3*u + 2];
  float fV2 = vertex (v) [3*(u+1) + 2];
  float fV3 = vertex (v+1) [3*u + 2];
  float fV4 = vertex (v+1) [3*(u+1) + 2];

  if ( filter && ( fV1 > distance) ) {
      mmap (v)[2*u + 0] = false;
  }
  else if ( ( fV1-fV2 ) > threshold || ( fV2-fV1 ) > threshold
  || ( fV1-fV3 ) > threshold || ( fV3-fV1 ) > threshold) {
      mmap (v)[2*u + 0] = false;
  }
  else {
      mmap (v)[2*u + 0] = true;
  }

  if ( filter && ( fV4 > distance) ) {
      mmap (v)[2*u + 1] = false;
  }
  else if ( ( fV4-fV2 ) > threshold || ( fV2-fV4 ) > threshold
|| ( fV4-fV3 ) > threshold || ( fV3-fV4 ) > threshold) {
      mmap (v)[2*u + 1] = false;
  }
  else {
      mmap (v)[2*u + 1] = true;
  }
}
```

Table 2. Create 3D mesh with Space of Interest in CUDA

Due to occlusion and noise near the boundary of the object, the depth map of Kinect sensor has some holes (see Figure 3(a)). The hole-filling algorithm is used to fill the holes inside the human shape — most commonly, for its simplicity and speed, the

bilateral filter. In Table 3, the 'HoleFilling' function is inputted the value of the original depth from a sensor. The bilateral filter calculates the position of the vertex in the hole by using the position of adjacent vertices. Figure 3(b) shows the results of the human mesh after hole-filling.

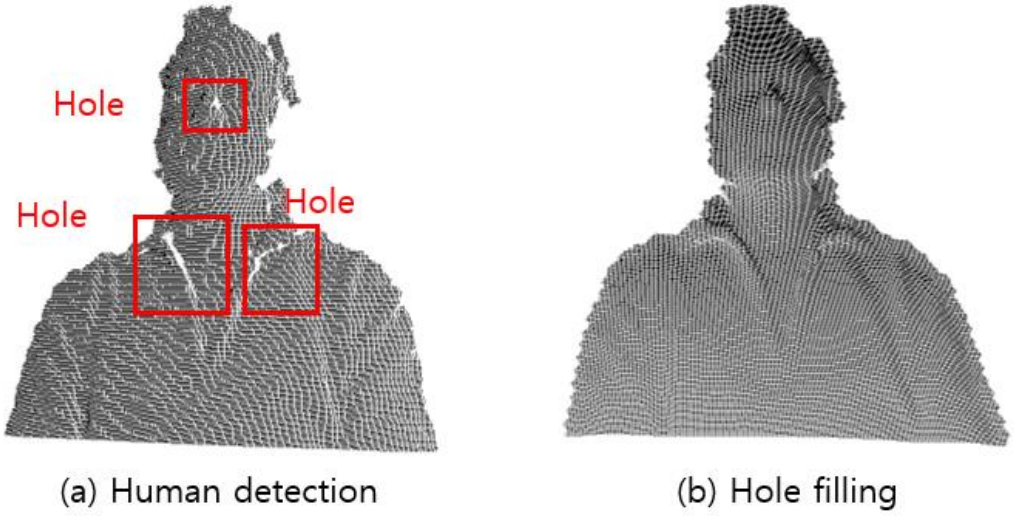


Figure 3. Human Mesh Visualization

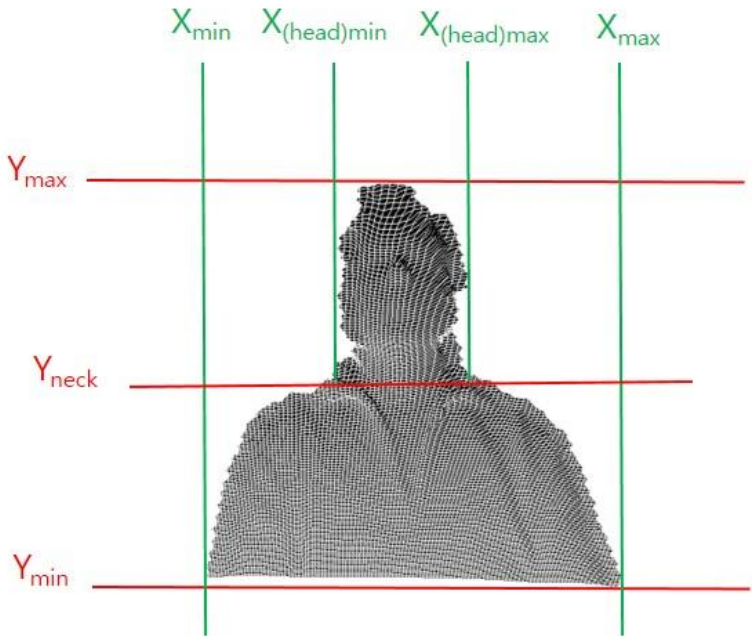


Figure 4. Separation of Face and Body

An interactive system has different purposes and uses specific body parts. Face and the body can be separated by using common geometric information of the human body. As shown Figure 4, four values (X_{min}, X_{max}, $Y_{min,}$ Y_{max}) are obtained by checking the

position of the human data. The values $X_{(head)min}$ and $X_{(head)max}$ can be estimated because the hair or ears have a maximum value of the x-axis. The extension of $X_{(head)min}$ and the human data meet at the shoulder and y value of the intersection point is Y_{neck}.

