Article

Methods and Software Tools for the Safety Movement of Agricultural Highly Automated Vehicle Based on Deep Learning Methods

Kirill Svyatov *, Roman Zhitkov, Vladislav Mikhailov, Imil Khayrullin

Posted Date: 30 November 2023

doi: 10.20944/preprints202311.1922.v1

Keywords: highly automated vehicles; robotics; modeling; obstacle avoidance

Article

Methods and Software Tools for the Safety Movement of Agricultural Highly Automated Vehicle Based on Deep Learning Methods

Kirill Svyatov *, Roman Zhitkov, Vladislav Mikhailov and Imil Khayrullin

Ulyanovsk State Technical University; roman73zh@gmail.com (R.Z.); vmixoks@gmail.com (V.M.); imil_khayrullin@mail.ru (I.K.)

* Correspondence: k.svyatov@ulstu.ru; Tel.: +78422778585

Abstract: When implementing a control system for a ground-based unmanned vehicle (UV) used in agriculture, an important factor is the low cost of the equipment used, so the use of cameras without lidars for navigation, avoiding obstacles, and ensuring movement safety seems promising. Ensuring the safety of movement of UV consists of developing methods and means that make it possible to bypass static and dynamic obstacles and make an emergency stop if it is impossible to bypass. The article describes methods and software that provide detection of objects surrounding the UV through the use of the YOLOv8 neural network, space free for movement through the use of semantic segmentation with the MobileNetV3 network architecture, tools for combining several data sources to build a local perception map, as well as linear and angular speed of the UV based on calculating the optimal direction of movement for avoiding obstacles. Testing of the developed methods and software was realized in the Webots simulation environment. Software is developed with ROS2.

Keywords: highly automated vehicles; robotics; modeling; obstacle avoidance

1. Introduction

The agricultural sector is among the most profitable and significant industries. As the volume of agricultural production expands, there is growing interest in reducing costs and decreasing the number of workers engaged in labor-intensive or health-hazardous conditions [1]. The use of unmanned ground vehicles for tasks such as spraying, fertilizer spreading, harvesting, plowing, and others is considered one of the most promising methods for improving labor efficiency [2]. Their application in agricultural tasks offers a range of important advantages:

- Increased productivity in field operations through the use of highly automated vehicles capable of continuous operation without breaks and able to work at night;
- Reduction of labor costs by decreasing the number of drivers needed to operate agricultural machinery;
- Lowering the overhead costs of producing agricultural crops, leading to a reduction in the price of the final product and strengthening the company's market position;
- Minimization of harmful effects on machine operators during the application of agrochemicals.

For large-scale implementation of autonomous driving technologies in agriculture, it is important to consider economic efficiency, particularly the cost of equipping unmanned vehicles.

One of the key elements of an unmanned vehicle control system is its ability to perceive the environment. Lidars, which provide high-precision three-dimensional scanning of the surroundings, are one of the most reliable means for this task [3]. However, they are also among the most expensive components of the system, which can significantly increase the overall cost of an unmanned vehicle and make its operation economically impractical for farms, especially those with limited budgets [4].

Using cameras instead of lidars can significantly reduce the costs of control system components [5], thereby making unmanned vehicles more affordable for farmers. This, in turn, can accelerate the adoption and scaling of unmanned technologies in the agricultural sector, contributing to increased efficiency and sustainability of agriculture.

 doi:10.20944/preprints202311.1922.v1

Currently, the main elements of unmanned agricultural vehicles are agronavigators integrated with steering devices. The most well-known solutions are from companies like Patchwork [6], Trimble [7], Ag Leader [8], SMAJAYU [9], solutions from John Deere, and others. Fundamentally, they have a system that, through precise tracking of equipment position, allows for controlled field navigation and accurate chemical application when integrated with executive devices. Additionally, some companies offer kits for retrofitting existing equipment for unmanned driving (Fieldin [10], Blue White Robotics [11], Braun Maschinenbau [12], GPX Solutions [13], ThornTek [14]). Most of them use lidars to prevent collisions with obstacles. Systems from Bear Flag Robotics [15], Sabanto [16], and Cognitive Agro Pilot [17] work with cameras without lidars.

One of the main challenges with control systems that use cameras instead of lidar is the lack of depth information in the images, making it difficult to estimate the distance to objects. From the perspective of constructing a sensory data processing system, combining data from multiple sources can improve the accuracy of the model. This approach is used for localization, for example, by combining GNSS data and the inertial system using Kalman filtering. This article proposes the combination of image analysis results by several deep learning models to obtain a local map of an unmanned vehicle used to implement a safety system for obstacle bypassing and emergency stopping. Such components are object detection and semantic segmentation of the image. To construct a local map that takes into account the distance to objects, a Birds-eye-View (BEV) projection is used. The advantage of this approach is that it can be combined with methods of analyzing spatial data obtained from lidar to improve the accuracy of obstacle distance recognition.

