Завдання передаварійного інтелектуального керування робота автономних транспортних засобів є дуже складною проблемою, особливо передаварійні умови транспортних засобів і в точках перетину в умовах реального часу.

Метою даного дослідження є розробка нового штучного інтелектуального адаптивного регулятора для системи передаварійної безпеки автономних транспортних засобів, а також модуля розпізнавання транспортних засобів та тестування в MATLAB, включаючи деякі деталізовані модулі. Були поставлені наступні завдання: пошук об'єктів за даними датчиків (Лідар, Радар), контроль швидкості та рульового управління, розпізнавання транспортних засобів з використанням згорткової нейронної мережі та Alexnet.

У даній дослідницькій роботі було реалізовано обробку зображень та лідарних даних в режимі реального часу. Спочатку була представлена система реального часу, яка складається з комплексних модулів, а саме: модулі виявлення тривимірних об'єктів, групування та пошуку об'єктів, видалення землі, глибинного навчання з використанням згорткових нейронних мереж. Починаючи з модуля найближчого транспортного засобу, завданням було – знайти найближчий попереду автомобіль і вважати його основною перешкодою.

У статті представлена адаптивна передаварійна система керування швидкістю та розпізнавання транспортних засобів. Модуль адаптивної передаварійної системи керування швидкістю залежить від даних глибинного навчання та лідарного датчика, які призначені для управління безрозсудною поведінкою водія на дорозі шляхом регулювання швидкості транспортного засобу для підтримки безпечної відстані від об'єктів попереду (таких як автомобілі, люди, велосипед або будь-який інший об'єкт), коли водій намагається підвищити швидкість. Наразі, модуль розпізнавання транспортних засобів виявляє і розпізнає транспортні засоби навколо автомобіля

Ключові слова: глибинне навчання, набір даних лідарного датчика, алгоритми K-мірного дерева, хмара точок, модуль розпізнавання транспортних засобів

UDC 621
DOI: 10.15587/1729-4061.2018.141298

A HYBRID LIAR/RADAR-BASED DEEP LEARNING AND VEHICLE RECOGNITION ENGINE FOR AUTONOMOUS VEHICLE PRECRASH CONTROL

Bassant Mohamed Elbagoury
PhD
Department of Artificial Intelligence and Robotics
Humboldt University in Berlin
Unter den Linden, 6, Berlin, Germany, 10099
E-mail: bassantai@yahoo.com
Rytis Maskeliunas
Professor
Department of Multimedia Engineering
Faculty of Informatics
Kaunas University of Technology
K. Donelaičio str., 73, Kaunas, Lithuania, 44249
Abdel Badeeh Mohamed M. Salem
PhD, Professor
University of Economics – Varna
Research Institute of the University of Economics – Varna
Knyaz Boris I blvd., 77, Varna, Bulgaria, 9002
E-mail: abmsalem@yahoo.com

1. Introduction

How many accidents we see every day due to car accidents? There are many causes to these accidents such as: Driver's inattention or poor insight about certain spots around the vehicles (i.e. blind spots) or what attracts us the most, the drivers with visual disorders such as, night blindness, color blindness and so on. Egypt loses about 12,000 lives due to road traffic crashes every year. It has a road traffic fatality rate of 42 deaths per 100,000 populations. This huge percentage was an inspiration behind the proposed system, where we hope that the following objectives to be met, to save as much lives on the road as we can. In this paper, we propose an intelligent system aiming at reducing the driver's workload by assisting the driver through interpreting the environment autonomously and hence supporting the driver. The proposed system features two main modules that have our concerns which are: Adaptive cruise pre-crash system and vehicle recognition each of which will be discussed in the upcoming sections. The goal of this research is to develop a complete intelligent adaptive controller for autonomous vehicle Pre-Crash system.

The coming section describes the main modules and experimental results of the presented research.

2. Literature review and problem statement

Currently, a collision avoidance system is a system of sensors that is placed within a car to warn its driver of any dangers that may lie ahead on the road. However, Recent Research is still missing Pre-Crash Planning, which needs Intelligent (Adaptive) Controller to be embedded for complete Car Safety and path-planning. The main research problem is focused on two main goals, which are path planning and some cases of intersection safety. Intersection safety covers applications related to approaching or passing intersections, with emphasis on the cooperation between vehicles, while Path planning problem usually exists in an environment which has many obstacles and constraints. The problem has been proven to be NP-Hard problem [1–6]. The path planning technology is an important aspect of AI and robotics. A star algorithm is a kind of path planning method which is applicable to the situation that the global environmental information is already known.

The frequency of vehicle crashes caused due to human errors (such as driver inattention or fatigue or the poor insight for the blind spots) can be decreased with the aid of an automatic system providing some mechanisms to avoid such accidents (Fig. 1).

