Service Robot for Deaf and Speech-impaired People

Demo video link:
https://www.space.ntu.edu.tw/navigate/s/B0677E24C6B042988FCDD705B77E351AQQY

Hsuan-Chu, Lin
Electrical Engineering Department
National Taiwan University
Taipei, Taiwan
b03901065@ntu.edu.tw

Yi-Chen, Hsu
Electrical Engineering Department
National Taiwan University
Taipei, Taiwan
b03901063@ntu.edu.tw

Wei-Li, Lin
Computer Science Department
National Taiwan University
Taipei, Taiwan
b03902047@ntu.edu.tw

Szu-Yu, Mo
Electrical Engineering Department
National Taiwan University
Taipei, Taiwan
b04901164@ntu.edu.tw

***Abstract*—In this project, we built an application for Pepper which can serve deaf and speech-impaired people. Deep learning is used to recognize sign language so that Pepper can understand and interact with those who cannot speak and hear. SLAM is also implemented to construct a map for the surroundings and for further navigation. With the help of this application, Pepper is able to guide or provide other services for those disabled people.**

Keywords—service robots, action recognition, computer vision, deep learning, SLAM, Pepper

I. INTRODUCTION

Social robots have recently become a popular issue in robotics field since they can interact with humans and serve them. There are plenty of social robots being commercialized, but few of them are able to serve deaf and speech-impaired people. Therefore, we aim to build an application on the social robot, Pepper, so that it can understand sign language and communicate with disabled people. We developed a deep learning model for sign language recognition and involved in every developmental progress, including data collection, image processing, and training neural network. Moreover, we used simultaneous mapping and localization to accomplish the navigation work of Pepper and applied landmark detection to localize its position more precisely. The techniques mentioned above and robot manipulation were integrated into a complete project. Details are shown in the following parts.

II. RELATED WORK

A. Human action recognition

The problem of sign language detection can be generalized as human action recognition, which has been long studied in computer vision area. Initially, handcrafted features have dominated this field for a long time. Improved Dense Trajectories (IDT) [1] is currently the one with best performance among them. It densely samples feature points in video frames and track them with optical flow, extracting different features along the trajectory. However, despite its good performance, this method is computationally intensive and can hardly be used in real-time applications.

With recent availability of powerful parallel machines such as GPUs, deep convolutional neural networks (CNN) [2] have been shown to be successful on image recognition tasks [3]. To recognize actions in video, 3D convolutional networks [4] (Figure 1.b) with 3D kernels (filters extended along the time axis) extracting features from both spatial and temporal dimensions are introduced, so that spatial-temporal information and motions in adjacent frames can be captured. Based on these 3D kernels, Tran et al. [5] attempt to find generic descriptors for videos. They show that a network with $3 \times 3 \times 3$ homogeneous filters performs best among various architectures for 3D convolutional neural networks.

Besides 3D CNN, some studies employ different structure such as recurrent neural networks (RNN) (Figure 1.a) to exploit the temporal information. Baccouche et al. [6] tackle the problem of action recognition by adding a specific RNN known as Long-Short Term Memory (LSTM) [7] after a series of cascading CNNs. On the other hand, many recent works adopt multiple stream design [8] (Fig. 1). With external computed optical flow information, they can achieve very high performance on existing benchmarks. However, compared to 3D CNN, both of these models require additional computation, and hence are not suitable for our real-time sign language recognition system.

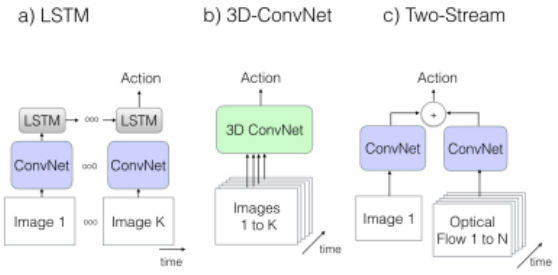

Fig. 1. Common action recognition model architectures

B. Simultaneous localization and mapping

The simultaneous localization and mapping (SLAM) problem of humanoid robots has frequently been investigated. Some have implemented localization and navigation capabilities on the Pepper using the ROS middleware, combining ROS Indigo with the Pepper NAOqi framework[9]. Others have tried to train neural networks to control the robot to reach the target location in urban dynamic environments. The robot has to rely on GPS and compass sensor to navigate from the starting point to the goal location in an environment with moving obstacles[10]. Still, SLAM is even improved with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling[11]. However, few of them integrate human interaction within, not to mention the target group of our project, the deaf and visually-impaired people.