In BEV projection the geometry of objects is restored and the path is planned. The key advantage of an autonomous vehicle control system that uses only cameras and constructs a BEV projection, in addition to a frontal projection, lies in its enhanced perception of depth and distance to objects. This advantage encompasses several positive effects:

- The BEV projection provides a top view of the environment, allowing the system to more accurately determine the depth and distance to objects. This is crucial for safe maneuvering and collision avoidance.
- The BEV projection offers a wide view of the surroundings, enabling the control system to see more than what is possible with just a frontal camera. This improves the decision-making capabilities in complex traffic situations.
- With the depth and distance to objects taken into account, the system can more effectively plan paths and maneuver in complex conditions, such as at intersections or in heavy traffic.
- Utilizing only cameras with the capability of creating a BEV projection potentially reduces the need for additional sensors like LiDAR or radar, which can decrease the cost and complexity of the system.
- The BEV projection may be more adaptable to changes in the environment, such as variations in lighting or weather conditions, compared to other perception methods.

2. Obstacle detection and avoidance method

To implement a complex of obstacle avoidance systems integrated with linear and angular velocity control, it is necessary to implement four modules working together:

- Obstacle detection module and determination of their coordinates.
- Module for determining the space free for movement.
- Module for constructing a local map of the surrounding space of the unmanned vehicle.
- Module for controlling the linear and angular velocity of the unmanned vehicle.

The obstacle detection and coordinate determination module is responsible for detecting obstacles in the surrounding space of the unmanned vehicle. It uses sensor data (images from a depth camera) to detect and localize objects and obstacles using the YOLOv8 library [19]. The data is processed to determine the position and size of obstacles relative to the unmanned vehicle. The effectiveness of this module is critically important for the safety and reliability of the system.

Let's describe the mathematical formulation of the object detection task.

 doi:10.20944/preprints202311.1922.v1

Let there be I - a frame obtained from the camera, used for building models, $O = \{t_x, t_y, t_w, t_h, p_0, p_1, p_2, .., p_k\}$ – the resulting vector. It is necessary to select such a vector of parameters a of the neural network model $A: F \rightarrow O \times N$, which delivers the minimum to the quality functional J on the training sample X^n:

$$a^* = \underset{a}{\operatorname{argmin}} J(a, X^n), \tag{1}$$

where N – the number of objects in the image;

n – the volume of the training sample;

(t_x, t_y)– the center of the bounding box in the image;

(t_w, t_h) – the width and height of the bounding box;

p_0 – the probability of finding any object in this frame;

$(p_1, \ldots, p_k)$ – the probabilities of the object belonging to a certain class, $k = 80$. The total number of recognizable classes corresponds to the number of classes recognized by the YOLOv8 model, trained on the COCO data set (80 classes).

The module for determining the free space for movement analyzes data about the surrounding space and identifies which areas are free of obstacles and safe for movement using semantic segmentation methods implemented in the fastseg library [20]. Let's present the mathematical formulation of the task for semantic segmentation of the input frame. Image semantic segmentation is the process of assigning a label to each pixel of an image in such a way that pixels with the same labels belong to the same class of objects. Mathematically, the process of semantic segmentation can be described as follows:

Let I be the input frame of size $H \times W$, where H is the height and W is the width of the image. Each pixel in I has RGB values or other color spaces.

The image I is fed into a convolutional neural network (CNN). The CNN consists of a sequence of layers, including convolution layers (Conv), pooling layers (Pool), and fully connected layers (FC). Each layer transforms the input data as follows:

$$Conv: I' = \sigma(W * I + b) \tag{2}$$

$$Pool: I'' = pool(I') \tag{3}$$

$$FC: O = W''_{fc} I + b_{fc} \tag{4}$$

where $*$ is the convolution operation, σ is the activation function (e.g., ReLU), W and b are the weights and biases of the convolution layer, W_{fc} and b_{fc} are the weights and biases of the fully connected layer.

The output O from the CNN is transformed into a pixel classification map S of size $H \times W \times C$, where C is the number of classes. Each element $S_{i,j,k}$ represents the probability that the pixel at position (i, j) belongs to class k.

The class of each pixel is determined as the class with the maximum probability:

$$P_{i,j} = \underset{k}{argmax}\, S_{i,j,k} \tag{5}$$

where $P_{i,j}$ is the class label for the pixel at position (i, j).

Based on P, the output segmentation image is formed, where each pixel has a class label corresponding to the object it belongs to. A feature of the proposed method is that obstacle avoidance is based on a map with a "top-down" projection. When transforming the frontal projection into a top-down view, obstacles and objects become elongated and occupy pixels that are behind the object; such pixels should be marked as "free" or "unknown". To solve this problem, in the segmentation image, areas occupied by obstacle objects are replaced with free areas, so that later on the map, areas actually occupied by objects can be marked.

The mathematical description of the process of replacing pixels belonging to certain rectangular areas on the image can be presented as follows. Let S be the semantic segmentation image of size

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2023 doi:10.20944/preprints202311.1922.v1

$H \times W$, where each pixel $S_{i,j}$ has a class label $P_{i,j}$. Areas are defined by vectors $R_k = (x_k, y_k, w_k, h_k, c_k)$, where x_k and y_k are the coordinates of the middle point of the areaw_k and h_k are the width and height of the area respectively, and c_k is the class of the object to be replaced. Let there be N such vectors.

