Applied Artificial Intelligence

An International Journal

ISSN: 0883-9514 (Print) 1087-6545 (Online) Journal homepage: https://www.tandfonline.com/loi/uaai20

Mobile Robot Navigation Using an Object Recognition Software with RGBD Images and the YOLO Algorithm

Douglas Henke Dos Reis, Daniel Welfer, Marco Antonio De Souza Leite Cuadros & Daniel Fernando Tello Gamarra

To cite this article: Douglas Henke Dos Reis, Daniel Welfer, Marco Antonio De Souza Leite Cuadros & Daniel Fernando Tello Gamarra (2019) Mobile Robot Navigation Using an Object Recognition Software with RGBD Images and the YOLO Algorithm, Applied Artificial Intelligence, 33:14, 1290-1305, DOI: 10.1080/08839514.2019.1684778

To link to this article: https://doi.org/10.1080/08839514.2019.1684778

Published online: 01 Nov 2019.

Submit your article to this journal

Article views: 3234

View related articles

View Crossmark data

Citing articles: 23 View citing articles

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=uaai20

APPLIED ARTIFICIAL INTELLIGENCE
2019, VOL. 33, NO. 14, 1290–1305
https://doi.org/10.1080/08839514.2019.1684778

Taylor & Francis
Taylor & Francis Group

Check for updates

Mobile Robot Navigation Using an Object Recognition Software with RGBD Images and the YOLO Algorithm

Douglas Henke Dos Reis[a], Daniel Welfer[b], Marco Antonio De Souza Leite Cuadros[c], and Daniel Fernando Tello Gamarra[a]

[a]Control and Automation Engineering Course, Federal University of Santa Maria, Santa Maria (UFSM), Santa Maria, Rio Grande do Sul, Brazil; [b]Department of Applied Computing, Federal University of Santa Maria, Santa Maria (UFSM), Brazil; [c]Federal Institute of Espirito Santo (IFES), Professional Master Program in Control and Automation Engineering, Vitoria, Espirito Santo, Brazil

ABSTRACT

This work presents a vision system based on the YOLO algorithm to identify static objects that could be obstacles in the path of a mobile robot. In order to identify the objects and its distances, a Microsoft Kinect sensor was used. In addition, a Nvidia Jetson TX2 GPU was used to increase the image processing algorithm performance. Our experimental results indicate that the YOLO network has detected all the predefined obstacles for which it has been trained with good reliability and the calculus of the distance using the depth information returned by the Microsoft Kinect camera had an error below of 3,64%.

Introduction

Humans have a surprising ability for pattern recognition and object identification, in order to mimic this ability different algorithms have been proposed based on deep learning (Guo et al. 2016). There are a lot of algorithms for computational vision based on convolutional neural networks (Data Science Academy Team, Deep Learning Book), but one of the most advanced techniques proposed to classify objects in real time is the You Only Look Once (YOLO) (Redmon et al. 2016) algorithm. Also, the Microsoft Kinect is a depth sensor that captures RGBD images, initially it was designed for video games, but its low cost has opened new ways and possibilities of its use in robotics. Considering the evolutions in the technology currently the number of applications that uses mobile robots is increasing (Bezerra 2004).

The work described in this article aims to integrate the Microsoft Kinect with the YOLO algorithm, so the proposed system can locate and classify the objects information (i.e. obstacles) in the images returned by the sensor. After recognizing the objects, it is intended to identify the object's distance from the robot, so it could be possible to create a control algorithm for the navigation of a mobile robot.

CONTACT Daniel Fernando Tello Gamarra daniel.gamarra@ufsm.br Control and Automation Engineering Course, Federal University of Santa Maria (UFSM), Santa Maria, Rio Grande do Sul, Brazil

Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/uaai.

© 2019 Taylor & Francis

Related Work

In (Moura 2014) anthropometric data were obtained using the Microsoft Kinect sensor and its application for detecting the dimensions of the human body which are evaluated along with depth information. The idea is a solution for physiotherapists in order to facilitate the identification of posture-related problems. The mathematical models applied obtained satisfactory results, however much of the depth information returned by the Microsoft Kinect could have some precision error (Moura 2014). It is not necessary to know the size of the object, the work focuses on the recognition of posture problems, such as the height difference of two shoulders to facilitate the work of professionals of the area.

A control system for the visual tracking of a robot using Microsoft Kinect and MATLAB shows good results in the identification of the chosen target in the image and in calculating the distance with the information returned from the Microsoft Kinect is described in Pegas work (Pegas 2014).

There are other applications for the Microsoft Kinect sensor in robotics. The use of the Microsoft Kinect sensor to detect the movements in order to control the Khepera III robot (Wang et al. 2013). In the work of (Fadli, Machbub, and Hidayat 2017), there are some examples of the robotic applications with Microsoft Kinect.

The YOLO algorithm is a state of art algorithm in real-time object recognition. We can see some applications of YOLO like in the system of the image recognition using YOLOv2 (Redmon and Farhadi, 2017) for robotic platforms like in Tenguria's work (Tenguria et al. 2017). Two Raspberry Pi microprocessors were used for the tests. Neural network tests were performed with on board processing, but the result was very poor. A test was done on CPU and GPU. Using GPU (both Nvidia GTX 750 Ti and Nvidia GTX 860M) the processing time obtained was less than 0.5 s per image and on the CPU (Intel i7) a processing time of approximately 9.45 s per image. This work demonstrates that a solution using hardware without good processing power and, especially, that is not equipped with a graphic card is not attractive.