```
int filter[8][2] = { {-1, -1}, {0, -1}, {1, -1}, {-1, 0},
                     {1, 0}, {-1, 1}, {0, 1}, {1, 1} };

HoleFilling(vector<unsigned short> depth, vector<unsigned short>
            holefill)
{
  int u = threadIdx.x + (blockIdx.x * blockDim.x);
  int v = threadIdx.y + (blockIdx.y * blockDim.y);

  if (depth (v)[u] != 0 ) {
       int nCnt = 0, nDepthSum = 0;

       for(int i = 0; i < 8; i++) {
             int nX = u + filter [i][0];
             int nY = v + filter [i][1];

             if ( depth (nY)[nX] ) {
                   nDepthSum += depth (nY)[nX];
                   nCnt++;
             }
       }

       if ( nCnt != 0 )  {
             holefill (v)[u] = (nDepthSum / nCnt);
       }
       else {
             holefill (v)[u] = 0;
       }
    }
}
```

Table 3. Hole Filling in CUDA

The Face($X_{(head)min}$, $X_{(head)max}$, Y_{neck}, Y_{max}) and the Body(X_{min}, X_{max}, Y_{neck}, Y_{min}) can be separated by seven lines (see Figure 4). The facial component can be extracted by examining the points of the face. The nose tip can be extracted by using the information that the nose is the most protruding component of the face. However, if a user is wearing a hat, the most protruding facial component is not the nose. After calculating the center of the circle surrounding the circle of face, the candidate circle of the nose with a radius of 5 cm can be generated (see Figure 5). The nose tip can be found inside of the candidate circle of the nose.

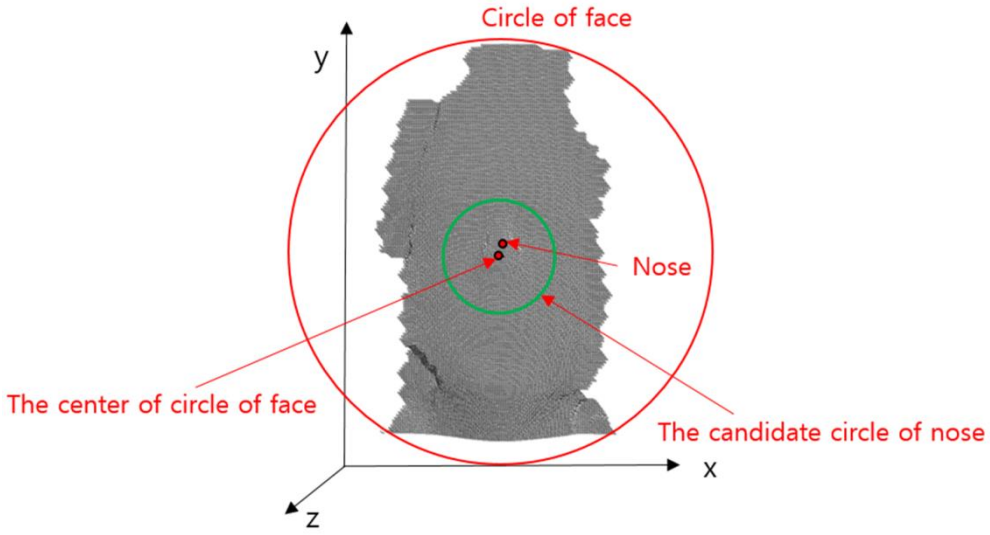


Figure 5. Separation of Face and Body

The projected curve of points in x plane and y plane based on the feature points is used to efficiently analyze the surface of the face (see Figure 6). The glabella and the center of the forehead and the lips can be extracted by analyzing the depth curve in x plane including the position of the nose tip. Other facial components can be detected using the depth curves in y plane including the glabella and the center of the forehead and the lips.

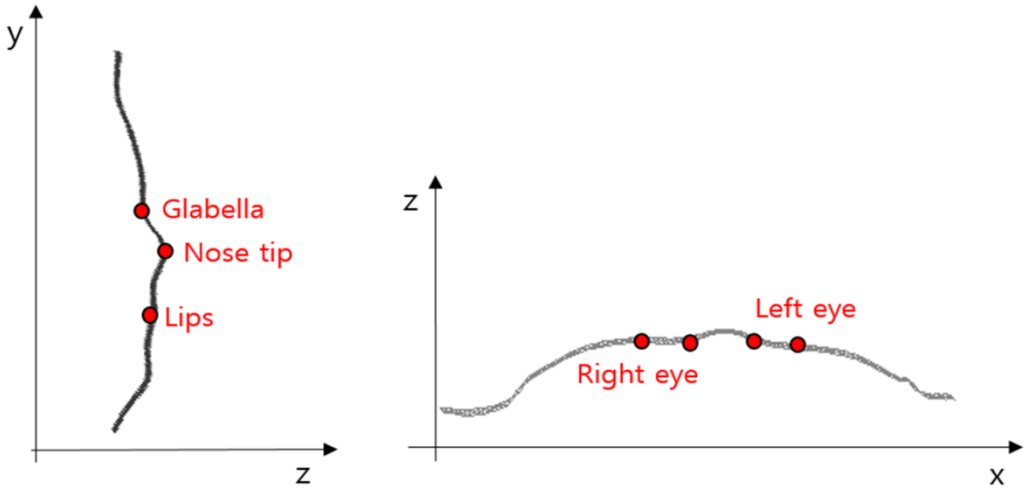


Figure 6. Facial Component Detection

3. Result

Extraction of human body components is necessary for natural interaction. In case of the interactive art exhibition in a dark environment, a depth sensor is more suitable than an image sensor. The depth-based human detection algorithm differs from the image-based algorithm. The human and the background points are separated by using the

distance of the sensor. The surface of human is generated to connect adjacent points. The head and the body points are separated by using the position of a three-dimensional human shape. The nose tip is extracted from the face surface and the forehead; the eyes and the lips are detected from the surface by using depth curve in x- and y-coordination based on the nose tip. In order to recognize the pose facing in various directions, front face reconstruction should be applied to facial component detection in future studies.

