Humanoid Robotic Systems - ”Turning NAO into an English Butler”

Chengjie Yuan Cong Wang Helin Cao Yansong Wu

I. PROJECT DESCRIPTION

The project aims to turn the robot NAO into an English butler. In the task, Nao will look for an object in a predefined workspace and go to the target position to grasp it. The robot should avoid obstacles while moving to the desired position as well. Finally, upon grasping the goal object, the robot should find the target person and hand over the object.

In detail, once the front button is pressed, NAO will first say ”Can I help you” and waits for a further command to make sure which object to find and grasp as well as which person to hand over the object. Upon understanding the command, the robot starts to look for the target object and detect the surrounding. The obstacles and boundaries of the workspace are marked with ArUco markers. Based on these ArUco markers, the robot will plan a routine itself to avoid obstacles while approaching the goal position. Once the desired position is reached, the robot will bend its body to grab the object. Afterward, it will look for the target person by turning around. Upon finding the known person, it will stop and hand over the object. Otherwise, it will stop if the target person is not found after rotating 4 times.

II. TECHNICAL PART

According to the project description above, the whole project is divided into different subtasks i.e. speech recognition, visual detection, localization, navigation, grasping along with face recognition and handover. Which means the robot will execute different subtasks in different situations.

A. Speech Recognition

This is the first task, the robot needs to ask for a command from the people. From the command, the robot should get two important information, one is the person who it should hand over the object to, the other is the object it needs to detect and grasp.

The vocabulary should be set at the very beginning. The vocabulary composes of the verb ”bring”, two names of the people along with one object name. We do not set other object names considering the complexity of the object detection part as different objects hold different characteristics needing to be figured out.

Based on the vocabulary, the recognition is working, we should judge whether the word is accurately recognized or not, the confidence value is used, which is one characteristic of the recognized word. Based on the testing, the threshold of the confidence value is set to 0.45. If the value is lower than the threshold, the word will be defined as ”Unknown”. Along with the accurately recognized words, they are combined into a sentence and will be compared to a certain command which we defined in advance (e.g. ”bring jack cup”). If the combined sentence matches the predefined command, the second string will be compared with the list of names that we set in vocabulary. When the known name is obtained, it will be published to the topic ”/facename”, which will be used in the final face recognition part. On the contrary, if the combined sentence does not match, which means the robot cannot understand the command. The robot will repeat the process and ask for command again until it can understand. When finishing this subtask, it will stop speech recognition and call the next subtask visual detection.

B. Visual Detection

Because NAO does not have a depth camera, the distance between the object and NAO is calculated from the pixel coordinates. In our project, the pixel height of the object is measured as a reference quantity to calculate the distance. The reason why we used the pixel height to calculate rather than the pixel area is that, the error in horizontal pixels is huge when the light changes during the experiment.

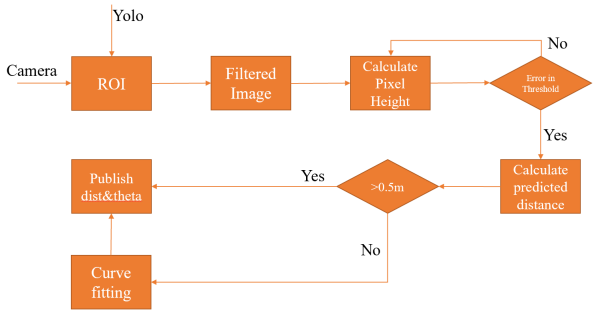

Fig. 1. Flowchart of visual detection

The ROI of the object is obtained through YOLO. And a threshold is set to acquire the filtered image of the object(Figure 2) [1]. Many processes are implemented in order to eliminate the accidental error. Only when the difference of two adjacent observed pixel height is less than three, then this value of pixel height is taken into calculation. After five times of calculation, the average of the five calculated distance is taken as the estimated distance $dist_p$. After getting the estimated distance $dist_p$ according to the robot base-footprint, the polar angle of the object is calculated through the following formula:

