Experience from the operation of the Pepper humanoid robots

Arkadiusz Gardecki
Institute of Drive Systems and Robotics
Opole University of Technology
Opole, Poland
a.gardecki@po.opole.pl

Michal Podpora
Institute of Computer Science
Opole University of Technology
Opole, Poland
m.podpora@po.opole.pl

***Abstract*—This paper summarizes authors' experience with the operation of both versions of autonomous humanoid robot Pepper. The robot's construction, as well as its capabilities and limitations are discussed and compared to the NAO robot. Practical background of working with Pepper robots and several years of experience with NAO, result in specific know-how, which the authors would like to share in this article. It reviews not only the robots' technical aspects, but also practical use-cases that the robot has proven to be perfect for (or not).**

Keywords—robotics; humanoid robots; NAO; Pepper; HMI

I. Introduction

Nearly six-year experience of operating NAO humanoid robots and one year of working with the Pepper humanoid robots has motivated the authors to summarize the practical aspects of working with Pepper (Fig. 1).

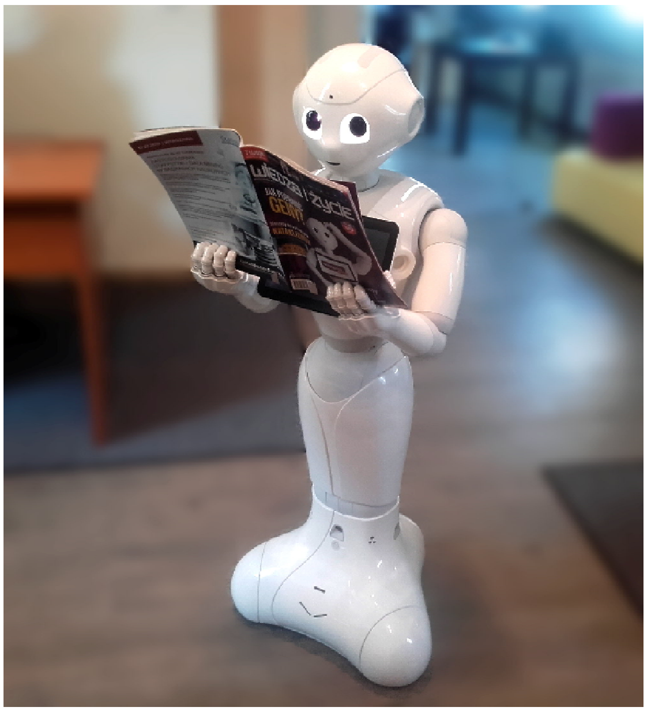


Fig. 1. Pepper humanoid robot, version Y20 V10

Pepper is the successor of the NAO robot - undoubtedly one of the most popular humanoids designed for interacting with human [1]. The popularity of NAO has mainly been caused by its usability and functionality. Although robotics education can start with e.g. LEGO Mindstorms or basic programmable toys, this approach does not have much real-world application in it [2]. On the other hand, there are industry-level platforms, which might be too complex for the beginning, or completely inadequate if the robot is expected to be used explicitly for interaction with human (HMI). Although NAO fills this gap perfectly, Pepper's functionality brings the human-robot interaction to a new level. Of course Pepper does have limitations, which will be also indicated in this paper.

Pepper is an autonomous humanoid robot designed by Aldebaran Robotics, and released in 2015 by SoftBank Robotics (SoftBank acquired Aldebaran Robotics in 2015). In 2015 Pepper was available only in Japan [3], in June 2016 also in Europe [4], and in November 2016 in the US [5]. Main target market as well as the most natural use case for Pepper is interaction with human. Its height and physical proportions and dimensions make it appear more similar to a human being than NAO (yet not crossing the dreaded Uncanny Valley [6]) resulting in better HMI User Experience [7]. Although the Pepper robot is a relatively new construction, it already proved to be successful in various areas of application. Its main role is to engage people [8] - primarily in business: interactively provide information on a company's offer, to greet and amuse customers, or to influence the prominence of a company. Pepper seems to be one of the best options for implementing and research on HMI, due to the sensors, technologies and functionalities included in its design and preprogrammed in a form of an API (application programming interface). A significant advantage of this design is its full programmability. The manufacturer provides frequently updated documentation and dedicated development environment Choregraphe [9].