For each area R_k, the following steps are performed:

1. The boundaries of the area are determined:

$$x_{start} = x_k - \frac{w_k}{2};\ x_{end} = x_k + \frac{w_k}{2} \tag{6}$$

$$y_{start} = y_k - \frac{h_k}{2};\ y_{end} = y_k + \frac{h_k}{2} \tag{7}$$

2. For each pixel in this area, if the pixel belongs to class c_k, its value is replaced:

$$S_{i,j} = 100;\ for\ x_{start} \le i < x_{end},;\ y_{start} \le j < y_{end},;\ if\ S_{i,j} = c_k \tag{8}$$

The result of this operation is a modified segmentation image S', where pixels in the specified areas belonging to the indicated class are replaced with the value 100.

This process can be implemented programmatically using loops to iterate through the areas and the corresponding pixels within each area. It is important to ensure that the indices and boundaries remain within the dimensions of the image.

The module for constructing a local map of the surrounding area and path planning creates a dynamic map of the surrounding area, including detected obstacles and safe routes using the "Inverse Perspective Mapping" method. This method is used in image processing and computer vision to transform images obtained from a perspective projection into images as if they were obtained from an orthographic (or planar) projection.

To transform the entire image into a Birds-eye-view (BEV) projection using a homography matrix [21], it is necessary to apply the transformation to each point of the image. In this case, the transformation can be described in the following general form:

1. Let S' be the original image, and S'' be the transformed image.
2. For each point (x, y) in the image S, the homography matrix H is applied to find the corresponding point (x'', y'') in the image S'':

$$\begin{pmatrix} x'' \\ y'' \\ w'' \end{pmatrix} = H \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{9}$$

3. After applying the homography matrix, the coordinates are normalized to convert them into Euclidean coordinates:

$$x_{\text{norm}} = \frac{x''}{w''} \tag{10}$$

$$y_{\text{norm}} = \frac{y''}{w''} \tag{11}$$

4. The pixel value at the new coordinates $(x_{\text{norm}}, y_{\text{norm}})$ is assigned to the corresponding pixel in the image S''. This process is repeated for each point of the original image S' to form the complete image S'' in the Birds-Eye-View projection.

To determine the position of the vehicle itself on the local map, it is assumed that the point with coordinates $(w/2, h)$ is the vehicle's position point (x_{ego}, y_{ego}). To determine the vehicle's coordinates on the local map (x_{ego}', y_{ego}') using the homography matrix, the function bev is used:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2023 doi:10.20944/preprints202311.1922.v1

$$bev(x,y) = \left(\frac{H_{11}x + H_{12}y + H_{13}}{H_{31}x + H_{32}y + H_{33}}, \frac{H_{21}x_{ego} + H_{22}y + H_{23}}{H_{31}x + H_{32}y + H_{33}}\right) \quad (12)$$

$$(x_{ego}', y_{ego}') = bev(x_{ego}, y_{ego}) \quad (13)$$

In the next step, objects previously removed are marked on the local map so that their geometric characteristics correspond as closely as possible to their real values [22]:

1. Let S'' be the Birds-Eye-View (BEV) projection image, where areas occupied by recognized obstacles have already been replaced with the value 100.
2. Each recognized object is defined by a bounding rectangle with parameters: the rectangle's center $(x_{\text{mid}}, y_{\text{mid}})$, width w, height h, and the object class c.
3. For each bounding rectangle, the coordinates of the two lower points (P_1, P_2) in the BEV projection are calculated:

$$P_1 = bev\left(x_{\text{mid}} - \frac{w}{2}, y_{\text{mid}} + \frac{h}{2}\right) \quad (14)$$

$$P_2 = bev\left(x_{\text{mid}} + \frac{w}{2}, y_{\text{mid}} + \frac{h}{2}\right) \quad (15)$$

1. The area for replacement is calculated as follows:

$$\text{- Left boundary: } x_{\text{left}} = x_{\text{mid}} - \frac{w}{2} \quad (16)$$

$$\text{- Right boundary: } x_{\text{right}} = x_{\text{mid}} + \frac{w}{2} \quad (17)$$

$$\text{- Bottom boundary: } y_{\text{bottom}} = y_{\text{mid}} + \frac{h}{2} \quad (18)$$

$$\text{- Top boundary: } y_{\text{top}} = y_{\text{bottom}} - 2w \quad (19)$$

2. For each point (x, y) in the specified area on the image S'', if it is within the boundaries of the area:

$$x_{\text{left}} \leq x \leq x_{\text{right}} \quad (20)$$

$$y_{\text{top}} \leq y \leq y_{\text{bottom}} \quad (21)$$

the pixel value is replaced with the object class c.

3. These steps are repeated for each recognized object from O on the original image I.

The module for controlling the linear and angular velocity of the autonomous vehicle is responsible for executing control commands based on the output of the path planning module. It regulates the speed and direction of the vehicle's movement, ensuring adherence to the planned route and avoiding collisions. To achieve this, the module uses PID (Proportional-Integral-Derivative) control for precise angular velocity management.

Based on the obtained map, the module plans an optimal route for the autonomous vehicle, considering set goals and constraints. Route planning is carried out using the RRT* algorithm [23] on the local map, where areas free for movement are marked with the value 100, and all other values represent occupied cells. The target point is determined by matching the local map and the global map, considering the vehicle's position on the global map, defined by the tuple $(x_{global}, y_{global}, \theta)$.

