Real-Time Head Detection with Kinect for Driving Fatigue Detection

Yang Cao[1] and Bao-Liang Lu[1,2]

[1] Center for Brain-like Computing and Machine Intelligence
Department of Computer Science and Engineering
[2] MOE-Microsoft Key Lab. for Intelligent Computing and Intelligent Systems
Shanghai Jiao Tong University
800 Dongchuan Road, Shanghai 200240, P.R. China

Abstract. Nowadays, depth cameras such Microsoft Kinect make it easier and cheaper for us to capture depth images. It becomes practical to use depth images for detection in consumer-grade products. In this paper, we propose a novel and simple real-time method to detect human head in depth image for our driving fatigue detection system, based on the elliptical shape of human head. Experiments show that our method can successfully detect human head in different light conditions and across different head poses. We integrate this detection algorithm into our driving fatigue detection system, and see remarkable improvements both in detection rate and detection speed.

Keywords: Head Detection, Depth Image, Kinect, Fatigue Detection.

1 Introduction

Human head detection is often the fundamental step in many computer vision applications, such as head or face tracking, face recognition, face expression analysis and gender classification. Given a color image, an infra-red image, a depth image or a combination of them, the goal of head detection is to find the locations and sizes of all human heads in the image. Head detection is difficult if different light conditions and different head poses are taken into consideration.

Traditionally, human head detection is accomplished on color images. Since human face has the most significant features in head, we often do face detection as a way of head detection. Face detection has been studied for decades. In [2] face detection methods before year 2000 are nicely surveyed. In the 2000's, Viola and Jones made a great contribution to this field by their excellent work in [1]. Many later works are based on or inspired by the approach in [1]. Works after year 2000 are surveyed in [3].

Human head detection on depth image has also been studied for a long time, which can be broadly classified into two types. One type of head detection methods on depth images tries to detect features of faces such as nose tips, eyes and cheeks. For example, Colombo *et al.* [4] proposed to detect human face on depth image through an analysis of the curvature of face. Chew *et al.* [5] proposed to

M. Lee et al. (Eds.): ICONIP 2013, Part III, LNCS 8228, pp. 600–607, 2013.

detect nose tips in a depth image by calculating effective energy. For this type of head detection methods, the advantage is that organs on faces are detected directly, which provides us more information such as locations of each organ instead of just head location and size. The disadvantage is that detection rate often decreases dramatically for depth image with moderate noise. The other type of head detection methods on depth images tries to detect human heads using general information such as the elliptical shape of head. Xia *et al.* [7] used 2D chamfer distance matching to find candidate head locations according to the contour of head and shoulder, and then used a hemisphere to fit the head. Suau *et al.* [8] first extracted foreground pixels in the depth image, and then used a binary elliptical template to search for the head location. This type of head detection methods can tolerate moderate noise in the depth image, and is often more robust to different head poses.

Our driving fatigue detection system operates according to the facial expression of the driver, so accurate and fast head detection is an important step. Generally speaking, color image can provide more information than depth image, but is more sensitive to light condition. Considering the complex light condition during driving, the head detection method proposed in this paper utilizes depth information only. We ignore facial details in the depth image, and use the elliptical shape of head as a clue. Methods proposed in [7,8] also use the same clue to detect human head. The difference between their methods and our method is that, their methods both include a matching stage, *i.e.* shifting a template in the depth image pixel by pixel to find a best match, which greatly slow down the speed of head detection. For example, the method in [8] can only do real-time tracking at resolution 160 × 120. However, combined with a tracking trick, our method can achieve real-time tracking at resolution 640 × 480.

Our method contains three key steps: depth image split, contour extraction and ellipse fitting. Section 2 describes our method in detail. Section 3 evaluates our method on the Kinect face database proposed in [6]. Section 4 shows the integration of our driving fatigue detection system and the method proposed in this paper, followed by the conclusion and discussions in section 5.

2 Method

2.1 Assumptions

In our head detection method, we make two heuristic assumptions about human head.

a) Approximately, human head has an elliptical shape.
b) The depth values of human head are continuous.

Strictly speaking, these two assumptions doesn't hold for every people every time. For example, if a person has very bushy hair or wears accessories such as caps, his/her head may not have an elliptical shape in depth image. However, we find these two assumptions do hold in most cases. In fact, both [7] and [8] make assumptions which are similar with ours.

