Real Time Face Detection using Geometric Constraints, Navigation and Depth-based Skin Segmentation on Mobile Robots

Vo Duc My
Cognitive Systems Group
Computer Science Department
University of Tuebingen
Sand 1, D-72076 Tuebingen, Germany
Email: duc-my.vo@uni-tuebingen.de

Andreas Masselli
Cognitive Systems Group
Computer Science Department
University of Tuebingen
Sand 1, D-72076 Tuebingen, Germany
Email: andreas.masselli@uni-tuebingen.de

Andreas Zell
Cognitive Systems Group
Computer Science Department
University of Tuebingen
Sand 1, D-72076 Tuebingen, Germany
Email: andreas.zell@uni-tuebingen.de

***Abstract*—Face detection is an important component for mobile robots to interact with humans in a natural way. Various face detection algorithms for mobile robots have been proposed; however, almost all of them have not yet met the requirements of the accuracy and the speed to run in real time on a robot platform. In this paper, we present a method of combining color and depth images provided by a Kinect camera and navigation information for face detection on mobile robots. This method is shown to be very fast and accurate and runs in real time in indoor environments.**

I. Introduction

Face detection is a necessary step for many other algorithms of face analysis such as face recognition, face tracking and facial expression recognition. With increased development of sensor technology, robots will be equipped with more and more different sensing modules, such as laser range scanners, sonar sensors and Kinect cameras. Among them, the Kinect is a relatively new device from which we can extract the depth values and color values of an arbitrary position in the image. Therefore, the Kinect becomes a powerful device to help robots to explore objects in 3D real world space. In this paper, we describe the way to combine color and depth images from the Kinect with navigation data to build a real time system of face detection. This combination gives us some advantages. First, we can compute 3D geometric constraints of objects based on depth values from the Kinect, which are used for removing non-face regions. Second, we can apply a new technique of depth-based skin segmentation for improving the efficiency of finding human skin as well as increasing the speed of detecting faces. By combining depth values, we can isolate skin areas in different objects and in different distances. Furthermore, we can determine the distance from them to the Kinect camera; and therefore, we can limit the size of potential faces in every skin region to reduce processing time. Third, the combination of depth values and navigation data gives robots an opportunity to determine 3D coordinates of every position in real world space. Thus, robots can easily ignore background regions, and only focus on the potential facial regions. We tested our method and achieved remarkable results of computational speed and accuracy. Figure 1 shows an example of our face detection which runs on the mobile robot platform SCITOS G5.

Fig. 1. Our face detection is able to run in real time on a robot platform in an indoor environment. By using geometric constraints, navigation and depth-based skin segmentation, the average processing time can be reduced by a factor of 50, and false positives are decreased by several tens of times

The remainder of this paper is organized as follows: We present previous researches on face detection in Section II. In Section III, our method is described in detail. In Section V, the experimental results obtained from our database are shown and conclusions are given in Section VI.

II. Related work

There is a long history of researching face detection in images and in videos. Some of these approaches can be applied to detect faces in real time, such as using skin color [8], depth information [11] and machine learning techniques [10], [9].

Especially, in applications for mobile robots, range data is used to make face detection faster. Blanco [3] utilized laser range scanners to determine potential candidates before applying face detection methods. Fritsch [7] used a stereo microphone to localize the voice of a talking person and detect the persons legs using a laser range finder. Such methods that use 2D laser range scanners do not give highly accurate results because lasers provide poor features that let robots confuse human legs with table legs or chair legs.

Additionally, geometric constraints based on depth information or lasers are also used to limit the search of facial regions, which was proved in [6]. Their approach uses geometric

constraints on possible human height and human facial size and can remarkably reduce the amount of average computation as well as decrease a large number of false positives. Even though this approach can reduce the computational cost by 85 percent, it has not yet met the real-time requirements for an image resolution of 640×480.

One further approach of face detection using depth information is reported in [4]. Their method uses depth data from a stereo camera to calculate the corresponding size of faces, applies distance thresholding to avoid detecting faces in the areas that are too far from the camera with too few face pixels. In addition, they suppose that with traditional stereo cameras, which provide sparse information, it is impossible to calculate depth values in the areas without texture, and in such areas, face detection methods do not work. Therefore, the locations that do not contain depth information are unnecessary to detect. In addition, the combination of depth values and context information helps a robot to avoid finding faces in irrelevant locations, such as ceiling, floor plane, etc. This method improved the processing time of face detection significantly . It is able to decrease the computational cost by 80 percent, but it reduces detection rate in the case that faces are far from the camera or the faces are close to the image bolder.