Initially, the target direction of movement is determined using the first five points of the path $P_i = (x_i, y_i)\ for\ (i = 1,2,\ldots,5)$, which form part of the vehicle's path. Vectors between these

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2023 doi:10.20944/preprints202311.1922.v1

points are calculated: $\vec{V_i} = P_{i+1} - P_i$ for $i = 1, 2, 3, 4$, and the average direction of these vectors is determined.

$$\theta_{avg} = \frac{1}{n-1}\sum_{i=1}^{n-1} \text{atan2}(y_{i+1} - y_i, x_{i+1} - x_i) \quad (22)$$

This formula assumes that θ_{avg} will be the average value of angles between consecutive pairs of points on the path.

The PID controller is used to minimize the difference between the current orientation angle of the vehicle θ and the target angle θ_{avg}.

The error is defined as $e(t) = \theta_{avg} - \theta$.

The proportional controller calculates the corrective action $u(t)$ as:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)d\tau + K_d \cdot \frac{de(t)}{dt} \quad (23)$$

where K_p, K_i, K_d are the coefficients of the proportional, integral, and differential components of the PID controller, respectively. These coefficients are determined experimentally.

This algorithm allows the autonomous vehicle to automatically adjust its course to follow the designated path while avoiding obstacles, using the direction information determined from the five path points and the PID controller for precise angle control. An emergency stop is executed if no path is found. The vehicle remains stationary until the objects obstructing the path are cleared.

2. Development of software and results

To implement the proposed models, the Webots simulation environment integrated with ROS2 was used. For testing, a scene was developed in which a car and surrounding objects that needed to be driven around were located on an agricultural field: a person, a truck, animals (Figure 1).

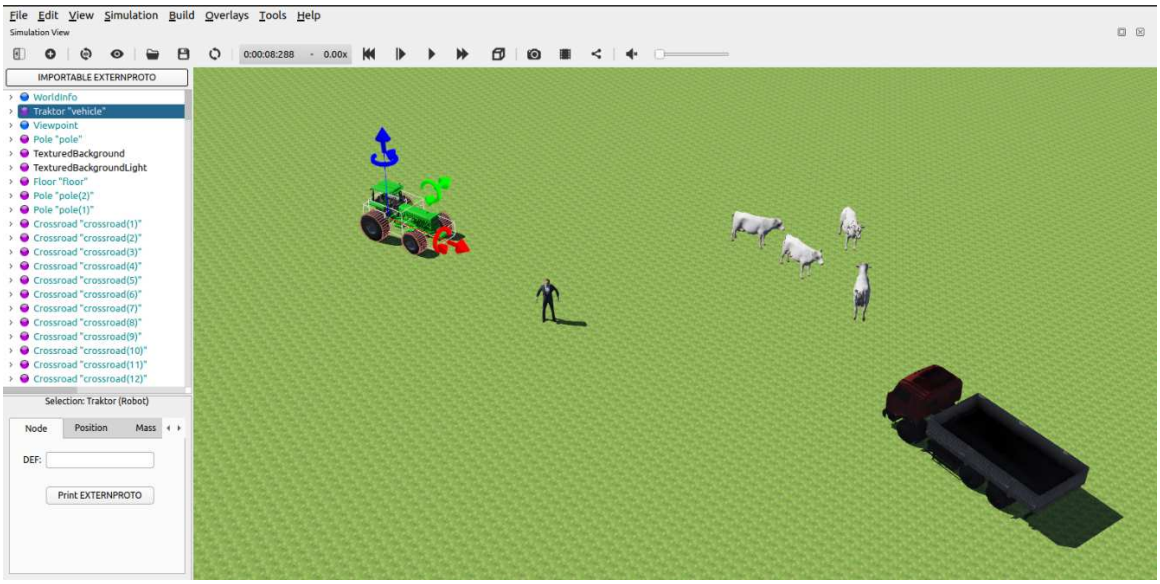

Figure 1. Webots simulator scene.

To control an unmanned vehicle, a software package was created in the ROS2 environment, which includes the following ROS2 nodes and functional modules:

- node_sensors_webots (Vehicle Sensor Driver) – a node that is responsible for collecting data from sensors in the Webots environment and generates messages in the appropriate topics with pre-processing:

  /vehicle/camera/image_color – RGB image from the camera

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2023 doi:10.20944/preprints202311.1922.v1

/vehicle/range_finder/image – image from the depth camera
/vehicle/gps_nav – current global coordinates
/odom – message about the vehicle's coordinates and orientation

- node_localmap – a node that collects data from sensors to implement security functions and calls methods of the MapBuilder class to build a local map and transmit control commands to the car.
- Vehicle controller, which is responsible for sending vehicle control commands to the Webots simulation environment.

A sequence diagram for data processing is presented in Figure 2.

