Marco? Polo!

*Play hide and seek with a NAO-Robot

Yu Mu
yu.mu@tum.de

Soeren Pohl
soeren.pohl@tum.de

Max Schuran
max.schuran@tum.de

Zhouyi Gu
zhouyi.gu@tum.de

I. Introduction

The idea of this project is to try to make people feel less lonely in quarantine by playing hide and seek with a NAO. Therefore the NAO must be capable of searching and finding an object and additionally be able to hide himself.

For this project a switch-Mode Architecture with an reactive system is usable, because the NAO should react on the different environmental influences. The detailed system architecture can be found in the *Appendix* at the end of the document. For the different operations, either searching or hiding, the NAO runs different loops and checks the environment by using his perception modules for information which could lead to a successful fulfilment of his task. For the perception of the room the top camera and the microphones have been chosen. The bottom camera is not exactly suited for this project because the information that can be generated by it's respective image is not sufficient enough as it is not showing the whole space in front of the robot. The microphones are matching the needed criteria because it is possible to perceive where a sound is coming from, since they are located towards the front and the back of the robot.

II. System Modules

A. Vision

This subsection contains two vision tasks. The first is utilization of the face recognition through top camera of the NAO. The second is to find the local dark spot in the room using NAO's top camera and comparing histograms of two photos in order to locate the dark spot in a three dimensional space.

1) face recognition: For the seeking task the NAO searches the human. The NAO should say their name to notify the person that they have been found. The idea is using neural network namely darknet(YOLO) for face recognition. Since darknet should be trained for custom object detection, it is necessary to build a custom face dataset. Around 500 face images were generated and manually labeled as two face classes. To accelerate the training process and get higher FPS at face recognition, CUDA and OpenCV dependencies were added into darknet package. After hyperparameter configuration darknet is able to train custom dataset and generates weights in each specific training step. In consideration of the small training set it is necessary to validate the performance of different weights to overcome underfitting and overfitting. With the help of darknet_ros package [1] with proper configuration file and weights from darknet package, the darknet can be executed with roslaunch, then the NAO's top camera topic can be subscribed and detected face can be drawn with corresponding label and bounding box. Meanwhile the darknet_ros package publishes bounding_box topic, which can be subscribed to get the BoundingBoxes message with information about detected face such as class, probability etc. In practical test NAO detects the person correctly and the probability is around 60 percent. After comparison with class from BoundingBoxes message and default classes values NAO should be able to distinguish between participants and say their names with *ALTextToSpeech* from naoqi.

2) Dark spot detection: For the hiding process, the robot needs to find the darkest spot in the room and walk towards it. After walking one meter, it should stop and compare the current view with the previous view to confirm that the old dark spot is still the darkest one, otherwise it will move to the new spot in the room. While the robot moves to the global dark spot inside the room, it will stop and perform the squatting and hiding motion, which will be decoded in the subsection Motion. To find the dark spot in a photo, the pixel in the photo with smallest RGB value and its coordinate in the photo frame need to be figured out. For eliminating the effect of noise the photo should be pre-processed by Gaussian blur, with which the noise, that is the pixel with very high values, will be averaged by the neighbouring pixels. Also it is preferable, that the top one third of the captured photo should be removed before the processing in order to confirm that the found dark spot is reachable. Otherwise, it is possible to find a dark spot which hangs on the wall.

After getting the coordinate of the dark spot in the photo frame, the walking direction towards the dark spot should be computed through the value of the spot on the x-axis of the dark spot. Unlike the two dimensional Cartesian coordinate, the origin of the OpenCV coordinate locates at the top left corner of the photo. The x-axis along the width of the photo goes from the origin to the right and the y-axis along the height of the photo from top to bottom. The projected point of the camera is at the center of the photo. The coordinate is denoted as (x_{cam},y_{cam}). x_{cam} is equal to the half width of the photo and y_{cam} half height.

