# The robot-guide for indoor navigation

Anna D. Volgina, Denis S. Kirillov, Andrew N. Kravtsov, Svetlana S. Dyakonova , Anton I. Kanev
Department of the Informatics and Control systems
Bauman Moscow State Technical University
Moscow, Russian Federation
anneta.volgina@mail.ru, d.turchin@bk.ru, kravtandr@gmail.com, dyakonova.s.s@yandex.ru, aikanev@bmstu.ru

***Abstract* — Robotics, speech recognition and computer vision are relevant problems of computing, signal processing and analysis. Using algorithms for audio and visual information processing allows to control a real mechanism in order to create an autonomous robot-helper. The paper describes the working principles of hardware and software of the robot-guide, their interaction and the implementation of the robot-server interconnection through existing World Wide Web infrastructure. The authors used the Raspberry Pi as a robot control plate, the system of speech recognition module based on PocketSphinx library and the sensor RPLIDAR A1 as the navigation module for computer vision implementation.**

***Keywords — robot, Raspberry Pi, lidar, speech recognition, navigation.***

## I. Introduction

Robotics is the most important technical basis for the development of modern production and services. The main goal of creating robots is to automate routine or dangerous works for humans. But they can also be used in a social environment to increase the comfort level of a person in the area of robot's responsibility, as in the case of the guide-robot, the project under consideration.

Computer vision and natural language processing play an important role in the development of robots designed for continuous interaction with the real environment. For example, advanced deep learning based computer vision algorithms are applied to enable real-time on-board sensor processing for small UAVs [1] and automatic speech recognizers are used in air traffic control communications [2].

For robots developers the task of choosing algorithms for the satisfying behavior according to the task and robots hardware capabilities is always relevant [3].

This work aims to study several methods of speech recognition, navigation and mapping as well as modern and most effective technical methods for conducting such studies and testing algorithms.

## II. Hardware architecture of the platform

The basis of the robot's chassis was chosen is a two-wheel differential drive equipped with a passive ground support, as it provides the best maneuverability in a room with a flat floor.

Raspberry Pi (ARM64) is used as a control board. To connect additional devices, the Troyka HAT expansion board is used. It adds additional pins via GPIO with hardware PWM making it possible to the Raspberry Pi to control encoder motors. The power source is 7.2V 5200 mAh Li-ion battery.

The WM8960 audio codec is engaged in solving the problems of recognizing voice commands and generating speech from text.

The RPLIDAR A1 lidar has 360° viewing angle and was chosen as a positioning tool. It provides the robot with information about the environment allowing to location determination.

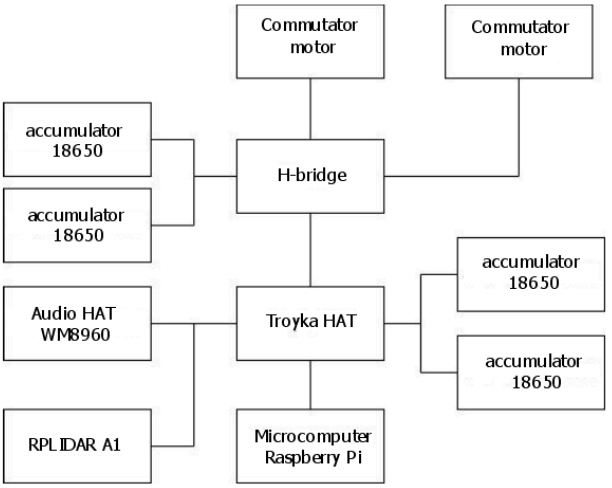


Fig. 1. Functional scheme.

## III. Software

The robot is based on the popular specialized Robot Operating System – ROS. It provides all the necessary functionality for the standard robots components. ROS is an add-on on top of LINUX with an additional package of utilities and services necessary for the robots development. It can be equated to a full-fledged OS by the fact that it provides all the standard operating system features such as hardware abstraction, low-level device and high-end package management and inter-process message mailing. We have installed ROS on a Raspberry Pi and a desktop computer, and then connected the ROS systems into a single network. The robot receives and transmits control signals and data from sensors through the system of modules called ROS-topics.