References

[1] C. B. Ng, Y. H. Tay and B. M. Goi, "Vision-based Human Gender Recognition: A Survey", arXiv preprint arXiv, **(2012)**, pp. 1204.1611.
[2] J. Chin and W.T. Fu, T. Kannampallil, "Adaptive information search: age-dependent interactions between cognitive profiles and strategies", Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, **(2009)**.
[3] Y. S. Wang, M. C. Wu and H.Y. Wang, "Investigating the determinants and age and gender differences in the acceptance of mobile learning", British Journal of Educational Technology, vol. 40, no. 1, **(2009)**.
[4] L. Ballihi, B. Ben Amor, M. Daoudi, A. Srivastava and D. Aboutajdine, "Boosting 3D-geometric features for efficient face recognition and gender classification", IEEE Transactions on Information Forensics and Security, vol. 7, no. 6, **(2012)**.
[5] X. Lu, H. Chen and A. K. Jain, "Multimodal facial gender and ethnicity identification", Advances in Biometrics **(2005)**, pp. 3832.
[6] Y. Kim, M. Lee, S. Nam and J. Park, "User interface of interactive media art in a stereoscopic environment", Human Interface and the Management of Information, Information and Interaction for Learning, Culture, Collaboration and Business, **(2013)**.
[7] T. Darrell, G. Gordon, M. Harville and J. Woodfill, "Integrated person tracking using stereo, color, and pattern detection", International Journal of Computer Vision, vol. 37, no. 2, **(2000)**.
[8] V. Ganapathi, C. Plagemann, D. Koller and S. Thrun, "Real time motion capture using a single time-of-flight camera", Proceedings of the Computer Vision and Pattern Recognition (CVPR) **(2010)**.
[9] C. Zhang and Z. Zhang, "Calibration between depth and color sensors for commodity depth cameras", Proceedings of IEEE International Conference on Multimedia and Expo (ICME), **(2011)**.

Authors

SangHun Nam received the B.S degree in mechanical design from Chung-Ang University. He received M.S and Ph. D degree in Image science from graduate school of advanced imaging science, multimedia & film in Chung-Ang University. His major is computer graphics & virtual environments. He completed a post-doctoral course at virtual environment Lab in Chung-Ang University from 2012 to 2013. His research area includes virtual environments, spatial sketch and interaction, interactive media art.

YoungEun Kim is a media artist working on 3D stereoscopic installations in Korea. She received the M.A degree in Fine Art from the Chelsea College of Art & Design in London. She is Ph. D candidate in Multimedia Art at graduate school of advanced imaging science, multimedia & film in Chung-Ang University. Her research area includes interactive media art, stereoscopic art and interaction.

Jung-Yoon Kim received the B.S degree from Hoseo University Korea in 2001 and the M.S. degree in Game engineering from Hoseo University Korea in 2006. He is Ph.D. degree at Chung-Ang University in 2013 Korea. His current research interests are in the areas for Game Design, Game ART, and Game AI.