III. Approach

In this section, we describe our approach of real time face detection in detail. Our approach includes six steps: First, we use a strategy of sampling to reduce computational costs while not affecting the accuracy of the algorithm. Instead of scanning the whole image, we only collect data from several hundred sampling points that span the whole image. In the second step, we evaluate these sampling points under geometric constraints to select appropriate points for finding potential face regions. Similar to the second step, in our third step, we use constraints of navigation to extract the sampling points that belong to foreground and remove those which belong to the background. In the fourth step, skin detection is applied to the regions that are around the filtered sampling points. If the density of the skin pixels around a sampling point is over a given threshold, this sampling point will be kept for the next step. In the fifth step, all selected sampling points are used for depth-based skin segmentation to isolate potential face regions as well as to estimate the sizes of the faces that possibly occur in these regions. In the last step, these regions are scanned by a face detector to localize human faces.

A. Sampling

A popular and efficient technique that we applied in this paper is a sampling technique. For not losing the information of potential face areas, the sampling interval must be small enough to detect the smallest faces, which in our experiments have the size of 20×20 pixels . We choose a sampling interval of 10 pixels in both horizontal and vertical directions as described in Figure 2. In addition, the sampling positions in the color image and in the depth image must have the same coordinates. In every sampling point, the information set of depth value, skin region and navigation data is collected and processed to serve the next preprocessing steps.

B. Geometric constraints

Based on the combination of color and depth images from the Kinect and knowledge of the camera's geometry, we can compute 3D coordinates of every point. The robot can now estimate the appropriate size of a human face at a certain sampling point. A sampling point which is too far from the camera can be ignored because the robot can not detect any faces there even if they occur. The robot is also able to ignore irrelevant regions which do not contain human faces for instance, floor and ceiling. Furthermore, we limit the height of the search area because we know the minimum and maximum height of humans when they sit on a chair, stand or walk. We now describe how to apply geometric constraints for a mobile robot to limit the search space by using depth values collected at sampling points:

First, we present the way to compute the transformation between coordinates of the robot frame and 2D coordinates of the images. We denote the 3D coordinates of an arbitrary point in the robot frame with respect to the Kinect camera with $({}^{c}x_p, {}^{c}y_p, {}^{c}z_p)$; ${}^{i}x_p$ is the depth value of this point in the depth image; ${}^{i}y_p$ and ${}^{i}z_p$ are 2D coordinates of this point in the color image. We mount the Kinect camera on the robot such that the normal vector of the depth image plane is parallel to the ${}^{c}x_p$ axis and ${}^{c}y_p$ and ${}^{c}z_p$ axes are parallel to ${}^{i}y_p$ and ${}^{i}z_p$ axes of depth image, respectively. We can now compute the $({}^{c}x_p, {}^{c}y_p, {}^{c}z_p)$ coordinates

$$ {}^{c}x_p = {}^{i}x_p \tag{1} $$

$$ {}^{c}y_p = \frac{({}^{i}y_o - {}^{i}y_p){}^{i}x_p}{f_y} \tag{2} $$

$$ {}^{c}z_p = \frac{({}^{i}z_o - {}^{i}z_p){}^{i}x_p}{f_z} \tag{3} $$

where ${}^{i}y_o$ and ${}^{i}z_o$ are 2D coordinates of the principal point, f_y and f_z are the focus lengths of depth camera. Now the height in robot frame coordinates of sampling points in the depth image can be computed. The sampling points which are out of range from minimum height h_{min} to maximum height h_{max} are ignored since they are unlikely in potential facial areas. Furthermore, the sampling points that are too far from the camera are also ignored, as mentioned above

In addition, the robot can estimate the appropriate size of human faces in different distances; therefore, it can limit the range of scales to detect faces. We assume that average size of a human face is 0.15 meters; therefore, the formula to estimate the size of faces in images is

$$ s_{face} = \frac{0.15 f_y}{d} \tag{4} $$

where s_{face} is the average size of faces in the distance of d meters. The constraint of face size contributes to a significant reduction of processing time because the robot only has to scan potential facial areas in one scale instead of multiple scales.