Fig. 1. Truck Accident

An autonomous car (driverless car, self-driving car, robotic car) is a vehicle that is capable of sensing its environment, and navigating without human input. Autonomous vehicles detect surroundings using radar, GPS and computer vision. It is fundamentally defined as a passenger vehicle that drives by itself. It is also referred to as an autopilot, driverless car, auto-drive car, most prototypes that have been built so far performed automatic steering that were based on sensing the painted lines in the road. Today's researchers are using sensors and advanced software together with other custom-made hardware in order to assemble autonomous cars [7–10].

Although the prototypes seem to be very successful, a fully autonomous car that is reliable enough to be on the streets has not been constructed yet. This is mostly because of the difficulties involved in controlling a vehicle in the unpredictable traffic conditions of urban areas. While better hardware is being developed there are important limitations on the artificial intelligence side of the research. It would be fair to say that the future of the autonomous cars mostly depends on the development of better artificial intelligence software. On the other hand, most hardware that is being used on the cars seems to be doing well in terms of reliability, response time and accuracy.

Intelligent or Driverless cars/vehicles will be an inevitable thing in the future, thus we should focus our efforts in revitalizing the technologies and methodologies used to bring their deployment on a sooner timeline than anticipated.

Not only Fortune 500 Companies like Google and Tesla are undergoing research in this field, but they continuously find new ways every day to make cars fully autonomous and self-reliant.

Two main debacles set in this field are related to A) Path & Trajectory Planning and Calculations & B) Sensory and Environmental Awareness of surroundings. These two are the two main things we are going to focus on in our research.

Few of the most notable developers of autonomous vehicles and their systems are:

– Ford [11–15];
– Tesla;
– KITTI (German Research – University Based) [14].

Both Ford & KITTI [14] are still in the early stages of development as tests are inducte each and every other day to ensue optimizations are applied to their algorithms properly and that they would be able to push and scale their methodologies for the greater good of autonomous driving.

Tesla is ahead in the sense that they have been investing more resources and time into this field for a long time now; we see them as one of the first to bring Level 1 Autonomy to the table of driverless cars. Because not only they develop and patent their hardware that will eventually lead them to full autonomy, they also think how would they make it safe for every human riding a car that they develop. A car that they would eventually sell to the average end-consumers which means they will be the first to tackle this field in a way appropriate to real life application.

So, without further ado, we will tackle both the issues we aforementioned at the start. And that is the path planning algorithm and the environment sensory subsidiaries. We will be implementing a LiDAR based Path Planning Algorithm and alongside this we will implemented Deep Learning for Vehicle Recognition. Other algorithms are required for Environment mapping and localization like SLAM [9] and EKF [16].

3. The aim and objectives of the study

This research paper aims at presenting a new artificial intelligent adaptive controller for autonomous vehicle Pre-Crash system along with vehicle recognition module and tested in MATLAB including some detailed modules.

To achieve the objective, the following tasks were set:

1. Finding Objects in sensor Data (LiDAR. RADAR).
2. Speed and Steering control.
3. Vehicle Recognition using convolution neural network and Alexnet.

4. Proposed System Architecture and Modules Engines

Regarding to our presented research paper, we are mostly interested in the implementation of the autonomous vehicles modules (Vehicle Detection and Recognition, Speed and Steering control).

How we do it scientifically? We use LIDAR and cameras with wide-angle lens that captures frames of high speed

videos, and then we analyze each frame by applying different algorithms. Main system architecture is shown in Fig. 2, each will be described in details in coming sections.

