# DeepSpatial: Intelligent Spatial Sensor to Perception of Things

Marco Antonio Simões Teixeira, Flávio Neves-JR, Anis Koubaa, *Member, IEEE*, Lúcia Valéria Ramos de Arruda, *Member, IEEE*, and André Schneider de Oliveira, *Member, IEEE*

***Abstract*—This paper discusses a spatial sensor to identify and track objects in the environment. The sensor is composed of an RGB-D camera that provides point cloud and RGB images and an egomotion sensor able to identify its displacement in the environment. The proposed sensor also incorporates a data processing strategy developed by the authors to conferring to the sensor different skills. The adopted approach is based on four analysis steps: egomotive, lexical, syntax, and prediction analysis. As a result, the proposed sensor can identify objects in the environment, track these objects, calculate their direction, speed, and acceleration, and also predict their future positions. The on-line detector YOLO is used as a tool to identify objects, and its output is combined with the point cloud information to obtain the spatial location of each identified object. The sensor can operate with higher precision and a lower update rate, using YOLOv2, or with a higher update rate, and a smaller accuracy using YOLOv3-tiny. The object tracking, egomotion, and collision prediction skills are tested and validated using a mobile robot having a precise speed control. The presented results show that the proposed sensor (hardware + software) achieves a satisfactory accuracy and usage rate, powering its use to mobile robotic. This paper's contribution is developing an algorithm for identifying, tracking, and predicting the future position of objects embedded in a compact hardware. Thus, the contribution of this paper is to convert raw data from traditional sensors into useful information.**

***Index Terms*— Spatial sensor, egomotion, YOLO, mobile robot.**

## I. Introduction

ADVANCESin sensing techniques and technologies have allowed the development of small and useful sensors capable of providing a large amount of data. A class of such sensors is the RGB-D type that provides spatial information about the environment. The distance information is linked to pixels of the image such that each pixel can be represented by its coordinates X, Y, Z in a Cartesian plane, relative to the center of the sensor. This operation generates a data point cloud.

Manuscript received October 2, 2020; accepted October 29, 2020. Date of publication November 4, 2020; date of current version January 15, 2021. This work was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil, under Grant 001. The associate editor coordinating the review of this article and approving it for publication was Dr. Amitava Chatterjee. *(Corresponding author: Marco Antonio Simões Teixeira.)*

Marco Antonio Simões Teixeira, Flávio Neves-JR, Lúcia Valéria Ramos de Arruda, and André Schneider de Oliveira are with the Graduate School of Electrical Engineering and Computer Science (CPGEI), Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba 80230-901, Brazil (e-mail: marcoteixeira@alunos.utfpr.edu.br; neves@utfpr.edu.br; lvrarruda@utfpr.edu.br; andreoliveira@utfpr.edu.br).

Anis Koubaa is with the Robotics and Internet-of-Things Unit (RIOTU), Prince Sultan University (PSU), Riyadh 11586, Saudi Arabia, and also with the CISTER/INESC and ISEP-IPP, 4200-135 Porto, Portugal (e-mail: akoubaa@psu.edu.sa).

This article has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/JSEN.2020.3035355

These RGB-D sensors are used for a multitude of applications, such as facial recognition [1], [2], object measurement and classification in industries [3] among several other applications. Specially in mobile robotics, RGB-D sensors are usually used for navigation tasks and environment mapping [4], [5]. Moreover, they can also be used for robust tasks, such as object tracking [6], [7], and identification of the robot's position in the environment [8], [9]. However, the successful accomplishment of all these activities involves reliable data processing beyond the mere data capture by the RGB-D sensor.

Indeed, RGB-D sensors provide only distance data associated with an RGB image. This raw data is not enough to support any decision made by a robot, or to identify the person passing in front of the sensor, for example. Thus, It is necessary to process this data to generate useful information, such as an occupation map for mobile robots, or identify a person's face for a security system.

In general, computer vision techniques are applied to RGB images to extract such useful knowledge. For example, deep learning methods allow us to perform advanced tasks such as identification of objects or people, identification of anomalies, among many others [10]–[14]. One of the techniques that stands out in objects' recognition from RGB images is YOLO (You only look once) [15], [16]. This technique does not need robust hardware for online running and promotes a good

trade-off between processing power and precision of results. Besides, the used computer vision technique must also be able to deal with the point cloud data corresponding to the pixel coordinates generated by the RGB-D sensors [17]–[19].

In this paper, an embedded intelligent sensor, named *Intelligent Spatial Sensor to Perception of Things* (DeepSpatial sensor), is developed. The proposed sensor is an arrangement of different perception sources, which are merged to produce concise and reliable information. An RGB-D sensor is used to obtain a cloud of points containing the distance of the objects around it and also giving spatial notion of the environment. An egomotion sensor is used to identify sensor displacement and provide its linear and angular velocity. All pre-processing and merge-processing are carried out on an embedded CPU. The YOLO is used as a tool to object identification, and information matching for different perception sources, being processed over an embedded GPU that supports deep learning techniques.

The work [20] presents a comparison between CNN object detection techniques being performed in embedded systems, such as Jetson TX1, TX2, and Xavier. YoLo is the algorithm with the highest update rate in embedded systems, according to the authors. Thus, YoLo was chosen as an object detection tool. In the results section (sec::Performance), a comparison is presented between the different versions of YoLo running on Nvidia Jetson Nano. Our main contribution lies in integrating some well-known sensing hardware into a single perception system able to identify objects in the environment, track them, and predict their future positions.

Some works discuss similar approaches for sensing systems, also presenting solutions for objects tracking in the spatial environment using YOLO [21]–[24]. However, these works are not concerned with the technique's materialization into an embedded solution, with the identification and prediction of detected objects motion, and with the reference movement (i.e., the sensor motion by itself). As proposed in this paper, these techniques also use known deep learning tools for the development of a new sensing strategy. However, different from these cited works that only use computer vision techniques to track objects in the environment, the sensor herein proposed is capable of tracking objects using the Point Cloud and it also provides spatial object disposition based on the distance between objects in the environment, predicting the future positions of such objects.

There also are some papers concerned with the prediction of the objects' trajectory, as [25], [26]. Especially in [25], a time difference strategy similar to that presented in this work is used. However, in these works, the authors are not concerned with the acceleration and possible collision calculation between dynamic objects and the object being tracked, as it will be done in this paper.

Finally, our recent paper [27] presents a software strategy very similar to that developed herein. However, this recent paper is not worrying about the sensor displacement by itself in the environment and about the processing capacity or hardware requirements. In summary, this work aims to develop a perception system (in terms of hardware and software) using deep learning techniques as a tool for object identification and tracking, predicting their future position, and identifying the relative sensor displacement in the environment. The result will be a compact equipment capable of carrying out all the proposed actions.

## II. DeepSpatial Sensor

This paper aims to develop an intelligent sensor to identify objects in the environment, track them, and predict their future positions. The sensor is called *Intelligent Spatial Sensor to Perception of Things* or simply DeepSpatial sensor. The DeepSpatial development is presented through two steps. First, the proposed sensor architecture and used hardware will be presented in section II-A. Thus the software procedures implementing the sensing intelligent approach is discussed in section II-B.

### A. DeepSpatial Hardware

The proposed sensor hardware has four components. The first is a small computer implementing three tasks: data processing, information exchange with the user (a mobile robot in this paper), and Wireless network creation and management.

The chosen computer is the Intel Nuc NUC5i5RYH due to its processing power, small size, and low battery consumption. An Nvidia Jetson Nano board is the second component. It is used to run computer vision procedures for object identification in the environment (YoLo). This graphics processor is ideal for performing tasks in parallels, such as deep learning techniques and other Artificial Intelligence (AI) applications.

The other two hardware components are sensing elements. The Intel RealSense D435i sensor is used to perceive the environment. This component has an RGB camera to collect images of the environment and infrared sensors to obtain spatial information. This information is used to identify objects and their positions around the sensor. The Intel RealSense Tracking Camera T265 is used to capture the DeepSpatial displacement. This camera measures its movement allowing to infer information such as speed and travel direction of DeepSpatial, for example.