II. Specifications and construction of the Pepper robot

Pepper robots have been offered generally in one version, but its last modification (ver. 2.5.5) includes major changes to the API and the software as well as an upgraded tablet. Thus, old programs (implemented for API older than 2.5.5) may not function properly. Fig. 1 presents the general appearance of the robot, Fig. 2 indicates the (little) difference between Y20V10 and Y20V16 versions of the robot, where Y20 and V16 stand for: the number of degrees of freedom - 20, and the actual version of the robot - v. 1.6, respectively.

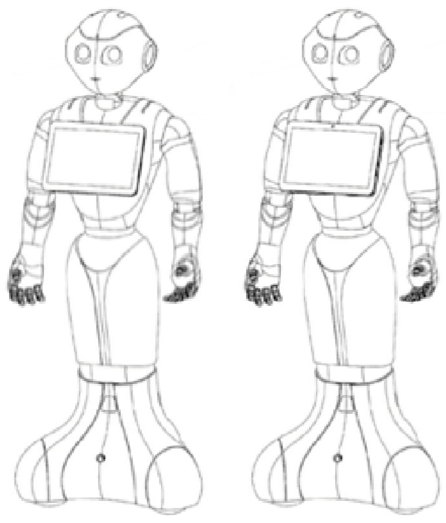


Fig. 2. Versions of the Pepper robot, version Y20V10 (left) [9] and Y20V16 – visually identical, except a thicker tablet with a camera (right)

The robot is controlled by a dedicated, linux-based operating system NAOqi. It contains several modules that comprise the library [9], enabling the developer to control the robot's resources. The NAOqi system of the Pepper robot is generally identical to the NAO robot's NAOqi, so switching from working with NAO to working with Pepper is fairly straightforward.

The table below summarizes important parameters of the V5 versions of NAO robot and Pepper.

TABLE I. SPECIFICATION OF NAO ROBOT V5 [9] COMPARED TO PEPPER Y20V16 [9]

NAO V5	**Pepper Y20V16**
Construction height: 574mm	Construction height: 1210mm
Construction weight: 4.8kg	Construction weight: 27.82kg
ATOM Z530, 1.6 GHz CPU, 1 GB RAM, 2 GB Flash, 8GB Micro SDHC	ATOM E3845, 1.9 GHz quad core, 1 GB RAM, 2 GB Flash, 8GB Micro SDHC
battery: Li-Ion 2.25 Ah	battery: Li-Ion 30.0 Ah
WiFi 802.11 a/b/g/n	
-	tablet: 1GHz, 1GB, FullHD, 10''
2 speakers and 4 microphones	
2 YUV cameras up to 1280p@30fps	2 YUV cameras up to 640p@30fps or 2560p@1fps and one 3D sensor: Asus XTION working at 320p@20fps
8 pressure sensors (4 per leg)	-
3-axis gyroscope, 3-axis accelerometer	
-	6 laser 2D sensors
-	2 Infra-Red sensors
Ultrasonic sensors (2 transmitters, 2 receivers)	
36 Magnetic Rotary Encoders	30 Magnetic Rotary Encoders

Mobile joints are powered by electric servo drives with high-speed DC motors (Brush DC Coreless, and Brushless DC), the power and torque of which are matched to the characteristics of a joint. Joints are equipped with a system of 12-bit magnetic rotary encoders using the Hall effect, informing about the current position of the robot.

It is important to keep in mind, that maintaining some specific manipulator positions (e.g. the robot's hand pulled out to the front) after few minutes significantly increases the joint's temperature. Overheating of the robot is monitored and signalized by the yellow LED bars flashing on the robot's shoulders, until the emergency shutdown. Therefore, it is very important to choose movements and duration of poses so that the joints would not overheat. This can be achieved simply by enabling the *Autonomous Life* mode, which constantly modifies the robot's pose so that it would look more natural. In this mode the robot simulates breathing movements (micromoves of arms), it is looking for a contact with a human (by moving head and whole body, reacting to sounds and voices, maintaining eye contact and following people), and it moves its joints to a temperature safe position. The *Autonomous Life* mode has a number of states, which refer to the level of "autonomy".