The figure 1 displays an example of the top view of the spatial relationship between the NAO and the photographed view. The origin indicates the origin in the photo coordinate. The $y_{darkspot}$ denotes the position of the spot on the y-axis. Suppose the distance between the robot and plane is denoted

as d. The angle α between two straight line from the camera can be calculated by equation 1. In the following operation, the angle will be used for turning the robot directly to the dark spot.

$$\alpha = \arctan(\frac{y_{darkspot} - y_{cam}}{d}) \quad (1)$$

In order to determine if the NAO moves to the hiding place already, the similarity of two images need to be compared. In this project, the Pearson product-moment correlation coefficient[4] is utilized for making this decision. The two images need to be converted to histograms initially. If the correlation coefficient is equal to 1.0, the two compared histograms can be considered as equal, and their two original photos are also seen as equal. Obviously the correlation coefficient need to be set smaller than 1.0, otherwise the NAO will hit the wall directly.

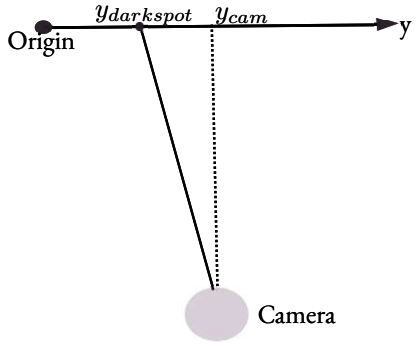

Fig. 1. Top view of the ground plane

B. Auditory

In this subsection it will be presented how the use of the Auditory and Speech engine which was integrated in the NAO to create a human-robot interface and to locate a player in the NAOs searching process.

Firstly it was necessary to establish a way to let the NAO follow a players command such that the NAO can determine whether it is the 'searcher' or the 'hider'. To accomplish this a dialogue was created after powering up the NAO. To implement a dialogue the *ALSpeechRecognition* and *ALTextToSpeech* Proxies from Softbank Robotics NAOqi API [2] were used. To avoid complexity the NAO only reacts to words instead of sentences such that it asks for some keywords and if the player responds with a predefined keyword the NAO would continue with an action. In that way the tasks 'hiding' and 'seeking' are two seperate outcomes of a certain question at one point of the initial dialogue. As one subscribes to the *ALSpeechRecognition* Proxy, data will be written into the *ALMemory* Proxy under the keyword 'WordRecognized' whenever a predefined word is recognized. The data contains the recognized word and the confidence of this word. It can be easily accessed by *ALMemory*'s method 'getData('keyword')'.

The Auditory engine of the NAO was furthermore used to locate a player when he hides himself. Therefore the *ALSound-Localization* proxy constitutes the method for calculating and obtaining the source of a sound. After the NAO would call 'MARCO' it expects to hear 'POLO' from the hiding player. Only if the NAO recognizes the keyword 'POLO' with the *ALSpeechRecognition* proxy, the data from *ALSoundLocalization* is read out which includes an angle, pointing into the direction where the sound-source is estimated with respect to the direction NAOs head is facing. The calculations are done by the API itself. The calculation uses the two or four (depending of the version of the NAO) microphones on NAOs head and compares the times a sound is recognized by each microphone. This method works best when only one sound source is present in the environment of the NAO but a sensitivity for the recognition can be set. During testing a medium sensitivity got the best results. A high sensitivity will yield to too much disturbance from even silent noises such as footsteps. On the other hand a low sensitivity forces the player to almost shout 'POLO' which then leads to too much disturbance. Furthermore it was crucial to start subscribing to *ALSoundLocalization* after the NAO spoke, otherwise it would recognize itself. This leads to another obstacle that the rather small size of the room presented. Soundwaves would reflect from the walls and in some cases lead to a computed direction facing in the opposite direction of the original sound source. To account for this challenge all angles pointing to NAOs back were ignored. To obtain the angle computed by *ALSoundLocalization* the variable in *ALMemory* with the keyword 'ALSoundLocalization/SoundLocated' has to be read similar to the sound recognition, this variable will be updated only if one subscribes to the *ALSoundLocalization* proxy.

C. Motion

This section elaborates the overall motion tasks of the NAO robot. This includes walking towards the hiding spot, in the sound direction and squat down to hide.

To walk in a specific direction for the movement-tasks, the "walk to hideout" and "walk to sound" and also for the squatting process the *ALMotion* package has been used. For the movement the NAO detects a the direction of the sound or the dark spot and should walk to it with the *ALMotionProxy::moveTo()*-command (see fig. 2). Therefore the NAO turns about the angle $theta$ around the z-axis which is pointing out of the plane until the x-axis points in the desired direction. During testing it showed that the turning motion had to be done in a spiral motion because as the NAO turned on a spot it would loose its balance.