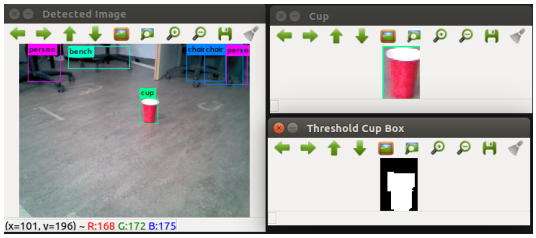

Fig. 2. Detected image

$$theta = \arctan(\frac{object_h \cdot object_c}{dist_p \cdot pixel_h}) \quad (1)$$

where $object_h$ is the height of the object in centimetre, $pixel_h$ is the height of the object in pixel coordinate, and the $object_c$ is the centre of the object in pixel coordinate.

This estimated distance $dist_p$ and the polar angle is sent out after the NAO find the object, and it is continuously being sent out while NAO is walking. Because the view angle has a great influence on the result, two different calculation methods are implemented during the program. When NAO reaches about 0.5m before the object, the view angle will have a great influence on the visual detection and calculation. As a result, when the distance is less than 0.5m, the pixel height is no longer taken into the calculation of distance. A fitting curve is plotted in terms of the pixel coordinates and the cartesian coordinates. Using the fitted curve, we can get the estimated distance from the pixel coordinate, as shown in Figure 3 4.

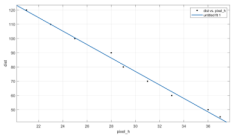

Fig. 3. Fitting curve far from the object

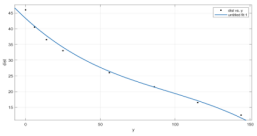

Fig. 4. Fitting curve near the object

C. Approaching and Grasping

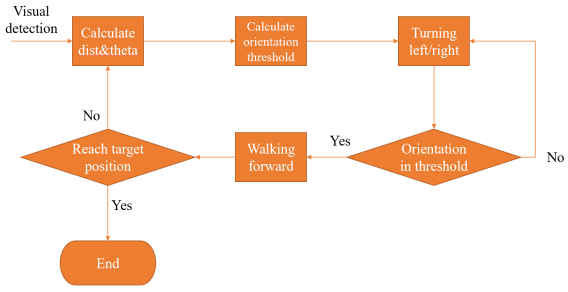

Fig. 5. Flowchart of walking part

When there is no obstacle between NAO and the object, NAO will start the approaching movement. After calculating the distance, NAO will at first adjust the orientation depending on an orientation threshold, as shown in Figure 5. This threshold also depends on the distance. The threshold angle is large when NAO is far from the object, and it is small when NAO is close. After NAO adjusts the orientation, a forward movement will be executed. The distance that NAO moves depends on the estimated distance calculated before. These approaching methods will be executed many times, until NAO reaches the pre-grasp position. In order to eliminate the error while walking, a differential control is also implemented in the control program. The whole flowchart of the detection and approach part is shown in Figure 6

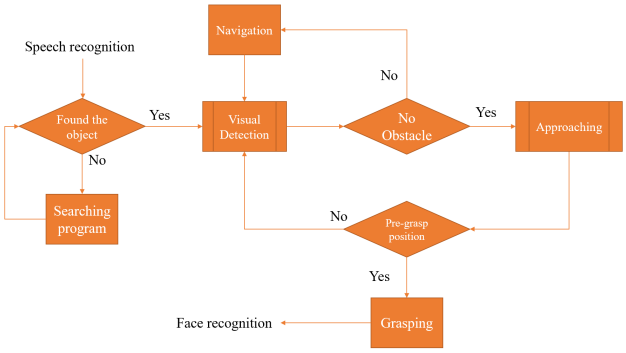

Fig. 6. Flowchart of detection and approaching part

After NAO reaches the pre-grasp position, a service will be called to execute the grasping movement. The grasping movement is pre-set in joint-space control [2], as shown in Figure 7.