The communication between all components of DeepSpatial sensor is implemented through Ethernet and USB interfaces, as shown in Figure 1. A direct connection between the NUC computer and the Intel RealSense D435i via a USB 3.0 is established. The same occurs with the Intel RealSense Tracking Camera T265 sensor. The communication between Jetson Nano and the NUC computer is carried out via an Ethernet network. Finally, the NUC computer creates a wireless network allowing access to information from DeepSpatial sensor and communication with other equipment, such as the mobile robot.

The entire proposed strategy runs entirely on the DeepSpatial sensor. External equipment, such as a computer, can collect sensor data, but it is not necessary to employ it. This work aims to propose a novel embedded and independent equipment to spatial perception.

Finally, the integration and management of all hardware components is carried out through "Robot Operating System"

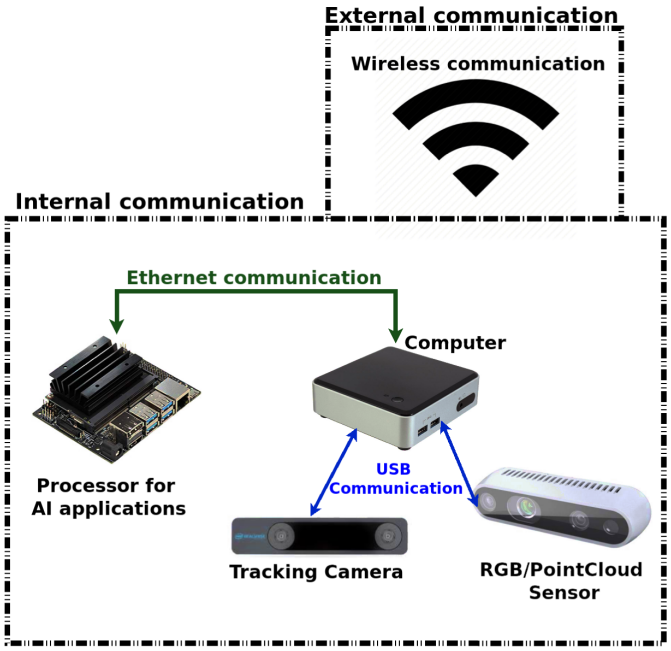



Fig. 1. Representation of communication between DeepSpatial sensor components. For AI processing, it is using an Nvidia Jetson Nano. The tracking sensor is an Intel RealSense Tracking Camera T265, and the RGB/Pointcloud sensor is an Intel RealSense D435i.

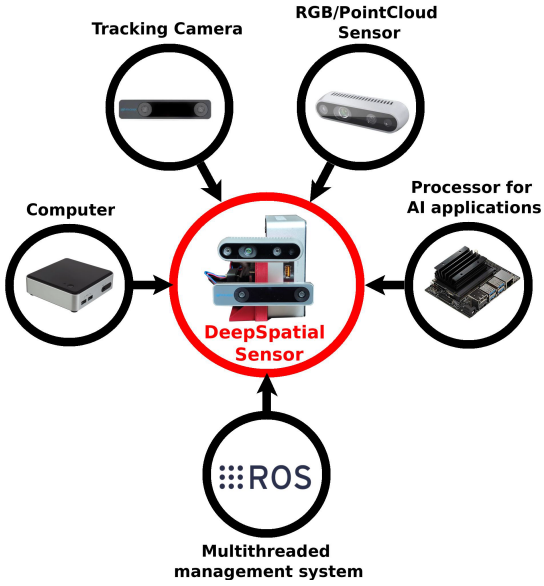



Fig. 2. All hardware components of the DeepSpatial sensor. For AI processing, it is using an Nvidia Jetson Nano. The tracking sensor is an Intel RealSense Tracking Camera T265, and the RGB/Pointcloud sensor is an Intel RealSense D435i.

(ROS) [28], [29] as can be seen in Figure 2. The center of this figure shows the DeepSpatial view, including all hardware components. It is worthwhile to note that the specific hardware components above cited are not mandatory. They can be replaced by any similar component with the same processing and sensing capacities and playing the same role.

## B. DeepSpatial Software

The *Intelligent Spatial Sensor to Perception of Things* is able to identify objects in the environment, classify them into static and dynamic types, and track all of them. These tasks are accomplished through four analyzes: egomotion, lexical, syntax, and prediction analysis. The developed software strategy adopts a follow-up approach, where each analysis provides information to the next ones. However, it is also possible to collect all data and independently perform the analysis, according to the desired use of the DeepSpatial sensor. The software aspects will be presented and discussed in the following sections.

*1) Egomotion Analysis:* One of the main problems in the development of the DeepSpatial sensor is its spatial displacement. As the robot moves, errors can be generated in the calculation of the movement of the identified objects, since movement direction is based on the spatial displacement of the object concerning the sensor. In this way, a static object can have a misrepresentation of movement.

The Intel RealSense Tracking Camera T265 sensor is adopted to correct the influence of the sensor displacement. This equipment can support visual odometry techniques, like those presented in [30]–[32], which allow calculating the displacement of the sensor in the environment (egomotion). In this way, it is possible to obtain the linear speed (LVS) and angular speed (AVS) of the DeepSpatial sensor. This information will be used in the next Syntax analysis to compensate errors due to DeepSpatial displacement in the environment when the direction, speed, and acceleration of the identified objects are calculated.

*2) Lexical Analysis:* The lexical analysis aims to survey the characteristics of objects in the environment. The objects identified by the Lexical Analysis will be stored in tokens.The token's attributes are related to the object characteristics. Such attributes are object's class ($C_t$), probability ($PRO_t$), object's center in the image ($C_{timg}$), object's height in the image ($H_{img}$), object's width in the image ($W_{img}$), 3D position of the object in the real environment ($P_t$) and the object's life time ($T_t$).

- *Object's class and probability* These information are generated by YOLO, which can identify objects in an image. For each object detected by YOLO, a box is created around the object and an object name (class) is given. A degree of probability (probability) of the detected object belongs to the given class is also added. Both attributes (class and probability) are directly taken from YOLO. The image is collected by the D435i sensor, inputted to YOLO, processed and the YOLO answer is stored.
- *Center, height, and width of the object in the image* These data refer to the shape of the object identified in the image. This shape is computed from the object's box provided by YOLO. The number of pixels forming the height ($H_{timg}$) and the width ($W_timg$) of the object's box are computed, as also the position in the image of the box's center ($C_timg$). These measures correspond to the height and width of the object in pixels and to the position of the center of the object in the image. These data are used to extract future information and can be employed by the sensor user to carry other specific tasks.
- *3D position of the object* The position of the object in the real world is one of the most critical information to be

captured. The previous attributes (center of the object in the image, height and width in pixels) and the information from the Intel RealSense D435i sensor are used to identify the object's position in the world. D435i sensor provides a cloud of points where each pixel of the image has an associated spatial point, given by spatial coordinates [x, y, z] indicating the distance among the pixel and the center of the D435i sensor. Thus, each pixel [i, j] has a distance data x, y, z in the point cloud (PC [i, j, x, y, z]).

• *Life time* that is the time values when the object is detected by YOLO. Time is running continuously, starting at zero when the DeepSpatial is turned on.

