Mappino - Open Source Robot for Learning about IR Range Scan Matching

Albert Diosi
This paper is the result of independent, self-sponsored research
Samorin
Slovakia
Email: albert.diosi@gmail.com

Abstract—**Localization and mapping is a basic building block for many mobile robots. Scan matching is one way of implementing such a functionality. Currently a significant commitment of time and financial resources are likely to be necessary to delve into this area of robotics. This paper aims to bring scan matching closer to robotics enthusiasts and students by introducing an inexpensive, open source robot programmable through the accessible Arduino programming environment. In the presented robot, rotating infra red range sensors are used instead of a laser scanner. Simple example code is available for the robot to facilitate a quicker familiarization process. Preliminary experimental results demonstrate the scanning and scan matching capability of the robot.**

I. Introduction

In the future, mobile robots will likely play an important role in people's lives. Beside being useful, mobile robots can be fun to build and exciting to experiment with. Building and experimentation may encourage youth in a playful manner to learn about programming, maths, physics and engineering in general.

For decades, robotics enthusiast have been building mobile robots. However, their efforts often stopped before creating truly useful ones. Perhaps this has been caused by the lack of mapping and localization capability. It is often easier to solve a practical problem if the robot knows its pose and has access to a map of its environment. As an example let us consider a plant watering robot. If one assumes that the pots holding the plants are stationary (a reasonable assumption to make if the robot has to work only when the owners are on holiday) the problem of making the robot to know where to water can be reduced to the problem of knowing where the robot is and where the pots/plants are.

Perhaps one of the reasons why localization and mapping did not catch on in the circles of robotics enthusiasts is due to the large initial commitment required in time and often in finances to complete a mobile robot capable of self pose awareness. Furthermore the complexities of the maths involved may scare people away from implementing a localization and mapping functionality.

In recent years, several off-the-shelf solutions have appeared for localization. If the modification of the environment is acceptable, then one can use for example the NorthStar system developed by Evolution Robotics[1]. Here a stationary unit projects a pattern of light on the ceiling (if there is one) which is observed by an optical sensor unit mounted on a mobile robot. From the the light pattern observation, the unit tells the robot its estimated pose. A similar localization device called StarGazer has been developed by the company Hagisonic[2]. However, instead of pattern projection, fiducials are placed on the ceiling (if there is one). The markers are then observed by a camera unit (including an illumination source) placed on the robot. Based on the observed appearance of the fiducials, the camera unit tells the robot its pose estimate. Unfortunately none of the mentioned approaches are inexpensive to buy, open source (do not facilitate learning and experimentation to a great extent) or provide information about obstacles and both require the modification of the robot's environment.

Popular solutions to the localization problem whithout requiring the modification of the environment include the use of sonars, cameras (2D and 3D) and laser range finders. Simple sonar systems are inexpensive, however due to wide beamwidth, specular reflections and cross-talk they are not straightforward to use for localization and mapping. An example of open source sonar localization (based on an existing map) can be found in CMU's Carmen robot software package[3].

Cameras seem to be newcomers compared to lasers and sonars, yet they gained rapid popularity in the area of localization and mapping. Cameras can be inexpensive, however they require a considerable amount of processing power for utilization. Furthermore, learning about how to use cameras for localization and mapping may demand a considerable amount of effort from the buddying roboticist. Luckily there are readily available open source software packages as Willow Garage's Robot Operating System (ROS)[4] implementing such localization functionality for stereo and 3D cameras. Just as Carmen, ROS is processing power hungry and the familiarization with ROS may take a considerable amount of effort.

Laser range finders provide a fairly simple and reliable way of localization and mapping. Their price used to be prohibitive for robotics enthusiasts and for classroom use, however as shown in [1], it is possible to build a $30 bill of material (BOM) laser range finder. The mentioned laser

[1]www.evolution.com/products/northstar (Accessed: 28 May 2011.)

[2]www.hagisonic.com (Accessed: 28 May 2011.)

[3]carmen.sourceforge.net (Accessed: 28 May 2011.)