Housing of the robot is made of durable plastic (ABS-PC), which makes it resistant to scratches and cracks. The ground on which the robot moves should be smooth, the robot should be used indoor only, inside a restricted region (due to laser sensors module, which requires a finite distance to a barrier).

Fully charged battery lasts for about 8 hours of operation depending on the intensity of tasks, although the manufacturer claims that the "*high capacity lithium-ion battery gives him approximately 12 hours of autonomy*" [10]. The robot also can be charged during operation, however the robot has its wheels disabled while charging, so it is not able to spin/turn or drive.

III. PROGRAMMING AND CONTROL OF THE ROBOT

Dedicated software, for programming and using basic functionalities of Pepper, consists of the multiplatform (Windows. Linux, MacOS) Choregraphe application (Fig. 3) and dedicated SDKs (Python, C++, Java, Android).

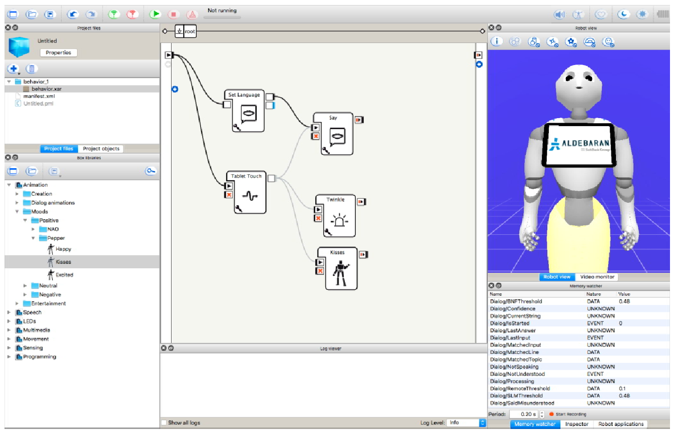
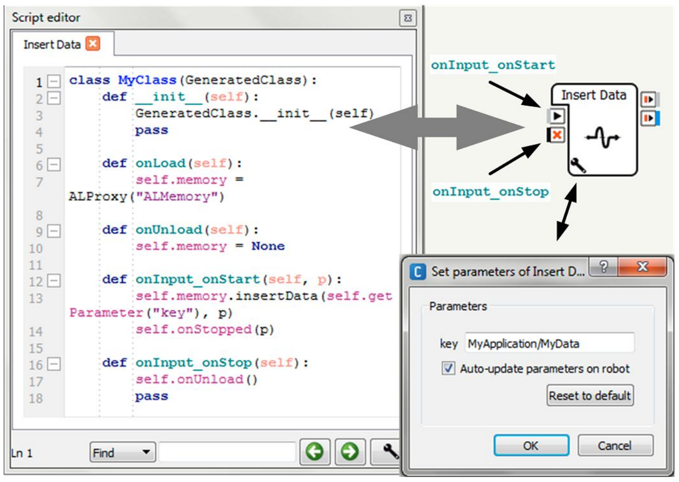


Fig. 3. View of the main window of the Choregraphe 2.5.5.5 environment. Left - object libraries; middle - construction area for designing applications; right - simulator view or the current robot position

The Choregraphe environment is supposed to be used for constructing an application by using some of the already existing blocks/objects, grouped into thematic libraries. It is also possible to modify functionality of a block/object by altering its Python code. These blocks/objects can be easily and intuitively used for building custom applications with varying degrees of complexity. The blocks/objects have a hidden layer, implemented in Python (fig. 4). At this level it is possible to have access to modules and variables representing states and signals from sensors and built-in algorithms controlling a robot.


```
class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self)
        pass

    def onLoad(self):
        self.memory =
ALProxy("ALMemory")

    def onUnload(self):
        self.memory = None

    def onInput_onStart(self, p):
        self.memory.insertData(self.get
Parameter("key"), p)
        self.onStopped(p)

    def onInput_onStop(self):
        self.onUnload()
        pass
```

Set parameters of Insert D...