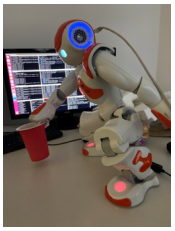

Fig. 7. Picking up the cup

D. Localization

Localization works as an important part of navigation. Because of the bias of actuators during the walk, the NAO needs to do a lot of relocalization during navigation. Besides, we need to know the position of the obstacle to drawing the global map. Some definitions of the global map can be presented as follows and are shown in Figure 8.

The global map [3] is a rectangle with size 1m*1.2m, the x-axis is along the long side, the y-axis is along the short side, θ is the angle between the positive direction of the x-axis of global map and robot frame, the positive direction of θ is anti-clockwise from a top view. The five localization ArUco markers [4] with id = 0,1,2,3,4 are located in (0,0), (0,50), (0,100), (60,0) and (60,100), respectively.

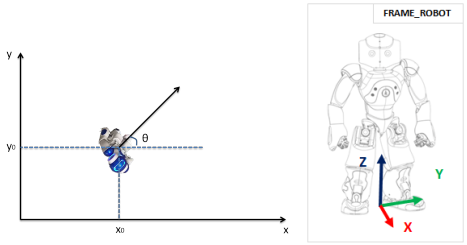

Fig. 8. The defination of global map and robot frame

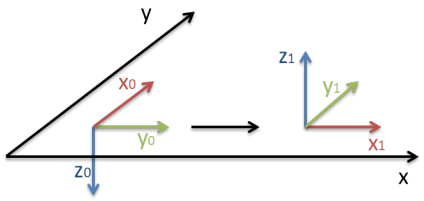

Fig. 9. The first part of rotation

1) Localization of NAO: The localization of NAO is based on frame transformation. Because the location and orientation of the marker are fixed in the global map, if we know the frame transformation from marker to the robot frame, the 2D position of NAO (x,y,θ) can be computed. The total transformation is composed of a series of transformations. The transformation chain is from the marker frame to the bottom optical frame to the bottom camera frame to the robot frame.The localization is based on the knowledge of transformation and inverse transformation of rigid body motion. The transformation matrix and inverse transformation matrix can be represented as

$$\boldsymbol{g} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{T} \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{2}$$

$$\boldsymbol{g}^{-1} = \begin{bmatrix} \boldsymbol{R^T} & \boldsymbol{-R^T T} \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{3}$$

Assume R_0 represents the transformation matrix from the marker frame to the robot frame.

$$\boldsymbol{g_0} = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{4}$$

The x,y and θ can be extracted from this matrix. For simplexity, the translation and rotation are supposed to be analyzed respectively. According to Figure 9, the localization marker frame is represented as (x_0, y_0, z_0), the robot frame is represented as (x_1, y_1, z_1), and the global map is the black line, so the y-axis of the marker corresponds to the x-axis of the map, which means the x of the 2D position is a_{13} and the y of 2D position is a_{03}. To calculate theta of 2D position, the rotation needs to be is decomposed into two-part. As shown in Figure9, the first part is the rotation R_1 from the marker frame to the robot frame, this robot frame means the theta is 0, then we can add a rotation along the z-axis ($\boldsymbol{R_z}$) as the second part.

Assuming the $\boldsymbol{R_0}$ represents the rotation part of $\boldsymbol{g_0}$, we can get the following equation:

$$\boldsymbol{R_1} * \boldsymbol{R_z} = \boldsymbol{R_0} \tag{5}$$

$$\boldsymbol{R_z} = \boldsymbol{R_1^T} * \boldsymbol{R_0} \tag{6}$$

It is known that a rotation along z-axis has following form:

$$\boldsymbol{R_z} = \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Finally, we get cos and sin of theta, the theta can be calculated as follows:

$$\theta = \begin{cases} -\arccos b_{00}, & \sin b_{01} \geq 0 \\ \arccos b_{00}, & \sin b_{01} < 0 \end{cases} \tag{8}$$

