ROS-based Autonomous Navigation Wheelchair using Omnidirectional Sensor

Yassine Nasri
ESIGELEC – IRSEEM
Technopole du Madrillet,
Av. Galilée, 76801 Saint
Etienne du Rouvray
Cedex Rouen, France

Vincent Vauchey
ESIGELEC – IRSEEM
Technopole du Madrillet,
Av. Galilée, 76801 Saint
Etienne du Rouvray
Cedex Rouen, France.

Redouane Khemmar
ESIGELEC – IRSEEM
Technopole du Madrillet,
Av. Galilée, 76801 Saint
Etienne du Rouvray
Cedex Rouen, France.

Nicolas Ragot
ESIGELEC – IRSEEM
Technopole du Madrillet,
Av. Galilée, 76801 Saint
Etienne du Rouvray
Cedex Rouen, France.

Konstantinos Sirlantzis
University of Kent
School of Engineering and
Digital Arts
Jennison Building,
Canterbury, Kent CT2 7NT, UK

Jean-Yves Ertaud
ESIGELEC – IRSEEM
Technopole du Madrillet,
Av. Galilée, 76801 Saint
Etienne du Rouvray
Cedex Rouen, France.

ABSTRACT
In the medical sector, and mainly for dependent patients with movement disabilities, controlling an electric powered wheelchair could prove a challenging task. Thus, implementing an autonomous navigation algorithm for static/dynamic environments could provide an easier way to move. Within this context, this paper presents innovative work on integrating a novel method of image-based geolocalization in a powered wheelchair. The work focuses on integrating the geolocalization algorithm within the Robot Operating System (ROS) framework. Tests are being conducted using an omnidirectional camera fixed on an automated wheelchair control system. Our results show low control errors both in straight line and curved paths. The proposed algorithm was developed by the ESIGELEC laboratory.

Keywords
Image-based geolocalization, automated robotic wheelchair, omnidirectional vision sensor.

1. INTRODUCTION
Over the last decades, motion estimation of moving objects and 3D reconstruction has been highly researched worldwide. The computer vision field has been studied and great results have been obtained. Recently, along with the predominance of mobile devices having access to localization services (google street view, google map, etc.), a new topic has surfaced; the image-based geolocalization. In our work, as our wheelchair navigation system is targeting mainly the indoor environment, we propose a ROS-based autonomous navigation wheelchair running a novel version of an indoor image-based geolocalization algorithm.

Within the framework of the EU, INTERREG IVA, COALAS1 (COgnitive Assisted Living Ambient System) project, we have been working on a semi/fully autonomous wheelchair design and prototyping [1]. To deal with the different specifications of COALAS project, a multi-sensor platform was created to assist the wheelchair intelligence with the necessary environment information. Thus, Ultrasound, Infrared, Laser Range Finder, etc. have been planned for use.

Considering the complexity that image-based geolocalization algorithms usually require [2], and wide range of other sensors that our wheelchair implements, and the huge amount of data to process, several methods have been proposed. In order to have a structured and efficient handling for events on the wheelchair platform, a ROS-based architecture has been chosen.

This paper is organized as follows. In Section 2, the ROS-based wheelchair navigation system is defined in which we include the wheelchair instrumentation, the localization algorithm based on omnivision, collision avoidance algorithm and ROS integration. In Section 3, we present and discuss the results obtained from indoor testing. Section 4 concludes this work and discusses possible extensions and future work.

2. ROS-BASED WHEELCHAIR NAVIGATION USING OMNIDIRECTIONAL SENSOR

2.1. Wheelchair Instrumentation
Although the wide range of commercialized electric powered wheelchairs available in the market, there has been little consideration to providing researchers with an embedded system which is fully compatible, and communicates seamlessly with current manufacturers' wheelchair systems. For our work, communication with the wheelchair embedded system is a crucial characteristic of the model to choose. The Bora Model (Fig 1) from the Invacare Corporation is an electric powered wheelchair [3] available in the market for a number of years and the company is supporting research on its platform by providing an interfacing board to communicate with its proprietary DX-System.

Fig 1: Bora wheelchair model from Invacare Inc.