[4]wwww.ros.org (Accessed: 28 May 2011.)

range finder later appeared in the robotic vacuum cleaner of Neato Robotics[5]. Laser scan matching is a popular way of performing localization and mapping. Software modules for performing scan matching are present in both Carmen and ROS. However existing laser scan matching approaches can be processor intensive, and fairly complex.

The goal of the work behind this paper is to develop a robot system minimalistic in size, cost and complexity for educational use, capable of performing localization and mapping in simple environments while running on a hard, flat surface. Initial results aimed at achieving this goal have been obtained by constructing a simple but limited laser range finder replacement embedded into a simple robot base. All processing on the robot is done using boards which can be programmed using the simple and popular Arduino integrated development environment (IDE)[6]. A laser scan matching algorithm, the Polar Scan Matching (PSM) [2] has been adapted to run on the utilized inexpensive 8-bit microcontroller. The software and hardware designs are open source which facilitates easy experimentation and modification. The electronic components of the robot have been designed with re-usability in mind for other projects.

The concept of the range scanner comprising of rotated IR range sensors (PSD - position sensitive device) is not new. There are many examples for it in the literature for example [3]. These sensor have even been used for Simultaneous Localization and Mapping (SLAM) for example through the observation of line segments as in [4].

The contributions of this paper are two fold: an open source platform for learning about localization and mapping for robotics enthusiast and students in high school or university classrooms. There is also an engineering contribution as there does not seem to be a precedence in the literature for performing scan matching on a small, 8 bit microcontroller. All the design files are available under a suitable open source license for free.[7]

The paper is structured as follows. First the robot platform is described followed by a brief description of the scan matching algorithm. Experimental results are shown next followed by discussion and conclusions.

II. Hardware Platform

A. Requirements

Prior to designing the robot the following lax requirements were set solely based on practical considerations of the author:

- Robot should enable easy learning about scan matching, mapping and localization.
- BOM should be below 150 Euros when building a single robot.
- The robot should be able to take 2D scans of simple environments while having controlled lighting conditions.

[5]wwww.neatorobotics.com (Accessed: 28 May 2011.)
[6]wwww.arduino.cc/en/Main/Software (Accessed: 28 May 2011.)
[7]www.diosirobotics.net/mappino.html

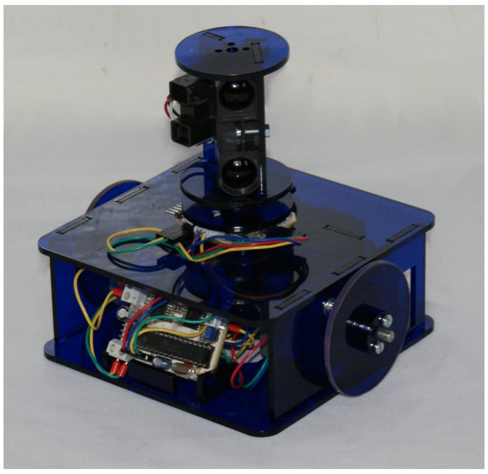
Fig. 1. Mappino v0.1.

TABLE I
Mappino Specifications

Size (WxDxH)	168x144x148mm
Weight	1100g
Motors	3 x Bipolar Stepper
Sensors	Sharp PSDs: GP2Y0A02YK, GP2Y0A710K
Batteries	6xAA 2450mAh NiMh
Estimated Runtime	≥20h on Standby, ≥2h while moving
Maximum Speed	≥10cm/s
Estimated Cost	220 euros

- Robot should be able to move indoors on flat hard floors and table tops.
- Robot run time should be longer than the duration of a double class (i.e. 2 hours).
- Minimum speed should be at least 10cm/s.
- Robot should be easily programmable, without learning much about microcontrollers.
- Robot should come with basic demos which provide a starting point for students.
- Individual robot parts should be modular, reusable and modifiable.
- Robot should be small and light for easy transportability.
- Robot should use easily available batteries.
- Electronics parts should be easily obtainable using a popular component distribution company such as Farnell.
- Mechanical parts should be easily obtainable by rapid prototyping services such as Ponoko.
- All software used in the design and for programming the robot should be freely obtainable, preferably open source. This is to ensure that users easily can modify the design.
- Robot should be programmable from three of the major operating systems (Linux/Mac/Windows).