Parameters

key: MyApplication/MyData

[✓] Auto-update parameters on robot

Fig. 4. The object *Insert Data* and its code in Python

The Choregraphe environment offers a variety of possibilities of adapting it to the programmer's needs, by inter alia control of code execution (by viewing the progress of application and the "bang signals" in the main window indicated by a moving dot or by using a logger), by control variables and events, by learning the robot (e.g. via face recognition), by design of movement sequences ("timeline") and many others.

While observing the development of this software a significant progress in improving its reliability and functionality can be seen. Unfortunately, newer versions are not error-free. Known bugs are reported in documentation [9]. However, there are commands that do not work properly.

The Monitor application offers the possibility to preview and record the most important parameters of the robot, e.g. changes in load of individual joints during movement. It works only during the connection with a real robot, not with a simulator.

Many years of the authors' experience in programming the NAO and Pepper robots (common NAOqi environment) indicates that implementing a stable application for a robot can and should be performed by using the dedicated SDKs (Python, C++, Java). The Python SDK is well documented by many examples, and the process of compilation and communication with the robot is very simple. A user has full access to all build-in modules and can expand robot's functionality with new, practically unlimited resources.

Fig. 5 shows a communication scheme and exemplary methods in modules.

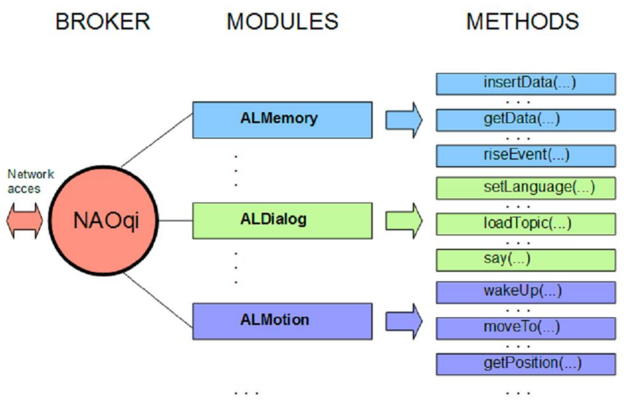


Fig. 5. The broker loads a preferences stored in file called *autoload.ini* that defines which libraries it will load. Each library contains one or more modules that use the broker to advertise their methods [9]

The broker loads a preferences stored in file called *autoload.ini* that defines which libraries it will load. Each library contains one or more modules that use the broker to advertise their methods [9].

To conclude this chapter, it is important to say that the development environment of the NAO/Pepper robot is a coherent and well-thought concept of the Aldebaran/SoftBank company. It offers the possibility to read signals from sensors, process these information and control servo drives. Such a programmable full access to the design is also associated with responsibility. We can damage the robot through improper steering. Manufacturers have also taken care of this, many critical operations are additionally protected programmatically, however these safeguards can be removed.

IV. Interaction with Human

Communication between human and the Pepper robot may take place in many ways, e.g. by using touch sensors (located on its head, on arms or palms), by using 2D or 3D image analysis from cameras, by interacting with touch tablet or by using speech recognition.

Compared to the NAO, the main difference is the presence of a touch tablet over the chest of Pepper (Fig. 2). It allows Pepper to display any visual or interactive content (images, videos, web pages) and thus to enhance the channels of human interaction. For voice-based applications, the tablet can be used to simultaneously address the issues of the conversation/ /interaction in order to enhance the message and overall quality of user experience. It is also useful when communicating with some impaired people or when voice communication fails for other reasons. Tablet can supplement or even replace verbal communication with the robot. This is particularly important in environments in which noise or sounds may disturb the operation of speech identification module.