- GPSB Interface

The GPSB interface, as shown in Fig 2, is an electronic board that allows control of the wheelchair using an instruction set to drive, change directions, change driving profiles, etc., and using only a serial interface (USB) to connect with a laptop.

Fig 2: GPSB board.

- Omnidirectional Sensor

In our work, we used an omnidirectional camera fixed on the top of the wheelchair (as is shows in Fig 3); the overall height of the camera center to the ground is 1.6 meters. For the calibration, we used the calibration tool developed by ESIGELEC laboratory.

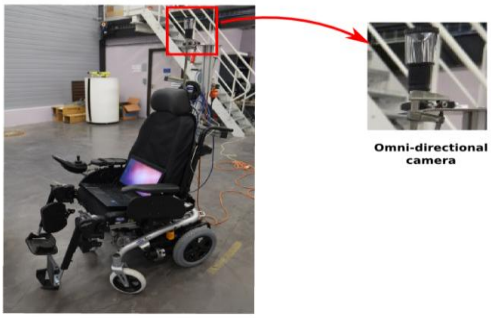

Fig 3: Testing wheelchair (Omnidirectional sensor embedded on top).

- Infrared and Ultrasound Platform

The IR/US platform implemented in the COALAS wheelchair is a pre-built system from the SYSSIAS IVA Interreg 2 Seas project. The SYSSIASs work has been integrated as is and the main contribution of our team in this is the integration of the collision avoidance algorithm developed by the School of Engineering and Digital Arts at the University of Kent to the ROS framework.

The hardware part consists of a combination of IR and US sensors on the wheelchair's outside edges, a set of 11 US sensors and 12 IRs have been setup to provide ranging information about the wheelchair surroundings and distances from other objects in almost 360 degrees (as is shows in Fig 5 and 6). Data from the sensors is collected directly by 4 Arduino Mini boards controlling 3 US and 3 IR sensors each. Measurements are then sent back to an Arduino Due board using an RJ458 serial link. The Due format the data collected from the sensors to a specific data protocol appends the joystick input and sends it to the ROS framework for collision avoidance processing. The Arduino Due board is also responsible of receiving control commands from the ROS framework and writing it on the wheelchair control bus using the GPSB interface (Fig 4).

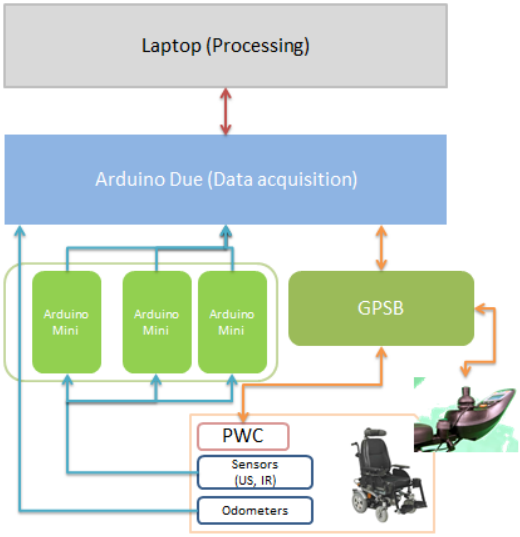

Fig 4: Collision Avoidance system following COALAS specification

The position of the sensors implemented on this platform can be shown on Fig 5 and Fig 6:

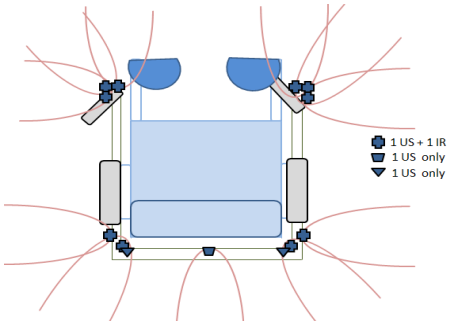

Fig 5: Ultrasound and Infrared sensors positions.

Fig 6: Ultrasound and Infrared sensors mouting on the back (left) and left front side (right) of the wheelchair.

- Multi-sensor fusion Platform

Navigation and location technologies are continually progressing, allowing ever higher accuracies and operation under ever more challenging conditions [4]. The development of such technologies requires the rapid evaluation of a large number of sensors and related utilization strategies [5]. For autonomous and assisted navigation systems, a wide range of sensors could be used: Laser range finders, Kinect RGB-D cameras, Ultrasound sensors, Infrared sensors, etc. These sensors allow more accuracy and provide rich information about the environment to the robot (wheelchair), thus a better

understanding of the surroundings and better algorithm design.