979-8-3503-9952-3/23/$31.00 ©2023 IEEE

## IV. Technical methods of conducting researches

Our team has studied and applied modern concepts for the mobile robotic platforms development and testing.

So, we have created and applied a web interface for tracking the odometry, intermediate sensors signals processing and other real-time algorithms. Metrics from the robot are sent to the web application server using ROS-topics, which are installed on the workstation paired with the robot and initiate communication with the server via WebSocket protocol. The workstation application is a part of the ROS concept. It is installed on the user's computer and connected to the robots' managed control board via a high-level XML-like XMLPRC protocol based on TCP. This allows to delegate resource-intensive calculations to a more efficient server.

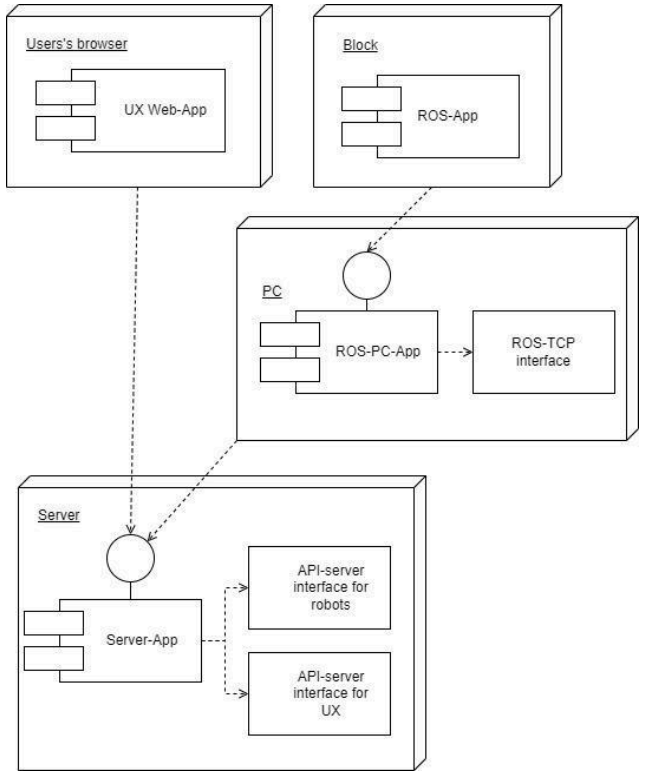


Fig. 2. The component architecture of the system.

## V. Speech recognition

### A. PocketSphinx

Voice commands are recognized by the PocketSphinx system, a part of CMU Sphinx, an open source speech recognition toolkit.

It is a software engine specialized for speaker-independent continuous speech recognition. An audio file (.wav) is converted to a defined language in text form using a pre-trained acoustic model in order to determine the source and destination language for speech-to-text conversion. The sample audio files are taken from large scale speech repository [4].

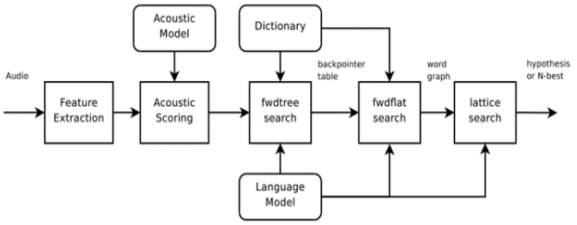


Fig. 3. PocketSphinx structure.

### B. Comparison with other systems

The most popular speech recognition systems were selected for consideration are CMU Sphinx, HTK, iAtros, Julius, Kaldi и RWTH ASR. They were compared in terms of recognition accuracy and speed, ease of use, and internal structure [5]. The accuracy of the systems was compared by the most common metrics: Word Recognition Rate (WRR), Word Error Rate (WER), which are calculated using the following formulas:

$$WER = (S + I + D) / T \quad (1.1)$$

$$WRR = 1 - WER \quad (1.2)$$

where:

- *S* is the number of word replacement operations,
- *I* is the number of word insertion operations,
- *D* is the number of word deletion operations from the recognized phrase to obtain the original phrase,
- *T* is the number of words in the original phrase.