An interesting mechanism, which can also be interpreted as touch sensor, is the use of signals from actuators in joints. Each change in the position of joint is registered, and thus it can be used for sensing/detecting e.g. change in the position of hand joints caused by the movement during handshaking with the robot. Such use of these sensors allows for implementation of interesting and useful algorithms/behaviors for interaction between human and robot. Some parameters, like the degree of stiffness of joints can be adjusted, and in combination with information from encoders and actuators it is possible to control the movement of a specific joint (or the whole robot) by interpreting the position or readings from a joint, e.g. algorithm for leading the robot by the hand.

The robot has two 2D switchable cameras and one 3D sensor (see Table 1). Image from a camera (only one camera can be selected at a time) can be used as a starting point for author's image processing algorithm or for use in any block or built-in module. The *NAOqi Vision* and *NAOqi People Perception* APIs contain a variety of modules, which offer many interesting and potentially useful vision recognition algorithms.

In algorithms, the purpose of which is to interact with humans, it is very important to recognize human presence and then to maintain visual contact. In that context, *ALFaceDetection* and *ALPeoplePerception* are very useful modules. *ALFaceDetection* is a module in which the robot tries to detect faces in front of him. *ALPeoplePerception* is an extractor, which keeps track of the people around the robot and provides basic information about them. These modules work satisfactorily to keep eye contact with humans. However, there are situations in which these algorithms do not work properly. These are weak lighting or too strong light source (bright window, bright lamp).

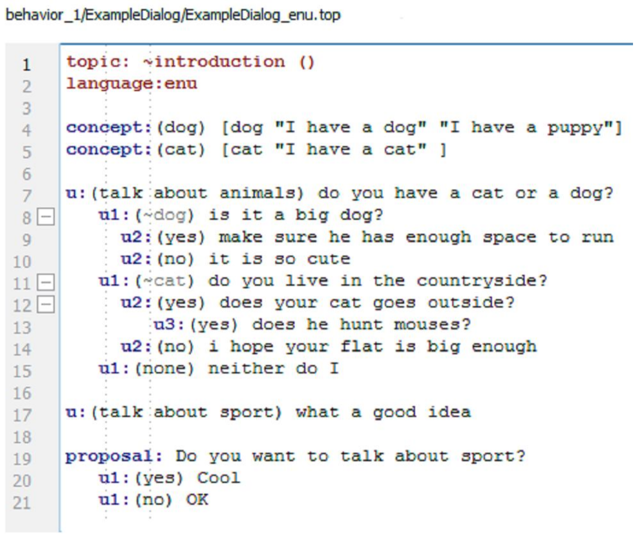
Pepper also has visual modules (eg. *ALFaceCharacteristics*) responsible for recognizing the features of the caller or his emotions: age, gender, smile, sadness and so on. These algorithms do not work very well and in practical applications are not very useful.

By default, Pepper is delivered with one language preinstalled (English) and one additional language can be installed. The list of the currently available languages is being expanded, and there are some languages that have limited usability (e.g. limited speech recognition capabilities). There are also languages that can be accessed as a "developer version", e.g. Polish. Any additional languages (i.e. more than two) can be installed, but for a fee.

The NAOqi's built-in speech recognition mechanism (*ALSpeechRecognition and ALDialog*, which rely on sophisticated speech recognition technologies provided by NUANCE) works quite well in laboratory conditions and acceptably under real life conditions. When used in noisy conditions, or when there are many sources of sounds, some mistakes may appear but the quality is still sufficient to communicate with robot. The robot also has the ability to generate speech. *ALTextToSpeech* module actually works very well.

The *ALDialog* module expands the robot possibilities with conversational skills by using a list of "rules" written and categorized in an appropriate way. By using *ALDialog* module it is possible to build a customized chatbot. In current version (2.5.5.5) many bugs were removed from the previous version. In *ALDialog* we can create a basic knowledge database for conversational skills. Extensive and plagiarized syntax makes it possible to design a very complex chatbot algorithm. It is possible to include multiple alternative words and phrases and introducing randomness in response. Due to this possibility it is possible to use Pepper for example for a simple customer support or human resources projects.