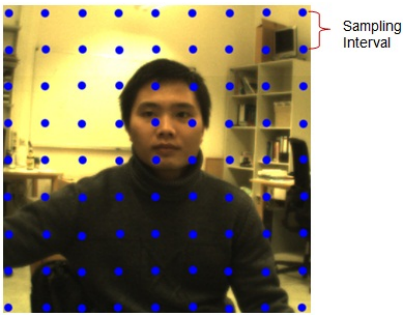

Fig. 2. Example of sampling. The sampling interval is 10 pixels in both horizontal and vertical directions

C. Navigation

One of the advantages of current robot software is that it contains different modules, including cameras and other sensors, motion control, navigation, etc. These modules run concurrently and are able to share data. In our face detection task, we utilize the navigation module for reducing the space of searching faces. We found that the Kinect camera allows a robot to determine 3D coordinates of every object with respect to robot coordinates, while the navigation module provides a robot with its 3D coordinates with respect to world coordinates. Therefore, we can track human activity around the robot. The fusion of these modules gives a robot the ability to map the points in Kinect images to cells of an occupancy grid map. This mapping helps the robot to avoid searching the background regions whose corresponding cells are occupied in the grid map, and focus on the regions of human activities, whose corresponding cells are free.

We note that $({}^{r}x_c, {}^{r}y_c, {}^{r}z_c)$ are 3D coordinates of the Kinect camera with respect to robot coordinates, $({}^{w}x_r, {}^{w}y_r, {}^{w}z_r)$ are the coordinates of the robot with respect to the origin of the navigation map and θ is the rotating angle of the robot. Thus, the coordinates of an arbitrary point in the world space with respect to robot coordinates, $({}^{r}x_p, {}^{r}y_p, {}^{r}z_p)$, are computed as

$$\begin{pmatrix} {}^{r}x_p \\ {}^{r}y_p \\ {}^{r}z_p \end{pmatrix} = \begin{pmatrix} {}^{r}x_c \\ {}^{r}y_c \\ {}^{r}z_c \end{pmatrix} + \begin{pmatrix} {}^{c}x_p \\ {}^{c}y_p \\ {}^{c}z_p \end{pmatrix} \tag{5}$$

After that the coordinates of this point with respect to the origin of the grid map, $({}^{w}x_p, {}^{w}y_p, {}^{w}z_p)$, are computed by the following formulas:

$$\begin{pmatrix} {}^{w}x_p \\ {}^{w}y_p \\ {}^{w}z_p \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} {}^{r}x_p \\ {}^{r}y_p \\ {}^{r}z_p \end{pmatrix} + \begin{pmatrix} {}^{w}x_r \\ {}^{w}y_r \\ {}^{w}z_r \end{pmatrix} \tag{6}$$

These coordinates are mapped to an occupancy grid map to determine the corresponding cell. As a result, we use the above formulas for updating navigation information for every sampling point. A sampling point would not be used for segmenting potential face regions when it is mapped to an occupied cell in the grid map because it belongs to a certain background area.

D. Skin detection

In this section, we explain a skin detection technique mentioned in [8]. In general, the idea of this method is that it tries to eliminate the influence of non-standard illumination before using a set of simple rules which are very efficient to classify skin and non-skin pixels under standard illumination conditions. In order quickly eliminate non-standard illumination from images we use the Gray World assumption of color compensation. The Gray World theory is built on the presumption that the average surface color on the image is achromatic. With this presumption, we try to find the scale factors which adjust the color channels R, G and B in such a way that their mean values are equal to the ones under standard illumination. The scale factors are calculated as follow

$$s_n = \frac{t_n}{a_n} \tag{7}$$

where s_n is the scale factor of channel n, t_n is the standard gray value of channel n, and a_n is the average value of channel n. Because the computation of the gray world method in the whole image is a quite expensive while the illumination does not change too much in short time, the scale factors are only adjusted every second. After eliminating non-standard illumination, image pixels are classified as skin or non-skin using a set of fast and simple rules. So we denote $I(i,j)$ as the skin value in a point at row i and column j in the color image. $I(i,j)$ is equal to 1 if following rules [8] are satisfied and equal to 0 if not:

$$R > 95 \text{ AND } G > 40 \text{ AND } B > 20 \text{ AND}$$

$$\text{Max}\{\text{R, G, B}\} - \text{Min}\{\text{R, G, B}\} > 15 \text{ AND}$$

$$|R - G| > 15 \text{ AND } R > G \text{ AND } R > B$$

where R, G, B are three values of red, green, and blue elements in an arbitrary pixel, respectively. Such skin detection method will be applied to determine whether the area around a sampling point in color image is skin or not. $T(m,n)$ is defined as the sum of skin values in a set of pixels around the sampling point at coordinates (m,n) and is computed by the following formula:

$$T(m,n) = \sum_{i=-2}^{2} \sum_{j=-2}^{2} I(sm+2i, sn+2j) \tag{8}$$

where s is the sampling interval. The area around the sampling point at coordinates (m,n) is a skin area if $T(m,n)$ is higher than a threshold, otherwise it is non-skin area. The calculation of skin areas around sampling points improves the skin segmentation that will be mentioned in the next session.

E. Depth-based skin segmentation

Color-based skin segmentation is known to be a fast and efficient method of finding potential facial regions. However, a drawback of this method is that it is much influenced by different illumination sources and also detects skin-tone color objects (wood, hair, some clothes...), which both cause the segmentation to fail. In many cases, segmented areas would occupy a big part of the image. It makes the face detection system run very slowly and less accurate. Furthermore, the previous method of skin segmentation is used for classifying

Fig. 3. Result of depth-based skin segmentation. The skin regions such as face, hand are segmented while false positives in background are removed by constraints of geometry and navigation

every pixel into differently labelled skin regions, which takes a lot of time. In this section, we describe our technique of depth-based skin segmentation which improves efficiency of finding human skin as well as reduces the computational cost. Instead of classifying every pixel, we try to label every sampling point in such a way that two sampling points have the same label if their sets of conditions of depth values, geometric constraints, navigation constraints, and skin values are similar. The technique of skin segmentation using sampling points remarkably reduces the processing time, in our case by a factor of 100. The number of sampling points would be reduced further if we use constraints of geometry and navigation together with skin detection to filter sampling points out of interest ranges. After using such constraints, unnecessary sampling points are removed, and we use a fast and accurate algorithm to segment these filtered sampling points. This algorithm is used to classify filtered sampling points into different labelled regions instead of classifying image pixels. In our algorithm, two filtered sampling points A and B will have the same label if the distance between them is under a threshold, which is set to a half of the average size of a face estimated at the sampling points. This threshold is applied to remove the effect of disconnected skin regions on a face which are made by noise and concave areas around eyes. With such a strategy, all filtered sampling points are classified rapidly into different skin regions as described in Figure 3. In this figure, the false positives in the background are removed by using constraints of geometry and navigation, and the depth-based skin segmentation is applied to isolate potential face regions.

The algorithm of depth-based skin segmentation is shown to be faster and more accurate than previous algorithms of connected components since we can remove false positives in background and eliminate affect of noise without using expensive operators of erosion and dilation

F. Face detection

In our system, we use the Viola-Jones algorithm [10] for our system of face detection because it can process images extremely quickly and is very accurate. There are three key contributions that result in the extremely efficient Viola-Jones algorithm: the integral image, Adaboost learning, and the cascade architecture. The integral image is an effective image representation to reduce computational costs when testing or training Haar features. Adaboost learning allows to train a fast and accurate classifier by selecting a small number of the best features from tens of thousands of possible features. Finally, these classifiers are combined to make a cascade architecture model to quickly reject a large number of false positives in the stages until achieving a high detection rate. Based on these contributions, the Viola-Jones method is likely the fastest machine learning method for face detection. However, when dealing with a high resolution image, for instance, 640×480 in our system, the Viola-Jones method alone does not meet the requirement of a real time face detection system for mobile robots. Therefore, in our system, we apply the above preprocessing steps to reduce the computational cost before using the Viola-Jones face detector.