III. Methodology

A. Sign Language Data Collecting

It is not possible to include the whole set of sign language into our project since we have to collect all the data by ourselves. Therefore, we choose 9 commands (Yes, No, I want to go to restroom, I want to checkout, I want to drink that, I want to buy meat, I want to buy vegetable, I want to buy eggs and I want to buy fruits) that our robot should recognize. We record our video data with Kinect 2 as the built-in camera in pepper does not provide sufficient field of view (FOV) as well as resolution. Each video is 5 seconds long, or 25 frames with fps equals to 5. However, including RGBA, depth and infrared channels, the raw video size of Kinect 2 is 25 x 424 x 512 x 6, which is way too large for our model to handle. Therefore, we decide to keep only intensity (grayscale), depth and infrared information, crop to 320 x 320, and then down-sample it by a factor of 2, resulting in 1/17 of original size. Finally, we get 180 videos after each of us records 5 videos for each command.

B. Sign Language Data Preprocessing

Since our data contains depth information, we can generate a mask that corresponds to the foreground part by thresholding at a certain depth value. However, there are many "holes" in the depth image, and thus in the generated mask. Hence, we perform binary closing then binary opening on the mask to remove these holes. Then, we fill the holes in original data with the local maximum value in the neighborhood, which can be done by grayscale closing. Finally, the hole-removed data is masked by the hole-removed mask, resulting in a clear foreground image. For more details, please refer to results section.

C. Sign Language Model Training

Our model architecture is shown in the Table 1. In order to minimize the time for prediction, our model contains only convolution, ReLU, pooling and fully connected layer, which can be highly parallelized. We do not use recurrent model like LSTM, and do not require external computation like optical flow. Following the principle [5] of designing 3D convolutional neural networks, the convolution filters are all of shape 3 x 3 x 3. The input layer may consist of 1 to 3 channels, depending on how much information we use for training and prediction. We have tested 7 different configurations and found that using grayscale together with infrared gives best result. For more details, please refer to results section. Also, data augmentation including translation (along all three axes) and flipping is used to alleviate the problem of limited amount of training data.

Table 1. Model architecture for sign language recognition.

Layer	Output Shape	Param
Input	(batch size, 25, 160, 160, 1~3)	0
Conv3D	(batch size, 25, 160, 160, 32)	2624
ReLU	(batch size, 25, 160, 160, 32)	0
MaxPooling3D	(batch size, 12, 80, 80, 32)	0
Conv3D	(batch size, 12, 80, 80, 32)	27680
ReLU	(batch size, 12, 80, 80, 32)	0
MaxPooling3D	(batch size, 6, 40, 40, 32)	0
Conv3D	(batch size, 6, 40, 40, 64)	55360
ReLU	(batch size, 6, 40, 40, 64)	0
MaxPooling3D	(batch size, 3, 20, 20, 64)	0
Conv3D	(batch size, 3, 20, 20, 64)	110656
ReLU	(batch size, 3, 20, 20, 64)	0
MaxPooling3D	(batch size, 1, 10, 10, 64)	0
Flatten	(batch size, 6400)	0
Fully Connected	(batch size, 64)	409664
Fully Connected	(batch size, 9)	585
Total Parameters: 606569		

D. Face Detection

Pepper is ready to offer service at any time. To trigger the social service of Pepper, face detection is implemented to detect a human. It is done by applying the face detection module in NAOqi API[12]. Once a face is detected by Pepper, the ALTracker, provided by NAOqi API as well, will turn its head and keep an eye contact with the person in front. In that way, people would be aware that he or she is detected by Pepper, and thus following interactions could start.

E. Tablet Image Showing

Usually, Pepper talks to people by text-to-speech function. However, due to the fact that the deaf aren't able to listen to speech, the way to communicate with human is replaced by showing images on the tablet right in front of Pepper's chest. Each image contains a sentence together with some pictures if necessary.

This work is done by first uploading all the images that we would like to show to a new project in Choregraphe[13], a NAO software for connection between computer and Pepper. The project would then be uploaded to Pepper, transferring those image files to Pepper's own memory. By that time, we can show those images on the tablet by applying image module in NAOqi API.

F. Mapping and Localization

To fulfill the navigation goal of our project, SLAM has to be implemented to get a map of the environment. We used the

exploration API to construct a map as Fig. 2 shows. The exploration range is set to be at most 8 meters far from the original point.