B. The Robot Mechanics

The implemented solution (see fig. 1,2) consist of a robot base into which a rotating range finder (see fig. 3) module is embedded. The components of Mappino can be seen in fig. 4. The specifications of Mappino are shown in tab. I.

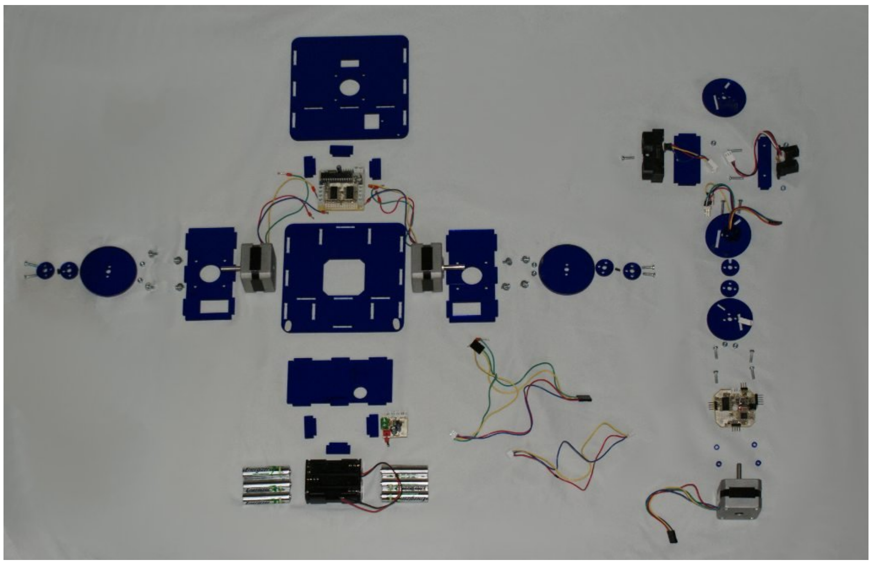
Fig. 4. Most of Mappino's components laid out as they are assembled.

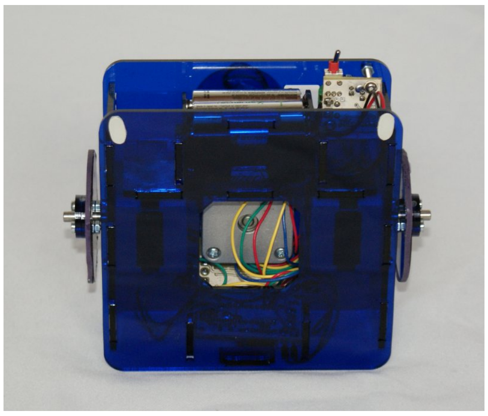
Fig. 2. Mappino viewed from underneath. Notice the power board and the mouse skids.

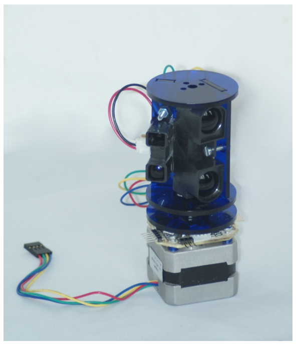
Fig. 3. The range scanning module.

Most mechanical parts except nuts, bolts and motors are laser cut from 3mm thick blue acrylic. Through careful design, most parts just snap together. However if extra strength is necessary one can apply glue as well. To improve traction, rubber bands are glued onto the wheels. This may be changed to O-rings in the future to ease assembly. The back of the robot is weighted down by the 6 AA batteries used as a power source. Two mouse skids mounted on the bottom of the robot under the batteries ensure easy gliding on smooth surfaces.

The robot is driven by two bipolar stepper motors micro-stepped to 800 steps per revolution. Stepper motors were chosen to simplify the control of the robot and to increase the odometry accuracy.