The object's localization is estimated through an average of the points belonging to the box of the identified object. As YOLO does not perform segmentation of the object, the average is carried out only with 20% of the box's height and width pixels, thus only the center of the identified object is obtained, ensuring that the points used for the average of its 3D position are referring to the object. Equations [1,2,3,4] present the calculation used to obtain the 3D position of each object identified by the sensor, where the variable PC [i, j, x, y, z] refers to all 3D points identified by the RGB-D sensor, being i and j their dimensions in a 2D matrix, and (x, y, z) are the distance data. Moreover equation 5 computes the limits over the coordinate axes that are used in the summation. In this equation, $CH_{start}$ and $CH_{end}$ correspond to the height ranges used in the RGB-D data, and $CW_{start}$ and $CW_{end}$ limit the width ranges. These intervals correspond to 10% of the size of the width and height taken from the center of the identified object. Finally, the spatial position of the object is a 3D point ($P_t$) having the positions $(x, y, z)$ of the identified object. To cut only the center of the box, its size is multiplied by 0.1 (10%) (Equations 23 and 4) of its height and width from the center. In this way, the box is converted to 20% of its total height, and 20% of its total width.

$$P_t = (P_t[x], P_t[y], P_t[z]) \tag{1}$$

where

$$P_t[x] = \frac{1}{CH_{end}} \frac{1}{CW_{end}} \sum_{i=CH_{start}}^{CH_{end}} \sum_{j=CW_{start}}^{CW_{end}} PC[i, j, x] \tag{2}$$

$$P_t[y] = \frac{1}{CH_{end}} \frac{1}{CW_{end}} \sum_{i=CH_{start}}^{CH_{end}} \sum_{j=CW_{start}}^{CW_{end}} PC[i, j, y] \tag{3}$$

$$P_t[z] = \frac{1}{CH_{end}} \frac{1}{CW_{end}} \sum_{i=CH_{start}}^{CH_{end}} \sum_{j=CW_{start}}^{CW_{end}} PC[i, j, z] \tag{4}$$

and

$$\begin{aligned} CH_{end} &= (C_{timg} + (H_{timg} * 0.1)) \\ CH_{start} &= (C_{timg} - (H_{timg} * 0.1)) \\ CW_{end} &= (C_{timg} + (w_{timg} * 0.1)) \\ CW_{start} &= (C_{timg} - (w_{timg} * 0.1)) \end{aligned} \tag{5}$$

*3) Syntax Analysis:* Syntax analysis is responsible for converting the data of each token into useful information. The information generated in this step is the speed, direction, and acceleration of each identified dynamic object. An object is considered dynamic or not, according to the pre-defined classification given in Table I. This table also indicates which dynamic objects are alive. Thus, at the end of this analysis, each identified object will have information such as dynamic or static, alive or not, velocity, direction, and acceleration if they are in motion.

TABLE I
INFERRED KNOWLEDGE IN OBJECTS

| | ***Object class*** |
|---|---|
| ***Is it dynamic?*** | person, bicycle, car, motorbike, aeroplane, bus, train, truck, boat, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe, frisbee, snowboard, sports ball, skateboard, surfboard tennis racket, chair |
| ***Is it alive?*** | person, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe |

• *Identification of the same token* The displacement of the object in space allows the computation of speed, direction, and acceleration. Thus, it is necessary to have the position of the same object in two instants of time, to calculate these variables. The strategy to identify the same object in two moments is given by a simple comparison between all tokens of two objects at two instants of time,present time ($T_{present}$) and past time ($T_{past}$). In this way, the identification of the same token in both times is made by comparing all objects from time $T_{present}$ with objects from time $T_{pass}$. First, it is compared whether the two tokens are dynamic and belong to the same class, if it is true, the Euclidean distance between the real position of the objects, identified through the point cloud, is compared. If the Euclidean distance is bigger than 0.15, we assume that it is not the same object. This value was defined empirically, after proving to be enough to not lose the movement of an object and to prevent different grouping tokens.

• *Velocity Calculation* eThe object's speed, direction, and acceleration are based on the same token identification in two moments.. The speed calculation is given by the displacement of the object in the environment ($\delta Distance$) divided by time ($\delta Time$). The distance is obtained by calculating the Euclidean distance between $(P_t[x, y, z]_{T_{pass}})$ and $(P_t[x, y, z]_{T_{present}})$. Time is obtained by the difference between $T_{tT_{present}}$ and $T_{tT_{pass}}$, where $T_{present}$ refers to the time of the last set of tokens collected, and $T_{pass}$ refers to the tokens previously received. Finally, the sensor speed, obtained by the egomotion analysis, is subtracted from the speed of the token. In this way, the calculation of the linear token velocity ($LV_t$) is given by the equation 6.

$$LV_t = \frac{\delta Distance}{\delta Time} - LVS \tag{6}$$

where $LVS$ is the linear speed of the sensor.

• *Direction calculation* The direction is used to check the object's trajectory in the environment. This calculation is done by measuring the angle between the same token in two moments using the function $atan2$ [33]. The direction is calculated in 2D, thus only $(P_t[x, y]_{T_{pass}})$ and $(P_t[x, y]_{T_{present}})$

are used, despising the $z$ information for each identified object. The $atan2$ function returns the angle between the two points in radian. This information is added to the angular speed of the DeepSpatial sensor obtained in the analysis of egomotion. The direction ($D_t$) calculation is given by the equation 7.

$$D_t = atan2((P_t[y]_{T_{present}} - P_t[y]_{T_{pass}}), (P_t[x]_{T_{present}} - P_t[x]_{T_{pass}}) - AVS \quad (7)$$

where $AVS$ is the angular speed of the sensor.

• *Acceleration calculation* The acceleration of an object is due to the speed difference during two instants of time divided by time. In this way, it is only possible to calculate the acceleration of objects that already have speed. Therefore, to perform the acceleration calculation ($ACC_t$), it is first checked whether the token has speed at two instants of time, if so, equation 8 is computed, which corresponds to the speed variation ($\delta Vel$) by the time variation ($Time$).

$$ACC_t = \frac{LV_tT_{present} - LV_tT_{pass}}{\delta Time} \quad (8)$$

*4) Prediction Analysis:* Prediction analysis uses egomotion data, lexical, and syntax analyses, to infer the future position of the object. The prediction of the next object, position is computed through the speed, acceleration, and direction of a dynamic object. These future positions can be used to prevent potential collisions between dynamic objects and the DeepSpatial sensor.

The future position of dynamic objects at a different time in the future ($T_p$) is computed based on acceleration information. If an object has an acceleration value, equation 9 is used to identify the object's displacement in space ($Ob_d$). If the object has no acceleration, its speed, considered as constant, is used, multiplying $T_p$ by $LV_t$.

$$Ob_d = LV_t * T_p + ACC_t * \frac{T_p^2}{2} \quad (9)$$

After calculating the displacement of the object in the environment, it is possible to calculate its future position after $T_p$ seconds. Having the object's spatial position ($P_t[x, y, z]$) its displacement in the environment ($Ob_d$) and its direction ($D_t$), it is possible to calculate its new position in the environment ($P_p[x, y]$) as presented in Equation 10. The displacement of the object on the Z-axis is not considered.

$$\begin{aligned} P_p[x] &= P_t[x] + \sin(Ob_d + 90) * Ob_d; \\ P_p[y] &= P_t[y] + \cos(Ob_d + 90) * Ob_d; \end{aligned} \quad (10)$$

As all collected data are related to the DeepSpatial sensor center, the calculation of the future position of the sensor is applied using the equation 9 considering, instead of $ACC_t$, the linear speed of the sensor $LVS$. The points used to represent the initial position of the sensor are [0,0,0] because the sensor is considered the origin of the coordinate plane. The new predicted position of the sensor in $T_p$ time is defined as $Sp_p$.

• *Collision prediction* Having the future position of all the identified dynamic objects and the DeepSpatial sensor position, at time $T_p$, it is possible to predict possible collision paths. The collision predaiction is carried out by calculating the Euclidean distance between the future position of the DeepSpatial sensor, and all future positions of the dynamic objects identified in the environment. If this distance is less than a predetermined threshold, it means that the sensor and the dynamic object will be very close to each other in the future, signaling a possible collision.

By default, the future position of identified objects is continually calculated with $T_p$ taking values of 1, 3, and 5 seconds. After the next location of all objects is obtained, the collision prediction is also carried out for each instant of time. If possible collisions are predicted, an alert is published, so that the DeepSpatial sensor user can take the appropriate actions.

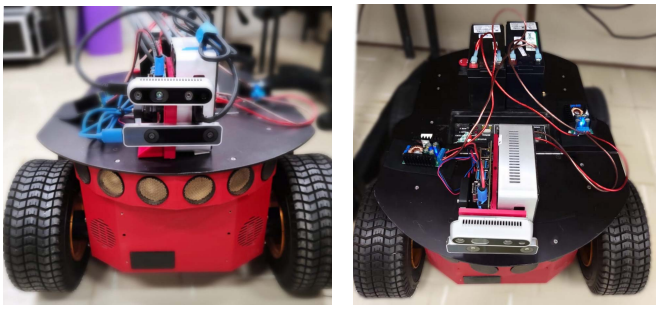


Fig. 3. Sensor attached to the Pioneer P3-AT robot. Left, front view of the robot. Right, top view of the robot, where it is possible to view the batteries for powering the sensor.