IV. Experimental Setup

In this section, we present the experimental evaluation of our system by using different constraints. To evaluate the accuracy as well as the speed of our algorithm, we compared it with the performance of Viola-Jones algorithm of face detection built in the Intel OpenCV library [1]. All experiments are based on our database collected from two kinds of mobile robots, PR2 and SCITOS G5, and from different indoor environments, including offices, corridors, kitchen, museum and laboratory. The robots are equipped with Kinect sensors and Sick S300 laser scanners. We use a PC with a 2.4 GHz Intel Core 2 CPU to test our algorithms in these experiments.

A. Datasets

To evaluate our face detection algorithm, we used two challenge datasets. The first one is the Michigan dataset which comprises seven ROS log files spanning from one to three minutes, which are selected from datasets of detecting and tracking humans [5]. The selected log files must contain front faces in color images in different indoor environments, and in different illuminations conditions. These log files in the Michigan dataset recorded only Kinect color and depth images at 30 frames per second, but do not contain an occupancy grid map; therefore, we just use it for the first experiment which does not require constraints of navigation. All these log files are collected from the a Willow Garage mobile robot PR2 that moves around an office building with kitchens, corridors, meeting rooms and departments to detect human front faces. The mobile robot PR2 is equipped with a Kinect camera that is mounted 2.0 meters high on the top of robot's head, and looks downward. The second dataset is the Tuebingen dataset which consists of seven ROS log files collected from a MetraLabs mobile robot SCITOS G5. Every log file lasts from one minute to three minutes, and contains Kinect color and depth images, tf transform messages, and laser range data. This mobile robot used a Kinect camera mounted 1.1 meters high and looking forward. To record the log files, the mobile robot had to move around in our office building, including the laboratory, the corridors and the museum with different backgrounds,

and different illumination conditions. Moreover, we provided an occupancy grid map of our building for mobile robots; therefore, the Tuebingen dataset can be used for both of the experiments to test face detectors with or without navigation.

B. Implementation Details

To evaluate the efficiency of the constraints in improving the processing time and accuracy, we carried out two different experiments, in which we compared our algorithm with the OpenCV face detector. The OpenCV face detector was run from a smallest size of 20×20 pixels and scaled up by a factor 1.2 in whole images. The first experiment was implemented to evaluate the role of geometric constraints and depth-based skin segmentation in improving the face detection performance. A face detector using geometric constraints, depth-based skin segmentation but not navigation is shown to can run efficiently in real time on mobile robot platforms as well as in surveillance systems. This detector was compared with the OpenCV face detector in accuracy and processing time. In the second experiment, a second face detector using geometric constraints, navigation and depth-based skin segmentation was compared with the above face detector without using navigation and the OpenCV face detector. This experiment was performed to demonstrate the efficiency of navigation information in reducing the number of false positives and the computational cost.

Our whole system is programmed based on the ROS system [2]. ROS is a powerful system for software developers which provides libraries, tools, message-passing, package management for building robot applications. Our robot system consists of many ROS nodes such as the node of controlling the laser range-finder, the node of performing localization, the node of managing the Kinect operation, the node of providing the graphical view, and the node of process odometry information. By using ROS, the color and depth images from the Kinect can be synchronized and fused with other sensors, and robot coordinates are also updated frequently based on the tf package which is responsible for computing 3D coordinate frames. Furthermore, an occupancy grid map of our building is used to provide the information of background for mobile robots.

Our face detection system can run at 7.9 milliseconds per frame on average with an accuracy of over 95 % in our datasets

V. Results

In this section, we compare the results of the accuracy and speed between our face detection algorithm and the OpenCV face detector. The first experiment of testing the efficiency of geometric constraints and depth-based skin segmentation was implemented in both datasets. The second experiment for testing the influence of navigation ran on the Tuebingen dataset which includes the occupancy grid map.

A. Processing time

Table I shows a distinguished difference between our detector and the OpenCV face detector in processing time when they are run on both datasets. We denote the first method as the face detection method using geometric constraints and depth-based skin segmentation.

TABLE I
Comparison of processing time between the OpenCV face detector and the first method.

Datasets	OpenCV	First Method
Michigan	500.1 ms	8.7 ms
Tuebingen	530.9 ms	13.1 ms

TABLE II
Comparison of processing time among the OpenCV face detector, the first method and the second method.