After turning to the desired direction the NAO walks a predefined distance along the x-axis. The walking distance must be chosen according to the environmental conditions i.e. due to the small size of the lab room, 0.5m has been chosen.

The NAO hides himself by squatting down and closing his eyes with his hands. Therefore his feet are constrained to stay on the ground during the whole motion. To keep the Center of Mass between the two feet, to not let the NAO fall over, the NAO will bent a little forward during the down squatting. To have a smooth motion the *angleInterpolation*-method is used and additionally a long time span to avoid unnecessary inertial forces.

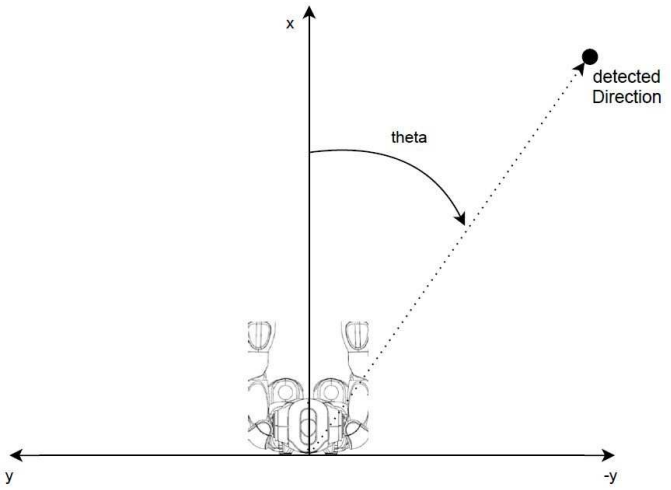

Fig. 2. NAO Robot_Frame walking direction

D. Navigation

In this subsection, the mechanism of localization and path planner will be presented. While the robot is moving inside the room, it needs to avoid the obstacles and localize itself inside the room. After it finishes the hiding and seeking, it should walk out of the room. For the navigation, A* path planner[6] combined with ArUco markers[3], [5] is implemented.

1) Localization: In this project, the ArUco markers with ID from 1 to 4 are used for localization and the markers with ID 5 and 6 are obstacles. The spatial relationship between the markers and the NAO is visualized by the package *tf* and *visualization_msgs/Marker Message* in *rviz*. The ArUco markers can give the NAO its own coordinate in the NAO's camera frame. For estimating the pose of markers on the map it is necessary to transform the markers' coordinate from its camera frame to robot frame, where the homogeneous transform from the camera frame to the robot frame can be obtained by the Cartesian control API *ALMotionProxy::getTransform*. Therefore, the markers are utilized technically only during navigating out of the room.

2) Path planner: The A* search algorithm as presented in the flowchart3 is used for finding the most efficient path on the unknown map from start to end. n denotes the node on the map, some of the nodes are unreachable due to the obstacles. $g(n)$ denotes the cost from specific node to the start, the cost can be described as travelled distance in this case. $h(n)$ denotes the heuristic function, which states the predicted required cost from the specific node to the end node. The evaluation function $f(n)$ is computed by $f(n) = g(n) + h(n)$. While moving to the next node, the node with smallest $f(n)$ should be chosen. The search requires two sets for storing information about the node, O denotes the open set and saves the nodes for further expansion. C is the closed set, which stores the explored nodes. $c(n_1, n_2)$ is the length of an edge connecting between node n_1 and n_2. $b(n_1) = n_2$ states, that node n_2 is assigned as the parent node of n_1.

In this project the map with the size of $3meter \times 3meter$ is discretized by 30×30 nodes. The width of each node is approximately equal to the size of the feet of the robot. As planned, the origin of the map lies at the artificial hide corner. Four ArUco markers, which function as landmarks in the map, are placed at the four sides of the map. The door, which is the end node on the map, has fixed coordinates on the map. The obstacles can be casually placed inside the map, their coordinates will be detected by the camera and published after the transformation.

Because the Locomotion control API *ALMotion-Proxy::moveTo* is used for walking in this project, the angle and destination as input need to be updated after each step during the navigation.