An application that only relies on depth information is shown in Lucian's work (Lucian et al. 2018). The depth information of a pair of legs is returned by the Microsoft Kinect to form an image using YOLOv2. The goal is to propose a solution to problems in areas of low to medium traffic of people in real-time applications. Very satisfactory results were obtained both for the execution of YOLOv2 in Nvidia Jetson TX2 and for the detection efficiency of YOLOv2 (Lucian et al. 2018).

Bersan proposes an experiment that involves the elaboration of a map of the environment where the robot is moving along with the detection of the position of the objects previously trained to be detected by the neural network (Bersan et al. 2018). A 2D laser sensor, odometers and an RGBD camera were used with the

YOLO algorithm for the detection of objects. Also, a camera with a depth sensor more powerful than the Microsoft Kinect was used.

YOLO was also used for a NAO humanoid robot for object recognition and tracking. Some environment tests showed that the neural network really helped the robot in real-time object recognition and tracking (Zhou et al. 2018). Another use of YOLO is in a service robot application that needs to recognize the tennis ball in order to plan the navigation path to collect it (Gu et al. 2018). YOLO showed a precise object recognition of a tennis ball in real time.

A more complex YOLO application can be found for object classification in order to establish correlation between the objects and the person based on the distance between them (Zapf et al. 2018). For example, if YOLO detects a cup of coffee and a person, and both are close to each other, then the program would determine that there is a person drinking coffee in the image. YOLO performed its task perfectly. Another application is the use of YOLO to detect and classify the furniture and the household objects in order to do the mapping and estimate the localization of the robot on the map (Maolanon et al. 2019). In this application is used YOLOv3 and the SLAM algorithm, both are combined in a ROS application that does the mapping and object detection simultaneously.

There is another work where is created a modified version of YOLO algorithm aiming a better efficiency in 3D object recognition (Zhao, Jia, and Ni 2018). One of the modifications is that the bounding box becomes the cluster box. The main function of the cluster box is to encompass all the object inside its borders. The M-YOLO (as is called the modified YOLO) do not use depth information to build and recognize the 3D object, the recognition and the training is still based in 2D images, and the cluster box in 2D coordinates are mapped and transformed to 3D coordinates. The Zhao's work shows another application for the YOLO algorithm that can be turned into a powerful 3D object recognition algorithm.

In this paper, we propose a new navigation system based on YOLOv2 and on the Microsoft Kinect sensor in order to recognize objects and calculate its distances to the robot in order to help a mobile robot to accomplish a navigation task. Our paper is focused on object recognition and the distance calculus for a robot navigation task different from other works that have previously used YOLOv2. Also, our work is intended to be applied in a reactive navigation task different from a planning navigation approach in robotics that tries to construct a map.

Materials

Hardware Configuration

For the initial tests, it was first used a notebook, a Microsoft Kinect, a television and a modem. Through the notebook, a connection is made to the card through

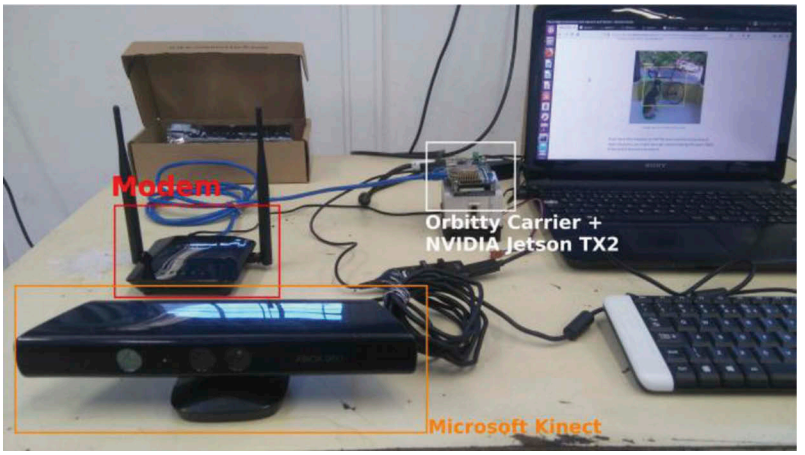

Figure 1. The hardware used in our first tests.

the network and through it, the commands and programs are executed on the card. The output images of the programs appear on the television and the terminal outputs appear in the notebook. The setup that was used for the initial tests is shown in Figure 1.

Nvidia Jetson TX2

Companies have been investing in smaller and more efficient computers, such as Raspberry Pi, which is widely used and was one of the pioneers in this environment. The Nvidia Jetson TX2 (Figure 2) comes from the same concept of having a small computer to facilitate practical applications, but it has a differential because it is directed to applications that involve artificial intelligence.

Nvidia Jetson TX2 is a reminiscent of Raspberry Pi, but its processing power is even greater because it has a very powerful graphics card onboard. With this board, Nvidia aims to be able to process all the data on the board itself without needing to use external resources.

There are no connectors on the Nvidia Jetson TX2 as it does on Raspberry Pi. In order to be able to use the board, it was possible to acquire the Orbitty Carrier card from Connect Tech Inc. (Figure 3).

Mobile Robot (POTTER)

For application of the current work was used the mobile robot named Potter. It is a mobile robot built by the research group GARRA (Automation and Applied Robotics Group) of the UFSM (Federal University of Santa Maria) (Valente et al. 2017).