2) Localization of Obstacle: After the calculation above, we have got the 2D position of NAO. The localization of obstacle is based on the current 2D position (x_0, y_0, θ_0) of NAO. Firstly, we can calculate the transformation matrix from the robot frame to the marker frame, which is just the inverse of g_0. Then the translation $T = (t_0, t_1, t_2)$ can be extrated from the transformation matrix. Then the position of obstacle can be calculated as follows:

$$\begin{aligned} x &= x_0 + t_0 * \cos\theta_0 - t_1 * \sin\theta_0 \\ y &= y_0 + t_0 * \sin\theta_0 + t_1 * \cos\theta_0 \end{aligned} \tag{9}$$

3) Error analysis: Accurate localization is the premise of navigation. But even if the previous theory is right, the result of localization still has large errors(more than 10 cm), which can not support successful navigation. So, the error needs to be analysed.

- Calibration: Camera Parameter can influence the accuracy of marker detection extremely. So we do the calibration with a chessboard. Calibration is the process of estimating the intrinsic and extrinsic parameters of the camera. But because of the high non-linearity of the camera parameter, the results of calibration in the different experiments are very different from each other, so we performed many calibrations and choose the result which is close to mean.
- Unstable marker frame: The marker detection algorithm has a drawback. The marker frame is not fixed on the marker, which means even if the marker is stationary with respect to the camera, the marker frame may change suddenly, especially for some orientation, it can not be solved by calibration. To deal with this problem, we have to choose a more stable view.
- Measurement error: Because of the limitation of laboratory conditions, we have to establish a space for the experiment. The biggest problem is that it is hard to make sure that the localization markers are coplanar, a small error in orientation will lead to a huge error in

localization. Besides, it is hard to check if the localization is accurate because we can only measure the length manually with a ruler, and the angle can only be measured visually. To modify this error, we have to add some bias in position.

As a result, the error of localization can be limited within 5cm, which is usable.

E. Navigation

To go through the predefined workspace with obstacles and build a reconfigurable map of the robot in its environment, NAO needs to get the position of itself, obstacles and the target. Based on the position information, a trajectory is planned and a reconfigurable map is generated. Firstly, NAO looks around to localize and determine the positions of the obstacles. Then, a trajectory plan is generated and the map is printed on the rviz. NAO moves along the trajectory. After NAO moves several steps each time, it relocates and adjusts the movement to walks along the trajectory until the object is in front of NAO and there is no obstacle between them. Figure 10 shows the whole structure of the navigation part. In the following subsection, it will be explained how the positions information is obtained or how to plan a trajectory and build the map.

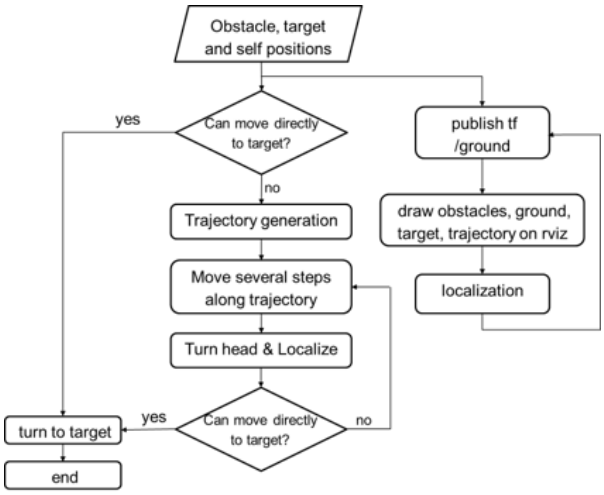

Fig. 10. Flowchart of Navigation

1) Obtain position information: The accuracy of the position information is of vital importance in the navigation part. The following approaches are used to improve the quality of position information.

a. As for obstacle detection, because of the limits on viewing angle of NAO, it is easy to miss obstacles. What is worse, the detected orientation is always very unstable, when the marker locates to some degree. So real-time obstacle detection always causes a terrible result. To avoid the huge potential danger caused be detection mistakes. It doesn't detect the obstacles in real-time, rather detecting all the obstacles at the beginning of navigation in a more stable way. NAO stands in situ and turns his head at a fixed angle and then detection the ArUco markers refer to the obstacle (Detection while turning the head always causes a wrong value). NAO repeats this "Turn head – Detect" process five times to ensure a relatively accurate result.

b. For the localization part, to ensure NAO could find at least one localization marker. "Turn head – Detect" strategy is also very necessary here. In addition, after each time NAO moves several steps, it turns back to keep the robot's orientation the same as the orientation at the beginning of the navigation. In this way, the probability of localization failure is reduced is greatly reduced.