In the odometry center of the robot a pair of rotating range sensors is mounted. The range sensors are rotated by the same type of stepper motors as used for propelling the robot. As range sensors a long range (100-550cm) Sharp distance sensor (PSD) is used in conjunction with a shorter range (20-150cm) PSD. The shorter range PSD is offset by 45°. The purpose of the shorter range PSD is to provide distance measurements to objects closer than the minimum range of the long range sensor (100cm).

To reduce cost and to simplify the design the robot does not have a bump sensor. Instead the range sensors are to be used to detect close obstacles. However, users may add their own bump sensors if necessary.

The laser cut acrylic used in the robot providing the structural elements may be replaced in the next revision with a less brittle material such as Lexan or laser cut aluminium to increase the life expectancy of the base. Given the considerable weight of the robot due to the 3 stepper motors and the batteries, the robot can not be handled roughly without risking breaking the acrylic.

The next version of the robot may entail a reduction of size. Grub screws are also considered to enable solid mounting of the wheels and the sensor head to the motor shafts. Currently, frequent mounting and removing of the wheels and sensor head may loosen the tight fit of the plastics and cause slipping. There are other minor tweaks considered as extra holes for mounting, cabling and for making some of the screws easier

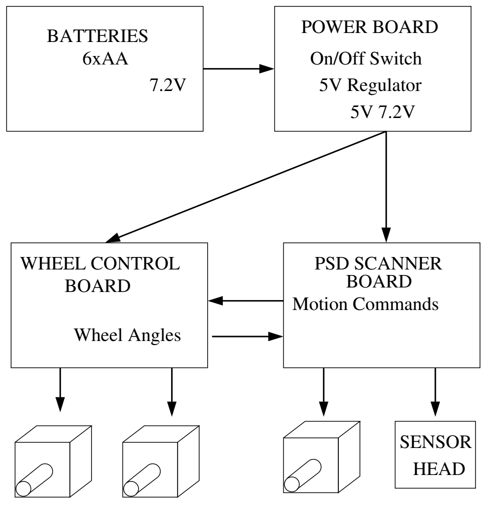

Fig. 5. Overview of the electronics. The wheel control and scanner boards can be programmed through the Arduino IDE.

to access with a screw driver.

C. Electronics

The electrical connection of the robot can be seen in fig. 5. As already mentioned, the robot is powered by 6 AA, preferably NiMh rechargeable batteries. Power from the batteries is lead to a power distribution board which hosts a resettable fuse, an on/off switch and a 5V regulator. The battery voltage and the regulated 5V voltages are connected to the dual bipolar stepper motor control board and the PSD scanner board.

The motor control board accepts serial line commands from the PSD scanner board and reports back motor orientations. The board can micro-step two bipolar stepper motors. The current into the motor windings can be set by software.

The PSD scanner electronics is mounted on top of the stepper motor, just under the sensing head. The board controls the panning stepper motor, evaluates a position reference signal and samples the analog signals provided by the PSDs. It is also the job of the PSD scanner board to generate the scans, perform scan matching and to generate the desired commands for the motion control board.

To enable energy consumption control, motors and sensors can be turned off by software. The current consumption while having the motors and sensors turned off is 60mA. By turning on the sensors and the panning motor, this current goes up to 200mA. In motion, the current consumption increases to 680mA. The current consumption is the highest with 870mA when the traction motors are activated but stationary. In the demo code supplied, motors and sensors are turned off when not in use, thus one can have the robot sitting on a desk in the "on" state for tens of hours when using 2450mA capacity batteries. In motion, the batteries will hopefully last for more than two hours even at the end of their life cycle.

In the spirit of a reusable/modifiable design, the PSD scanner can be used as a standalone unit, independent of the robot base. Furthermore the sensing head can be easily replaced with other sensors. The motor controller and the motors can be be used on other projects as well. There are prototyping areas on the motor control and PSD scanner boards to allow adding extra components.

As the parts have been designed using freely available software as QCad[8] for the mechanics and GEDA's gSchem[9] and PCB for the electronics, users can customize the design of the robot. The author used Linux throughout all of the design and programming process, however there are reports of the tools working on other operating systems as well.