In terms of recognition speed, a comparison was made using the Real Time Factor - an indicator of the ratio of recognition time to the duration of a recognized signal, also known as the Speed Factor (SF). This indicator can be calculated using the formula:

$$SF = T_{rec} / T \quad (1.3)$$

where:

- $T_{rec}$ is the signal recognition time,
- *T* is its duration and is measured in fractions of real time.

All systems were trained using the WSJ1 (Wall Street Journal 1) speech corpus containing about 160 hours of training data and 10 hours of test data. This speech corpus includes recordings of speakers of both sexes in English. After the experiment and processing of the results, the following table was obtained

TABLE I. METRICS COMPARISON

| System | WER, % | |
|---|---|---|
| HTK | 19,8 | |
| CMU Sphinx (pocketsphinx/sphinx4) | 21,4/22,7 | |
| Kaldi | 6,5 | |
| Julius | 23,1 | |
| iAtros | 16,1 | |

| | |
|---|---|
| RWTH ASR | 15,5 |

The system implementation language, the algorithms used in recognition, input and output data formats, and the internal structure of the system software implementation were chosen as criteria for comparing structures.

TABLE II. ALGORITHMS COMPARISON RESULTS

| System | Feature extraction | Acoustic modeling | Language modeling | Recognition |
|---|---|---|---|---|
| HTK | MFCC | HMM | N-gramm | Viterbi algorithm |
| CMU Sphinx (pocketsphinx/sphinx4) | MFCC, PLP | HMM | N-gramm, FST | Viterbi algorithm, bushderby |
| Kaldi | MFCC, PLP | HMM, GMM, SGMM, DNN | FST, converter N-gramm -> FST | Forward-backward algorithm |
| Julius | MFCC, PLP | HMM | N-gramm, Rule-based | Viterbi algorithm |
| iAtros | MFCC | HMM, GMM | N-gramm, FST | Viterbi algorithm |
| RWTH ASR | MFCC, PLP, voicedness | HMM, GMM | N-gramm, WFST | Viterbi algorithm |

CMU Sphinx shows mean recognition accuracy (WER~22%) but the best recognition rate of all reviewed (SF=0.5). As a result, this system works faster on weak computers like Raspberry Pi. Structurally, this system uses many modern approaches to speech recognition, including a modified Viterbi algorithm, although there are fewer approaches used than Kaldi. In terms of ease of use, CMU Sphinx is ahead of Kaldi, since in addition to the console interface it provides an API, which greatly simplifies the process of embedding the system into a third-party application. It also has detailed documentation, which, unlike Kaldi, is aimed at a novice developer, which greatly simplifies the process of getting to know the system. Thus, PocketSphinx was chosen for the project.

### C. *Speech recognition*

Our team has conducted a research on speech recognition machine learning model based on PocketSphinx in attempt to solve the Russian language commands recognition task.

The key phrase in the project is a Russian appeal to the robot “Робот” ("robot"). Commands are built using a phonetic dictionary - a set of all words that the robot must recognize, and grammar - a set of rules by which word combinations are formed from them.

Having heard the correct command, the robot repeats it. To make an intelligent dialogue with the user autoregressive generative models trained on language modeling tasks can be used for question generation based on the given text[6].

To study the accuracy of recognition, an experiment was conducted: the commands of each class under study were spoken at least 20 times and also several incorrect phrases were spoken a few times into the microphone, the True Positive, True Negative, False Positive and False Negative indicators for the classes were counted. Further, according to the obtained values, the Precision, Recall, F1 and Accuracy indicators were calculated.