## III. Results and Discussions

This section aims to present the results obtained with the developed *Intelligent Spatial Sensor to Perception of Things*. A first experiment is presented in order to analyze as the operating data are collected and processed by DeepSpatial sensor. Then, an experiment with the DeepSpatial sensor embedded in a mobile robot is carried out. The mobile robot has linear and angular speed control, making it possible to perform a comparison between the speeds obtained by the robot and by the proposed DeepSpatial sensor.

The mobile robot Pioneer P3-AT was used (Figure 3). This robot is compatible with ROS, and it has been connected to the DeepSpatial sensor. From the wireless network created by the DeepSpatial sensor, it was possible to collect data from the DeepSpatial sensor and send commands to the robot. The robot has linear and angular speed control, besides encoders used to calculate these speeds logically. All the described experiments were carried out with the DeepSpatial sensor embedded to the robot, powered by batteries and communicating through the wireless network created by the DeepSpatial sensor.

### A. Knowledge Extraction From Collected Data

All information processed by the DeepSpatial sensor can be visually obtained throughout a user interface. Thus it is possible to view the identified objects, their positions around the sensor, their predicted positions, and the possible collision paths. This information is also available in a textual form, through a topic from ROS. In this way, the information can be

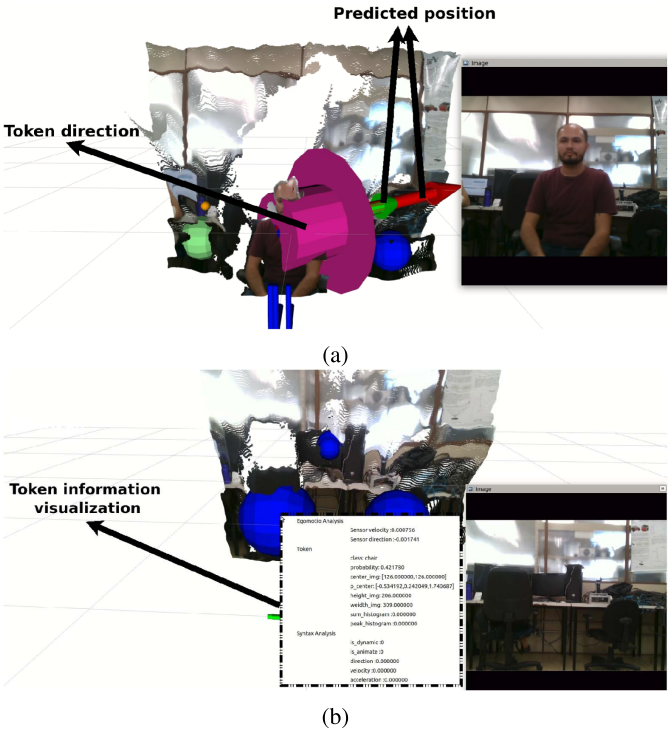



Fig. 4. Graphical display provided by the DeepSpatial sensor. (a) Representation of a dynamic object, its movement and future positions. (b) Token information.

directly read from the DeepSpatial sensor in order to support actions and decisions, such as stopping the robot or deflect it if a future collision is predicted, or look for a specific object in the environment and others. This DeepSpatial sensor opens up a wide range of options in the field of mobile robotics.

Figure 4 and Figure 5 show the information provided by the DeepSpatial sensor. In Figure 4, the sensor is directly connected to a monitor, where the information is displayed. First, the image is processed by YOLO, then identified object information such as object's position in the world is calculated based on data coming from the RGB-D view. The appearance of dynamic objects changes from sphere to an arrow, pointing to the calculated direction for object displacement. The information about predicted positions are also displayed as arrows, pointing to the possible future positions of the object. All future positions can be visible or it is possible to filter them for 1, 3, and 5 seconds. Finally, when a possible collision between objects is inferred, a black sphere is generated around the possible collision locus.

## B. Performance Analysis

All experiments are carried out at a frequency of 10 Hertz. After 30 minutes from the beginning of the operation, the DeepSpatial sensor CPU (Intel Nuc) is operating at 53.4% and using only 10.42% of memory. The CPU is handily running; however, the entire system is limited by YOLO's update rate. If YOLO operates at 2 Hertz, the whole system will work at the same frequency. There are different versions of YOLO, the last being YOLOv3. A smaller version, but with less precision, is the YOLOv3-tiny, it operates at a higher

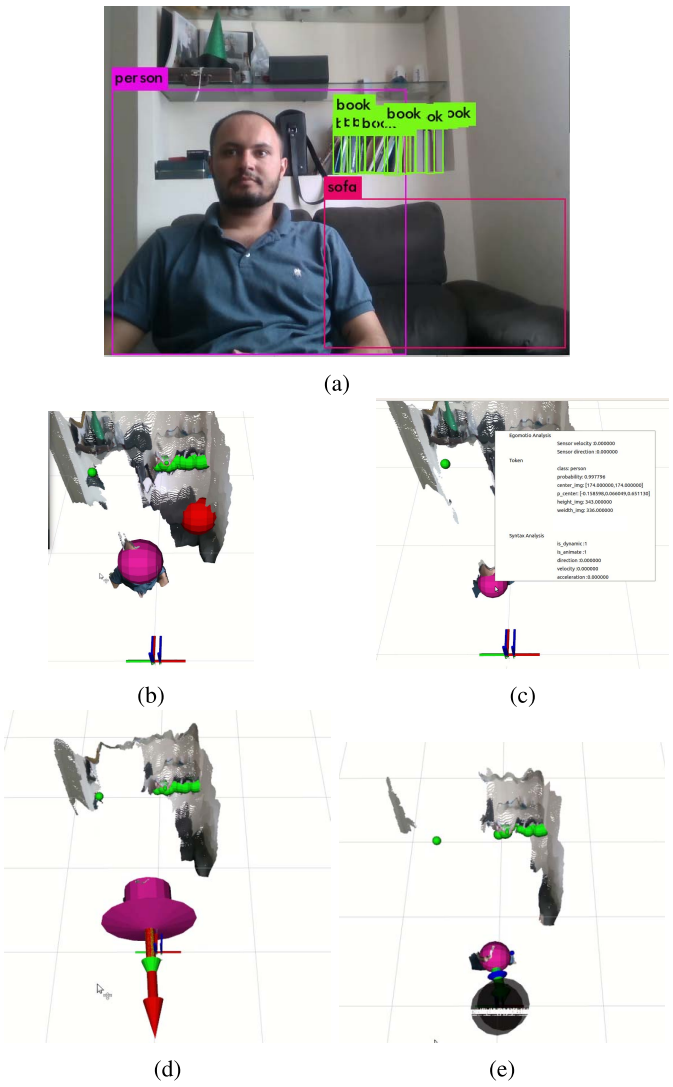



Fig. 5. Graphical display provided by the DeepSpatial sensor. The grid is represented in meters. (a) YOLOv3-tiny. (b) 3D position of the objects identified in the environment. (c) Information about a specific object. (d) A vector pointing to the predicted positions in 1 (blue), 3 (green) and 5 (red) seconds. (e) Possible collision warning.

frequency but with a lower mean Average Precision (mAP). Figure 6 presents two bar plots, where the first one shows the operating frequencies of the YOLO version running on the DeepSpatial sensor, and the second shows the mAP of all versions, according to its developer [15], [16]. YOLOv3 has the highest mAP, but its update rate is minor (1.40). YOLOv2 has a good mAP and an acceptable refresh rate in some situations. YOLOv3-tiny offers a reasonable update rate and an adequate mAP. GPU usage remains 99% regardless of the chosen version of YOLO.