It is equipped with two direct current motors with incremental encoders in them, so that it is possible to read the wheel position and with this information

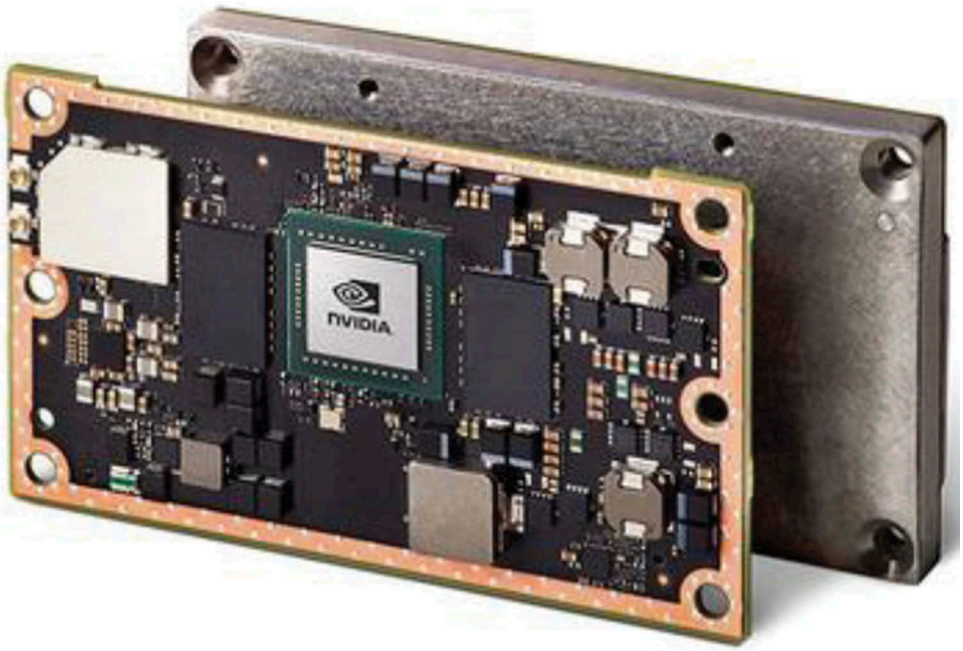

Figure 2. Nvidia Jetson TX2.

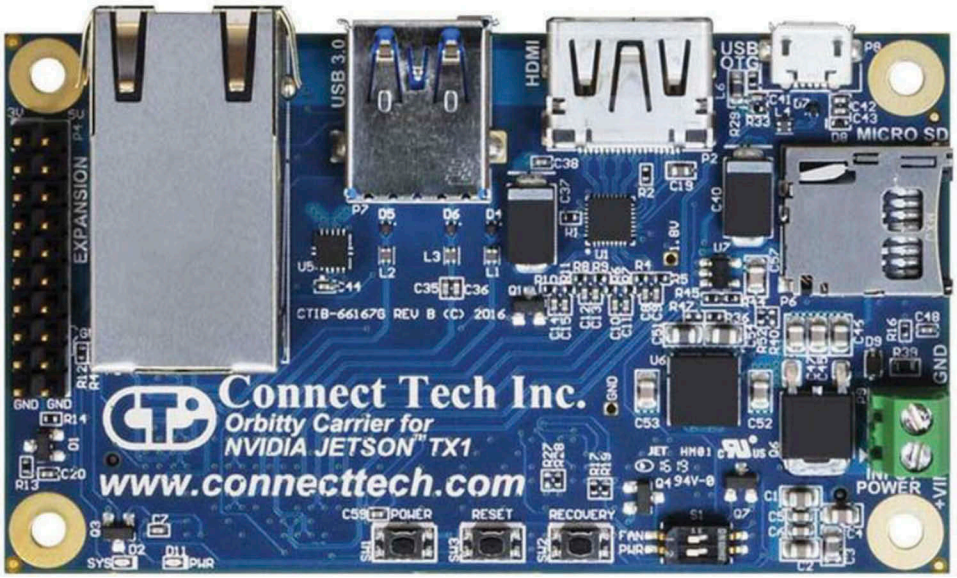

Figure 3. Orbitty carrier board.

calculate de distance traveled by the robot. A MD49 driver does the control and protection of the motors. It also has mounted on its frame ultrasonic sensors, a laser sensor and a Microsoft Kinect. The device that reads and controls all the devices in the robot is an Arduino Mega 2560 that performs the low-level tasks.