Precision is the fraction of relevant instances among the retrieved instances

$$\text{Precision} = TP / (TP + FP) \quad (2.1)$$

Recall is the fraction of relevant instances that were retrieved

$$\text{Recall} = TP / (TP + FN) \quad (2.2)$$

The F1-score is the harmonic mean of the precision and recall

$$F1 = 2 / (1 / \text{Precision} + 1 / \text{Recall}) \quad (2.3)$$

Accuracy is how close a given set of observations are to their true value

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (2.4)$$

TABLE III. POCKETSPHINX BASED TRAINED MODEL METRICS

| Command classes | Precision | Recall | F1 |
|---|---|---|---|
| 901 | 0,962962963 | 0,787878788 | 0,866666667 |
| 901.1 | 1,000000000 | 0,888888889 | 0,941176471 |
| 901.2 | 1,000000000 | 0,851851852 | 0,920000000 |
| 903 | 1,000000000 | 0,682926829 | 0,811594203 |
| 904 | 0,846153846 | 0,785714286 | 0,814814815 |
| 905 | 0,928571429 | 0,634146341 | 0,753623188 |
| 906 | 0,961538462 | 1,000000000 | 0,980392157 |
| 907 | 1,000000000 | 0,911764706 | 0,953846154 |
| elevator | 0,590909091 | 1,000000000 | 0,742857143 |
| staircase | 0,347826087 | 0,923076923 | 0,505263158 |
| hanger | 0,870967742 | 0,964285714 | 0,915254237 |

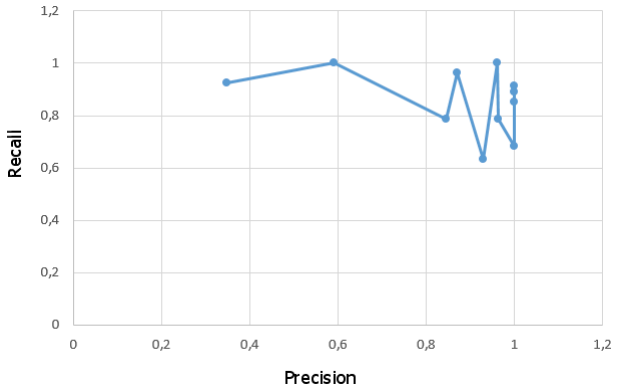


Fig. 4. Recall and precision graph.

The command recognition accuracy for the considered data set is 75%.

## VI. NAVIGATION

One of the points of our work is the navigation task, we need to teach the robot to move in space, namely to get to

the chosen destination. To solve this problem, we need data about the environment represented by point cloud [7], which we will get with the help of a lidar. The developed robot uses a lidar model RPLIDAR A1. Using it, the robot receives information about the surrounding space in the form of a cloud of points.

Based on the collected data, 3 navigation tasks shown in the figure 5 are solved.

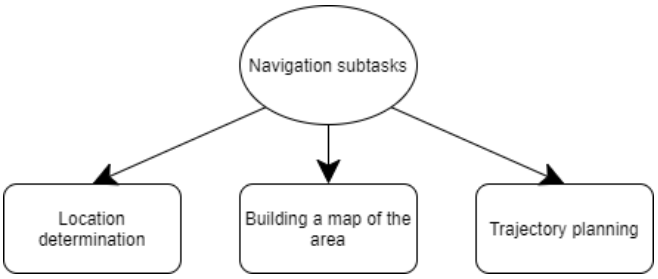


Fig. 5. Navigation subtasks.

As a rule, separate modules are used to solve these tasks, for example, to solve the trajectory planning subtask, various decision-making systems are used. One option is a decision-making system based on Mivar expert systems[8, 9]. However, the basic SLAM navigation algorithm solves two of them at once - localization and construction of the environment map, which distinguishes it from other solutions. The SLAM navigation algorithm is described in figure , where C is the control of the robot at time t, L is the obtained location of the robot at time t, E is the information about the environment around the robot at time t, and M is the constructed map.

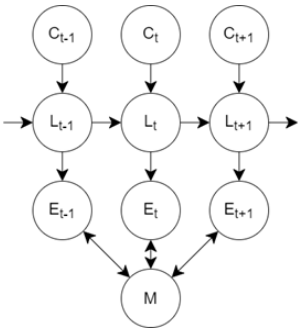


Fig. 6. SLAM navigation algorithm.

At time *t-1* the robot is at point $X_{t-1}$ and receives information about the environment $E_{t-1}$. The robot compares the received information with the existing map *M* and changes the map if it finds a difference. Then we control the robot with the command $C_{t-1}$ and get to the point $X_t$, then the algorithm repeats again.