The syntax used in *ALDialog* text file is shown in the figure below.


behavior_1/ExampleDialog/ExampleDialog_enu.top

```
topic: ~introduction ()
language:enu

concept:(dog) [dog "I have a dog" "I have a puppy"]
concept:(cat) [cat "I have a cat" ]

u:(talk about animals) do you have a cat or a dog?
   u1:(~dog) is it a big dog?
     u2:(yes) make sure he has enough space to run
     u2:(no) it is so cute
   u1:(~cat) do you live in the countryside?
     u2:(yes) does your cat goes outside?
        u3:(yes) does he hunt mouses?
     u2:(no) i hope your flat is big enough
   u1:(none) neither do I

u:(talk about sport) what a good idea

proposal: Do you want to talk about sport?
   u1:(yes) Cool
   u1:(no) OK
```

Fig. 6. A example file containing the conversation logic syntax consistent with the syntax of the *ALDialog* module

The robot can also interact with humans by other technical means, e.g. by changing LED colors. Key areas containing LEDs are the eyes of the robot. There are several built-in mechanisms that change these LEDs (e.g. during *Dialog* robot changes eyes color to blue to indicate listening). Some of the Choregraphe's blocks utilize the robot's LEDs for preprogrammed behaviors (e.g. "blink") or for expressing mood [11]. The LEDs can also be freely controlled in software, to reinforce robot's actions and communication with its visual appearance.

In addition to all built-in user interaction mechanisms, it is possible to extend the capabilities of the robot, e.g. by redirecting a sound or video stream to an external system [12]. The robot then becomes literally a Human-Machine Interface, and the potential uses are almost limitless.

V. PEPPER'S MOVEMENT

Pepper is equipped with a 3 omnidirectional wheels [13], enabling him to move around freely through 360°. NAOqi offers a few methods of moving. In the *ALMotion* module, the *moveTo()* method [9] is available. This method uses odometry only to estimate robot position, relative to a starting location. It is possible to define destination coordinates x,y and rotation angle.

The second method is also called *moveTo()*, but declared in the *ALVisualCompass* module [9]. This method uses a vision system to store keypoints extracted from the reference image and from the current camera position. The method's algorithm matches keypoints from the origin image with the current image. It computes the global deviation of all keypoints, which are in both images, compares it to the reference position, and determines the relative distance and the deviation of the robot.

The third approach concerns using methods *learnHome()* and *goToHome()* from the *ALLocalization* module [9]. These methods are based entirely on the visual system. The first step is performed using the *learnHome()* method: the robot takes a series of photos and composes a 360° panoramic view of the surrounding. Wherever the robot is, the method *goToHome()* can be executed - the robot will load the panoramic view from memory, perform a new panoramic photo about new environment, compare those two pieces of information and return to starting location.

The figures below (Figs. 7 – 8) show a comparison of accuracy of reaching destination position, for these three methods under real life conditions.

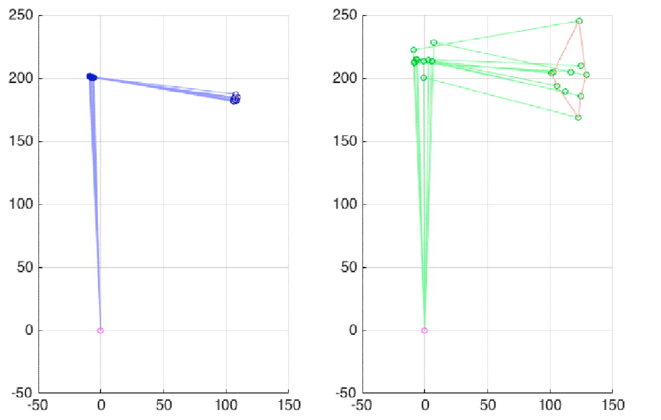


Fig. 7. Pepper's movement trajectories for different movement approaches - simple *moveTo()* using motors and encoders (left), and *compassMove()* using computer vision for navigation (right); (units: cm)