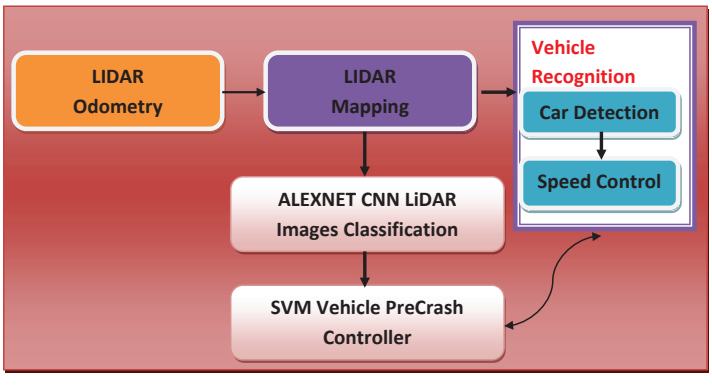


Fig. 2. Artificial Intelligent Engine for Vehicle PreCrash

4. 1. LIDAR Odometry Algorithm: Processing data from LIDAR dataset

Lidar Odometry Algorithms is shown in Fig. 3, it is based on LOAM algorithm presented by Zhang [ref.]. This algorithm took the point cloud from the last Scan as inputs, Lm, Is the growing point cloud of the present Scan, Lm+1, and constitute a shift from the last recursion, Pk l+1. If a new sweep is started, Pk l+1 sets to zero (line 4–6). Then, the algorithm works on feature extraction from Lm+1 to construct Nm+1 and Im+1 in line 7. For each feature point, its coincidence is found in Lm (line 9–19). The motion estimation is being adapted to a strong fitting [27]. In line 15, the algorithm assigns a bisquare weight for each feature point. The feature points that have larger distances to their coincidences which are designated with more small weights and points of landmarks with greater distances than the threshold are considered extreme values and are assigned zero weights. Then, in line 16, the pose transform is updated for one repetition. The nonlinear optimization process ends if convergence is reached or the maximum repetition no. is met. If the algorithm reaches the end of the scanning process, Lm+1 is reestablishment to new time stamp Lm+2 using the predestined motion during the scan. Further, only the transform Pk l+1 is returned for the new next round of The recursion. LiDAR processing results are presented in coming sections.

4. 2. LiDAR Sensor Dataset and Preprocessing

Regarding to our presented research paper, we are mostly interested in the implementation of the autonomous vehicles modules (Vehicle Detection and Recognition, Speed and Steering control).

How we do it scientifically? We use Light Detection and Ranging (LIDAR) and cameras with wide-angle lens that captures frames of high speed videos, and then we analyze each frame by applying different algorithms that will be described in details later on (Fig. 4).

Main components of the project:

1. Setup.

2. Coordinate Systems.

The coordinate systems are defined the following way, where directions are informally given from the driver's view, when looking forward onto the road:

– Camera: *x*: right, *y*: down, *z*: forward;

– Velodyne: *x*: forward, *y*: left, *z*: up;

– GPS/IMU: *x*: forward, *y*: left, *z*: up.

All coordinate systems are right-handed.

3. Lidar.

Lidar (also called LIDAR, LiDAR, and LADAR) is a surveying method that measures distance to a target by illuminating that target with a pulsed laser light, and measuring the reflected pulses with a sensor. Differences in laser return times and wavelengths can then be used to make digital representations of the target (Fig. 5).

– *LiDAR stands for Light Detection and Ranging.*

– *LiDAR is the most reliable sensor for object detection.*

– *High-speed detection & processing → real-time detection.*

– *Measures Time of Flight (TOF), the round-trip travel time for a laser pulse reflected off obstacles.*

LiDAR is more reliable than any existing sensing solution, including:

– Radar (all types);

– Video (all types, including IR);

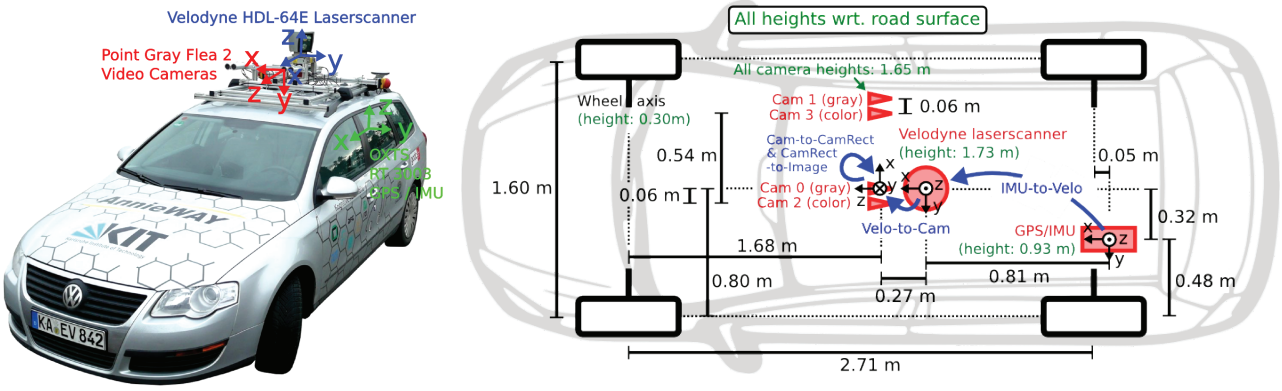
– Video+Radar;

– Video+Ultrasonic Sensors;

– Stereoscopic Cameras.

Step 1 : Input : Lm, Lm+1, Pk l +1 from the last recursion
Step 2 : output : Lm+1, newly computed Pk l+1
Step 3 : begin
Step 4 : if at the beginning of a sweep then
Step 5 : Pk l+1 0; Step 6 : end
Step 7 : Detect edge points and planar points in LM+1, put the points in N m+1 and I m+1, respectively;
Step 8 : for a number of iterations do
Step 9 : for each edge point in N m+1 do
Step 10 : Find an edge line as the correspondence, then compute point to line distance based on (9) and stack the equation to (11); Step 11 : end
Step 12 : for each planar point in I m+1 do
Step 13 : Find a planar patch as the correspondence, then compute point to plane distance based on (10) and stack the equation to (11); Step 14 : end
Step 15 : Compute a bisquare weight for each row of (11);
Step 16 : Update PLM+1 for a nonlinear iteration based on (12);
Step 17 : if the nonlinear optimization converges then
Step 18 : Break; Step 19 : end ; Step 20 : end
Step 21 : if at the end of a sweep then
Step 22 : Reproject each point in L m+1 to Pk+2 and form Pk+1;
Step 23 : Return Pl k +1 and L m+1; Step 24 : end Step 25 : else
Step 26 : Return P k l+1; Step 27 : end