Modern SLAM navigation packages, such as GMapping and Google Cartographer, implement solutions to all the above navigation subtasks at once. In addition, all these packages are supported and actively used by the community around the world with OS ROS, which will be the basis of our developed robot. Because of this, we will compare and consider these packages for use in controlling the robot.

GMapping is based on Rao-Blackwellized particle filter (RBPF) [10] that estimates a posterior $p(L_{(1:t)}|E_{(1:t)}, C_{(0:t)})$ about potential robot trajectories $L_{(1:t)}$ using observations $E_{(1:t)}$ and odometry $C_{(0:t)}$ data. The posterior is approximated with a set of points (particles) with corresponding probabilities (weights).The particle with the maximal weight is treated as the actual world state. The weight of a particle is updated with a correspondence measure between a new scan and the map estimated by a scan matcher.

Gradient descent is used in GMapping to match scans. At each iteration several predefined directions are checked. The starting position is the one with the maximum matching result. The scan score is computed as follows:

$$score(scan,\ map) = \sum_{p \in scan} e^{-\frac{1}{\sigma} * d(p,\ map)^2} \quad (3.1)$$

where:

- p – a scan point;
- d(p, map) – the minimal distance between p and an obstacle stored in the map (one obstacle point per cell);
- σ – the predefined dispersion.

Google Cartographer stores a map of an environment as a graph where every vertex presents a submap and scans got after the corresponding submap has been created. The edges represent transformations between corresponding submaps. So an optimization step to make the map consistent appears besides a scan matching process. Every cell of a Cartographer submap has a probability to be occupied. It is updated in case the submap has not been fully constructed yet. The following formula presents the rule of updating the cell value being hit (or miss):

$$M_{new}(cell) = odds^{-1}(odds(M_{old}(cell) * odds(p_{hit}) \quad (3.2)$$

where:

- $M_{old}(\cdot)$ – the old probability of the cell;
- $p_{hit}$ – the probability of the cell being hit;
- $odds(p) = \frac{p}{1-p}$ .

Minimizing functional costs is the basic idea of the scan matching approach. Thus, the scan matching process undergoes a minimization of the following functionality:

$$arg_{\xi} \min \sum_{k=1}^{K} (1 - M_{smooth}(T_{\xi} h_k))^2 \quad (3.3)$$

where:

- $M_{smooth}(x)$ – the value of a cell x smoothed by values in its neighborhood;
- $h_k$ – a cell involves a point from a laser scan;
- $T_{\xi}$ – the transformation matrix that shifts a point hk by ξ;
- ξ – the offset vector (ξx, ξy, ξθ)T .

This minimization is presented as a brute-force approach with a branch-and-bound modification.

Let us turn to the experiments given in the article [11], aimed at comparing the GMapping and Google Cartographer algorithms by analyzing the deviations from the trajectory at a certain distance. The results are presented in Table 4.

TABLE IV. GMAPPING AND GOOGLE CARTOGRAPHER COMPARISON

| Sequence | Length, m | Trajectory RMSE, m | | |
|---|---|---|---|---|
| | | *GMapping* | *Cartographer (online)* | *Cartographer (offline)* |
| 2011-01-19-07-49-38 | 68 | 0.216 ± 0.012 | 0.188 ± 0.023 | 0.191 ± 0.001 |

| | | | | |
|---|---|---|---|---|
| 2011-01-20-07-18-45 | 76 | 0.219 ± 0.012 | 0.219 ± 0.002 | 0.221 ± 0.001 |
| 2011-01-21-09-01-36 | 87 | 0.212 ± 0.029 | 0.217 ± 0.003 | 0.205 ± 0.001 |
| 2011-01-24-06-18-27 | 87 | 0.290 ± 0.035 | 0.217 ± 0.001 | 0.217 ± 0.001 |
| 2011-01-25-06-29-26 | 109 | 0.208 ± 0.008 | 0.232 ± 0.001 | 0.232 ± 0.002 |
| 2011-01-27-07-49-54 | 94 | 0.266 ± 0.012 | 0.266 ± 0.004 | - |
| 2011-01-28-06-37-23 | 145 | 2.388 ± 1.949 | 0.360 ± 0.069 | 0.354 ± 0.004 |
| 2011-03-11-06-48-23 | 245 | 0.365 ± 0.208 | 1.152 ± 0.601 | 1.348 ± 0.001 |
| 2011-03-18-06-22-35 | 80 | 0.145 ± 0.023 | 0.145 ± 0.021 | - |
| 2011-04-06-07-04-17 | 95 | 0.190 ± 0.002 | 0.201 ± 0.002 | - |
| 2011-10-20-11-38-39 | 264 | 0.352 ± 0.003 | 2.217 ± 0.021 | - |