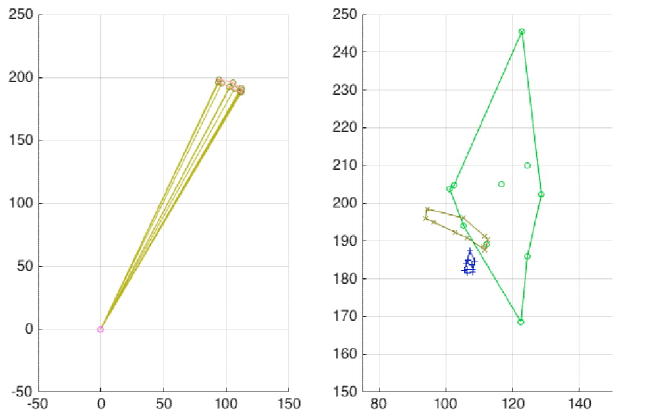


Fig. 8. Pepper's movement trajectories using *learnHome()* and *goToHome()* methods (left), and the dispersion of the reached coordinates of all three analyzed methods (rescaled, right); (units: cm)

The planned movement trajectory is the rotated letter "L" (with longer arm = 2m, shorter 1m, and angle 90°). In the *goToHome()* method the trajectory is calculated during movement using the vision system. In all cases starting point's coordinates are (0,0) and destination point's coordinates are (100,200).

To avoid collisions, the robot is equipped with a laser sensor. However, the lasers do not allow to perceive the environment 360° round. There are some "blind zones" (Fig. 9) between the areas covered by sensors, in which the robot cannot detect anything without moving its base or its head. In the blind zones, the arm speed (when moving) is reduced so that if it would collide with anything, it would not be dangerous. Additionally, the robot has bumpers. Bumpers are located inside every wheel cover and can detect only obstacles in direct contact with the robot.

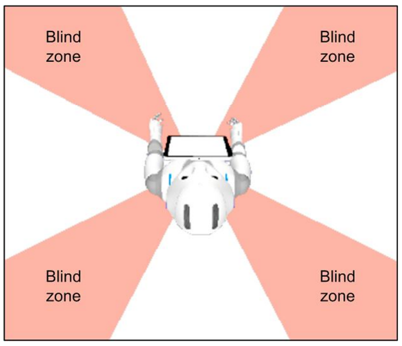


Fig. 9. Four blind zones (colored region) in which the robot cannot detect anything without moving its base or head [9]

In practice, the use of odometry is the most practical choice in terms of execution time. Repeatability and accuracy are satisfactory, but they depend to a large extent on the type of floor surface on which the robot moves.

The use of vision-based modules for basic navigation purposes gives more diverse results. Acceptable results (in terms of accuracy) are achieved by using the *ALLocalization* module, but for the *ALVisualCompass* method the scatter of the achieved target positions is significantly greater. In the case of the *ALLocalization* module, there is some inconvenience that the time required to complete a panoramic sequence of images takes about 30 seconds - it is hardly acceptable, particularly outside the laboratory environment.

VI. Summary

The Pepper robot proves to be great for entertainment or educational purposes [14], Pepper robots are liked by recipients regardless of their age. They still manage to surprise recipients with their movement capabilities and interaction with the environment and people [15, 16]. This design is also very promising for commercial uses, however additional software development is needed, which requires a team of Python developers. The SoftBank Robotics company provides sufficient support, concerning inter alia basic programming tutorials, themed interactive webinars, as well as a dedicated user forum, organized to help all independent developers - engineers of the SoftBank Robotics company are also active members of this community. Due to the fact, that the NAOqi operating system is identic in case of the NAO robot, countless solutions to Pepper's common issues or problems can be found in the NAO robot section. A large number of users registered in the developers community significantly contributes to the quality and number of available response resources.

The Pepper robot's design is sufficiently durable in typical use. The mapping of the mobility of its upper part to the human body mobility is sufficient. The simplification of the design of hands [17] to act as simple grippers is also not a problem. Lifting capacity of the robot is limited, but sufficient to hold a pen, or a small object. Sporadic problems during the operation of Pepper are acceptable in comparison with its benefits. However, fully unattended autonomous work of Pepper is still a challenge. The authors' experience shows, that the robot should be assisted (or at least supervised) by a living person.