After exploring the environment, the map is loaded to a localization module which can be found in the NAOqi API as well. During every localization, Pepper would first relocalize itself with the last coordinate that it has just reached. After that, Pepper would move to the destination position according to the exploration map, and return the position where Pepper believes it is in.

G. Landmark Detection

To make sure Pepper precisely return to its original point after guiding people to some destination, coordinate calibration is done by landmark detection. We placed six landmarks on the floor near the original point, each with a specific coordinate and a phase angle. After Pepper believes that it has returned to the original point using the localization method, it would start to look for landmarks on the floor. If any landmark is in Pepper's sight, Pepper would adjust its own position so that the relative position to the landmark is consistent with what it sees. The landmark detection would continue until Pepper arrives its original point.

H. Object Recognition

In order to grab objects, robots should detect whether the certain object appears within the range it could reach. Therefore, we applied the vision module in NAOqi API so that Pepper could understand different pictures and object sides and further recognized different objects.

First, Pepper was taught to recognize specific object by receiving its information, including the contour and visual key points. Information for each target object was stored in one XML file and accompanied with its respective images and placed in the memory of the robot. With the information, Pepper could start recognize objects based on the recognition of visual key points and check whether the specific object exist in its view. The recognition process is robust to distance but the resolution of images would strongly impact the performance of this module.

I. Robot Manipulation

Pepper should also move to complete each task. In this project, we applied the motion module in NAOqi API to control Pepper's pose and motion. The motion module provides several methods to facilitate robot movement. We chose joint control to make different poses and locomotion control to move the robot to certain places. The former directly controlled the positions of each individual robot joints. We specified the name of the joint, the target angle and the angular speed so that Pepper could hold a specific pose as we needed, such as grabbing a cup or waving. On the other hand, locomotion control enabled Pepper move to a target place on the ground plane by given the relative position data.

Fig. 2. Map constructed by exploration API.

Moreover, the motion module provides a function for Pepper that can avoid collision. The function could model the body of Pepper and calculate whether collision would happen. As long as this function is opened, we don't have to worry that the robot would be crashed while programming. To be noticed, this function should be closed when we try to make Pepper grab an object.

IV. Experimental Setup

A. Demo scenario

The scenario for out project is set at a supermarket, where Pepper is standing at the entrance awaiting some deaf and speech-impaired customers. Those people can ask some questions by posing sign language to a camera nearby. Pepper would then react to the query and offer the service for humans.

B. Hardware and software

The hardware we utilized in our project contains Kinect 2 for Xbox one, Pepper robot from SoftBank, and a laptop from Acer. The software we used includes Ubuntu 16.04, OpenCV, OpenGL, libfreenect2, pyfreenect2, cuda-8.0, tensorflow, Keras 2, Pepper SDK packages 2.5.5.5, Choregraphe 2.5.5.5. The following paragraph depicts the setup of these environments and how they connect to each other.

C. Demo workflow

Pepper is initially placed at the original point and ready for face detection. Once a face is detected by Pepper, the robot would be triggered and start to track the person's face. There would be an image showing "What can I do for you?" on the tablet. An image showing "Please look at the camera." would be displayed consecutively. The person would then turn to the Kinect camera which is placed at the right-hand side of Pepper, and pose the sign language of the command that Pepper can recognize. After approximately three seconds of video processing and recognition, Pepper would show an image on the tablet to confirm the request. The person should again turn to the Kinect camera to pose "yes" or "no". If the answer is "no", Pepper would show the image "What can I do for you?" to start another video recording. If the answer is "yes", Pepper would perform the request queried by the person. Among the nine commands trained, one is "I want to drink that". Pepper would operate object detection to make sure there are drinks on the table at the left-hand side of it, and then take the bottle for the person. Other commands are navigation requests. Pepper would lead you to the place you have asked for. Once Pepper arrives at the destination, it would turn around and