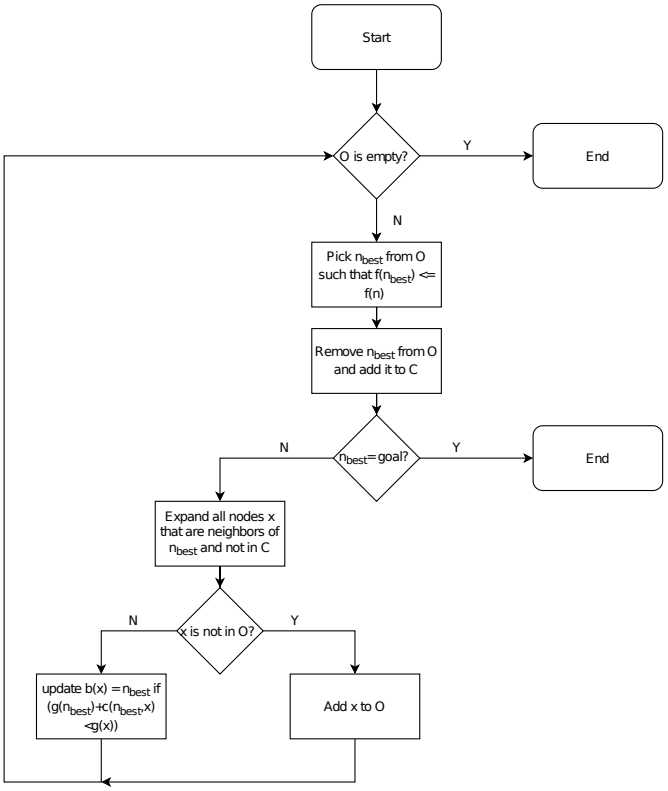

Fig. 3. A* star path planner

III. Problems

During the project we encountered several problems of different kind. Due to the corona-pandemic and the lack of experience in programming, it was not always possible or easy to test the code we produced. Nevertheless with webots we had a useful tool to test some code digitally. In the lab however we ran into several technical difficulties with the NAO robots. Since we tested everything in webots and it was working we thought it may be something wrong within the codes and how we run them on the physical working machine. It turned out that the NAOs themselves had some problems. The discovering process took a lot of time and we needed to change everything back to how it was. Regarding the different modes of the NAO

there have also been some challenges. The average training loss of darknet barely decreased and darknet marked an arbitrary face with wrong label in condition of weights generated after hundreds of training epochs. The reason could be a small and illficonditioned dataset so that darknet learned some strange features. The communication between different node handles could be very hard to implement, especially when some packages have conflict with each other. e.g. to actualize walking module, it was originally designed by using ROS actionlib. But due to the unknown conflict between OpenCV and actionlib, the walking was finally designed as a server. Regarding the SoundLocalization the echo from the wall and background noises were causing a wrongly identified sound direction. By adjusting the sensitivity of the NAO's microphones and speak in an adequate volume (not too loud, not too quite) the results turned out to be satisfying.

IV. Conclusion and outlook

A. Workload

Yu	Zhouyi
dark spot detection coordinate transformation moving to dark spot A star path planner localization visualization	facial recog.
Soeren	**Max**
hiding process coordinate transformation project proposer main documenter	auditory moving to sound

B. Conclusion

The project was a good opportunity to get to know how ROS and humanoid robots work. It showed how a robotic system should look like, the limits and the advantages/disadvantages of a NAO robot. All in all it was a great experience since we were allowed to come up with our own ideas for a project and problem solution. We are pretty thankful for the instituted trusted us in working with the robots independently and without strict supervision.

C. Outlook

Taking this project as a basis it is possible to extend it to a certain level. It could be possible to make two NAO robots play hide and seek with them self by adjusting the recognition algorithm. Additionally it could be possible to teach the NAO different hiding tactics i.e. moving to a different hiding spot after saying "Polo!" or using obstacles to hide behind. For the seeking part an occupancy grid map could be implemented which is searching on a map for the sound origin and tries to figure out the movement of the opponent by analysing the intensity and/or movement of the signal.

References

[1] M. Bjelonic. Yolo ros: Real-time object detection for ros. https://github.com/leggedrobotics/darknet_ros.