2.2 Work-Flow

Fig. 1 illustrates the overall process of our head detection method.

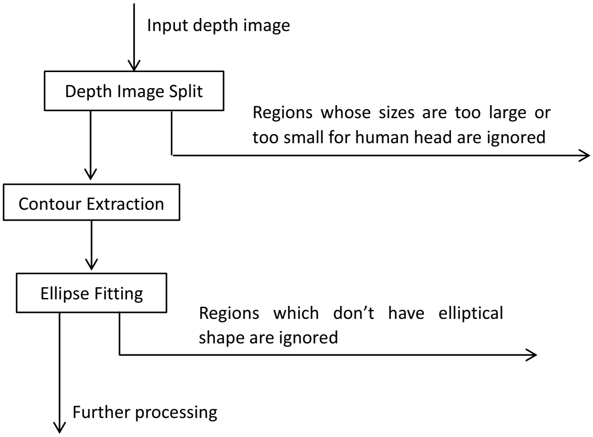
Input depth image
Depth Image Split
Regions whose sizes are too large or too small for human head are ignored
Contour Extraction
Ellipse Fitting
Regions which don't have elliptical shape are ignored
Further processing

Fig. 1. The work-flow of our head detection method

First, we split the depth image into regions according to the depth value of each pixel. Because we assume the depth values of human head are continuous, we can set a threshold and split depth image at locations where the change of depth is larger than the threshold. Regions whose size is much larger or much smaller than normal human head will be ignored. Second, we extract the contour of each region. For the extraction step, we use an algorithm which 'walks' along the contour of each region. Third, after we get the contour of each region, an ellipse fitting algorithm will be used and the similarity between the region and an ellipse will be calculated. Regions with an elliptical shape will be returned for further processing.

2.3 Depth Image Split

Image split is a kind of image segmentation. According to [9] there is quite a number of image split algorithms. Considering effectiveness, simplicity and speed, we choose image split based on computing connected components which is enough for our method.

In our design we use breadth first search to compute connected components. Neighboring pixels whose difference of depth is less than threshold will be treated as connected pixels. Fig. 2 is an example of image split. Here different regions are colored with different colors. We can see that the region of head has been split out. Notice that regions much larger or much smaller than normal human head will be removed.

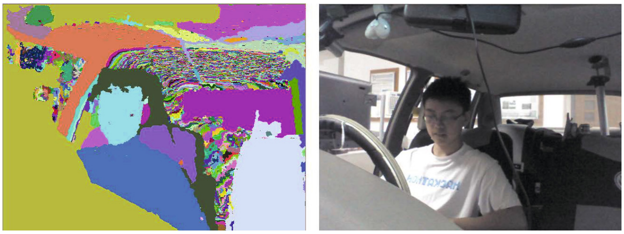

Fig. 2. An example of image split result (The right color image is for reference)

2.4 Contour Extraction

After the depth image is split into different regions, contour extraction is performed. Contours are expressed by contour points. Fig. 3 gives an example of a region (left figure) and it's corresponding contour points (right figure). One pixel in depth image is represented by one block in Fig. 3.

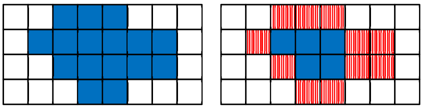

Fig. 3. A region in depth image and it's corresponding contour points

We use an contour extraction algorithm which 'walks' along the boundary of each region.

To represent contour points, a corresponding format is defined. This format contains not only the location of the point (*i.e.* the coordinates), but also the normal direction (*i.e.* the direction pointing to the 'outside' of the region). For example, if we express contour points in Fig. 3 using this format, we will have a list like this (starting from the top-left point)

$\langle 2, 0, \text{UP} \rangle \langle 2, 0, \text{LEFT} \rangle \langle 1, 1, \text{UP} \rangle \langle 1, 1, \text{LEFT} \rangle \langle 1, 1, \text{DOWN} \rangle \langle 2, 2, \text{LEFT} \rangle \ldots$

Each contour point is represented by three elements. The first two elements are the coordinates of the contour point, and the third element shows the normal direction of the contour point. Using this format, we can easily find and express the contour of a region.