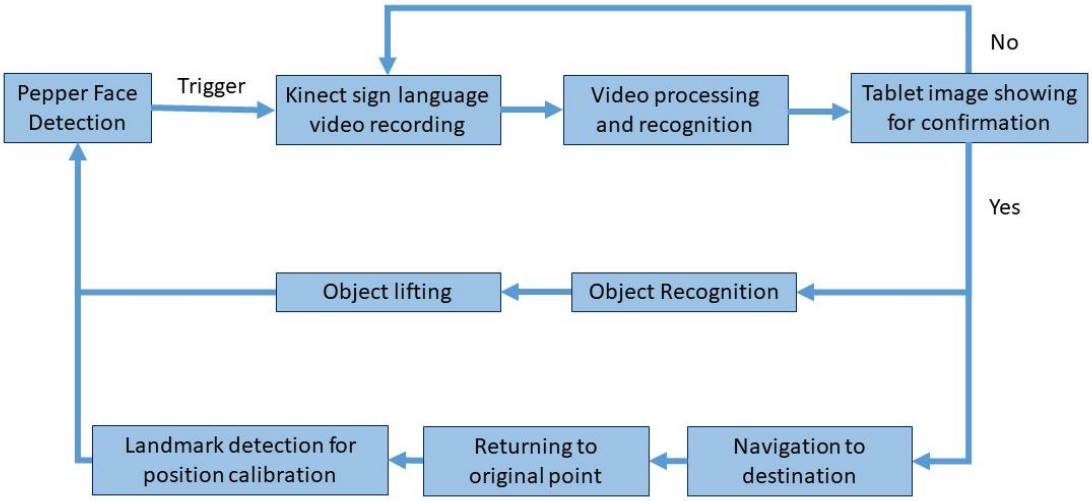

Fig. 3. The experimental setup of our project

perform face detection again to face at the person. Next, Pepper raises his hand and shows an image reading "here it is" to inform the person it has successfully reach the destination. Finally, Pepper performs localization one more time to get back to its original point. Landmark detection is conducted afterwards to precisely calibrate Pepper return to the exact home position. At that time, Pepper would be ready for another service then. The overall experimental setup is depicted in Fig. 3.

V. RESULTS

A. *Data preprocessing*

Raw data (Fig. 4) is a 25 frames, 5 seconds video, with 320 * 320 pixels per frame in three channels: gray scale, depth, and IR (infrared ray). For every frame, a mask (Fig. 5) is created by its depth data with a selected threshold value, 2000 (range 0 ~ 65535). To smooth the mask and avoid the noise, we apply a binary-closing algorithm, then with binary-opening algorithm to each mask. The mask is roughly the range of human in the video. Three channels (Fig. 6) will then be applied a closing algorithm and filtered by the mask. Finally, a linear normalization is applied to each channel, and it is ready to go into model for training and testing.