Fig. 3. LiDAR Odometry Algorithm [12]



a *b*

Fig. 4. Fully equipped vehicle_2: *a* — vehicle_1, *b* — vehicle_2

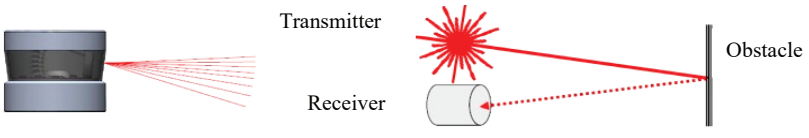


Fig. 5. LIDAR

Lidar applications presented in Fig. 6–9.

4. Stereo camera:

A stereo camera is a type of camera with two or more lenses with a separate image sensor or film frame for each lens. This allows the camera to simulate human binocular vision, and therefore gives it the ability to capture three-dimensional images, a process known as stereo photography.

Stereo cameras may be used for making stereo views and 3D pictures for movies, or for range imaging.

The distance between the lenses in a typical stereo camera (the intra-axial distance) is about the distance between one's eyes (known as the intra-ocular distance) and is about 6.35 cm, though a longer base line (greater inter-camera distance) produces more extreme 3-dimensionality.

Industrial – Factory/Warehouse Automation

Fig. 6. Lidar applications in Industrial

Simultaneous Localization & Mapping

Fig. 7. Lidar application in Simultaneous & mapping

Safety – Monitoring, Security – Surveillance

Fig. 8. Lidar application in Safety

Automotive – Full Awareness

Fig. 9. Lidar application in Automotive — full

5. Proposed System Architecture

Proposed System modules Steps presented in Fig. 10–12.

1) Reading data from LIDAR dataset.

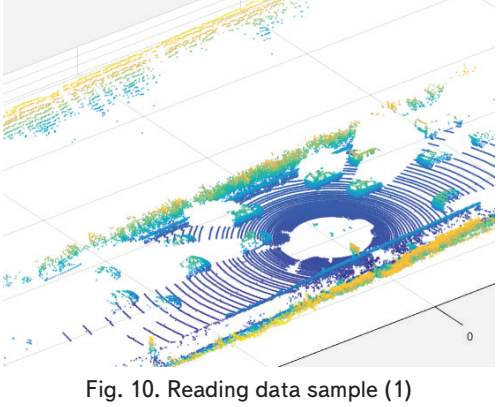


Fig. 10. Reading data sample (1)

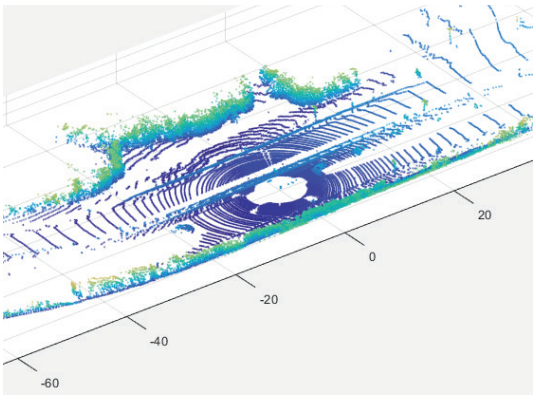


Fig. 11. Reading data sample (2)

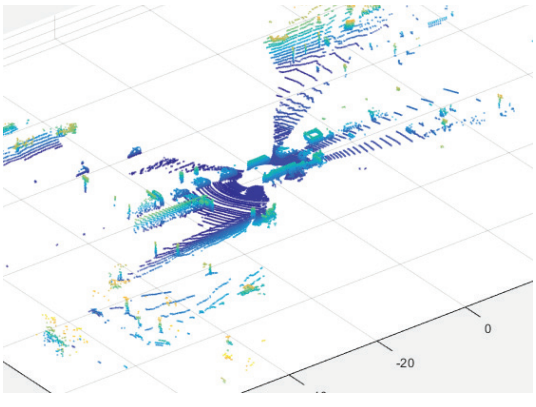


Fig. 12. Reading data sample (3)

2) Removing ground from the LIDAR dataset (Fig. 13–15).