TABLE II
APPROXIMATE COST BREAKDOWN IN EUROS.

Acrylic	30
PCBs	54
Motors	63
Sensors	55
Batteries	13
Other	5
Total	220

D. Assembly

Given pre-made boards and wiring harness, the robot can be assembled in few tens of minutes. However, making the wiring harness and assembling the boards requires skills and special tools. To reduce size and cost, surface mount parts were used.

E. Programming Environment

Until recently the programming of 8 bit microcontrollers using freely available software was often not an easy task. However, all has changed with the arrival of the Arduino boards and their programming environment (based on Wiring[10]). Arduino has enabled people with non-technical backgrounds to rapidly create embedded system prototypes. The programming language of the Arduino boards is a well documented, easy to use, simplified version of C/C++. There is also a vibrant and eager community of Arduino users all over the world. The only disadvantage is that the Arduino IDE does not support on-board debugging. Due to the virtues of the Arduino concept, the motion control board and the PSD scanner board can be programmed through the Arduino IDE and application programming interface (API).

F. Cost

When constructing only one robot, the BOM cost including value added tax and delivery fees is approximately 220 euros as can be seen in tab. II. The target price of 150 euros has been grossly overshot, however a significant portion of the cost are delivery fees. Furthermore the price includes the manufacture of 10 bare boards of each of the 3 boards. Educational institutions of some countries would not have to pay tax on the parts which considerably reduces the cost.

[8]www.qcad.org (Accessed: 1st of Aug. 2011.)
[9]www.gpleda.org (Accessed: 1st of Aug. 2011.)
[10]wiring.org.co (Accessed: 1st of June 2011.)

III. Scan Matching

Scan matching is a tool enabling localization and map building. 2D scans contain range measurements in a plane at given bearings. In scan matching an alignment of a current scan is sought which maximizes the overlap with a reference scan. The result of this alignment process can be used for localization as it provides the relative pose of the laser scanner when the current scan was taken with respect to the reference scan's pose. Even though scan matching is usually applied to data from laser range finders, it will be shown that this technique can be applied to PSD scans as well.

Scan matching has been around for a few decades starting with the work of [5]. However, in this paper a simplified version of the Polar Scan Matching (PSM) [2] approach is used as it is fairly simple and quick. However, there will be simplifications made which will preclude the reliable use in general indoor environments. The simplified scan matching will still demonstrate the principles behind laser scan matching in simple controlled environments.

Without delving into much detail for PSM (for more see [2]), the scan pre-processing steps involved for the 8-bit version are the following:

1) The PSD sensors are sampled over 267 steps of the full 360 degrees revolution of the sensing head.
2) The long range and short range measurements are combined together into one scan.
3) A median filter of width 3 is applied to the measurements to remove spurious readings.
4) The 267 readings sized scan is interpolated into a 256 readings scan. Having scans with 256 elements grossly simplifies the programming effort and processor load for matching.

During scan matching the following steps are executed:

1) A virtual scan is taken from the current scan as if the scanner was placed where the reference scan was taken. This step is called scan projection.
2) The projected scan's orientation is sought by searching for a rotation which minimizes the sum of absolute range difference between the projected scan and the reference scan.
3) The projected scan rotated by the estimated orientation.
4) The position of the current scan is estimated by minimizing the sum of square range residuals between the projected and reference scan's range readings. It is assumed that range readings in the projected scan describe the same planar points as the reference scan points at the same bearings.
5) Jump to 1) unless the changes are small, or a maximum number of steps have been achieved.

To enable a reasonable run time for scan matching on an 8-bit microcontroller with a very limited resources, the following compromises were made:

- Range readings are represented with 8-bits.
- Angles are represented with 8-bits.
- The unit range is 2cm.