## C. Egomotion

The DeepSpatial sensor's ability to capture and calculate its displacement is evaluated in the next experiments in which the DeepSpatial sensor is connected to the mobile robot. Visually, Figure 7 presents a representation of the robot in motion, and stopping in front of a person. In Figure 7.a and 7.c, it is possible to observe that the robot calculated its future

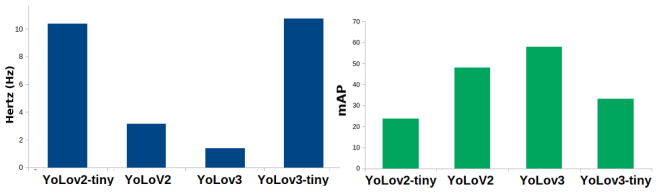



Fig. 6. On the left, frequency of operation in Hertz. On the right, Mean Average Precision (mAP).

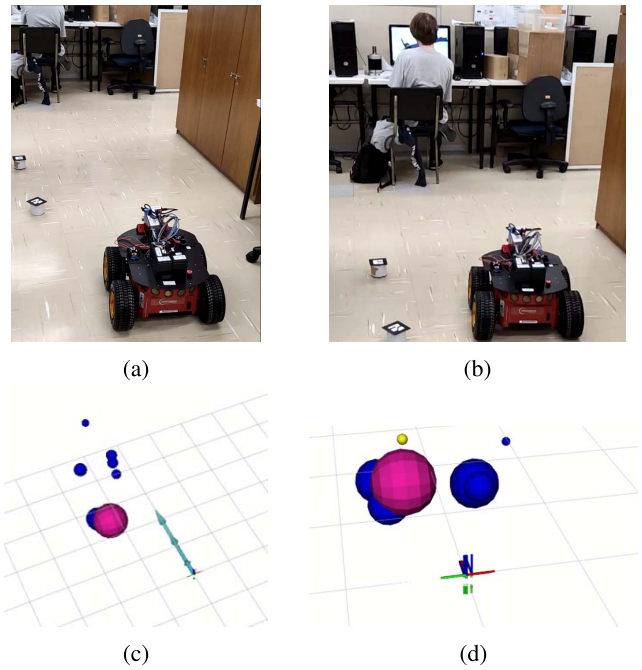



Fig. 7. DeepSpatial sensor embedded into the robot and all sensor information are collected over the wireless network. The grid is represented in meters. (a) Robot in motion. (b) Robot stopped. (c) Information calculated during robot motion. (d) Information calculated during robot stop.

position, but It did not consider the person as a moving object, this is because it compensated for its speed angular and linear with the calculated velocities for the dynamic object. In Figure 7.b and 7.d, future movements are not calculated for the robot, nor for the dynamic object, as both are still stoped.

The spatial motion capture is compared with odometry computed by the mobile robot. The robot is a standard mobile platform that estimates its relative displacement through a fusion of encoder odometry and inertial movement sensor. This estimation is susceptible to errors because it is based on dead-reckoning, with error accumulation. The results are presented in Figure 8, where "commands" represents the speed reference sent to the robot controller, "robot" represents the speed calculated by the robot's encoder and DeepSpatial sensor represents the speed calculated by the sensor DeepSpatial sensor.

The average error for the linear velocity calculated by the sensor was around 0.04 m/s, and for the angular velocity, it was around 0.06 m/s when compared to the speed obtained by the robot's encoder. Figure 9 presents a boxplot of the difference between speed data from both DeepSpatial sensor and the robot encoder. This error does not significantly affect

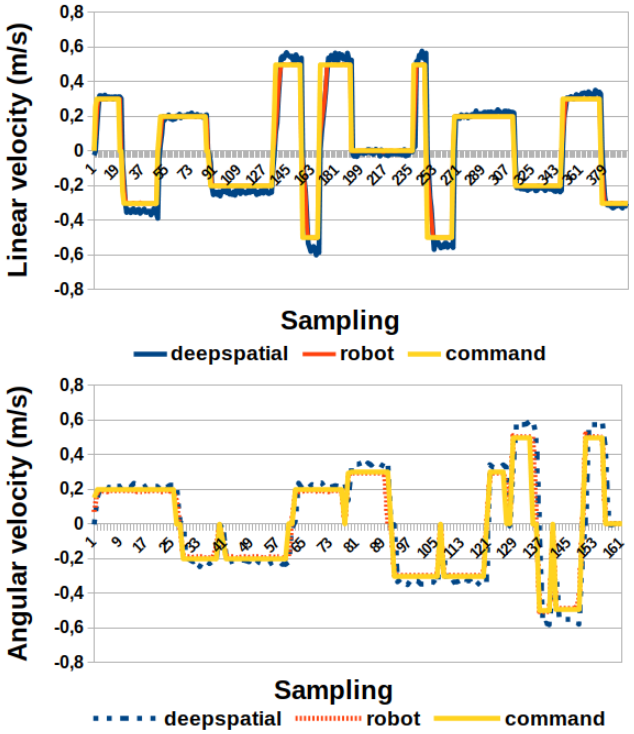



Fig. 8. Representation of linear and angular velocities measured during the experiment. commands are the velocity references sent to the robot controller. Robot is the speeds calculated by the robot's encoder. DeepSpatial sensor is the speeds calculated by the DeepSpatial sensor.

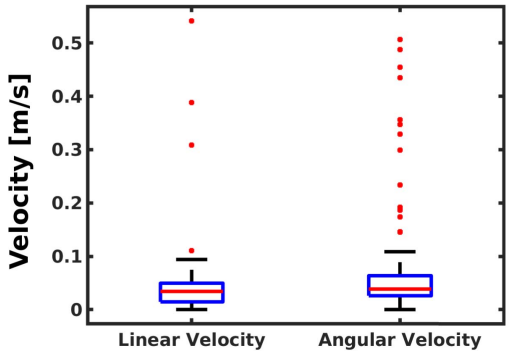



Fig. 9. Boxplot of the error between the speeds calculated by the DeepSpatial sensor and by the robot.

the calculation of the future speed for identified dynamic objects. For example, an object has been identified at 1 meter from the robot and it moves in $\delta t 1$, that is, 1 second in the future, it will be identified at 0.96 meters, causing the robot to detect the collision in advance.

### D. Object Tracking

This experiment aims to analyze the behavior of the proposed strategy and the Intel RealSense Tracking Camera T265 tracking sensor. Specific information about its sensor can be obtained at [34], [35].

One of the main difficulties in calculating the future position, speed, acceleration, and direction of an object, is to identify the same object in two moments. The strategy developed in this work uses only the Euclidean distance between the

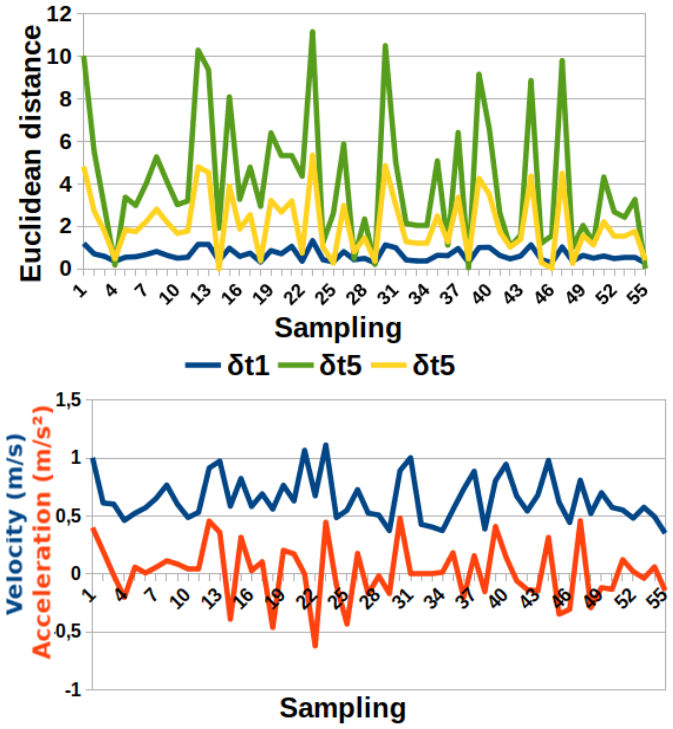



Fig. 10. Top figure: Euclidean distance between the current object position, and its predicted positions in 1, 3 and 5 seconds. Bottom figure: velocity and acceleration calculated for the object.Left Plot of the distance between the current position of the object, and its predicted positions in 1, 3 and 5 seconds.Right plots of velocity and acceleration calculated for the object.

position of dynamic objects in two moments. If this distance is less than 0.15 and more significant than 0.035, and the objects belong to the same class, then they are considered the same object. As the update frequency of the sensor is high, it is allowed to use a shorter distance, since the object does not move much between the two-time instants. The distance value greater than 0.035 is used to avoid false calculations, resulting from small movements of the object.