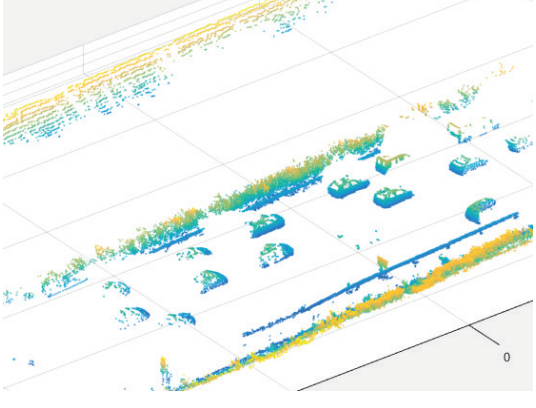


Fig. 13. Removing ground sample (1)

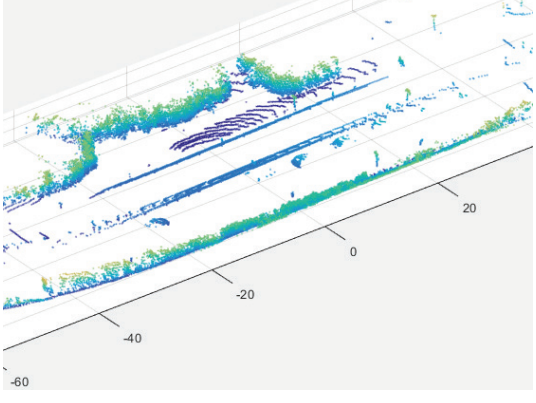


Fig. 14. Removing ground sample (2)

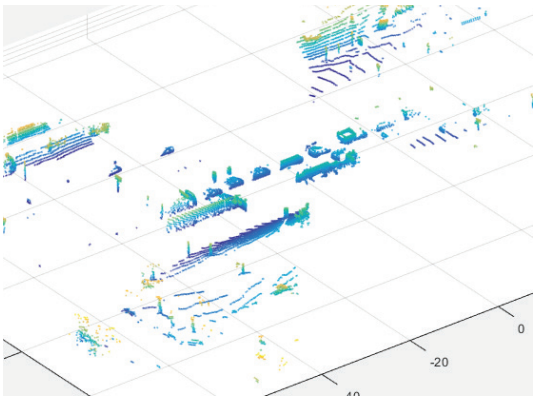


Fig. 15. Removing ground sample (3)

3) Clustering data based on nearest group points.

4) Projecting 3d points on camera and display 2d bounding boxes around them (Fig. 16–18).

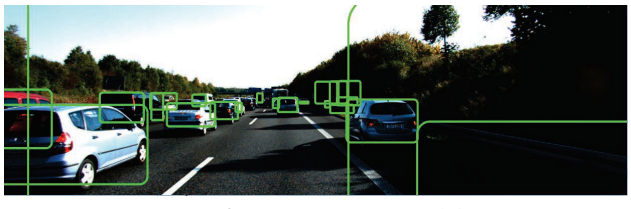


Fig. 16. Projecting sample (1)

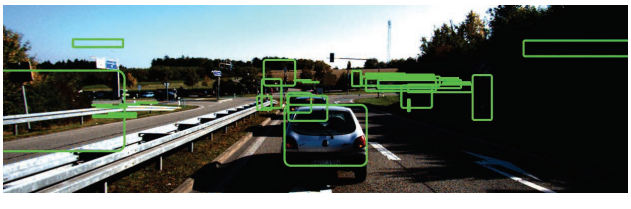


Fig. 17. Projecting sample (2)

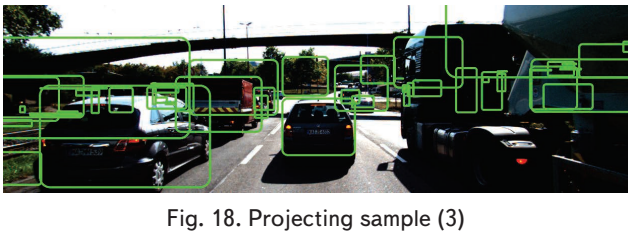


Fig. 18. Projecting sample (3)

5) Cropping object images.

6) Classifying cropped images between vehicles and non-vehicles using deep convolutional neural network with a support vector machine classifier.

7) Displaying only vehicles along with their distances from camera (Fig. 19–21).