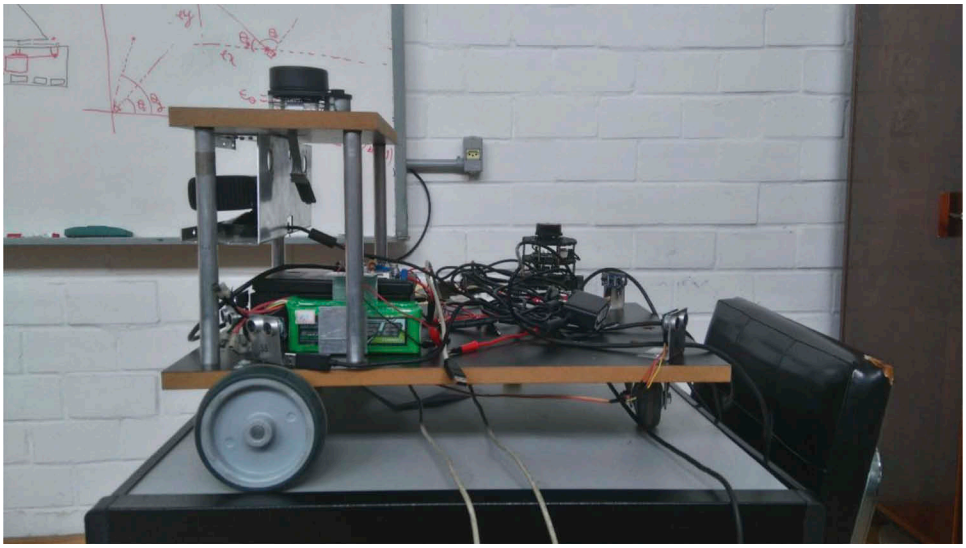

Figure 4. Mobile robot potter side view.

Currently, some additional equipment has been added such as the laser sensor and the batteries. Its navigation is realized by the two motors that realize a front traction in the robot. On its back, there is a wheel that is free to rotate in any direction. Figure 4 is a picture of the Potter mobile robot.

Software Architecture

For the execution of the work some tools and libraries were used to develop the software. The following is an explanation of the main tools and libraries used. Darknet is a neural network and YOLO is its optimized subdivision for real-time object detection. It is programmed in C language and CUDA, thus having support for CPU and Nvidia GPU. The YOLO that is used in this project is one of the branches of darknet. It was used YOLOv2.

PyDarknet is a wrapper for python language. It is imported as a library, making it possible to run YOLO in a script written in python. Its original name is "yolo34py" which means "YOLO 3 for python". It has this name because it was originally designed to run YOLO version 3 (YOLOv3), but it supports earlier versions. OpenCV is a multi-platform library, totally free for use (i.e. both academic and commercial), elaborated in C/C++ programming language that was originally developed for applications involving computer vision.

Freenect is an OpenKinect project that aims to connect Microsoft Kinect with computers so that it is possible to access the information returned from the equipment in various operating systems. The initial design for Microsoft Kinect was the use of body movements in video games on the Xbox 360 console. With the evolution of research and the increase need for embedded

ships that had certain attributes Microsoft Kinect was discovered as a good option by the existing equipment. Google Images Download is a python program in which you can do image search and download. With the help of this program a python script was created to download the necessary images for the database.

Methodology

Image Preprocessing

The first step is to prepare a database where it is necessary to obtain information, derived from the images. For this, a script was written in the python language where the Google search tool which retrieves a chosen number of images. In this step, a library that emulates the Google Chrome browser and then searches for the images was used.

For the organization of the images, after they have been downloaded, a mapping in the source folder is done by all the folders that contain in its name the name of the chosen label, and after recognizing all the folders it relocates the images contained in the folder, while renaming them sequentially. After the images are organized, we then created two more scripts to facilitate the treatment of the information (i.e. images) obtained. In the script are opened the images and the user has the possibility to select in the image the objects to be detected and if you do not have the desired object in the image or the image has a bad quality you can click the letter "X" to delete it. After selecting the objects in the image, click on the letter "Z", predefined by programming, to save the information of the object location in a text file and forward the next image. The program follows the flowchart logic of Figure 5.

The generated text files contain information on the number of objects that are in the image, and which are the upper left and lower right points of the rectangle that encompasses each object. After the text files are generated, a script must be run that converts the data contained in the files to the format that YOLO understands for training. The entire preparation of the database is summarized to the steps shown in the flowchart of Figure 6.

Distance Calculation

It was obtained the resolution of the sensor which is 2^{11} bits (2048 different values), the information transmitted by the Microsoft Kinect varies from 0 to 2047, in decimal values (OpenKinect website). Position 2047 is responsible for the warning that there was an error in the reading, which means that the object is outside the precision limits (very close or very far).

Another important information obtained on the site was regarding how to handle those bits that are transmitted by the sensor. There are two formulas in this

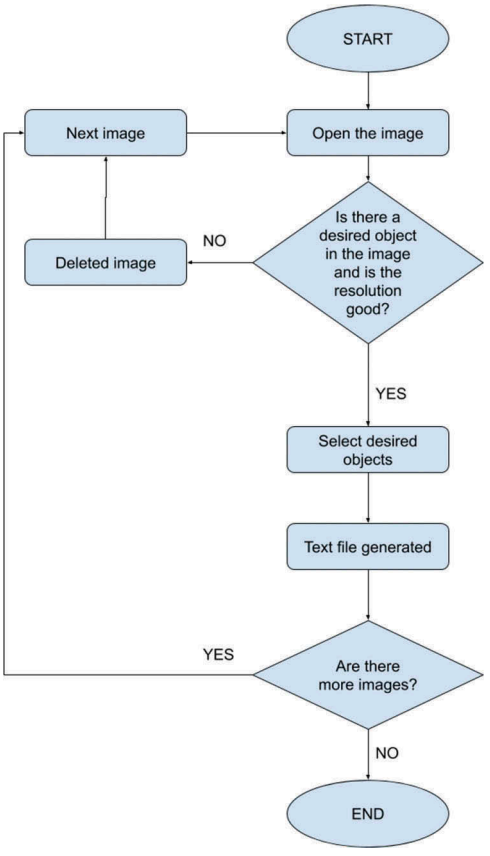

Figure 5. Image organizer flowchart.