REFERENCES

[1] SmashingRobotics, „Thirteen Advanced Humanoid Robots for Sale Today", updated 2016, available on-line (2017-03): https://www.smashingrobotics.com/thirteen-advanced-humanoid-robots-for-sale-today/

[2] Active8Robots, „Collaborative Robots in Higher Education", available on-line (2017-03): https://www.active8robots.com/sectors/collaborative-robots-in-higher-education/

[3] SoftBank Robotics, „SoftBank to Launch Sales of 'Pepper'" (press release), available on-line (2017-03): https://www.ald.softbankrobotics .com/en/Launch_Sales_of_Pepper

[4] SoftBank Robotics, „Press kit Pepper" (press release), available on-line (2017-03): https://www.ald.softbankrobotics.com/en/about-us/press/ press-kit

[5] M. Brophy, „Meet Pepper: Retail's First Robot", Independent Retailer, 2017, published on-line on 2017-03: http://independentretailer.com/ 2017/03/09/meet-pepper-retails-first-robot/

[6] M. Mori, K.F. MacDorman, and N. Kageki, „The uncanny valley [from the field]". IEEE Robotics & Automation Magazine, no. 19(2), pp. 98–100, 2012

[7] A. Rozanska, and M. Podpora, „Making eye contact with robot - exploring user experience in interacting with Pepper", submitted for publication, 2017

[8] C. Martin, „Pepper The Robot Looks To Engage Customers", MediaPost, 2017-01, available on-line (2017-03-27): http://www.mediapost.com/publications/article/292402/pepper-the-robot-looks-to-engage-customers.html

[9] SoftBank Robotics, „SoftBank Robotics Documentation", available on-line (2017-04-08): http://doc.aldebaran.com/2-5/index_dev_guide.html

[10] SoftBank Robotics, „Design that focuses on emotion", available on-line (2017-03-27): https://www.ald.softbankrobotics.com/en/cool-robots/ pepper/find-out-more-about-pepper

[11] SoftBank Robotics, „Who is Pepper?", available on-line (2017-04-08): https://www.ald.softbankrobotics.com/en/cool-robots/pepper

[12] A. Gardecki, and M. Podpora, „Extending vision understanding capabilities of NAO robot by connecting it to remote computational resource", Progress in Applied Electrical Engineering (PAEE), IEEE Xplore, 2016

[13] J. Lafaye, D. Gouaillier, and P.B. Wieber, „Linear Model Predictive Control of the locomotion of Pepper, a humanoid robot with omnidirectional wheels", 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids), available on-line (2017-07-07): https://www.researchgate.net/profile/David_Gouaillier/publication/281741354_Linear_Model_Predictive_Control_of_the_locomotion_of_Pepper_a_humanoid_robot_with_omnidirectional_wheels/links/55f6817b08ae1d9803976d2e.pdf , 2014

[14] F. Tanaka, K. Isshiki, F. Takahashi, M. Uekusa, R. Sei and K. Hayashi, „Pepper learns together with children: Development of an educational application", 15th IEEE-RAS International Conference on Humanoid Robots (Humanoids), Seoul, pp. 270–275, doi: 10.1109/HUMANOIDS.2015.7363546, IEEE Xplore, 2015

[15] A.H. Qureshi, Y. Nakamura, Y. Yoshikawa, and H. Ishiguro, „Robot gains social intelligence through multimodal deep reinforcement learning", 16th IEEE-RAS International Conference on Humanoid Robots (Humanoids), pp. 745–751, IEEE Xplore, available on-line (2017-07-07): https://arxiv.org/pdf/1702.07492.pdf , 2016

[16] V. Perera, T. Pereira, J. Conell, and M. Veloso, „Seting up Pepper for autonomous navigation and personalized interaction with users", arXiv preprint arXiv:1704.04797, 2017

[17] T. Hirano, M. Shiomi, T. Iio, M. Kimoto, T. Nagashio, I. Tanev, K. Shimohara, and N. Hagita, „Communication cues in a human-robot touch interaction", Proceedings of the Fourth International Conference on Human Agent Interaction, pp. 201–206, ACM, Spetember 2016