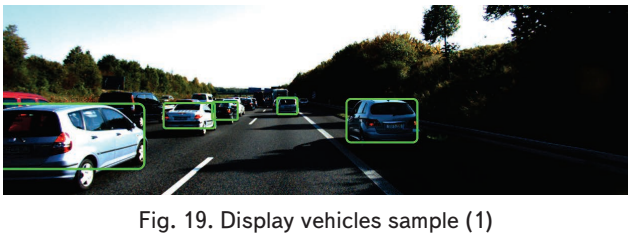


Fig. 19. Display vehicles sample (1)

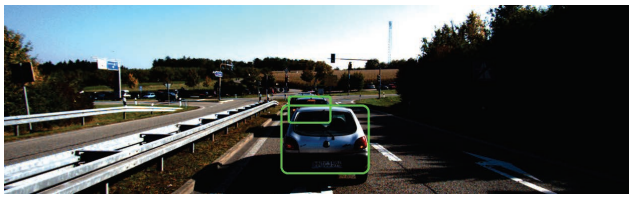


Fig. 20. Display vehicles sample (2)

Fig. 21. Display vehicles sample (3)

6. Proposed Algorithms

6. 1. Ransac algorithm

Random sample consensus (RANSAC) is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers are to be accorded no influence on the values of the estimates. Therefore, it also can be interpreted as an outlier detection method. It is a non-deterministic algorithm in the sense that it produces a reasonable result only with a certain probability, with this probability increasing as more iterations are allowed. The algorithm was first published by Fischer and Boles at SRI International in 1981. They used RANSAC to solve the Location Determination Problem (LDP), where the goal is to determine the points in the space that project onto an image into a set of landmarks with known locations.

A simple example is fitting of a line in two dimensions to a set of observations. Assuming that this set contains both inliers, i. e., points which approximately can be fitted to a line, and outliers, points which cannot be fitted to this line, a simple least squares method for line fitting will generally produce a line with a bad fit to the inliers. The reason is that it is optimally fitted to all points, including the outliers. RANSAC, on the other hand, can produce a model which is only computed from the inliers, provided that the probability of choosing only inliers in the selection of data is sufficiently high. There is no guarantee for this situation, [citation needed] [clarification needed] however, and there are a number of algorithm parameters which must be carefully chosen to keep the level of probability reasonably high (Fig. 22).

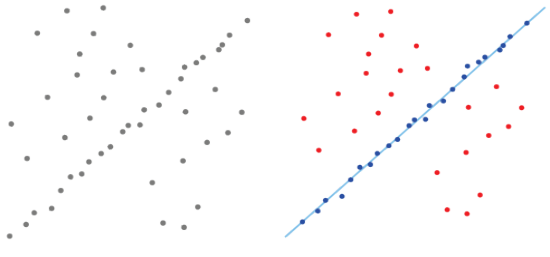


Fig. 22. Ransac algorithm

6. 2. *k-d* tree algorithm:

In computer science, a *k-d* tree (short for *k-dimensional tree*) is a space-partitioning data structure for organizing points in a *k*-dimensional space. *k-d* trees are a useful data structure for several applications, such as searches involving a multidimensional search key (e. g. range searches and nearest neighbor searches). *k-d* trees are a special case of binary space partitioning trees

The *k-d* tree is a binary tree in which every node is a *k*-dimensional point. Every non-leaf node can be thought of as implicitly generating a splitting hyperplane that divides the space into two parts, known as half-spaces. Points to the left of this hyperplane are represented by the left subtree of that node and points right of the hyperplane are represented by the right subtree. The hyperplane direction is chosen in the following way: every node in the tree is associated with one of the k-dimensions, with the hyperplane perpendicular to that dimension's axis. So, for example, if for a particular split the «*x*» axis is chosen, all points in the subtree with a smaller «*x*» value than the node will appear in the left subtree and all points with larger «*x*» value will be in the right subtree. In such a case, the hyperplane would be set by the *x*-value of the point, and its normal would be the unit *x*-axis

A 3-dimensional *k-d* tree. The first split (the red vertical plane) cuts the root cell (white) into two sub cells, each of which is then split (by the green horizontal planes) into two sub cells. Finally, those four cells are split (by the four blue vertical planes) into two sub cells. Since there is no more splitting, the final eight are called leaf cells.

7. Vehicle Recognition Module

This module consists of 2 parts:

– Extracting features from input image.

– Classifying those features between 2 categories vehicles and nonvehicle.

The training dataset contains more than 80,000 images categorized into 8 categories as following:

– 'Car';
– 'Person (sitting)';
– 'Cyclist';
– 'Van';
– 'Tram';
– 'Truck';
– 'Misc.';
– 'Pedestrian'.

We are interested only into the car category so we classify images into cars and non-cars.

Sample input images for vehicles (Fig. 23).

Fig. 23. Vehicles sample

Sample input images for Non-vehicles (Fig. 24).