The 'walking' procedure is a procedure of state transition, *i.e.* transition from a contour point to another contour point. The algorithm starts at a contour point of the region (top-left point of the region in our system), and 'walks' along the contour as the state transiting. When we get back to the start contour point, we know that we have found all the contour points of this region.

State transition depends on current contour point. We have four normal directions (UP, DOWN, LEFT, RIGHT), each direction has 3 situations, so we have 12 situations to consider in total. Fig. 4 gives us an example of the UP situation. DOWN, LEFT, RIGHT are very similar.

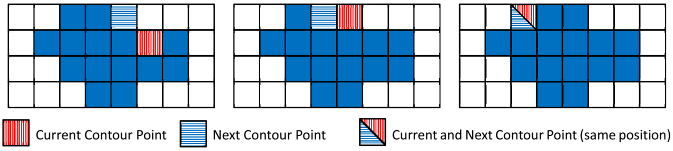

Fig. 4. Three cases of finding next contour point if current normal direction is UP

Notice that a certain location (coordinates) may contain more than one contour point. Fig. 5 is the contour extraction result of Fig. 2.

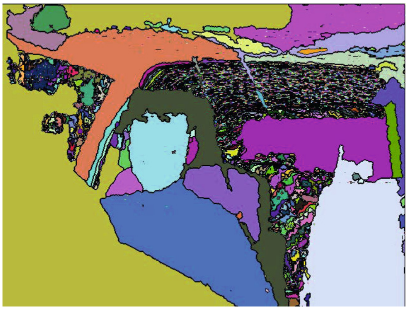

Fig. 5. An example of contour extraction result

2.5 Ellipse Fitting

Having found the contour of each region, we can use any ellipse fitting algorithm to fit an ellipse for the contour points. In our system, we choose the algorithm in [10]. We calculate fitness for each region using the formula below (less fitness value means higher fitness degree):

$$\frac{\sum_{i=1}^{n} \delta_i}{n \cdot h} \tag{1}$$

Here n is the number of contour points of the region, δ_i is the offset of the contour point to its corresponding point in the fitted ellipse (draw a line cross ellipse center and the contour point, the nearer intersection with the ellipse), and $h = \max\{\text{height of region}, \text{width of region}\}$. In conclusion, we calculate the normalized average offset of every contour points.

After we get the fitness of each region. We return the region whose fitness is less than a certain threshold as the human head we detected.

3 Experiments

We test our head detection method on a public available face database provided by Hg *et al.* [6]. In this database there are 31 persons, and each person has 17 poses (sitting in front of Kinect and looking at different positions with different facial expressions). For each pose, color image and depth image are taken at the same time for three times, so there are $31 \times 17 \times 3 = 1581$ color images and 1581 depth images. Both color images and depth images are taken using Microsoft Kinect, with a resolution 1280×960 for color images and a resolution 640×480 for depth images.

The test result is shown in Table 1. The detection rate is satisfying for most persons, and is not sensitive to head pose. But for persons with very bushy hair or beard, the detection rate drops heavily (boldface items in Table 1).

Table 1. Detection rate for each person in the database

Id	Detection Rate	Id	Detection Rate	Id	Detection Rate	Id	Detection Rate
1	88.24%	9	88.24%	**17**	**23.53%**	25	98.04%
2	90.20%	10	84.31%	18	100.00%	26	100.00%
3	100.00%	11	100.00%	**19**	**7.84%**	27	100.00%
4	100.00%	12	84.31%	20	100.00%	28	100.00%
5	64.71%	13	100.00%	21	100.00%	29	90.20%
6	**0.00%**	14	70.59%	22	100.00%	30	100.00%
7	100.00%	15	92.16%	23	94.12%	31	100.00%
8	**9.80%**	16	88.24%	24	100.00%	Average	83.05%

For detection speed, it takes about 170ms in average to do detection for an depth image of resolution 640×480 in my computer[1]. Compared with method proposed in [8], our method is about 2 times faster. The reason is that there are no template matching procedure in our method, which can greatly slow down the speed.

4 Integrate with Driving Fatigue Detection System

Fig. 6 shows the work-flow of our driving fatigue detection system. Here, head detection in depth image is an important step which speeds up the whole system. Originally, we detect human face in color image using Viola-Jones algorithm [1] directly, which requires about 1800ms for an image of resolution 1280×960. Now, we first detect human head in depth image, which requires about 170ms for an depth image of resolution 640×480, and then use traditional algorithm to detect face in the head region, which requires about 70ms.