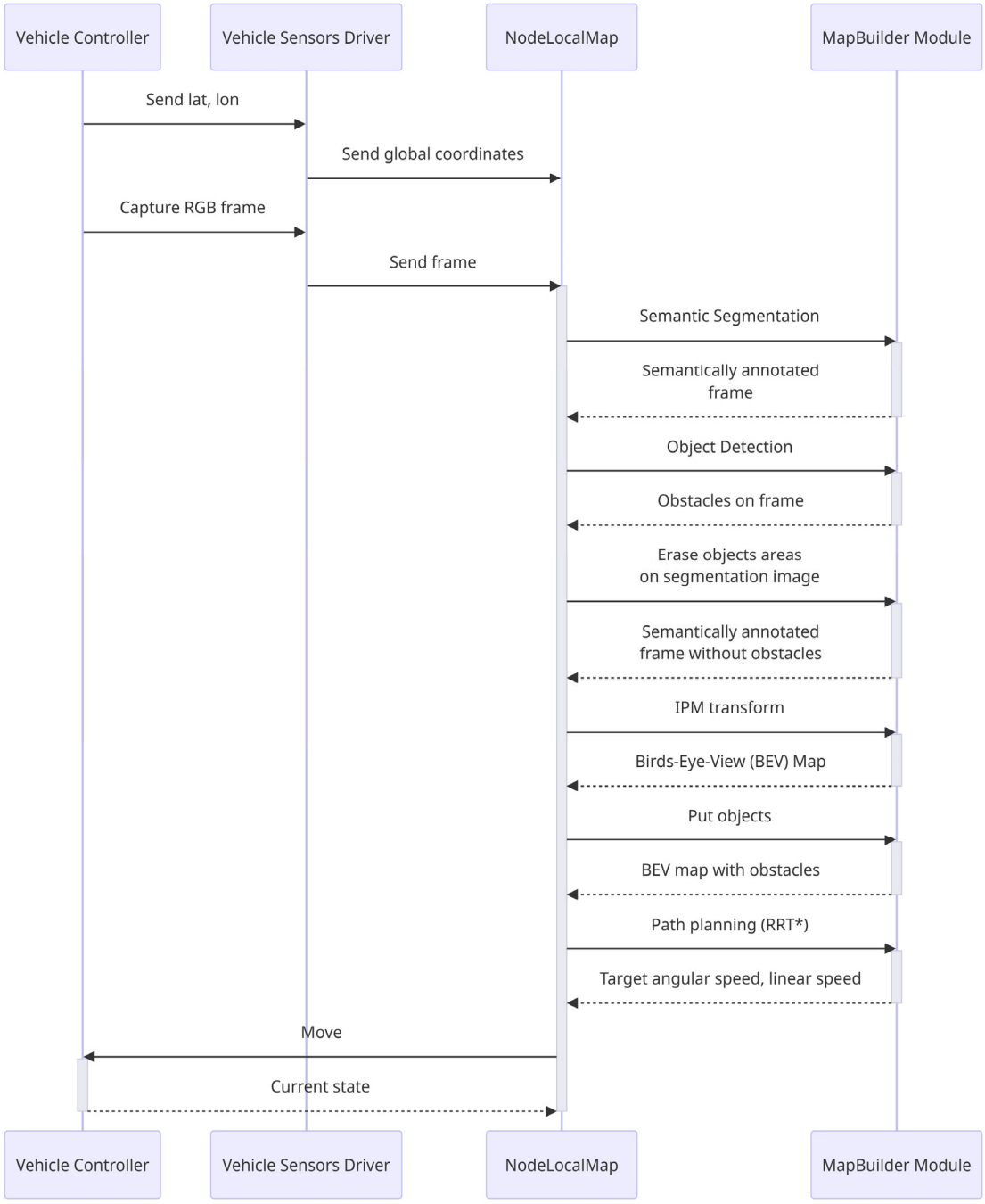

Figure 2. Sequence diagram for obstacle avoidance method.

Once the RGB frame is captured, __process_frame method from node_localmap is invoked. This method performs sequential processing of data as follows.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2023 doi:10.20944/preprints202311.1922.v1

Image preprocessing. The method receives three types of images: a standard RGB image (`image`), a segmentation image (`image_seg`). Each of these images is resized to a specific dimension provided by configuration file. This resizing is crucial for maintaining consistency in object detection and depth perception.

Object Detection (Figure 3). The method uses a pre-trained YOLOv8 (You Only Look Once) model for object detection. This model is known for its efficiency in detecting objects in real-time. The model identifies objects and their locations, returning bounding boxes (`tbs`) that encapsulate each detected object. Before performing Inverse Perspective Mapping (IPM) on an image with semantic segmentation, all objects identified by the object detector as obstacles are cut out (Figure 4). This is done to ensure that known objects do not occupy extra free space area during the projection in Bird's Eye View (BEV). Another necessary transformation is to replace the free space with a value different from 0 (in this case, 100, since no more than 100 object classes are recognized). Then, the coordinates of the detected objects are transformed into the coordinates of the local BEV map using a homography matrix and are displayed on the local map. This allows for more accurate path planning on the local map in the future.