Dataset	OpenCV	First Method	Second Method
Tuebingen	530.9 ms	13.1 ms	7.9 ms

We can find that processing time of the face detector using geometric constraints and depth-based skin segmentation is much less than that of the OpenCV face detector because its average processing time is only 8.7 ms in the Michigan dataset, and 13.1 ms in the Tuebingen dataset. At such speed, it runs as much as 41 to 57 times faster than the OpenCV face detector. It also means that the use of geometric constraints and depth-based skin segmentation can reduce the computational cost by 98 %.

To evaluate the contribution of navigation, we continued implementing the second experiment to compare the face detector using geometric constraints, navigation and depth-based skin segmentation with two above detectors. Table II shows the results of the three methods which are run in the Tuebingen dataset. We denote the second method as the face detection method using geometric constraints, navigation and depth-based skin segmentation.

With average processing time of 7.9 ms, the second method is even faster than the first method, and it also means that the computational cost are reduced by 99 %. Therefore, the navigation information plays an important role in avoiding searching irrelevant regions as well as avoiding unnecessary computations. That is one of the advantages of mobile robot in building such real time systems as face detection, or face recognition.

B. Accuracy

To evaluate the accuracy of a face detection method, we have to consider both the ratio of true positives and the ratio of false positives detected in the datasets. A good algorithm must not only achieve a high detection rate of true positives but also limit false positives in an image. In the experiments, we found that besides the advantage of processing speed, our methods can also improve the accuracy of face detection significantly: the ratio of detected true positives is similar to the OpenCV face detector, the ratio of false positives is much lower.

In the first experiment, we use both datasets without the occupancy grid map for testing the OpenCV face detector and the first method. Table III shows the comparison of the accuracy between these face detectors. The correct detection rate of our method is still high, 95 %, even though it is a little lower than the OpenCV face detector, 96 %. But our method improved remarkably the average false positives per frame, only 0.003, compared to the OpenCV face detector with 21 times more, 0.063. In general, the accuracy of our method is still more reliable than the unmodified OpenCV face detector.

TABLE III
COMPARISON OF THE ACCURACY BETWEEN THE OPENCV FACE DETECTOR AND THE FIRST METHOD.

Face Detectors	Correct Detection Rate (%)	Average False Positives per Frame
OpenCV	96	0.063
First Method	95	0.003

TABLE IV
COMPARISON OF THE ACCURACY AMONG THE OPENCV FACE DETECTOR, THE FIRST METHOD AND THE SECOND METHOD.

Face Detectors	Correct Detection Rate (%)	Average False Positives per Frame
OpenCV	97	0.079
First Method	95	0.005
Second Method	95	0.005

In the second experiment, we only use the Tuebingen dataset which contains the occupancy grid map for testing three face detectors. Table IV shows the comparison of the accuracy among them. We also compared the face detectors based on two criteria of the correct detection rate and the average false positives per frame. Both of the first and second methods achieve the correct detection rate of 95 %, which is slightly lower than the OpenCV face detector but their average false positives per frame are 16 times less than the OpenCV face detector. This showed that our face detectors are reliable for mobile robots. The reason is that the preprocessing steps eliminate almost all of the background which contains a lot of false positives, but do not affect the correct detection rate much.

Figure 4 shows the result of our face detection in different cases: a group of people standing at roughly the same distance; several people standing at different distances to the robot; people standing near a wall; people with different skin color; people at far distances.