section of the site that have been developed by researchers to handle the data. One of the formulas returns the approximate value in meters and another in centimeters. For the current project, it was found more interesting to work in centimeters to obtain a better accuracy of distance. The formula used is shown below.

$$\text{Distance} = \frac{100}{((-0.00307 * \text{Binary value in decimal} * 3.33))} \tag{1}$$

But this formula has a limitation associated with it, it only guarantees better precision between 10 cm of distance and 4 m of distance. In order to perform the first tests of the depth sensor, it was also necessary to discover how it was transmitted this data. For this it was necessary to do some experimental tests and, finally, it was discovered that the data is transmitted in a matrix with dimensions of 640 × 480. In each of the cells (i.e. which can be interpreted as a pixel) the transmitted matrix has a distance value of 2^{11} bits.

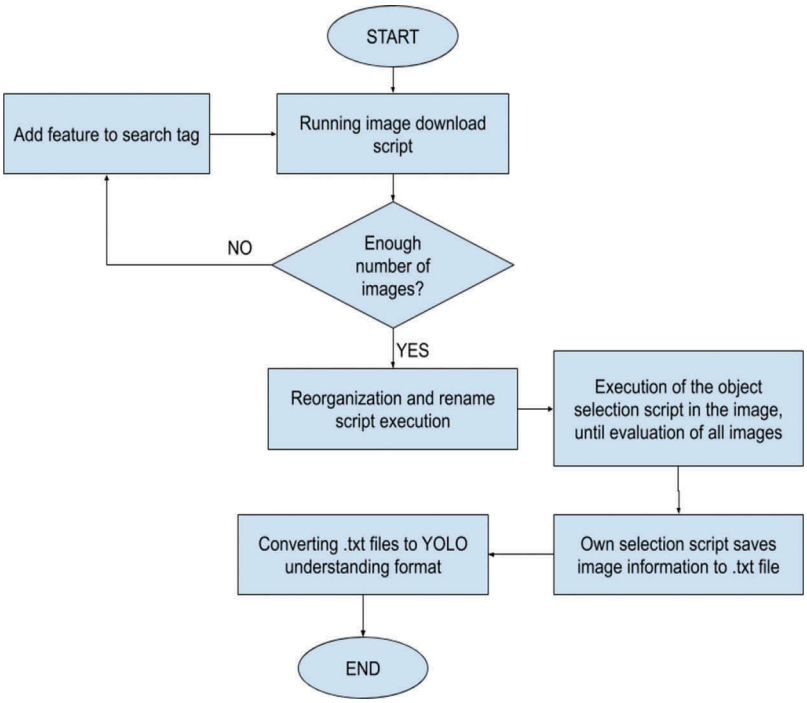

Figure 6. Database preparation flowchart.

Results

Training

In order to start the training, a machine has been created in the Google Cloud that enables the processing in the cloud. YOLO training lasted for about three days. Training was stopped when we reached an average error of 0.345359. For the training, 270 images were used for a common chair, 215 images for an office chair (with wheels), 390 images for a box and 244 images for a table in the databases. The graphic that results from the training is shown in Figure 7 that shows the results of the training process, in the X-axis we have the number of steps and in the Y-axis the error of each step.

Figure 8 shows the objects that were identified by the YOLO algorithm, in this case, the defined objects were a common chair, a wheeled chair, a box and a table. It can be observed in Figure 8 and Table 1 that the results of the application of the neural network with the weights generated in the training were satisfactory, except for the "Table" class which had a lower reliability than expected (i.e. obtained a reliability of 69%). However, it can be said that the validation of weights occurred successfully.

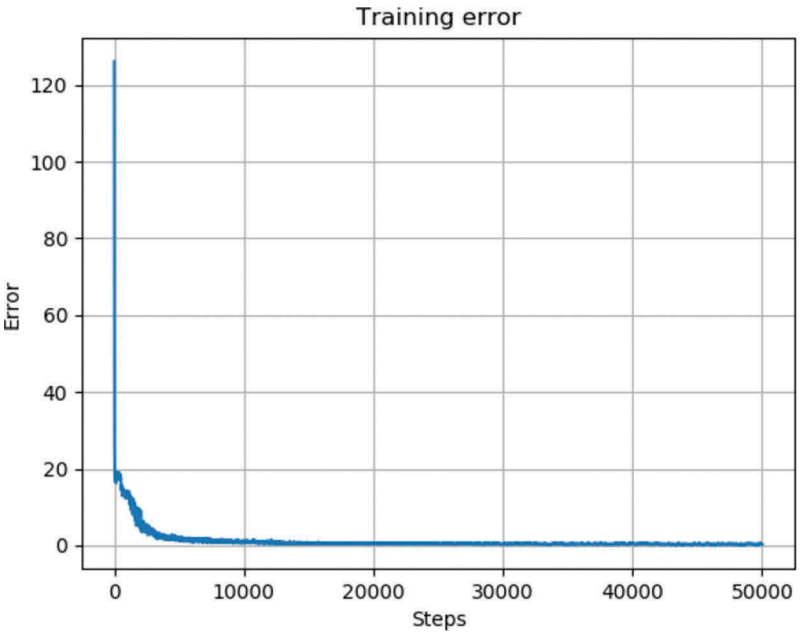

Figure 7. Training loss function.