In our case, to comply with COALAS project specifications, several sensors are used. The following list shows the most needed ones with their respective use:

- Omnidirectional camera: Autonomous navigation
- Kinect : Human fall detection
- Odometers: wheelchair movements tracking
- Camera and Microphone: Head and speech control of the wheelchair

All data coming from the above sensors should be integrated and processed seamlessly. A simple example of that is controlling the wheelchair by tracking the head movements of the user while running the collision avoidance algorithm to avoid obstacles on the way, or running the autonomous navigation system based on the omnidirectional camera while running the fall detection algorithm in parallel to detect a possible fall of a patient.

2.2.Vision-based Localization Algorithm

The Motion estimation from sets of images is a difficult task and requires multiple steps to achieve good results. The image-based geolocalization algorithm implemented in this work consists of two separate and complementing phases. The first phase is to store all the images, at a certain frequency, of the possible paths that the robot could pass through, thus creating a local or remote database, while the second phase is running the navigation algorithm using the images database created before.

For this work, an omnidirectional camera was used to create panoramic images. Once all the images representing our ground truth were acquired, we define the mathematical constraints to analyze them.

The proposed solution consists of three steps:

- **Camera calibration**: to get good results for our PIBG algorithm, we need to know how our sensor is set. This process is about finding the intrinsic parameters (focal length, pixel skew, principal point).
- **Features detection and correspondence**: As our algorithm relies on features detection, an essential step of this work is to choose our features extractor. Many solutions have been proposed in literature but we used Scale-Invariant Feature Transform (SIFT) [6].
- **Essential matrix and 3D triangulation**: The epipolar geometry (as shown in Fig 7) gives the relative position of one vision sensor with respect to another. Here we estimate from point clouds the essential matrix which completely describes the rigid geometric relationship between corresponding points of a stereo pair of cameras. The essential matrix is then decomposed – Singular Value Decomposition (SVD) – into two rotations matrices R1, R2 and two translation matrices t1, t2. In order to find the best combination out of the four possibilities, we must triangulate. This process consists of determining the three-dimensional world coordinates for an object given two dimensional views. As the intersection of the two vectors is almost impossible in the 3D world, we use an estimation to detect the closest point. As our navigation algorithm relies on a spherical camera model, after acquiring the 2D images, each pixel of the panoramic image 2D point (x, y /RGB) –is projected (mapped) to a 3D point on the unit sphere. For all the pixels we compute a spherical point cloud and therefore define the camera model. This model consists of having a projected image on a surface of a unit sphere centered by the camera.

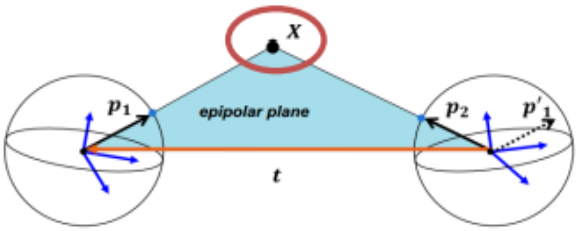

Fig 7: Position estimation using epipolar geometry and triangulation.

2.3. Collision Avoidance Algorithm

The collision avoidance algorithm is a part of the University of Kent SYSSIAS project. The algorithm consists of fusing data coming from the US/IR platform to detect and avoid obstacles along the wheelchair's way. This algorithm can be decomposed into three main steps; collecting data from the sensors using four separate Arduino mini boards, getting the user inputs by reading the joystick positions using the GPSB board, and finally, applying an artificial potential fields-based algorithm by an Arduino Due, and writing data back in the wheelchair control bus.

Our contribution is that in this work the corresponding collision avoidance algorithm is implemented within the ROS framework. For this purpose, the Arduino Due software is changed to collect data coming from the sensors and joystick data, and transfer it to the ROS system using a serial link with a PC. Also, some updates to the Arduino Minis code were necessary in order to comply with the COALAS specification (ROS framework) and allowing more control of the system. Fig 4 summarizes the COALAS Collision avoidance platform.