[1] CPU: Pentium T4400 2.2GHz, RAM: 4GB

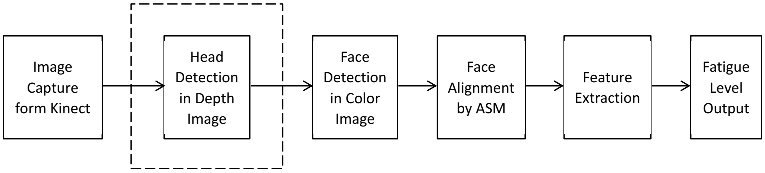
Image Capture form Kinect
Head Detection in Depth Image
Face Detection in Color Image
Face Alignment by ASM
Feature Extraction
Fatigue Level Output

Fig. 6. The work-flow of our driving fatigue detection system

After we get the accurate location of the driver's face, we use ASM (Active Shape Model) to determine the location of each organ on the face. This is done on color image. Then the appearance of each organ on the face can help us determine the facial expression of the driver. Currently, we use the shape of driver's mouth as a fatigue feature, which is expressed by a value:

$$d = \min\{100, \frac{h}{w} \times 100\} \tag{2}$$

Here h is the height of mouth, and w is the width of mouth. Therefore d increases as the driver opens his/her mouth. After we calculate d for each frame, we use formula below to calculate fatigue degree:

$$f = \min\{100, \frac{\sum_{i=1}^{n} d_i}{n} + 20t\} \tag{3}$$

Here n is the number of frames in 3 minutes, t is the number of yawns. For the calculation of t, we set a threshold for d. If d is larger than the threshold, we add 1 to t, and wait 10 seconds (we assume that yawn will not happen twice within 10 seconds or last for more than 10 seconds).

5 Conclusion and Future Work

In this paper, we proposed a novel human head detection method for depth image, which is both simple and robust. From the experiments, we can see that our method can achieve comparable detection rate and can reduce detection time.

This method utilizes depth information only, so it is not sensitive to light condition. The validity of the method is demonstrated by integrating the head detection method into our driving fatigue detection system. In the future, we plan to improve our method so that it can deal with very bushy hair and beard. For the driving fatigue detection system, we will add more features to get more accurate result.

Acknowledgements. This work was partially supported by the National Natural Science Foundation of China (Grant No. 61272248), the National Basic Research Program of China (Grant No. 2013CB329401) and the Science and Technology Commission of Shanghai Municipality (Grant No. 13511500200).

References

1. Viola, P., Jones, M.: Robust real-time face detection. International Journal of Computer Vision 57(2), 137–154 (2004)
2. Yang, M.H., Kriegman, D.J., Ahuja, N.: Detecting faces in images: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(1), 34–58 (2002)
3. Zhang, C., Zhang, Z.: A survey of recent advances in face detection. Microsoft Research (2010)
4. Colombo, A., Cusano, C., Schettini, R.: 3D face detection using curvature analysis. Pattern Recognition 39(3), 444–455 (2006)
5. Chew, W.J., Seng, K.P., Ang, L.M.: Nose tip detection on a three-dimensional face range image invariant to head pose. In: Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 1 (2009)
6. Hg, R.I., Jasek, P., Rofidal, C., Nasrollahi, K., Moeslund, T.B., Tranchet, G.: An RGB-D Database Using Microsoft's Kinect for Windows for Face Detection. In: International Conference on Signal Image Technology and Internet Based Systems, pp. 42–46 (2012)
7. Xia, L., Chen, C.C., Aggarwal, J.K.: Human detection using depth information by Kinect. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pp. 15–22 (2011)
8. Suau, X., Ruiz-Hidalgo, J., Casas, J.R.: Real-time head and hand tracking based on 2.5 D data. IEEE Transactions on Multimedia 14(3), 575–585 (2012)
9. Szeliski, R.: Computer vision: algorithms and applications. Springer (2010)
10. Fitzgibbon, A.W., Fisher, R.B.: A buyer's guide to conic fitting. In: Proceedings of the 6th British Conference on Machine Vision, vol. 2, pp. 513–522 (1995)