2) Trajectory planning: Figure 11 shows the strategy of trajectory planning. A boundary is set based on the size of NAO and the error of ArUco marker. At first, it detects if NAO can walk directly towards the target. If it can walk directly, then it turns to the target and the navigation ends. If not, it will merge the obstacles together and generate a trajectory (the blue dash line in the picture). To avoid detours, after each time NAO walks along the trajectory several steps, it will detect whether NAO can walk directly to the target. If there is no obstacle between NAO and target, the navigation plan is changed from the blue line to the green line. After that, NAO turns to the target and the navigation is finished. If not, it walks still along the planned trajectory until it reaches the last waypoint. Then it goes directly to the target and picks it up and the navigation is finished.

In the navigation part, we did not use the navigation stack[1]. Because the costmap_2d is based on the sonar/laser sensor. But NAO has neither of them. If we use the information from markers. We need to rewrite the whole package of costmap_2d. What's worse, many other packages for navigation also depend on the costmap_2d or uses some classes defined in the package costmap_2d. The map_server needs the result of costmap_2d to generate the map. The global and local costmap_2d are the basis of the planner. The compatibility with other packages of navigation stack also should be taken into consideration. It is a huge task.

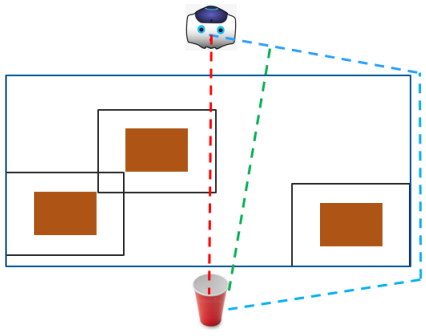

Fig. 11. Navigation strategy

[1]https://github.com/ros-planning/navigation

3) Map building: After NAO gets the position of obstacles and its position, a transformation frame named "/ground" is broadcasted. The ground, obstacles, and target are generated through marker in rviz with respect to this frame. In order for the map can be generated in real-time, the localization information is got all the time. Unfortunately, it often causes a localization mistake. In Figure 12 you can find the map. The green cubes refer to the obstacle, the pink point means the target and the red points are the waypoints of the planned trajectory.

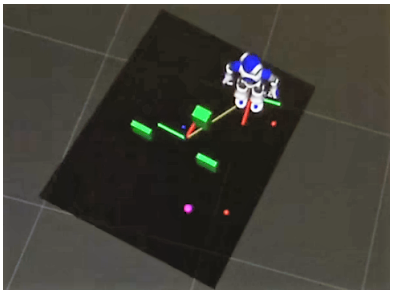

Fig. 12. Map

F. Face Recognition and Handover

This is the final subtask, which aims to find the target person and hand over the object. This part consists of two different modules, one is a subscriber, which is used to subscribe to the name of the target person. The other is a service, which will be called after finishing grasping the object.

The subscriber will always be running once the project started because it needs the information from the first talking task. When the service is called, the face recognition starts as well as movement. The robot will not stop rotating until it finds the target person or it has already rotated 4 times. This face recognition is based on online dedicated libraries[2]. It is built using dlib's[3] state-of-the-art face recognition built with deep learning [5]. As a result, the two libraries should be installed in advance. To do the face recognition, the images of the known people with names should be first stored in the test folder as a training set, the known faces will be encoded in advance. Once the recognition starts, the robot will detect if there is a person in its view, if so, the detected face will be encoded and compared with the known encoding. Meanwhile, the faces in the view will be located and bounded by a red rectangle with the labels below("Unknown" is a label if not match, the other labels are corresponding names). The result can be seen in Figure 13. The detected name will be compared with the name obtained by the subscriber as well. In this case, the robot could know who it needs to find. Figure 14 shows the whole process. After finding the target person, NAO will lift its arm to do the handover task.