2.4. ROS Integration

1) ROS Architecture

ROS is a Linux based software framework for operating robots. This framework uses the concept of packages, nodes, topics, messages and services. A node is a piece of software responsible for a very specific task, its output may be redirected to another node's input. The information which moves from node to node is called a message. Messages always travel anonymously via special ports called topics. A node which sends messages on a topic is called a publisher and the receiving node has to subscribe to the topic to receive that message, hence it is called a subscriber. All related nodes are combined in one package that can easily be compiled and ported to other computers. The packages are necessary to build a complete ROS-based autonomous robot control system.

Also, ROS architecture allows a more effective distributed inter-process/inter-machine communication and configuration using a language independent basis (C++, python, lisp, java, and more), thus, easier hardware abstraction and code reuse.

2) ROS Simulation Environment

Robot simulation (Fig 8) is an essential tool in every robotics' toolbox. A well-designed simulator makes it possible to rapidly test algorithms, design robots, and perform regression testing using realistic scenarios.

In order to test ROS scripts before uploading them to robots, ROS offers the use of two main simulators; RVIZ and Gazebo. While RVIZ is used mainly for its simplicity and for simple 3D environments, Gazebo offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. At your fingertips is a robust physics engine, high-quality graphics, and convenient programming and graphical interfaces.

In this work, as no wheelchair model has been already developed under the gazebo library, we had to create it. For creating new models for gazebo, SDF (Simulator Description Format), a new description format have been released to replace the old URDF (Unified Robot Description Format) description language. To implement the sensors on the modeled wheelchair, a simple way is too use the mesh description of each one and include it in the SDF model.

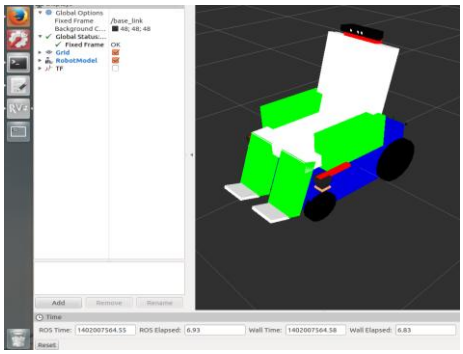

Fig 8: Simulated wheelchair model in RVIZ.

3) PIBG ROS Integration

a) PIBG ROS package dependencies

Software in ROS is organized in packages. A package might contain ROS nodes, a ROS-independent library, a dataset, configuration files, a third-party piece of software, or anything else that logically constitutes a useful module. The goal of these packages it to provide this useful functionality in an easy-to-consume manner so that software can be easily reused.

For the PIBG integration, an independent ROS package was implemented using the Catkin building tool. In this package, we created our PIBG nodes, topics, messages, etc.

The package dependencies are the core packages that are used by the PIBG package. For this work, we can cite among others:

- Vision_Opencv2: provides packaging of the popular OpenCV library for ROS. Also it implements the CV_bridge (bridge between ROS messages and OpenCV) and Image_geometry (Collection of methods for dealing with image and pixel geometry)
- Roscpp: a C++ implementation of ROS. It provides a client library that enables C++ programmers to quickly interface with ROS Topics, Services, and Parameters. Roscpp is the most widely used ROS client library and is designed to be the high-performance library for ROS.
- Pcl [7]: for point cloud processing – development. The PCL framework contains numerous state-of-the art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation.
- Image Transport: should always be used to subscribe to and publish images. It provides transparent support for transporting images in low-bandwidth compressed formats. Examples (provided by separate plugin packages) include JPEG/PNG compression and Theora streaming video.
- Other libraries Creation: the key power of the Catkin (Fig 6) build tool is how it makes it easier to build modular software without having to keep track of the specific build products of each package. Modularity, in this case, comes in the form of building specific functionality into libraries which can be used by other packages.

For this work, two pre-developed codes were used; the navigation algorithm code that implements the three main steps, calibrating the camera, features detection/ correspondence. For the essential matrix and 3D triangulation step, another Omnivision code was developed in the lab during a previous work. The two codes were developed using C++ language and in order to be able to use them in our work, we created a separate library for each code. Thus, we can use them in different, independent ROS packages, respecting the modularity of the ROS framework. Fig 9 presents the creation of these libraries.