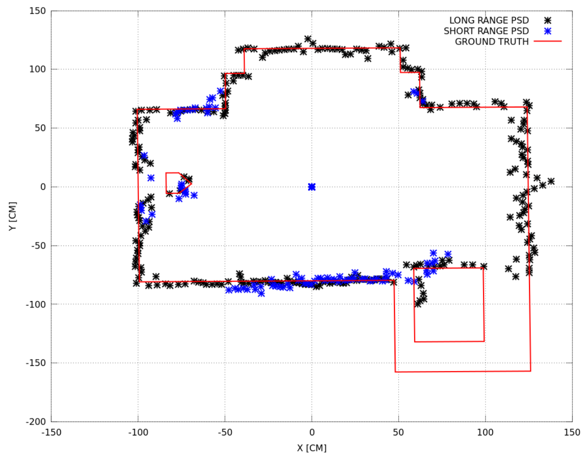

Fig. 6. A raw scan of a bathroom overlaid on the hand measured drawing (red) of the room. Points measured with the long range PSD are shown with black. Short range PSD points are shown with blue. Grid size is 50x50cm.

- Sine and Cosine look-up tables are 256 element large with 8bit resolution.
- In scan projection all readings are assumed to be good ones (bad ones are replaced with interpolated ones).

IV. Experimental Results

To show the basic capabilities of the robot, three experimental results are shown. In the first one a scan is compared to ground truth measurements, followed by a scan matching experiment. At last a series of scans are shown as collected by the moving robot. As the robot is still a work in progress, all the results are preliminary.

A. Scanning

In this experiment a raw scan (fig. 6) was taken by Mappino in a bathroom. The scan is compared to the ground truth outline of the room. The long and short range PSD voltage readings were downloaded into a PC, aligned, converted into distances and are shown separately.

Most surfaces in the bathroom were white tiles and reflected light back well. There were parts of the room which were only measured with just one of the range finders. All objects were within the maximum range of the long range PSD. The time needed to take the 360 degree scan was 4s. There were an additional 4 seconds used for determining the 0 orientation of the scanner.

The ideal surfaces resulted in a fairly good match of the scan with the ground truth.

B. Scan Matching

In this experiment on-board scan matching is demonstrated using scans taken in the same room as in the previous experiment. The robot was put down on the floor where it took a scan and stored it in its EEPROM. Then the robot moved a few tens of centimeters and took a second scan which

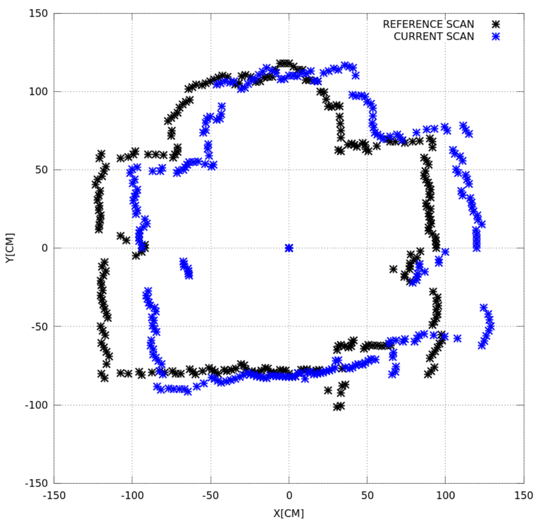

Fig. 7. Reference (black) and current (blue) scans prior matching. Grid size is 50x50cm.

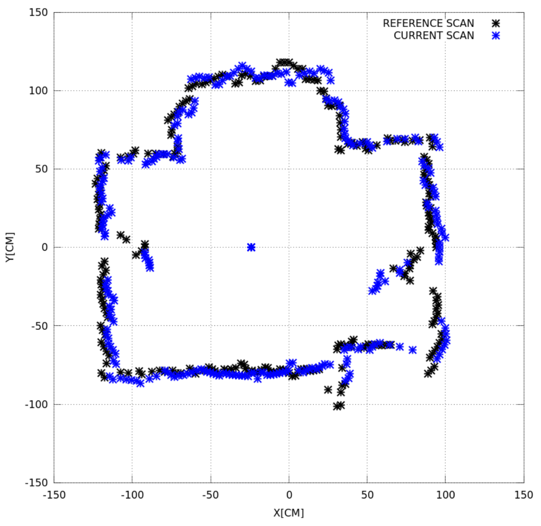

Fig. 8. Reference (black) and current (blue) scans aligned by the microcontroller performing scan matching. Grid size is 50x50cm.

was saved too. After matching these two scans, the scans and results were uploaded to a laptop through a serial line.