Figure 8. Applying YOLOv2 to an image to validate training.

Distance Calculation Using Microsoft Kinect

The algorithm uses Microsoft Kinect images as entries, the refereed images are processed with a Nvidia Jetson TX2 GPU. The error between the calculated distance and the actual distance reached a very low maximum, reaching a maximum of 3.64%. In Figure 9 is possible to see a graphic with information of the real distance value (i.e. blue line) and the distance value that was calculated (i.e. red line). Table 2 shows some of the returned distance values with the Kinect sensor usingEquation (1).

Table 1. Reliability returned from detection.

Class	Reliability
Table	69%
Common Chair	85%
Office (wheeled) chair	74%
Box	82%

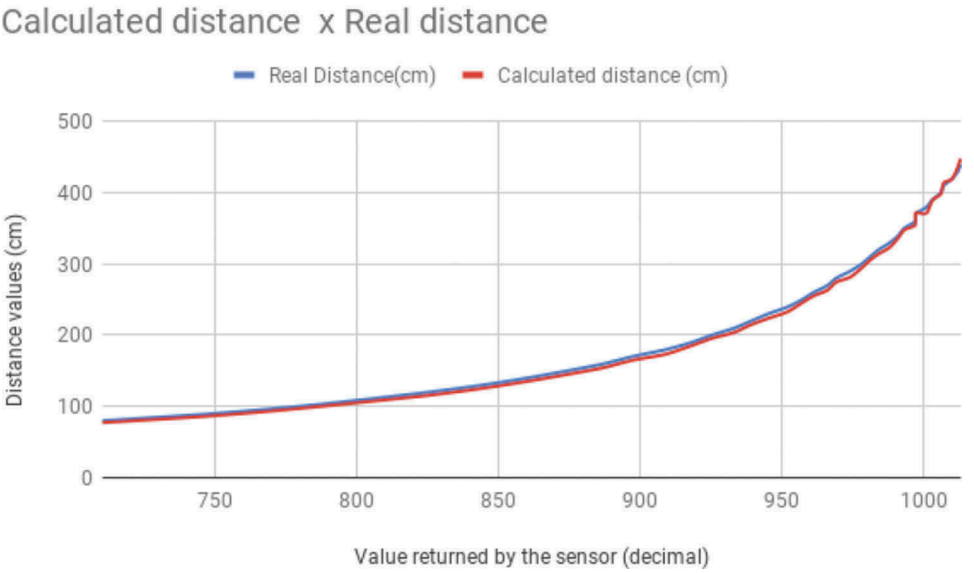

Figure 9. Real distances and calculated distances with the kinect sensor.

Integrating Depth with Object Detection

After running the program that includes depth calculation, there was a decrease of just 13% in the performance in relation to the display speed. Without the depth information processing we had approximately 4.31 FPS (i.e. frames per second) and after the inclusion of the processing and depth calculation we obtained approximately 3.76 FPS.

Figure 10 shows the result of detecting a common chair and an office chair, both objects were detected by the YOLOv2 algorithm and the distance information in centimeters from the robot to the object (in centimeters) was obtained using Equation (1) and the Microsoft Kinect sensor information. In Figures 11 and 12 we have the same information of the result of detecting a box and a table, respectively. In these figures, there is the information of the reliability that the YOLOv2 returns for each object detection.

Experiments in Mobile Robot Navigation

The experiment consists of assembling in the laboratory two situations for robot navigation where it does all the possible actions to avoid the obstacles with

Table 2. Distances calculated with the kinect sensor.

	Values calculated and returned		
Real distance (cm)	Value returned from Microsoft Kinect (decimal)	Calculation result (cm)	Error
80	664	77.43	3.21%
100	749	97.03	2.97%
120	803	115.64	3.64%
140	844	135.33	3.33%
160	874	154.60	3.37%
180	897	173.55	3.58%
200	918	195.41	2.29%
220	933	214.74	2.39%
240	945	233.18	2.84%
260	957	255.10	1.89%
280	966	274.44	1.99%
300	974	294.27	1.91%
320	981	314.14	1.83%
340	988	336.88	0.92%
360	993	355.25	1.32%
380	997	371.46	2.25%
400	1003	398.74	0.32%

Figure 10. YOLOv2 detecting common chair and office (wheeled) chair class.

a reactive control and detect the majority of the object that the neural network was trained to recognize. Although there are many control approaches like classic control approach (Mu et al. 2017) (Cuadros et al. 2014) (Jiang, Dagger, and Nijmeijer 1997) and heuristic control approach (Gamarra, Bastos Filho, and Sarcinelli Filho 2005; Fierro and Lewis 1998) in this work is used a reactive control.

Figure 11. YOLOv2 detecting box class.

Figure 12. YOLOv2 detecting table class.

The reactive control is very simple, the robot react in some way depending on the situation. To get the robot position during the tests was used an odometry algorithm like the one used in the work (Silva, Cuadros, and Gamarra 2018).

The performed test was a success. In both situations in which the robot was placed, it was able to complete a free collision trajectory successfully even though there were some variations in relation to each execution. In Figures 13 and 14 can be seen the results of the odometry program that was used in the tests in each situation that the robot was placed to navigate in an indoor environment. The positions of the objects were represented in the map with a rectangle that inside of it has the label for identification of which object it is. The positions of the objects were placed after the execution of the odometry tests, the odometry program just shows the robot navigation.