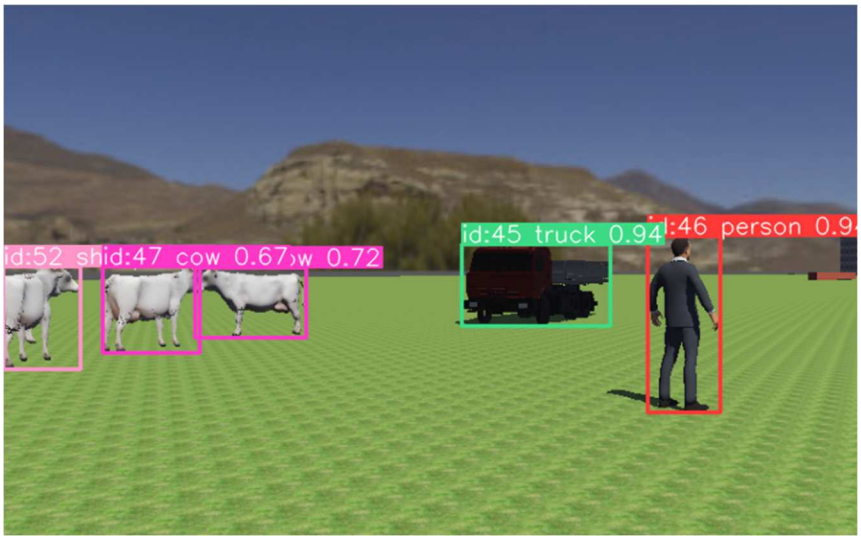

Figure 3. Obstacle detection.

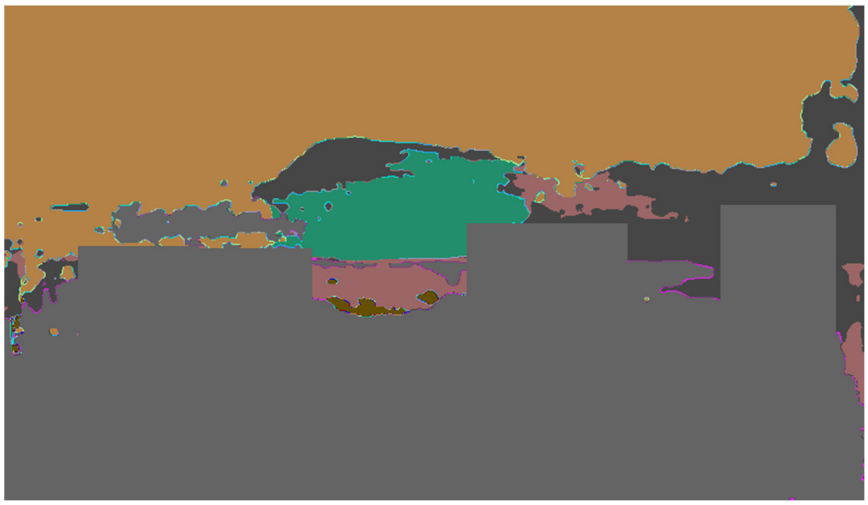

Figure 4. Semantically segmented image with removed obstacles.

After that **t**he coordinates of the detected objects' bounding boxes are transformed to align with the vehicle's perspective in method **put_objects**.

 doi:10.20944/preprints202311.1922.v1

Segmentation image (`image_seg`) undergoes an Inverse Perspective Mapping (IPM) transformation with opencv library. This transformation converts the image to a bird's-eye view, providing a top-down perspective of the road and surroundings (Figure 5). Homography matrix is provided by the configuration file.

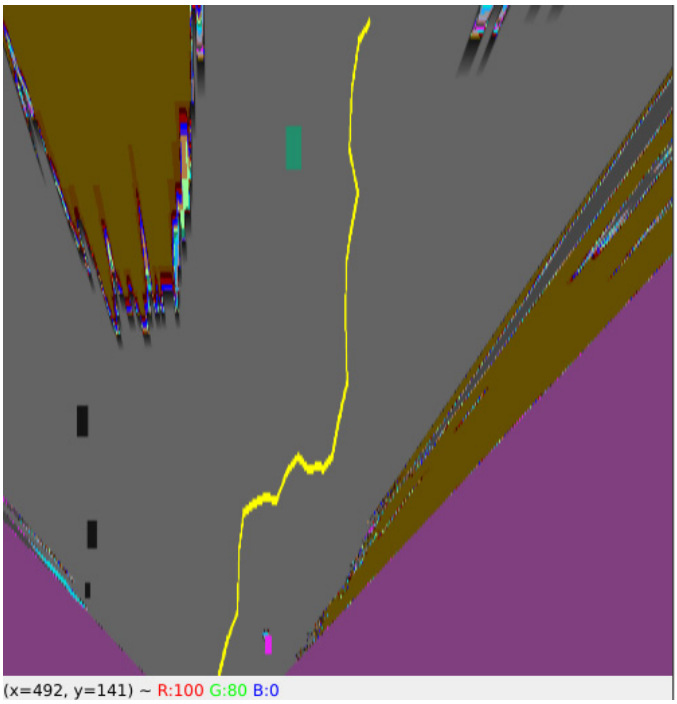

Figure 5. Local map with planned path and obstacles.

The local map is a scene model limited by the sensor's field of view. It is a multi-layered structure with a selected degree of detail, determined by the size of the map cell and the number of such cells in width and depth. Each layer is designed to store different aspects of the scene:

- The map of static obstacles and the road contour RM.
- The map of dynamic obstacles and models for behavioral analysis of these obstacles BM.
- The map of objects affecting the behavior of the autonomous vehicle itself (traffic lights, road signs).
- The local map is necessary for building a high-precision global map, which, in addition to the layer with coordinates of waypoints in the context of the task at hand, stores data about objects affecting the behavior of the autonomous vehicle itself (traffic lights, road signs) along the entire route, for reuse.

