Progress Report

1st Mykhailo Dmytrenko

2nd Antonio Sgorbissa

I. Introduction

In this brief overview the current progress of the project will be discussed, highlighting different points of interest, as well as plans for the future work.In the context of human SLAM for environmental disasters, there are currently several subjects of the main focus : **Karto - SLAM**, **Onthologies**, **Voice recognition** and usage of **extra sensors - Thingy91**, **Deterministic loop closure** and **Microsoft HoloLens map visualization**.

II. Karto - SLAM

Following the work of the Master thesis student who worked on this project, Karto-SLAM was chosen as a framework for the graph-based SLAM. The above-mentioned was improved significantly perfomance-wise and was able to successfully generate a graph based map real-time with a size of 120 by 55.6 meters with 2706 nodes in total with 1 forced loop closure and 7 automatic closures. The result can be seen in Fig. 1.

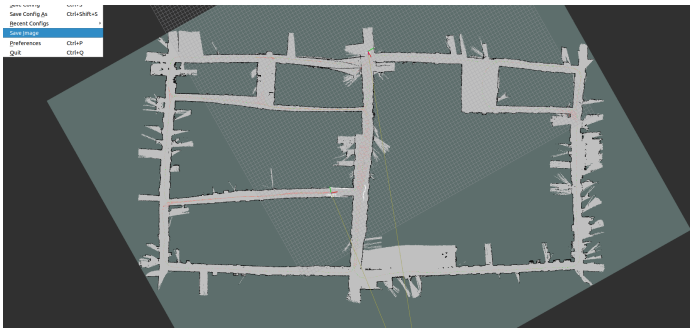

Fig. 1. Optimized Map built in real-time

III. Ontology

A major introduction to the loop closure algorithm are ontologies. The words were chosen according to the typical landmarks which are typically seen in the urban environment - in this case the historical centre of Genova. Example photo can be seen in Fig.2. Corresponding ontology tree is depicted on the Fig.3.

Several Python scripts automatically read and generate lists of words compatible with the speech recognition engine as well as with graph-based SLAM.

IV. Voice recognition - Picovoice

Finding a reliable ofline recognition engine presented a major challenge. Initially, CMU Sphinx recognizer [1] has been chosen. However, it failed to prove to be usable enough

Fig. 2. Example photo of the streetview

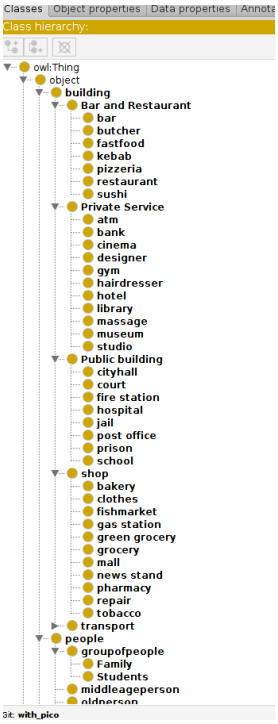

Fig. 3. Ontology generated

for the needed application. The main issues were unrecognizable words, false negatives and inability to recognize all words in the ontology at once. A different engine was chosen called **Picovoice** [2]. It is an efficient offline solution which uses compressed neural networks and can work on resource-restrained microcontrollers. It was successfully adapted to the current application in tandem with the ontology and it is able to recognize all 94 words of current ontology at once.

There are two main tecnologies included in Picovoice - **Porcupine** [3] wake word engine and **Rhino** [4] Speech-to-Intent Engine. Porcupine proved to be a very accurate wake word engine. It contains a list of pre-trained words, which can be used for the activation of the listening process. In addition to this, there is a possibility to train custom words using Picovoice console. Once the wake word is detected, Rhino engine gets activated. After a list of words to detect has been added, it is possible to train and download the model. Afterwards, the two models (from Porcupine and Rhino engines) can be used in a Python scripts on the local machine, hence once all models are downloaded the speech recognition process can work completely offline.

V. Extra sensors

In the context of environmental disasters there is also a need to gather extra information about the environment - temperature, vibrations, humidity etc. Thingy91 from Nordic Semiconductor [5] is a battery operated prototyping system with a multitude of sensors for motion, impact, air quality and much more. In addition to this, Nordic Semiconductors offers a cloud which gets all the data from the kit through the cellular data connection which than can be retrieved from anywhere. The kit supports NB-IoT and LTE-M, which are both communication technologies for IoT devices which work with cellular data. In Italy only NB-IoT is supported.

Fig. 4. Thingy91 development platform

The readily available data is visualized on the **nrfCloud**, which is seen on Fig. 5. This data can accessed as a JSON file with a curl command for further processing in Python script.

The main issue with this develiopment kit is that detailed accelerometer data is not available in the current software version. Instead, it is in the under developing section and is currently explored.

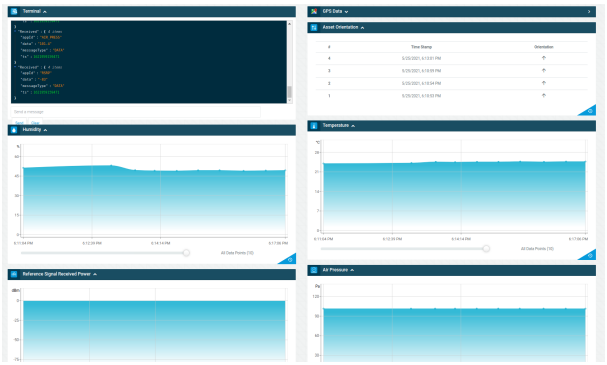

Fig. 5. nrfCloud data visualization

VI. Software overview

The general overview of the ROS nodes can be seen on Fig.6.For the audio input **Mapper** is listening for the word index which is then added to the graph node as a custom object. Then, in order to make use of semantic distances these indices are converted back into words where the ontologies information can be used again through a ROS service **dist_service**.

VII. Deterministic loop closure

Currently, deterministic loop closure algorithm uses 3 parameters to decide how to close the loop: laser scan response, physical distance and semantic distance. Each special node gets compared with the newly-added one. In case several criteria are met, the edge strength is determined proportionally to the above-mentioned parameters. E.g. the covarience of the link created is smaller the closer the nodes are to each other.

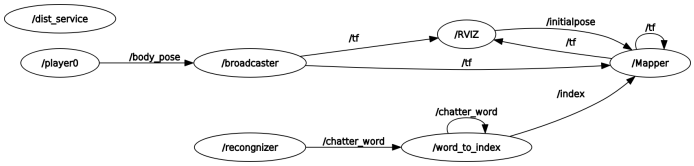

Fig. 6. Overview of ROS structure

VIII. Microsoft Hololens Map Visualization

There are some ready made tools available to save the map to the server in order to be visualized with Microsoft Hololens. One example is **map_server** in ROS, however it saves only the map without any frames or extra information, which can be a problem in the future. Instead it would be much more beneficial to save the entire RVIZ view, which can be done with **rviz_camera_stream**. This approach requires much more customization but more beneficial in the long run. [4]

References

[1] Sphinx home page. [Online]. Available: https://cmusphinx.github.io/
[2] Picovoice homepage. [Online]. Available: https://picovoice.ai/
[3] Porcupine introduction web page. [Online]. Available: https://picovoice.ai/docs/porcupine/

[4] Rhino introduction web page. [Online]. Available: https://picovoice.ai/docs/rhino/
[5] Thingy91 web-page. [Online]. Available: nordicsemi.com/Software-and-tools/Prototyping-platforms/Nordic-Thingy-91