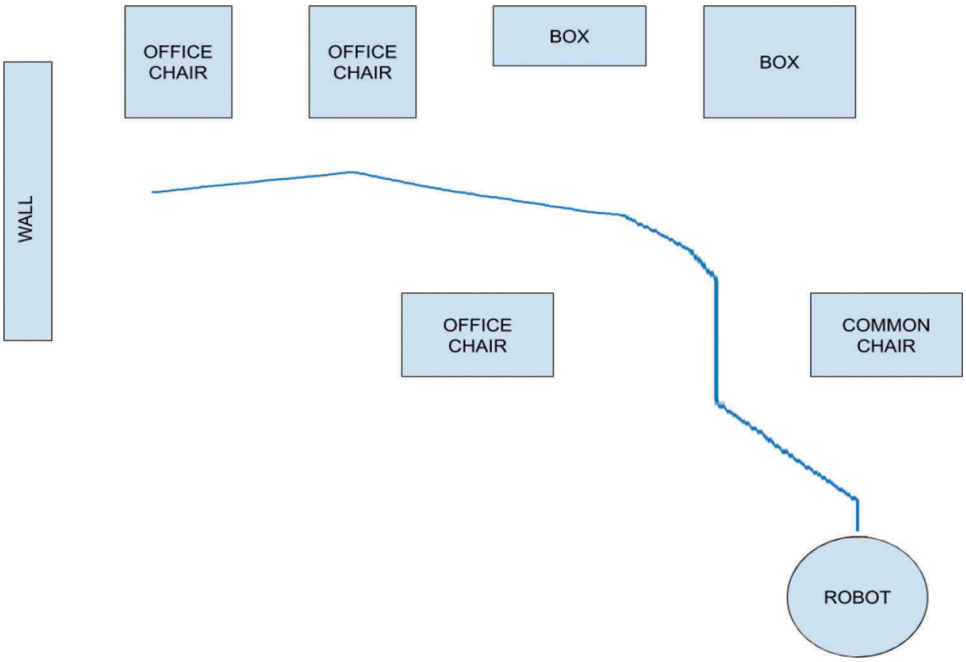

Figure 13. Result of the odometry in first obstacle track with the object positions.

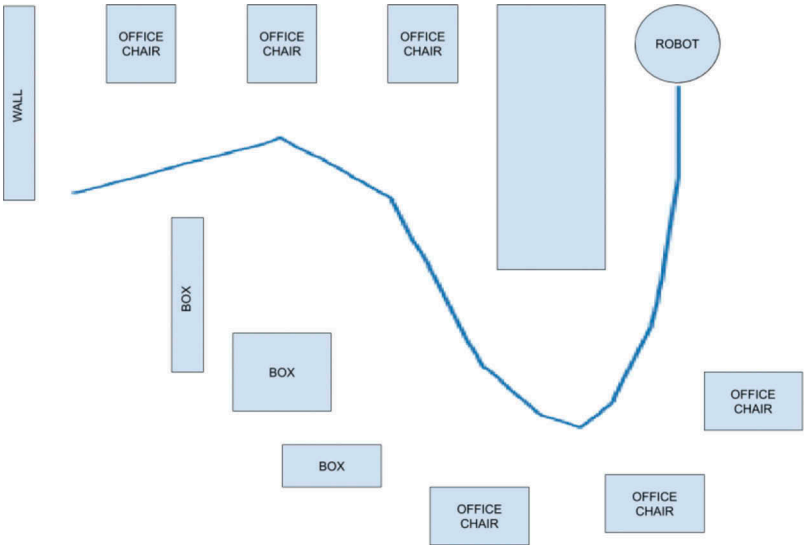

Figure 14. Result of the odometry in second obstacle track with the object positions.

It detected all the objects that it was trained for and performed all the actions that was programmed to avoid the obstacles. Tests were performed varying the speed of the motors, and it was detected that the faster the robot moves, the accuracy of object detections and its performance of obstacles avoidance decay.

Conclusion

In the current work, the main objectives of detecting a desired object using YOLO and Microsoft Kinect RGBD images, calculate the distance of the objects using the Microsoft Kinect sensor and then get all this information and make a simple navigation control were reached. Also, an integration of both sources of information was done, in order to run the system in real-time.

The information retrieved by Microsoft Kinect was successfully used. The YOLO neural network has detected all the predefined obstacles for which it has been trained with good reliability and the calculus of the distance using the depth information returned by Microsoft Kinect had an error below 3,64%.

The navigation control result was very satisfying. The robot recognized all the objects and executed all the actions that were proposed in the navigation control in order to avoid the obstacle. Even the performance of the neural network was lower than the expected, the robot was able to navigate without any problems. The odometry result helped to verify the control navigation performance.

Real-time object detection was also successful, although the execution of the final software was slow. In a Future work, we plan to fusion the Kinect information with others sensors like ultrasonic sensors in order to get a more robust architecture for a mobile robot navigation task.

Acknowledgments

We would like to acknowledge to the INPE (National Institute of Spatial Research) and the Professor Adriano Petry for their assistance and collaboration with this work.

References

Bersan, D., R. Martins, M. Campos, and E. R. Nascimento 2018. Semantic map augmentation for robot navigation: A learning approach based on visual and depth data. Article - 2018 Brazilian Symposium on Robotics and 2018 Workshop on Robotics in Education, Natal, Brazil.