Figure 10 shows the tracking of a moving person. On the left plot, the Euclidean distance between the person's current position and his predictions of future positions is shown, on the right plot, the calculated acceleration and speed profiles are presented. When both acceleration and speed are high, the future status of the object is calculated at a greater distance, as shown at position 24 of the plot. When we have a high speed, and low acceleration, the object's next location is considered to be less since the object is decelerating. In some cases, with a negative acceleration value, the object is deemed to stop in the future, position 4 of the plot.

The validation of dynamic object tracking by the DeepSpatial sensor is carried out by an experiment with the worst possible scenario: two dynamic objects of the same class are side by side. In this way, the proposed tracking algorithm must differentiate the objects to carry their tracking. The strategy used for this action was presented in the section II-B.3. During the experiment, two people walk side by side, and the Euclidean distance between these two positions obtained by the DeepSpatial sensor must be monitored. Figure 12 shows

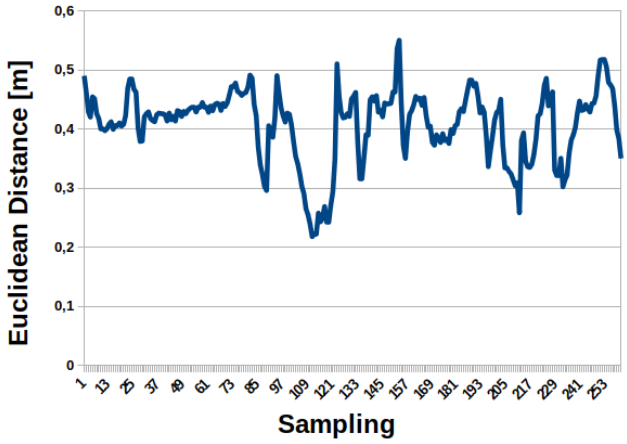



Fig. 11. Graph representing the distance between the two people during the experiment. People walked back and forth, side by side.

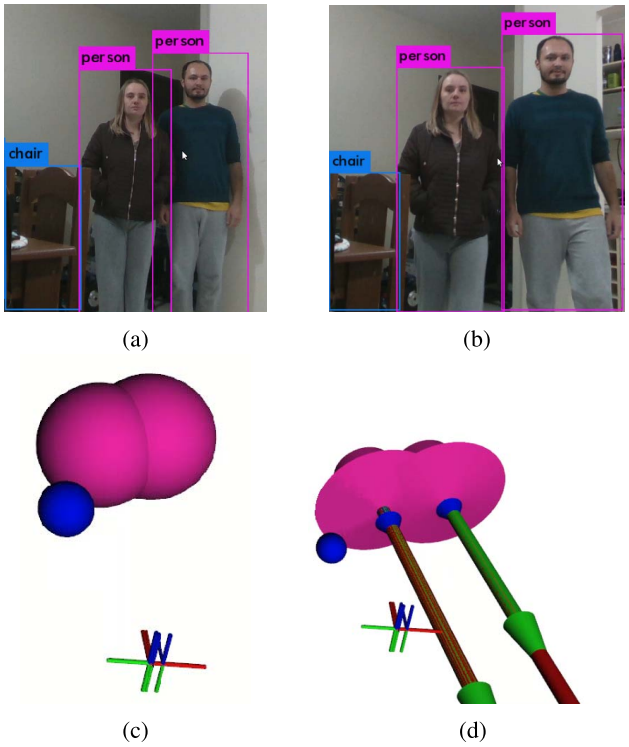
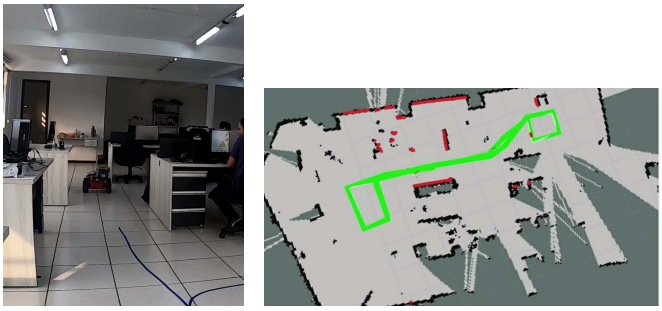



Fig. 12. DeepSpatial sensor view aimed at two people side by side. (a) People are standing still. (b) People are on the move. (c) Sensor output, people are standing still. (d) Sensor output, people in motion, with their future positions calculated.

the two people standing side by side, and then moving, where it is possible to observe the calculation of the future positions for both dynamic objects. Figure 11 presents the Euclidean distance between the objects (people) during the experiment. It is worth mentioning that during the entire monitoring, both people were correctly identified and tracked, validating the proposed tracking algorithm.

## IV. Application Example

The DeepSpatial sensor will be demonstrated in an example task To validate the approach proposed by this article. The equipment will be integrated into an autonomous navigation strategy and used as safety equipment to prevent accidents.



(a) (b)

Fig. 13. Pioneer P3-At robot performing autonomous navigation. (a) Robot in the test environment, being an office. (b) Map of the environment, the trajectory to be covered in green, and the robot's position on the map.

The Pioneer P3-AT robot is assigned to perform autonomous navigation from point to point. This is a traditional strategy for autonomous cargo transport robots [36]–[38] and, thus, validate the sensor during the execution of a conventional task in the area of mobile robotics. The robot's navigation circuit will be repetitive, so the robot will always navigate the same environment.

The DeepSpatial sensor will send environmental information to the robot, such as the need to perform an emergency stop. The robot will read the sensor's information in the token format and then decide to *stop* or *continue* according to the information provided by the DeepSpatial sensor. Some rules will be created for the robot's action based on the sensor's statement, which is presented below.

- If the prediction analysis identifies a collision in the present position or predicted positions in the future (1,3 and 5 seconds), the robot must save the navigation data for analysis.
- If the objective is in motion, but is not towards the robot, and is at a distance greater than 0.5 meters, the robot must continue its trajectory, and thus avoid an unnecessary stop.

In this way, the robot will perform navigation from point to point, repeating the points, and stopping when some of the conditions mentioned previously are reached. The distance between the object and robot at the time of stop will be stored, and the speeds of the object and robot, to evaluate the performance.

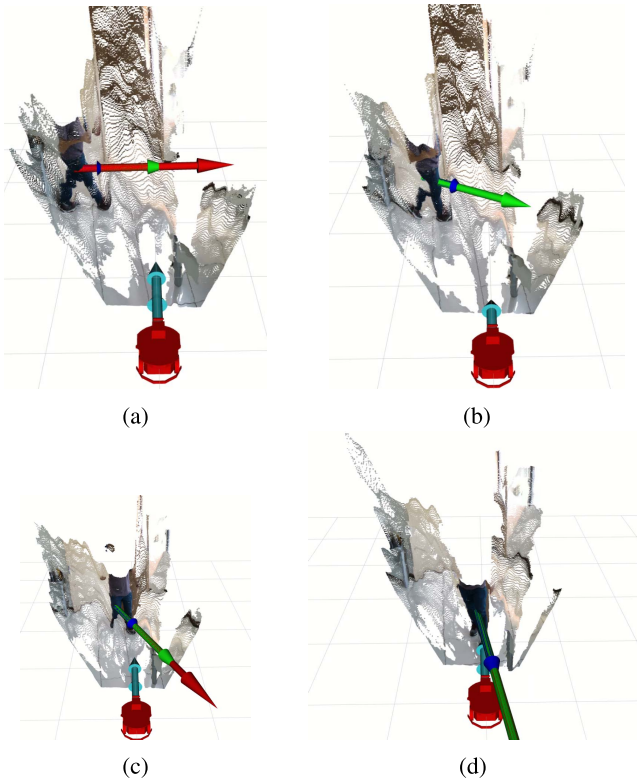
The robot sailed for 1 hour in an office, and 60 possible collisions were identified, being a possible collision in the present time, 17 potential collisions in one second, 26 collisions predicted in 3 seconds, and 16 predicted collisions for 5 seconds in the future, according to with the prediction analysis developed in this paper.