Based on the table, we conclude that the results of both methods are comparable with each other. That is, both methods are approximately equal in efficiency. However, Google Cartographer is supplied as a cloud solution, i.e. its computing power is located on Google servers. Because of this, this package is not suitable for us and preference is given to GMapping.

## VII. Conclusion

We have proved a chosen robot development toolkit with several researches. ROS is a great solution for Raspberry PI based mobile robots. The web app helps to read odometry in real time. Pocketsphinx library solves an offline speech recognition task efficiently. Google Cartographer and GMapping both turn to be a good SLAM algorithm implementation, however, the second one doesn't depend on external clouds. All these tools have contributed a lot in development speed of our guide-robot.

## Acknowledgment

We thank Valery Terekhov (Bauman Moscow State Technical University) for a financial contribution and assistance in publishing.

## References

[1] A. Palmas, P. Andronico, "Deep Learning Computer Vision Algorithms for Real-time UAVs On-board Camera Image Processing", NATO AVT-353 Research Workshop "Artificial Intelligence in Cockpits for UAVs", Turin, Italy, 2022.

[2] J. Zuluaga-Gomez, K. Veselý, I. Szöke, P. Motlicek, M. Kocour, M. Rigault, K. Choukri, A. Prasad, S. Sarfjoo, I. Nigmatulina, C. Cevenini, P. Kolcárek, A. Tart, J. Cernocký, "ATCO2 corpus: Large-Scale Dataset for Research on Automatic Speech Recognition and Natural Language Understanding of Air Traffic Control Communications", 2022

[3] A. Volkov, O. Varlamov "Method of creation of a two-level neural network structure for solving problems in mechanical engineering" 2021 J. Phys.: Conf. Ser. 2131 032003

[4] S. R. Poojara, C. K. Dehury, P. Jakovits, S. N. Srirama, Serverless data pipeline approaches for IoT data in fog and cloud computing, Elsevier FGCS, 2021.

[5] Andreeva A. R., Rudoman N. R., Sevanyan A. V., "Comparative analysis of speech signal conversion algorithms", Modern problems of physics, biophysics and infocommunication technologies, Krasnodar, 2018

[6] Marina A. Belyanova, Ark M. Andreev, Yuriy E. Gapanyuk, "Neural Text Question Generation for Russian Language Using Hybrid Intelligent Information Systems Approach", Bauman Moscow State Technical University, Moscow, Russia

[7] Elizaveta K. Sakharova, Dana D. Nurlyeva, Antonina A. Fedorova, Alexey R. Yakubov and Anton I. Kanev "Issues of Tree Species Classification from LiDAR Data using Deep Learning Model", Bauman Moscow State Technical University, Moscow, Russia

[8] Oleg O. Varlamov ""Brains" for Robots: Application of the Mivar Expert Systems for Implementation of Autonomous Intelligent Robots", Bauman Moscow State Technical University, Moscow, Russia

[9] D. V. Aladin, O. O. Varlamov, D. A. Chuvikov, V. M. Chernenkiy, E. A. Smelkova and A. V. Baldin, "Logic-based artificial intelligence in systems for monitoring the enforcing traffic regulations", Bauman Moscow State Technical University, Moscow, Russia

[10] A. Doucet, N. de Freitas, K. P. Murphy, S. J. Russell, "RaoBlackwellised Particle Filtering for Dynamic Bayesian Networks", in Proc. of the 16th Conference on Uncertainty in Artificial Intelligence, pp. 176-183, 2000.

[11] K.Krinkin, A.Filatov, A.Filatov, A.Huletski, D.Kartashov, Evaluation of Modern Laser Based Indoor SLAM Algorithms, in proceeding of the 22nd conference of fruct association.