Scan matching was initialized (see fig. 7) with the initial pose of (0cm,0cm,0°) of the current scan. The current scan's pose converged in two steps to (-24cm,0cm,-3°) as can be seen in fig. 9. The five iterations performed took just 154ms to calculate on the microcontroller even though the compilation of the code was optimized for size and not for speed. By visually observing the resulting overlap of the scans (fig. 8) one can only see a small angular misalignment. This is not surprising as angles are quantized at 1.4° increments.

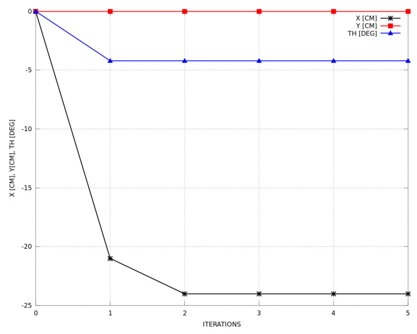

Fig. 9. The evolution of pose while matching the scan shown in fig. 7. X coordinate is shown with black, Y with red and orientation with blue. Grid size is 1x5.

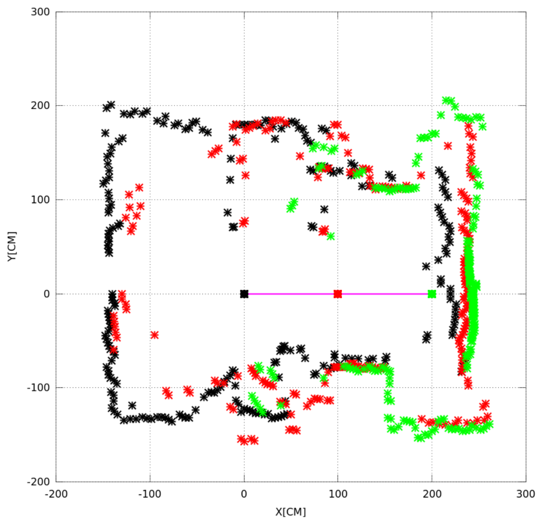

Fig. 10. Three scans collected approximately 100cm apart. Each scan is shown at the commanded robot pose with different colours. The commanded robot path is shown with magenta. Grid size is 50x50cm.

C. *Sequence of Scans*

To show what can one expect in less favorable conditions, the robot was commanded to travel 200cm in a kitchen while saving 3 scans into its EEPROM. The scans are cluttered with chair and table legs (see fig. 10 above the left half of the magenta line). Furthermore, horizontal edges as the bottom parts of radiators and cupboards which are aligned with the sensing axis of the sensors generate a lot of spurious measurements. As long range sensor is slightly tilted up to avoid getting measurements from the floor, the horizontal edges do not have to be at the same level as the sensor to influence measurements. Once the robot finished, it was hooked up to a PC and the scans were downloaded for visualization. The scans were plotted at the commanded robot poses which may differ from real poses.

V. Discussion and Classroom Use

The shown experimental results for scanning and scan matching indicate that Mappino has the potential to perform localization and mapping under laboratory conditions. The scan matching experiment has shown a surprisingly short scan matching time of 154ms on the employed 8 bit microcontroller on scans consisting of 256 range measurements.

The conservative estimates of Mappino's 20h standby and 2h of runtime battery life likely makes Mappino usable in classrooms.

However, there are limitations on the use of the robot. The internal EEPROM memory of the robot can only hold 4 scans. More memory can be hooked up through the exposed SPI port of the PSD scanner. The extra memory could also complement the 2KB RAM of the ATMEGA328 as it is barely enough for scan matching. Occupancy grid operations and path planning will likely not run at the same time as scan matching due to this memory limitation. Beside the use of external RAM, the upgrade of the ATMEGA328 is considered to a processor with 8KB of memory.

Even though there is a shortage of RAM, program memory seems abundant. In the current state of the robot only 12KB are taken up from the total of 32KB.