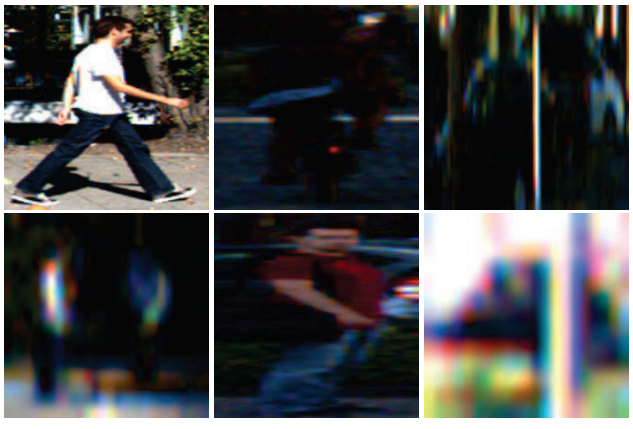


Fig. 24. Non-vehicles sample

8. Speed Planning

In trajectory generation for unstructured environment, or even structured with lake of speed information we need another step predict speeds on different parts of the path. some planning algorithm optimize for speed and position in the same step as in DARPA Urban challenge winner BOSS [5] also in some situation like intersections and city streets tight curves speed profiling will be useful for smoother and safer autonomous drive.

This step is executed after smoothing first step and simulation step.

The main idea is simple: we analyze the curvature along the generated trajectory when curvature is small so it is a tight turn and speed should be small and vice versa.

Parameter of this algorithm:

$s_{\max}$: Maximum speed limit.

N_{curve}: Number of points used for curvature calculation. By default we use 3, but according to how many points is generated in the smoothing step, this could produce misleading results, that why we check if all N_{curve} does not cover more than the cat wheel base length.

w: smoothness weight, to control how driving style how smooth driving through curves will be. If we can collect a lot of data with different driving style we can define high level parameter to control this weight (race, sports, comfort, super comfort). By applying polynomial regression this could be easily predicted.

9. Deep Learning: Convolutional Neural Networks ALEXNET

Alexnet is a convolutional neural network, it was trained over 1.3 million images and can classify between 1000 categories.

The neural network, which has 60 million parameters and 500,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and two globally connected layers with a final 1000-way SoftMax (Fig. 25).

We use alexnet only to extract features from input images.

We take the output from layer number 23 which has output of 4096 values then we use these values as input to a support vector machine to classify those features.

What is a Support vector machine?

In machine learning, support vector machines (SVMs, also support vector network**s**) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.

New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data are not labeled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The clustering algorithm which provides an improvement to the support vector machines is called support vector clustering and is often [citation needed] used in industrial applications either when data are not labeled or when only some data are labeled as a preprocessing for a classification pass.

More formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier (Fig. 26).

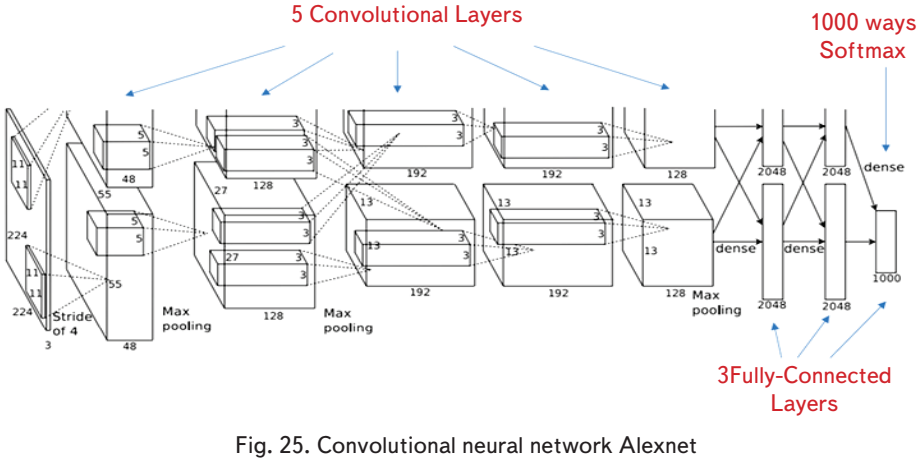


Fig. 25. Convolutional neural network Alexnet

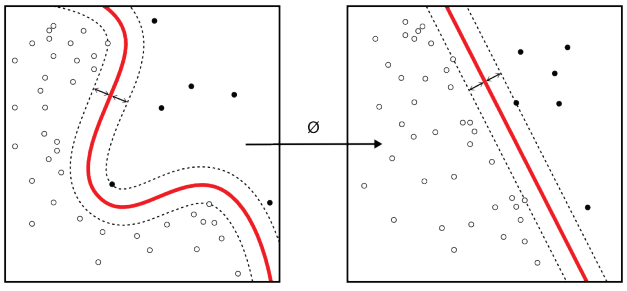


Fig. 26. Support vector machines (SVM)

10. Discussion

10. 1. System Implementation and Experimental Results

We have used MATLAB to implement our functions which consist of modules:

– Reading dataset from (Lidar/camera);

– Remove ground from Lidar;

– Cluster 3d points into objects based on Euclidean distance;

– Project each object on camera and draw a bounding box;

– Crop each object and extract its features using ALEXNET;

– Classify features using SVM.

«Remove ground»

This function removes ground from a LIDAR frame (Fig. 27).

«Cluster 3d»

This function divides 3d points from LIDAR into objects based on distance between points.