Fig. 4. A raw data with 25 frames and 3 channels

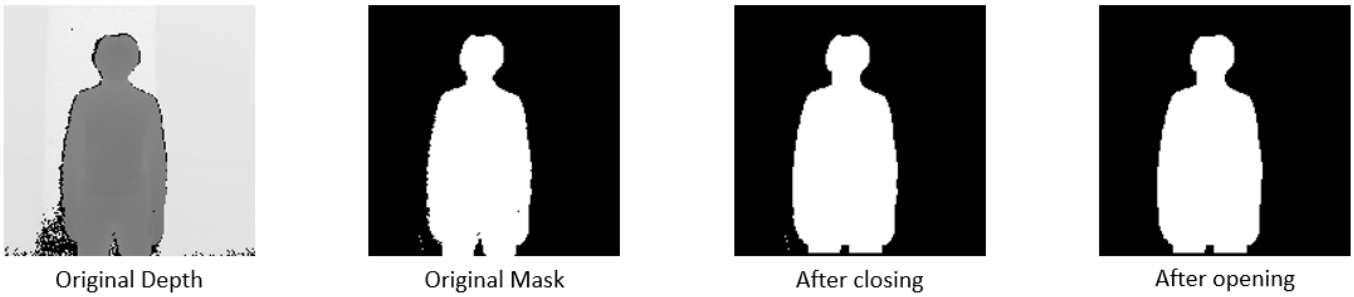

Fig. 5. Mask construction

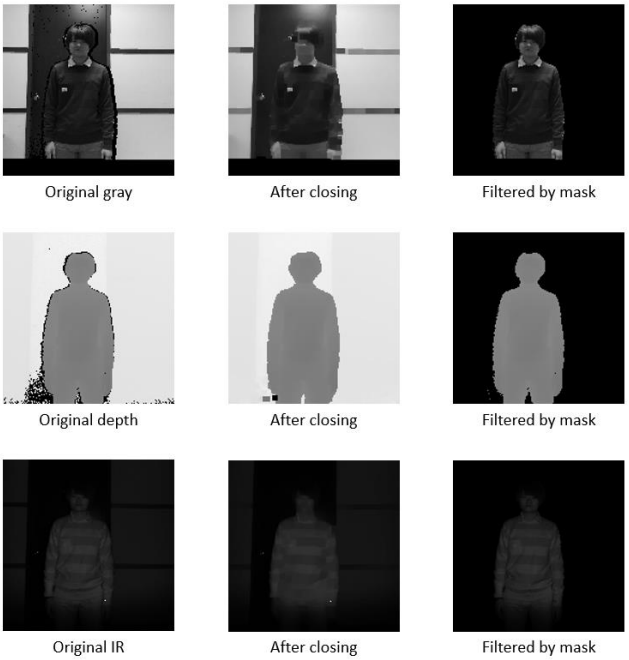

Fig. 6. Data preprocessing of three channels

B. Model with different input channels

The models were trained under different set of data, with all the possible combinations of channels. The one input channel model (Table 2) shows that every single channel is able to provide enough information for recognition. Depth data shows less capability in the model however it is the essential data for data preprocessing. In the end, the lowest loss one is the Gray+IR model with loss 0.0627, and with the probability of correct recognition higher than 99% on validation data.

C. Pepper simultaneous localization and mapping

After several times of exploration, the map is accurate enough for Pepper to arrive at the destination within radius of 50 centimeters. Although it is not 100-percent precise, the error is acceptable since humans can easily find the products they want if they are already at the zone nearby. Moreover, for the returning process of Pepper, landmark calibration is combined to enhance the position accuracy. We thus conclude the SLAM to be successful.

Table 2. The loss of each single input channel

Input channel	Loss
Gray	0.1654
Depth	0.2750
IR	0.1348
Gray+Depth	0.1427
Gray+IR	**0.0627**
Depth+IR	0.2597
Gray+Depth+IR	0.1170

VI. CONCLUSION

Utilizing deep learning, the nine commands consisted of consecutive sign languages have been successfully trained with high recognition accuracy. Pepper SLAM has been carried out with first exploring the surroundings, generating a map, and the localization during navigation afterwards. Furthermore, NAOqi APIs such as face detection, landmark detection, tablet image showing, object recognition, and robot manipulation are all integrated to complete the service. In sum, a service robot aimed for deaf and speech-impaired people is brought out in this project.

For future improvement, more variant data could be collected to train a model even more robust. ROS SLAM can also be implemented to make the navigation more precise. The size of the map can be expanded to satisfy the need in a real supermarket as well. However, although there remains some improvements to be made, the project done is considered a complete service for making deaf and speech-impaired people's life more convenient.

REFERENCES

[1] H. Wang and C. Schmid, "Action Recognition with Improved Trajectories." ICCV, 2013.

[2] LECUN, Y., BOTTOU, L., BENGIO, Y., AND HAFFNER, P. "Gradientbased learning applied to document recognition." Proceedings of the IEEE 86, 11 (1998).

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "Imagenet classification with deep convolutional neural networks." Proc. Advances in Neural Information Processing Systems (NIPS), pages 1097–1105, 2012.

[4] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition." IEEE Trans. Pattern Analysis and Machine Intelligence, 2013.

[5] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. "Learning spatiotemporal features with 3d convolutional networks." ICCV, 2015.

[6] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. "Sequential deep learning for human action recognition." In Proceedings of the Second International Conference on Human Behavior Unterstanding, 2011.

[7] Sepp Hochreiter and Jurgen Schmidhuber. "Long short-term memory." ¨ Neural Computation, 1997.

[8] Karen Simonyan and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos." In Proc. Advances in Neural Information Processing Systems (NIPS), pages 568–576, 2014.

[9] Vittorio Perera, Tiago Pereira, Jonathan Connell, and Manuela Veloso. "Setting Up Pepper For Autonomous Navigation And Personalized Interaction With Users." *CoRR*, abs/1704.04797, 2017.

[10] G.Capi, S.Kaneko, and B.Hua. "Neural Network based Guide Robot Navigation: An Evolutionary Approach." In Procedia Computer Science Volume 76, 2015, Pages 74-79.

[11] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling." In Proceedings of the 2005 IEEE International Conference on Robotics and Automation. IEEE, 2005, pp. 2432–2437.

[12] NAOqi API and Documentation "http://doc.aldebaran.com/2-4/index_dev_guide.html"

WORK DISTRIBUTION

Team member	works
Hsuan-Chu, Lin	SLAM, object recognition, robot manipulation, video editing
Szu-Yu, Mo	SLAM, face detection, landmark detection, tablet image showing, project integration
Wei-Li, Lin	environment setup, data preprocessing, model training, video editing
Yi-Chen, Hsu	environment setup, data preprocessing, model training, presentation