Color Processing and Cropping of IPM Image. The IPM image is processed for color to enhance visibility and differentiation of various elements and is cropped to focus on relevant areas for navigation, typically the road ahead and immediate surroundings.

The transformed **bounding boxes are drawn onto the colored IPM image**. This visualizes the location and size of detected objects from a top-down view. The method marks the current viewpoint and the target destination. This is essential for path planning.

Path Planning Using RRT. The Rapidly-exploring Random Tree (RRT) algorithm is used for path planning. RRT is effective in complex environments as it quickly explores the space while avoiding obstacles. If a viable path is found, it is shown on the IPM image, showing the proposed route from the current location to the target (yellow line on Figure 5).

In general, the RRT* algorithm is executed as follows:

- A tree consisting of a single vertex - the starting point - is created.
- On each iteration of the algorithm, a random point in the state space is generated. This point will be a potential new node in the tree.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2023 doi:10.20944/preprints202311.1922.v1

- It is necessary to find the vertex in the existing tree that is closest to the generated random point. This vertex will be called the "nearest vertex" or "nearest neighbor."
- A new vertex is created, connecting the nearest vertex with the generated point. This creates a new edge in the tree.
- After adding the new vertex, the cost and distance to all vertices in the tree that could be reached through the new edge are recalculated. This includes the nearest neighbors and their potential neighbors. If the recalculated cost to a certain vertex is less than its current cost, the cost of that vertex is updated, and the cost for its neighbors is recalculated to reflect the new information.
- Steps 2 through 5 are repeated until the target vertex is reached (or until the maximum number of iterations is reached).
- Once the target vertex is reached, the optimal path is reconstructed, moving from the target vertex to the starting point at the lowest cost, using the recalculated costs.
- The algorithm completes execution when the optimal path is found or the maximum number of iterations is performed.

The constructed path is not perfect (Figure 5., the constructed path is marked in yellow), but it allows for good obstacle avoidance. One of the hyperparameters of the algorithm is the maximum number of iterations. Through experimentation, a value of 1000 has been established.

Since the local map is updated with each frame received from the camera, the algorithm is constantly running. From the constructed path, target values of angular velocity are extracted, which are then transmitted to the car's controller.

Displaying Results. The method uses OpenCV (`cv2.imshow`) to display the processed images in different windows. This includes the original images, the IPM image with the path, and images with bounding boxes.

The emergency stop function is based on a pathfinding algorithm: if it is impossible to construct a path from the current point to the target, a command is sent to set the linear velocity to 0. This means stopping the car. Controlling the angular velocity allows for maneuvering around obstacles. Since the path is rebuilt with each frame, the time-dependent values of angular velocity, determined by formula (22), exhibit highly noisy data, which can lead to non-smooth movement. To ensure smooth steering control, a low-pass filter is applied to the values of angular velocity. Figure 6 shows the calculated values of the angle, as well as the filtered signal.

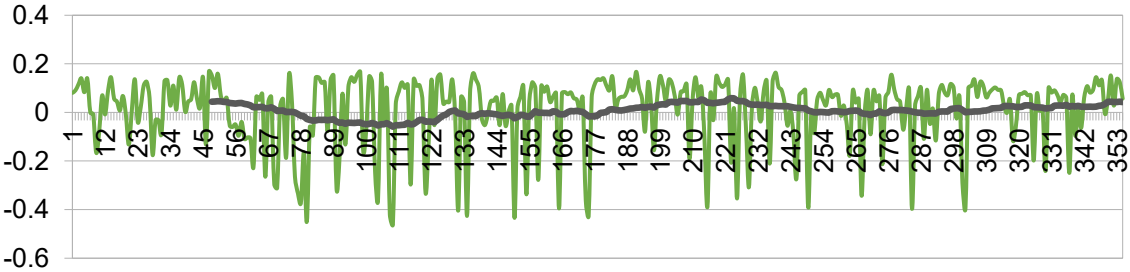

Figure 6. Angular speed corrections without filtering.

4. Discussion

This article describes models and software that enable an unmanned vehicle to circumnavigate obstacles and stop urgently if a collision cannot be avoided. The main idea is to combine the results of processing an RGB camera frame by several methods (object detection, semantic segmentation, inverse perspective mapping), the combined results of which allow for greater accuracy in determining the distance to objects. The result of data processing is an annotated local map, which is used for path planning and determining the steering wheel angle, which determines the angular velocity using a PID controller. To smooth changes in angular velocity, the data are smoothed with a low-pass filter.

The implemented method is relevant in the tasks of designing unmanned ground vehicles, especially those retrofitted for autonomous driving, where the low cost of the equipment used is important, as only cameras are used as sensors.

Further development of methods is planned. It is important to consider information not only from the front camera but also from the side cameras. This can be done using 360-degree cameras. Then an important aspect of safety and route planning is building a full map using SLAM. Since depth information is inaccurate, the development of a SLAM algorithm using semantic information about the objects surrounding the car is promising.