Bezerra, C. G. 2004. Localização de um robô móvel usando odometria e marcos naturais. MS thesis, Universidade Federal do Rio Grande do Norte.

Cuadros, M. A. S. L., P. L. S. De Souza, G. M. Almeida, R. A. Passos, and D. F. T. Gamarra 2014. Development of a mobile robotics platform for navigation tasks using image processing. Asia-Pacific Computer Science and Application Conference (CSAC 2014), Shangai-China.

Equipe Data Science Academy. Deep learning book. Accessed May 6, 2018. http://deeplearningbook.com.br/

Fadli, H., C. Machbub, and E. Hidayat 2017. Human gesture imitation on NAO humanoid robot using kinect based on inverse kinematics method. International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA), Surabaya.

Fierro, R., and F. L. Lewis. 1998. Control of a nonholonomic mobile robot using neural networks. *IEEE Transactions on Neural Networks* 9 (4):589–600. doi:10.1109/72.701173.

Gamarra, D. F. T., T. F. Bastos Filho, and M. Sarcinelli Filho 2005. Controlling the navigation of a mobile robot in a corridor with redundant controllers. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2005), Barcelona.

Gu, S., X. Chen, N. Zeng, and X. Wang 2018. A deep learning tennis ball collection robot and the implementation on Nvidia Jetson TX1 board. Conference on Advanced Intelligent Mechatronics (AIM), Auckland, New Zealand.

Guo, Y., Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew 2016. Deep learning for visual understanding: A review. Neurocomputing. Recent Developments on Deep Big Vision.

Jiang, A., Z.-P. Dagger, and H. Nijmeijer. 1997. Tracking control of mobile robots: A case study in backstepping. *Automatica* 33 (7):1393–99.

Lucian, A., A. Sandu, O. Radu, and D. Moldovan 2018Human leg detection from depth sensing. Article - R&D Innovation Center, Romênia.

Maolanon, P., K. Sukvichai, N. Chayopitak, and A. Takahashi 2019. Indoor room identify and mapping with virtual based SLAM using furnitures and household objects relationship based on CNNs. 10th International Conference of Information and Communication Technology foe Embedded Systems (IC-ICTES), Bangkok, Thailand.

Moura, R. G. 2014. Utilizando o microsoft kinect na obtenção de atributos antropométricos. Work of conclusion of course (Bachelor in Information Systems) - Lutheran University Center of Palmas, Palmas, Tocantis, Brazil.

Mu, J., X. G. Yan, S. K. Spurgeon, and Z. Mao. 2017. Nonlinear sliding mode control of a two-wheeled mobile robot system. *International Journal of Modelling Identification and Control* 27 (2):75–83. doi:10.1504/IJMIC.2017.082943.

OpenKinect. OpenKinect. Accessed May 6, 2018. https://openkinect.org/wiki/Main_Page

Pegas, G. L. 2014. Rastreamento visual para Robôs usando microsoft kinect. Completion of course work (Electrical Engineer) - Universidade Federal of São Carlos, São Carlos, SP.

Redmon, J., S. Divvala, R. Girshick, and A. Farhadi 2016. You only look once: Unified, real-time object detection. Conference on Computer Vision and Pattern Recognition - CVPR, Las Vegas, USA.

Redmon, J., and A. Farhadi 2017. YOLO9000: Better, faster, stronger. Conference on Computer Vision and Pattern Recognition - CVPR, Honolulu, USA.

Silva, R. M., M. A. S. L. Cuadros, and D. F. T. Gamarra. 2018. *Comparison of a backstepping and a fuzzy controller for tracking a trajectory with a mobile robot*. Vellore, India: Article – ISDA.

Tenguria, R., S. Parkhedar, N. Modak, R. Madan, and A. Tondwalkar 2017. Design framework for general purpose object recognition on a robotic platform. Article - International Conference on Communication and Signal Processing, India.

Valente, R. L., R. D. Schirmer, J. C. Jesus, B. Schuster, M. B. Severo, D. R. Richards, C. C. Conrad, R. S. Guerra, R. M. Silva, and D. F. T. Gamarra 2017. Construção de um Robô Móvel Para Ensino das Disciplinas dos Cursos de Engenharia. Article – XLV Brazilian Congress of Engineering Education. Joinville/SC, Brasil.

Wang, Y., G. Song, G. Qiao, Y. Zhang, J. Zhang, and W. Wang 2013. Wheeled robot control based on gesture recognition using the Kinect sensor. IEEE International Conference on Robotics and Biomimetics (ROBIO), Shenzhen.

Zapf, M. P., A. Gupta, L. Y. M. Saiki, and M. Kawanabe 2018. Data-driven, 3-D classification of person-object relationship and semantic context clustering for robotics and AI application. 27th IEEE International Symposium on Robot and Human Interactive Communication, Nanjing, China.

Zhao, X., H. Jia, and Y. Ni. 2018. A novel three-dimensional object detection with the modified you only look once method. *International Journal of Advanced Robotic Systems* 15: doi: 10.1177/1729881418765507.

Zhou, J., L. Feng, R. Chellali, and H. Zhu 2018. Detecting and tracking objects in HRI: YOLO networks for the NAO “I see you” Function”. 27th IEEE International Symposium on Robot and Human Interactive Communication, Nanjing, China.