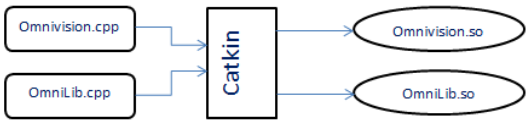

Fig 9: Catkin easy-way to create shared libraries.

b) PIBG Nodes

For implementing the PIBG algorithm and to comply with the ROS specification, we had to break up the algorithm logic into nodes. A node is a process that performs computation. Nodes are combined together into a graph and communicate with one another using streaming topics, RPC services, and the Parameter Server. These nodes are meant to operate at a fine-grained scale; a robot control system will usually comprise many nodes. In this work, the following ones have been employed:

- Omnivision_Server

This node implements the PIBG processing logic; it takes two successive images and calculates the relative distance and rotation between them. The output is a combined Translation/rotation vector.

- Omnivision_Publisher

This node is responsible of publishing the images from the image database to the subscribing image. This publish/subscribe is achieved using the OpenCV package to allow the transformation Image <-> ROS message as required by the ROS specification.

- Omnivision_Launcher

This node is the starter of the PIBG algorithm, loading the first image to the Omnivision_node and thus launching the image request calls.

- PIBG ROS chain

After explaining above all the components related to the PIBG implementation in ROS, the diagram in Fig 10 presents the whole chain of the PIBG process:

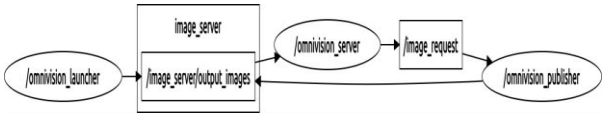

Fig 10: PIBG ROS Chain.

The /image server topic is the ROS topic where all images should be sent as ROS messages. First, the launcher sends the two first images to the server. After finishing its processing, the server sends a ROS message under the /image_request to the publisher that sends back in response the next image in the database or signals the end of processing if the path end is reached.

4) Collision Avoidance ROS package

Unlike the PIBG algorithm (Fig 11), the collision avoidance doesn't need any particular dependencies, the processing here doesn't in fact include any particular data type or specific exchange protocol, thus, no more than the Roscpp and the Rospy dependencies were added.

The ROS chain created consists of three nodes; the first node is responsible of handling the serial link between the Arduino Due and the laptop, it reads data (sensors measurements and joystick inputs) incoming from the Due board and writes data (control commands) on the bus to the Due for execution. The second node subscribes to the /raw_data topic, format it to sensors measurements and joystick inputs to publish under the /sensor_data and /joy_cmd topics respectively. The third node is the main part of our algorithm, it subscribes to both previous topics, applies the collision avoidance algorithm and finally publishes the control commands under the /cmd_vel topic. These messages are redirected under ROS to the first node to be written on the serial bus for execution.

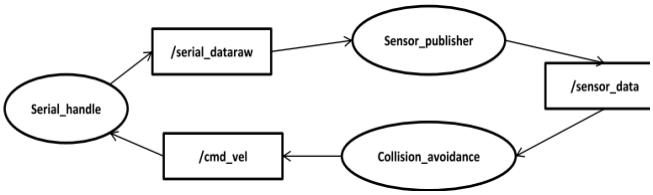

Fig 11: Collision Avoidance ROS chain.

Also, under ROS, it is possible to integrate QT interfaces. For analysis and visualization purposes, we developed a simple QT interface (

Fig *12*) to show data incoming from the US/IR platform, so giving us a feedback of the wheelchair surroundings.

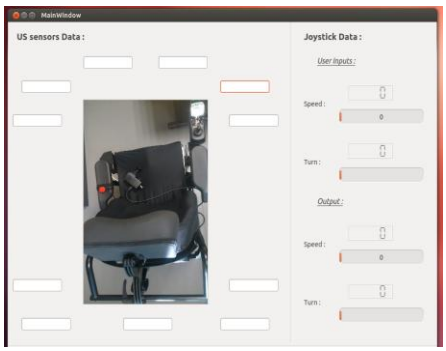

Fig 12: QT testing and vizualisation interface.