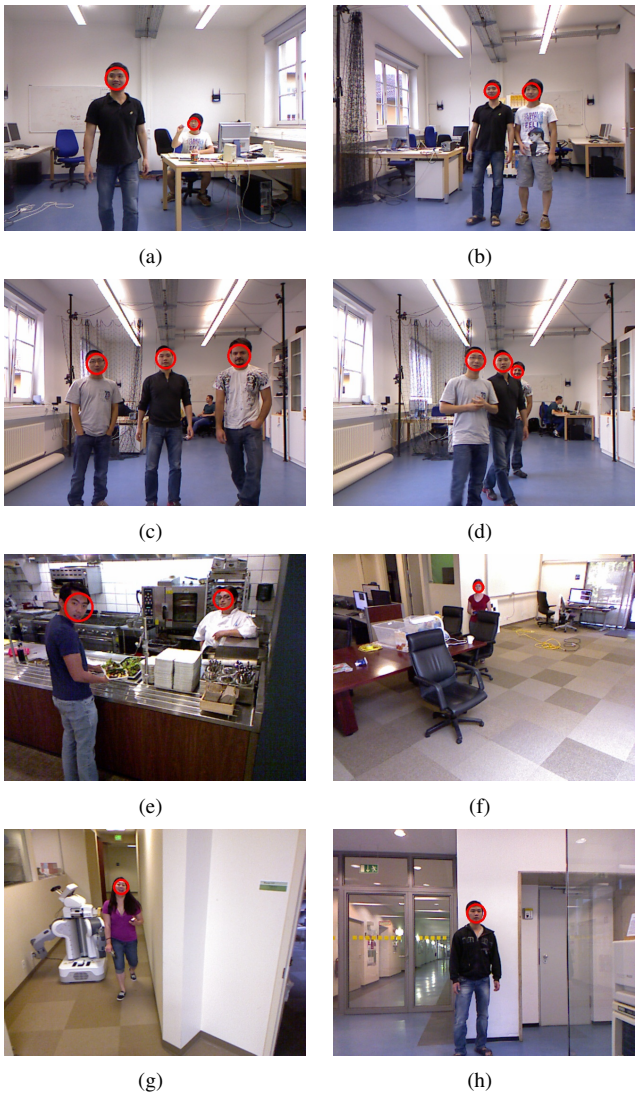

Fig. 4. Result of our face detection in different cases. 4(b), 4(c): A group of people standing at roughly the same distance; 4(a), 4(d), 4(e): Several people standing at different distances to the robot; 4(g), 4(h): People standing near a wall; 4(e): People with different skin color; 4(f): People at far distances

VI. CONCLUSION

This paper demonstrates that the contributions of geometric constraints, navigation information and depth-based skin segmentation to face detection are remarkable. Based on their combination, mobile robots are able to localize human faces in real time, and interact with humans in a more reliable way. In our future work, we focus on a promising direction of face tracking using navigation information. When human moving directions are repetitive, a robot can compute prior probabilities to predict the next face positions in 3D space.

REFERENCES

[1] Opencv, www.intel.com/research/mrl/research/opencv/.

[2] Ros, the robot operating system. http://www.ros.org/.

[3] J. Blanco, W. Burgard, R. Sanz, and J.L. Fernandez. Fast face detection for mobile robots by integrating laser range data with vision. In *Proc. of the International Conference on Advanced Robotics (ICAR)*, 2003.

[4] Walker Burgin, Caroline Pantofaru, and William D. Smart. Using depth information to improve face detection. In *Proceedings of the 6th ACM/IEEE International Conference on Human-Robot Interaction (Late Breaking Papers Track)*, Lausanne, Switzerland, 2011.

[5] Wongun Choi, C. Pantofaru, and S. Savarese. Detecting and tracking people using an rgb-d camera via multiple detector fusion. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1076 –1083, nov. 2011.

[6] M. Dixon, F. Heckel, R. Pless, and W.D. Smart. Faster and more accurate face detection on mobile robots using geometric constraints. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1041 –1046, 29 2007-nov. 2 2007.

[7] J. Fritsch, M. Kleinehagenbrock, S. Lang, G. A. Fink, and G. Sagerer. Audiovisual person tracking with a mobile robot. In *In Proc. Int. Conf. on Intelligent Autonomous Systems*, pages 898–906. IOS Press, 2004.

[8] J. Kovac, P. Peer, and F. Solina. Human skin color clustering for face detection. In *EUROCON 2003. Computer as a Tool. The IEEE Region 8*, volume 2, pages 144 – 148 vol.2, sept. 2003.

[9] A. Treptow and A. Zell. Combining adaboost learning and evolutionary search to select features for real-time object detection. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 2107 – 2113 Vol.2, june 2004.

[10] Paul Viola and Michael J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, May 2004.

[11] Haiyuan Wu, Kazumasa Suzuki, Toshikazu Wada, and Qian Chen. Accelerating face detection by using depth information. In *Proceedings of the 3rd Pacific Rim Symposium on Advances in Image and Video Technology*, PSIVT '09, pages 657–667, Berlin, Heidelberg, 2008. Springer-Verlag.