[2] S. R. Europe. Naoqi developer guide. http://doc.aldebaran.com/2-5/naoqi/index.html. Last accessed: 2021-02-23.

[3] S. Garrido-Jurado, R. Muoz-Salinas, F. Madrid-Cuevas, and R. Medina-Carnicer. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, 51:481–491, 2016.

[4] K. Pearson. Mathematical contributions to the theory of evolution.—iii. regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 187:253–318, 1896.

[5] F. J. Romero-Ramirez, R. Muoz-Salinas, and R. Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and Vision Computing*, 76:38–47, 2018.

[6] W. Zeng and R. L. Church. Finding shortest paths on real road networks: the case for a*. *International Journal of Geographical Information Science*, 23(4):531–543, 2009.

Appendix

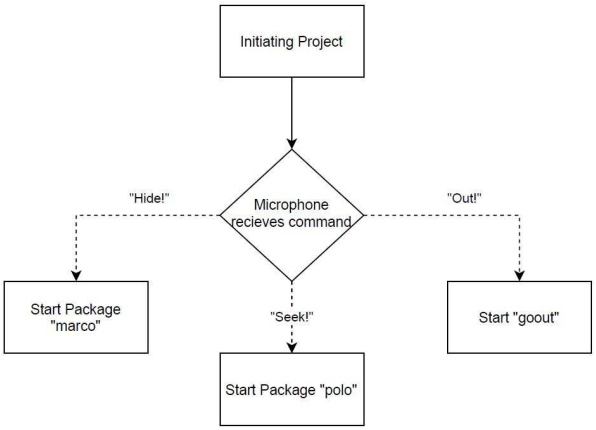

Fig. 4. Architecture Overview

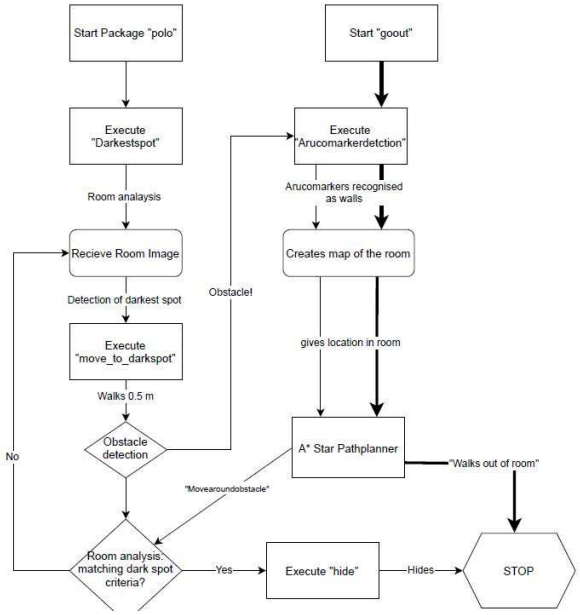

Fig. 5. Hide and Out

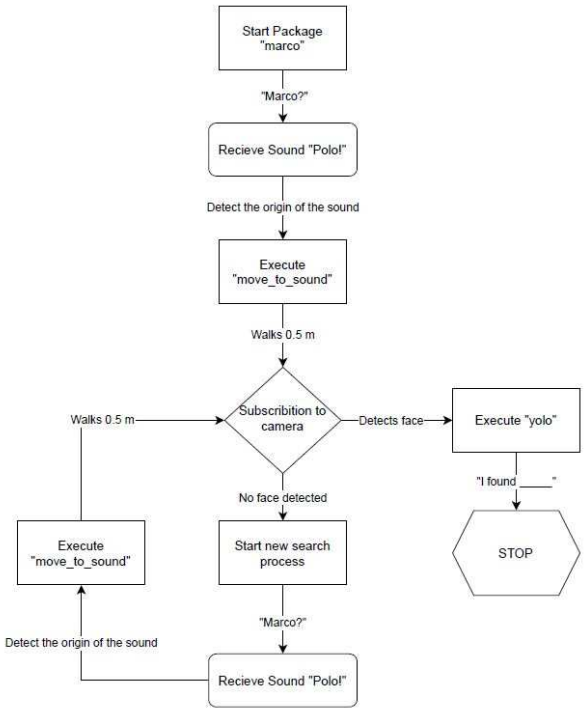

Fig. 6. Seek