3. RESULTS & ANALYSIS

As described before, the PIBG algorithm consists of two separate phases: a) creating path images database and, b) applying the algorithm. For the first part, we used RTMaps software to get data from the camera at a frequency of 0.5 Hz and create our path database. In addition, the covered path and positions of the wheelchair have been recorded using the Vicon system (3D localization system based on NIR cameras and time of flight calculation). For the second part of the test, we run the PIBG algorithm on a laptop carried by the wheelchair and also record the driven path.

In order to have reliable ground distance results, the tests were conducted on a constant speed. So, after obtaining the Translation/Rotation vector, the next relative position of the wheelchair can be deducted by a simple multiplication of its values by the distance scale.

3.1.Tests and Results

For software integration, we used Linux 12.04 (LTS version) and ROS Groovy and the wheelchair control (ROS compliant) package was employed. Finally,, to convert the Translation/Rotation vector obtained from the PIBG package to a real world pose, we implemented an intermediate node called "vect_to_pose". The resulting test nodes chain can be seen in Fig 13:

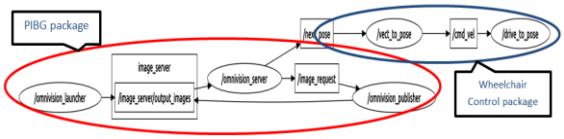

Fig 13 : Interface diagram of PIBG and wheelchair control ROS packages.

To test the PIBG algorithm in indoor environments, we implemented two different tests; the first consists of driving the wheelchair in a straight line and the second a Z form. The tests were carried in the ESIGELEC LNA lab and we used the Vicon system to record both driven paths positions. After that, and using the QtiPlot utility in Linux, we draw the Real/Driven paths of the wheelchair. Fig 14, 15, 16 and 17 present the results obtained:

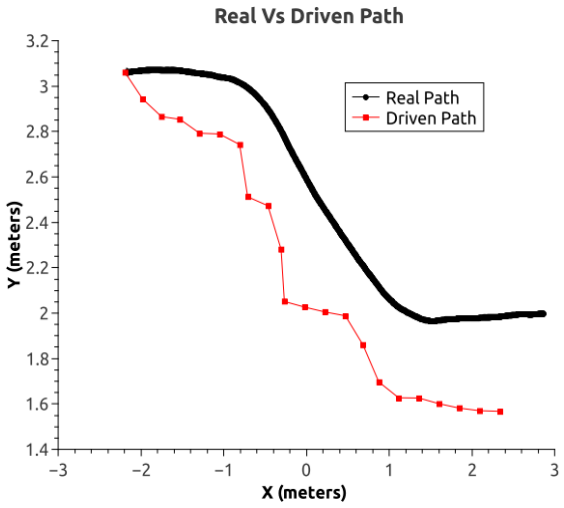

Fig 14: Real Vs Driven results for a Z form path.

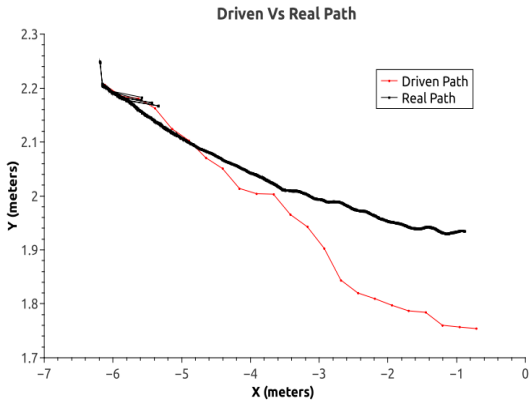

Fig 15: Real Vs Driven results for a straight forward path.

The error estimation was calculated as the average of errors between the ground truth positions and the ones resulting from our test runs. This error shows that for a straight line path, the average error between the ground truth position and the calculated one is only -0.18 meters in the X-axis and +0.2 meters on the Y-axis , while it is up to -0.36 in the X-axis and -0.6 meters on the Y-axis when the path is Z-like. This shows that the complexity of the path is affecting directly the algorithm's accuracy.