Figure 13 shows the robot navigating the defined circuit,where the robot makes a map of the environment, and then runs the SLAM. The navigation and localization technique is not interesting for this work, being used only and exclusively to validate DeepSpatial in a real application.

During navigation, the DeepSpatial sensor was turned on, and when it identified a possible collision, it wrote down the information. Table II presents an average containing the

TABLE II
COLLISION DATA DETECTED BY THE DEEPSPATIAL SENSOR DURING THE AUTONOMOUS NAVIGATION OF THE ROBOT. A TOTAL OF 60 COLLISIONS WERE IDENTIFIED, ONE AT TIME 0, 17 AT TIME 1, 26 AT TIME 3, AND 16 AT TIME 5

| Future time (seconds) | | Distance (Token future) (Robot prediction) | Distance (Token future) (Robot real) | Robot speed | Object speed |
|---|---|---|---|---|---|
| **0** | | **0,476 (Average)** | **0,476 (Average)** | **0,211 (Average)** | **0,391 (Average)** |
| | **Min:** | 0,476 | 0,476 | 0,211 | 0,391 |
| | **Max:** | 0,476 | 0,476 | 0,211 | 0,391 |
| **1** | | **0,222 (Average)** | **0,304 (Average)** | **0,189 (Average)** | **0,901 (Average)** |
| | **Min:** | 0,038 | 0,037 | 0,123 | 0,301 |
| | **Max:** | 0,487 | 1,040 | 0,231 | 1,366 |
| **3** | | **0,338 (Average)** | **0,705 (Average)** | **0,174 (Average)** | **0,461 (Average)** |
| | **Min:** | 0,026 | 0,188 | 0,093 | 0,014 |
| | **Max:** | 0,497 | 1,163 | 0,254 | 0,951 |
| **5** | | **0,298 (Average)** | **0,852 (Average)** | **0,181 (Average)** | **0,469 (Average)** |
| | **Min:** | 0,149 | 0,300 | 0,094 | -0,017 |
| | **Max:** | 0,499 | 1,501 | 0,232 | 1,226 |



(a) (b) (c) (d)

Fig. 14. Image provided by DeepSpatial during its operation. The identification, tracking, and prediction of the position of a person on the movement is presented. (a) The person is perpendicular to the sensor. (b, c) The person is performing a turning action. (d) The person is moving towards the sensor.

distance between the robot and the Token at the time of the collision is detected, both in the present time (0) and in the future (1.3 and 5 seconds). The robot's actual distance and the predicted distance to the object are also shown in the table. It is worth mentioning that the higher the object's speed and

the robot, the greater the distance from the predicted collision, as the forecast will consider that the object is accelerating.

As DeepSpatial calculates the identified objects' speed and direction, it considers that a collision will only happen if the object is in its direction, thus avoiding unnecessary stops and accidents, anticipating a stop or slow down. Figure 14 presents an identified trajectory of a person, where first, he is going in a direction perpendicular to the robot. Then he performs a contour action and goes towards the robot.

This section presented the software and hardware of the DeepSpatial sensor proposed by this work, also discusses the sensor's advantages in a traditional application in mobile robotics and sensing.

## V. Conclusion

This work has developed an embedded sensor, composed of a set of components, where the raw data of each component is collected and gathered, generating useful information for several applications. With the proposed sensor, it is possible to develop an intelligent equipment capable of identifying dynamic objects and tracking them, in addition it also provides information such as, for example, a bottle is on a table, which can be used by a household robot, for example. In this way, the development of such intelligent equipment can be concerned with treating the information from the DeepSpatial sensor and not trying to collect them from the environment.

The sensor processing runs on Intel Nuc NUC5i5RYH, and it is observed that after 30 minutes of uninterrupted use, the computer remained with only 53.4 % of its processing capacity and 10.42 % of its occupied memory. The Jetson Nano was used to perform object detection. When using YOLO to identify objects, Jetson Nano used 99% of its GPU. However, this use does not represent a risk during execution, since it has a CPU, leaving the GPU dedicated to YOLO. In terms of hardware, the sensor proved to be satisfactory, having no problems at run-time, always running online.

The logical approach is organized in some steps, *egomotion analysis*, *lexical analysis*, *syntax analysis*, and *prediction analysis*. In egomotion analysis, the Intel RealSense Tracking Camera T265 is used to identify the sensor's movement. The ability to identify movements was verified, where its presentedresults have attained an average error concerning the data obtained by the robot of 0.04 at linear speed, and 0.06 for angular speed. Then the lexical analysis is performed, where all the information of the object is collected, using the YOLO and the RGB-D depth sensor. In syntax analysis, the data collected from the objects is used to calculate their displacement in the environment, direction, speed, and acceleration. In prediction analysis, a prediction of the future position of all dynamic objects is carried out. This prediction is able to prevent a possible collision between two dynamic objects in the environment.

The experiment results have showed that the DeepSpatial sensor performance was satisfactory. Its limited frequency of operation is directly linked to YOLO. However, a new setting can be done since we choose between using YOLOv2 for greater accuracy, and operating at a low rate, or losing efficiency using the YOLOv3-tiny and operating at a frequency of 10 hertz. In future works, the replacement of either Jetson Nano or YOLO will be considered to seek a reasonable rate with a better accuracy on the identified objects. Finally, the sensor was proposed and used in a real application. Thus, this article proposed not only creating the sensor, in terms of hardware and software, but also brought examples of application.

## Acknowledgment

The authors gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

## References

[1] L. Jiang, J. Zhang, and B. Deng, "Robust RGB-D face recognition using attribute-aware loss," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 10, pp. 2552–2566, Oct. 2020.

[2] L. Ren, J. Lu, J. Feng, and J. Zhou, "Multi-modal uniform deep learning for RGB-D person re-identification," *Pattern Recognit.*, vol. 72, pp. 446–457, Dec. 2017.

[3] R. Méndez Perez, F. A. Cheein, and J. R. Rosell-Polo, "Flexible system of multiple RGB-D sensors for measuring and classifying fruits in agri-food industry," *Comput. Electron. Agricult.*, vol. 139, pp. 231–242, Jun. 2017.

[4] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[5] Y. Sun, M. Liu, and M. Q.-H. Meng, "Improving RGB-D SLAM in dynamic environments: A motion removal approach," *Robot. Auto. Syst.*, vol. 89, pp. 110–122, Mar. 2017.

[6] Y. Liu, X.-Y. Jing, J. Nie, H. Gao, J. Liu, and G.-P. Jiang, "Context-aware three-dimensional mean-shift with occlusion handling for robust object tracking in RGB-D videos," *IEEE Trans. Multimedia*, vol. 21, no. 3, pp. 664–677, Mar. 2019.

[7] M. Jiang, Z. Pan, and Z. Tang, "Visual object tracking based on cross-modality Gaussian-Bernoulli deep Boltzmann machines with RGB-D sensors," *Sensors*, vol. 17, no. 12, p. 121, Jan. 2017.

[8] P. F. Proença and Y. Gao, "Probabilistic RGB-D odometry based on points, lines and planes under depth uncertainty," *Robot. Auto. Syst.*, vol. 104, pp. 25–39, Jun. 2018.

[9] P. F. Proenca and Y. Gao, "SPLODE: Semi-probabilistic point and line odometry with depth estimation from RGB-D camera motion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 1594–1601.

[10] T. Wilaiprasitporn, A. Ditthapron, K. Matchaparn, T. Tongbuasirilai, N. Banluesombatkul, and E. Chuangsuwanich, "Affective EEG-based person identification using the deep learning approach," *IEEE Trans. Cognit. Develop. Syst.*, vol. 12, no. 3, pp. 486–496, Sep. 2020.