Overall, the use of only cameras allows building a control system for an unmanned vehicle on cheap sensors. Such systems are less likely to be used in urban environments, where the complexity of the road scene is high, but can be useful in agriculture or in manufacturing, where traffic is not so complex.

Supplementary Materials: The following are available online at https://github.com/ulstu/cad-self-driving/.

Author Contributions: Conceptualization, K.S.; methodology, K.S.; software, K.S., R.Z., V.M.; validation, I.K.; resources, I.K.; data curation, K.S.; writing—original draft preparation, K.S.; writing—review and editing, R.Z.; project administration, K.S.; funding acquisition, K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by Russian Science Foundation grant No. 23-11-00265, https://rscf.ru/project/23-11-00265/.

Data Availability Statement: Not applicable.

Acknowledgments: Authors are grateful for a GAZ, UAZ, SimbirSoft, Avion, «Precision farming systems» and RITG companies.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Kuutti, S.; Bowden, R.; Jin, Y.; Barber, P.; Fallah, S. A Survey of Deep Learning Applications to Autonomous Vehicle Control. IEEE Transactions on Intelligent Transportation Systems 2021, 22, 712–733, doi:10.1109/TITS.2019.2962338.
2. Peng, Chen, Peng Wei, Zhenghao Fei, Yuankai Zhu, and Stavros G. Vougioukas. "Optimization-Based Motion Planning for Autonomous Agricultural Vehicles Turning in Constrained Headlands." arXiv preprint arXiv:2308.01117 (2023).
3. Sensors | Free Full-Text | Robust Fusion of LiDAR and Wide-Angle Camera Data for Autonomous Mobile Robots Available online: https://www.mdpi.com/1424-8220/18/8/2730 (accessed on 22 November 2023).
4. Liu, L., Lu, S., Zhong, R., Wu, B., Yao, Y., Zhang, Q., & Shi, W. (2020). Computing systems for autonomous driving: State of the art and challenges. IEEE Internet of Things Journal, 8(8), 6469-6486.
5. Stoma, M.; Dudziak, A.; Caban, J.; Droździel, P. The Future of Autonomous Vehicles in the Opinion of Automotive Market Users. Energies 2021, 14, 4777, doi:10.3390/en14164777.
6. GPS Systems for Agriculture | Patchwork Technology | Usk Available online: https://www.patchworkgps.com (accessed on 22 November 2023).
7. Autonomous Agriculture Solutions | Trimble Autonomy | Autonomy Available online: https://autonomy.trimble.com/en/agriculture (accessed on 22 November 2023).
8. Ag Leader | Agricultural Technology Solutions Available online: https://www.agleader.com/?locale=en (accessed on 22 November 2023).
9. Solution - Smajayu Available online: https://www.smajayu.com/category/solution/ (accessed on 22 November 2023).
10. High Value Crops Smart Farming. Available online: https://fieldin.com/home/ (accessed on 22 November 2023).
11. Bluewhite. Available online: https://www.bluewhite.co/ (accessed on 22 November 2023).
12. Braun Mechanical Engineering. Available online: https://braun-maschinenbau.info/ (accessed on 22 November 2023).

13. GPX Intelligence - Track What Matters Most Available online: https://gpx.co/ (accessed on 22 November 2023).
14. Home - Thorntek Available online: https://thorntek.com.au/, https://thorntek.com.au/ (accessed on 22 November 2023).
15. Autonomous Tractor Fleets. Available online: https://www.bearflagrobotics.com/ (accessed on 22 November 2023).
16. Autonomous Tractor Upgrade | Sabanto Steward Available online: https://sabantoag.com/ (accessed on 22 November 2023).
17. Cognitive Pilot - Autonomous Driving Technologies for Ground Transport Available online: https://en.cognitivepilot.com/ (accessed on 22 November 2023).
18. BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation Available online: https://hanlab.mit.edu/projects/bevfusion (accessed on 22 November 2023).
19. Jocher, G.; Chaurasia, A.; Qiu, J. YOLO by Ultralytics Available online: https://github.com/ultralytics/ultralytics (accessed on 22 November 2023).
20. Ekzhang/Fastseg: PyTorch Implementation of MobileNetV3 for Real-Time Semantic Segmentation, with Pretrained Weights & State-of-the-Art Performance. Available online: https://github.com/ekzhang/fastseg (accessed on 22 November 2023).
21. Fu, Zhongtao, et al. "Homography matrix based trajectory planning method for robot uncalibrated visual servoing." 2023 IEEE International Conference on Real-time Computing and Robotics (RCAR). IEEE, 2023.
22. Liu, Chang, et al. "YOLO-BEV: Generating Bird's-Eye View in the Same Way as 2D Object Detection." arXiv preprint arXiv:2310.17379 (2023).
23. Xanthidis, Marios; Esposito, Joel M.; Rekleitis, Ioannis; O'Kane, Jason M. (2020-12-01). "Motion Planning by Sampling in Subspaces of Progressively Increasing Dimension". Journal of Intelligent & Robotic Systems. 100 (3): 777–789. doi:10.1007/s10846-020-01217-w. ISSN 1573-0409. S2CID 3622004