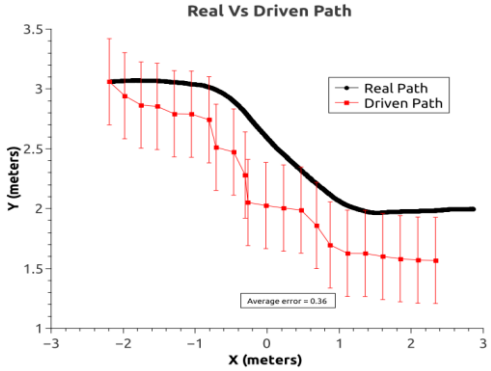

Fig 16: Real Vs Driven results for a Z form path + error intervals.

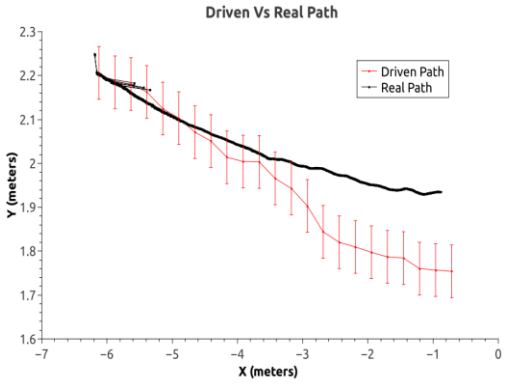

Fig 17: Real Vs Driven results for a straight forward path + error intervals.

4. CONCLUSION

Today, the world count between 110 million and 190 million adults who have significant difficulties in walking [8] and billions have been spent in the wheelchair industry. However, wheelchair automation is still a new concept. In this article, we present a novel model of automated wheelchairs. This model based on the ROS framework allows a new vision towards this potential future industry. In order to test the effectiveness of the ROS framework on the wheelchair, we implemented an autonomous navigation system based on an image-based geolocalization algorithm. Tests have been conducted using an omnidirectional sensor in an indoor environment. In the future, an optimization of the proposed solution will be done to enable the processing of larger amount of data and optimal path choice.

5. ACKNOWLEDGMENTS

This research is supported by the Franco-British research project Cognitive Assisted Living Ambient System (COALAS, Project Nr. 4194, URL: http://coalas-project.eu/), that has been selected in the context of the INTERREG IVA, France (Channel) England, European cross-border co-operation program.

The authors wish to thank the Technological Resource Center (TRC) of ESIGELEC/IRSEEM for his help in the testing phase. We also thank the Autonomous Navigation Laboratory (ANL) staff for the significant contribution during the different tests.

6. REFERENCES

[1] N. Ragot, and G. Caron, and M. Sakel, and K. Sirlantzis, "A EU multidisciplinary research project for assistive robotics neuro-rehabilitation", IEEE/RSJ International Conference on Intelligent Robots (IROS), Workshop on Rehabilitation and Assistive Robotics: Bridging the Gap Between Clinicians and Roboticists, Chicago, USA, September 12-18, 2014.

[2] F. Dellaert, and W. Burgard, and D. Fox, and S. Thrun, "Using the condensation algorithm for robust, vision-based mobile robot localization". In Proceedings 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, June 23-25, 1999.

[3] Invacare® Bora User manual REV1.6.

[4] J. Llinas, and D. L. Hall, "An introduction to multi-sensor data fusion". ISCAS '98, In Proceedings of the 1998 IEEE International Symposium on Circuits and Systems. Vol. 6, pp. 537-540, May 31-June 3, 1998.

[5] W. Meeussen, and M. Wise, and S. Glaser, and S. Chitta, and C. McGann, and P. Mihelich, and E. Marder-Eppstein, and E. Muja, and V. Eruhimov, and T. Foote, and J. Hsu, and R.B. Rusu, and B. Marthi, and G. Bradski, and K. Konolige, and B. Gerkey, and E. Berger, "Autonomous door opening and plugging in with a personal robot", In Proceedings of 2010 IEEE International Conference on Robotics and Automation (ICRA), pp.729-736, May 3-7, 2010.

[6] A.C. Murillo, and J.J. Guerrero, and C. Sagues, "SURF features for efficient robot localization with omnidirectional images", In Proceedings of 2007 IEEE International Conference on Robotics and Automation, pp. 3901-3907, April 10-14, 2007.

[7] R.B. Rusu, and S. Cousins, "3d is here: Point cloud library (PCL)". In 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 1-4, 2011.

[8] World Health Organization Report, Fact Sheet N°352, September 2013.