[2]https://github.com/ageitgey/face_recognition
[3]http://dlib.net/

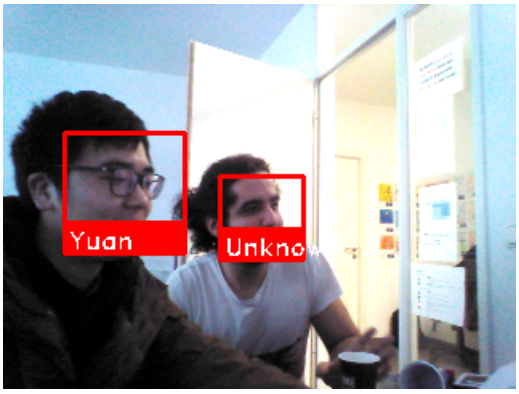

Fig. 13. Result of face recognition

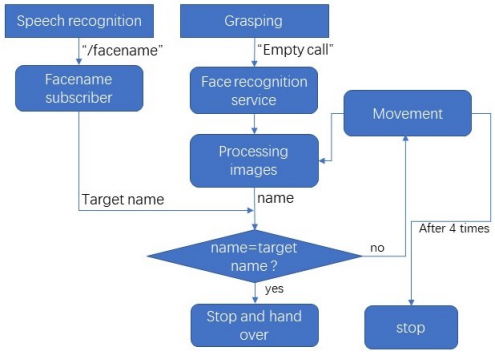

Fig. 14. Flowchart of Face recognition

issues and solutions: One issue for this part is the processing speed for each frame. When this programme is running, we cannot get a good real-time face recognition since the frame in the video will be stuck once detecting faces. As a result, the video is not fluent. Considering the previous method, each frame will be processed, it costs a lot of time. To figure out this issue, we do processing every 5 frames. In this case, the performance is better. Another issue is when we want the robot to move, the default configuration of its hand is open, it will drop the grasped object when moving. Hence, a new walking pose is set to make sure that the robot firmly holds its hand while moving.

Drawbacks and Future Work

The visual detection method lacks the universality because it only fits for specific objects and needs pre-set parameters considering the effect of light. Also, the grasping method lacks feedback while grasping, and the joint-state control method is not robust sometimes. Therefore, we could find approaches to decreases the effects of light, in those cases, our method could adapt to more general cases. Additionally, visual feedback could be taken into account for better performance. For the navigation part, it could use a classic algorithm to improve navigation performance. The number of markers can be increased to improve the accuracy of localization. Finally,

one improvement to the project is the final handover part. In our case, the robot directly executes the handover task when detecting a known person without knowing the accurate position of the person. Hence, depending on the size of faces in the image, we could try to calculate the depth between NAO and the person. A controller could be generated base on it to make NAO walk to an appropriate position to do the handover task.

REFERENCES

[1] Dawson-Howe K. A practical introduction to computer vision with opencv[M]. John Wiley & Sons, 2014.

[2] Gouaillier D, Hugel V, Blazevic P, et al. Mechatronic design of NAO humanoid[C]//2009 IEEE International Conference on Robotics and Automation. IEEE, 2009: 769-774.

[3] George L, Mazel A. Humanoid robot indoor navigation based on 2D bar codes: Application to the NAO robot[C]//2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids). IEEE, 2013: 329-335.

[4] Salinas R M. ArUco: An efficient library for detection of planar markers and camera pose estimation[J]. 2019.

[5] Geitgey A. Machine learning is fun! part 4: modern face recognition with deep learning[J]. Medium. Medium Corporation, 2016, 24.