[11] A. Zhavoronkov *et al.*, "Deep learning enables rapid identification of potent DDR1 kinase inhibitors," *Nature Biotechnol.*, vol. 37, no. 9, pp. 1038–1040, Sep. 2019.

[12] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019, *arXiv:1901.03407*. [Online]. Available: http://arxiv.org/abs/1901.03407

[13] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, "Car detection using unmanned aerial vehicles: Comparison between faster R-CNN and YOLOV3," in *Proc. 1st Int. Conf. Unmanned Vehicle Syst.-Oman*, 2019, pp. 1–6.

[14] B. Benjdira, Y. Bazi, A. Koubaa, and K. Ouni, "Unsupervised domain adaptation using generative adversarial networks for semantic segmentation of aerial images," *Remote Sens.*, vol. 11, no. 11, p. 1369, Jun. 2019.

[15] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*. [Online]. Available: http://arxiv.org/abs/1804.02767

[16] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7263–7271.

[17] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, Nov. 2019.

[18] X. Liu, M. Yan, and J. Bohg, "MeteorNet: Deep learning on dynamic 3D point cloud sequences," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9246–9255.
[19] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 770–779.
[20] S. Hossain and D. J. Lee, "Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with GPU-based embedded devices," *Sensors*, vol. 19, no. 15, p. 3371, Jul. 2019.
[21] G. Ning *et al.*, "Spatially supervised recurrent convolutional neural networks for visual object tracking," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.
[22] X. Lv *et al.*, "A robust real-time detecting and tracking framework for multiple kinds of unmarked object," *Sensors*, vol. 20, no. 1, p. 2, Dec. 2019.
[23] M. Jiang, T. Hai, Z. Pan, H. Wang, Y. Jia, and C. Deng, "Multi-agent deep reinforcement learning for multi-object tracker," *IEEE Access*, vol. 7, pp. 32400–32407, 2019.
[24] Y. Yoon *et al.*, "Analyzing basketball movements and pass relationships using realtime object tracking techniques based on deep learning," *IEEE Access*, vol. 7, pp. 56564–56576, 2019.
[25] M. A. Simoes Teixeira, N. Dalmedico, A. S. de Oliveira, L. V. Ramos de Arruda, and F. Neves, "A pose prediction approach to mobile objects in 2D costmaps," in *Proc. Latin Amer. Robot. Symp. (LARS) Brazilian Symp. Robot. (SBR)*, Nov. 2017, pp. 1–6.
[26] J. Zhong, H. Sun, W. Cao, and Z. He, "Pedestrian motion trajectory prediction with stereo-based 3D deep pose estimation and trajectory learning," *IEEE Access*, vol. 8, pp. 23480–23486, 2020.
[27] M. Teixeira *et al.*, "Intelligent 3D perception system for semantic description and dynamic interaction," *Sensors*, vol. 19, no. 17, p. 3764, Aug. 2019.
[28] M. Quigley *et al.*, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, Kobe, Japan, vol. 3, no. 3.2, 2009, p. 5.
[29] A. Koubaa, *Robot Operating Syst. (ROS): The Complete Reference*, vol. 4, 4th ed. Cham, Switzerland: Springer, 2020.
[30] S.-H. Tsao and S.-S. Jan, "Observability analysis and performance evaluation of EKF-based visual-inertial odometry with online intrinsic camera parameter calibration," *IEEE Sensors J.*, vol. 19, no. 7, pp. 2695–2703, Apr. 2019.
[31] W. Ci, Y. Huang, and X. Hu, "Stereo visual odometry based on motion decoupling and special feature screening for navigation of autonomous vehicles," *IEEE Sensors J.*, vol. 19, no. 18, pp. 8047–8056, Sep. 2019.
[32] M. Aladem and S. A. Rawashdeh, "A combined vision-based multiple object tracking and visual odometry system," *IEEE Sensors J.*, vol. 19, no. 23, pp. 11714–11720, Dec. 2019.
[33] Z. Luo, J. Ding, L. Zhao, and M. Wu, "An enhanced non-coherent pre-filter design for tracking error estimation in GNSS receivers," *Sensors*, vol. 17, no. 11, p. 2668, Nov. 2017.
[34] A. Alapetite, Z. Wang, J. P. Hansen, M. Zajączkowski, and M. Patalan, "Comparison of three off-the-shelf visual odometry systems," *Robotics*, vol. 9, no. 3, p. 56, 2020.
[35] O. Mise, R. Madison, and B. Haight, "A comparison of SWaP-limited, visual-inertial odometry systems for GPS-denied navigation," *Proc. SPIE*, vol. 11424, Apr. 2020, Art. no. 1142406.
[36] A. Saffiotti, "The uses of fuzzy logic in autonomous robot navigation," *Soft Comput. Fusion Found., Methodologies Appl.*, vol. 1, no. 4, pp. 180–197, Dec. 1997.
[37] E. Krell, A. Sheta, A. P. R. Balasubramanian, and S. A. King, "Collision-free autonomous robot navigation in unknown environments utilizing PSO for path planning," *J. Artif. Intell. Soft Comput. Res.*, vol. 9, no. 4, pp. 267–282, Oct. 2019.
[38] M. M. Ejaz, T. B. Tang, and C.-K. Lu, "Vision-based autonomous navigation approach for a tracked robot using deep reinforcement learning," *IEEE Sensors J.*, early access, Aug. 13, 2020, doi: 10.1109/JSEN.2020.3016299.

**Marco Antonio Simões Teixeira** received the M.Sc. degree in electrical and computer engineering from the Federal University of Technology at Parana (UTFPR), Parana, Brazil, in 2017, where he is currently Ph.D. degree in electrical and computer engineering. His research interests include mobile robots in the areas of perception and intelligent systems, computer vision, navigation, mapping, and applied AI.

**Flávio Neves-JR** received the B.Sc. and M.Sc. degrees in electrical engineer from the Federal University of Technology, Parana, Brazil, in 1987 and 1989, respectively, and the Ph.D. degree in electrical engineering from the University Paul Sabatier (LAAS), France, in 1998. Since 1992, he has been with the Federal University of Technology, Parana, where he is a Full Professor. His research interests include hardware and software to instrumentation and automation.

**Anis Koubaa** (Member, IEEE) is currently a Professor of Computer Science and the Leader of the Robotics and Internet of Things Research Laboratory, Prince Sultan University. He is also a Senior Researcher with CISTER and ISEP-IPP, Porto, Portugal. His current research interests include providing solutions toward the integration of robots and drones into the Internet of Things (IoT) and clouds, in the context of cloud robotics, deep learning, robot operating systems (ROS), cloud robotics, and unmanned aerial systems. He is also a Senior Fellow of the Higher Education Academy (HEA), U.K. He has been the Chair of the ACM Chapter in Saudi Arabia, since 2014.

**Lúcia Valéria Ramos de Arruda** (Member, IEEE) received the degree in electrical engineer from the Federal University of Ceara, Brazil, in 1985, the M.Sc. degree from the Graduate School of Electrical Engineering, Campinas State University (FEE/UNICAMP), Brazil, in 1988, and the Ph.D. degree in electrical engineering from the University of Nice-Sophia Antipolis, France, in 1992. Since 1995, she has been with the Federal University of Technology, Parana, where she is a Full Professor. Her research interests include soft computing methods to model and control of dynamic systems.

**André Schneider de Oliveira** (Member, IEEE) received the M.Sc. degree in mechanical engineering, focused in force control of rigid manipulators, from the Federal University of Santa Catarina (UFSC) in 2007, and the Ph.D. degree in engineering of automation and systems, in 2011, with thesis focused on differential kinematics through dualquaternions for vehicle-manipulator systems. He is currently an Adjunct Professor with the Federal University of Technology, Parana (UTFPR). He is a member of the Advanced Laboratory of Robotics and Embedded Systems (LASER) and Automation and Advanced Control System Laboratory (LASCA). His research interests include robotics, mechatronics, and automation with special focus in navigation and localization of mobile robots, autonomous and intelligent systems, perception and environment identification, cognition, deliberative decisions, human-interaction, and navigation control.