There are limitations on the environment as well due to the PSDs used for measuring range. These sensors do not work in all lighting conditions. One can expect interference from the sun and other light sources. Black surfaces may not reflect back enough light either for the sensor to work properly. Non-smooth surfaces may generate false measurements as well. The users therefore should start in simple, controlled environments, and as they advance their knowledge, they may be able to handle more and more difficult environments.

The stepper motors and wheel geometry enable accurate odometry, but the low motor torque, low robot clearance and low wheel traction only work well on hard, smooth and level surfaces. Beside table tops and linoleum flooring, the robot is expected to work on hard floor and tiles, but not on carpet. As the odometry of the robot is fairly accurate, scans could be taken during motion and then converted into virtual scans taken from one spot for matching purpose. This exercise, however is left for the user.

Regarding the educational value of the presented robot, there are several topics high school/university students or robotics enthusiasts can learn by using Mappino. The basic demo code provided shows the possibilities of platform and provides motivation to start working with the robot. After familiarization, they may decide to start by adding simple code for avoiding obstacles or following a person. One may continue by generating occupancy grids from a few scans. Once having an occupancy grid, path planning can provide heaps of fun. As users learn more and more about robotics, they may decide to improve odometry estimation and motion control. After grasping how scan matching works, they may get scan matching running on a moving robot as well. The modification of the provided scan matching code to work with bad readings could be also rewarding. Of course, one does not need to rely on points-based scan matching for localization. It is also possible for the users to learn about feature based localization and mapping in structured environments by extracting lines and corners from scans. Users may make hardware modifications as well. Extra electronics can be added to the prototyping areas. One can also redesign the mechanical bits of the robot and then re-use the electronics.

For younger kids the robot can be quite easily converted into a drawing robot. The PSD scanner can be replaced with a pen pointed at the floor. The motor and control electronics of the scanner can be converted into a pen lifting mechanism.

There are other possibilities regarding the robot. The programming serial cable can be replaced with a wireless link which would enable remote monitoring of the robot from a PC. The next step could involve familiarization with ROS by writing drivers for the robot and having ROS controlling the robot from a PC.

VI. Conclusion

In this paper the open source mobile robot Mappino was presented. Mappino is being developed to provide a fairly easy entry into the realms of robot localization and mapping using scan matching. Mappino achieves this goal by sporting two rotating IR range sensors as an inexpensive replacement for a laser range finder. Mappino can be programmed through the easy to use Arduino environment. The laser cut acrylic body is propelled forward by a pair of stepper motors powered by 6 AA batteries.

The shown experimental result indicates that the quality of scans taken by the robot is adequate for mapping and localization in controlled, well chosen environments. It has been shown that it is possible to perform scan matching on an 8-bit microcontroller.

Future work involves refining the mechanical design, addressing the problem of having inadequate amount of available RAM and creating more demos as proper on-the-fly localization using scan matching. The addition of a wireless communication link is also considered.

Acknowledgments

The author is very grateful for the reviewers' comments. Geoffrey Taylor's comments on the draft are gratefully acknowledged. The Taylor family's help of non-technical character during part of the development process is appreciated.

References

[1] K. Konolige, J. Augenbraun, N. Donaldson, C. Fiebig, and P. Shah, "A low-cost laser distance sensor," in *ICRA'08*, 2008, pp. 3002 – 3008.

[2] A. Diosi and L. Kleeman, "Fast laser scan matching using polar coordinates," *The International Journal of Robotics Research*, vol. 26(10), pp. 1125–1153, 2007.

[3] C. Joo and Y. Ryoo, "Self localization of mobile robot using multiple scanning infrared range system," in *ISIS'07*, 2007.

[4] Y.-H. Choi, T.-K. Lee, and S.-Y. Oh, "A line feature based slam with low grade range sensors using geometric constraints and active exploration for mobile robot," *Autonomous Robots*, vol. 24, pp. 13–27, 2008.

[5] I. J. Cox, "Blanche–an experiment in guidance and navigation of an autonomous robot vehicle," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, pp. 193–203, april 1991.