«Nearest object»

This function gets the nearest heading car.

«Training images preparation»

This function crops labeled images and prepare them to be the same size as AlexNet input size (Fig. 28).

«Train neural network»

This function trains the SVM classifier on the features obtained from ALexNet.

«Detect car»

This function detects heading car to get its distance and velocity (Fig. 29).

«Speed limit detector»

This function calculates self-speed to estimate the Braking distance (Fig. 30).

«Vehicle recognition»

This function detects and recognizes cars in the scene (Fig. 31).

```
function [out,in] = gp_removeGround(ptCloud,maxDistance)
pc = pointCloud(ptCloud);
% Crop the point cloud to only contain points within the specified region.

% Fit the ground plane.
% maxDistance = 0.2; % in meters
referenceVector = [0, 0, 1];
[~, inPlanePointIndices, outliers] = pcfitplane(pc, maxDistance,
referenceVector);
% colorLabels(inPlanePointIndices) = greenIdx;

pcWithoutGround = select(pc, outliers);
out = pcWithoutGround.Location;
in = select(pc, inPlanePointIndices);
in = in.Location;
end
```

Fig. 27. Function «Remove ground»

```
function prepare_training_images()
    root_dir = 'E:\FCIS\4th\Second Term\1GP\Training dataset';
    data_set = 'training';
    tic;
    % get sub-directories
    cam = 2; % 2 = left color camera
    image_dir = fullfile(root_dir,[data_set '/image_' num2str(cam)]);
    label_dir = fullfile(root_dir,[data_set '/label_' num2str(cam)]);
    indices = zeros(2);
   for i = 0:7480
        objects = readLabels(label_dir,i);
        img = imread(sprintf('%s/%06d.png',image_dir,i));
        for obj_idx=1:numel(objects)

            % plot 2D bounding box
            object = objects(obj_idx);
            pos = [object.x1,object.y1,object.x2-object.x1+1,object.y2-
object.y1+1];
            imgToSave = imcrop(img,pos);
            imgToSave = imresize(imgToSave,[227 227]);
            filename = '';
            if  strcmp(object.type,'Car') == 1 && object.occlusion <= 1
                filename = sprintf('E:\\FCIS\\4th\\Second Term\\1GP\\Training
dataset\\training\\cropped2\\%s\\%010d.png','Car', indices(1));
                indices(1) = indices(1)+1;
            elseif strcmp(object.type,'Car') == 0
                filename = sprintf('E:\\FCIS\\4th\\Second Term\\1GP\\Training
dataset\\training\\cropped2\\%s\\%010d.png','DontCare', indices(2));
                indices(2) = indices(2)+1;
            else
                continue;
            end
            imwrite(imgToSave,filename);
        end
        fprintf('%d ',i);
        toc;

    end
end
```

Fig. 28. Function «Training images preparation»

```
function detect_car(base_dir,calib_dir)
    frame      = 0; % 0-based index
    fid =
fopen(sprintf('%s/velodyne_points/data/%010d.bin',base_dir,frame),'rb');
%frame i
    velo = fread(fid,[4 inf],'single')';
    fclose(fid);
    velo(:,4) = [];
    ptCloud = velo;
    ptCloud = cut(ptCloud,50,1,1.5);
    idx = ptCloud(:,1)>-2 & ptCloud(:,1)<2.6;
    ptCloud(idx,:) = [];
    idx = ptCloud(:,1)<0;
    ptCloud(idx,:) = [];
    [m,i] = min(ptCloud);
    nearestCar = ptCloud(i(1),:);
    idx = ptCloud(:,1)>nearestCar(1)+1;
    ptCloud(idx,:) = [];
    pcshow(ptCloud);
    % options (modify this to select your sequence)
    if nargin<1
      base_dir  =
'/mn/karlsruhe_dataset/2011_09_26/2011_09_26_drive_0009_sync';
    end
    if nargin<2
      calib_dir = '/mnt/karlsruhe_dataset/2011_09_26';
    end
    cam        = 2; % 0-based index

    % load calibration
    calib =
loadCalibrationCamToCam(fullfile(calib_dir,'calib_cam_to_cam.txt'));
    Tr_velo_to_cam =
loadCalibrationRigid(fullfile(calib_dir,'calib_velo_to_cam.txt'));

    % compute projection matrix velodyne->image plane
    R_cam_to_rect = eye(4);
    R_cam_to_rect(1:3,1:3) = calib.R_rect{1};
    P_velo_to_img = calib.P_rect{cam+1}*R_cam_to_rect*Tr_velo_to_cam;

    % load and display image
    velo_img = project(ptCloud(:,1:3),P_velo_to_img);
    img = imread(sprintf('%s/image_%02d/data/%010d.png',base_dir,cam,frame));
    fig = figure('Position',[20 100 size(img,2) size(img,1)]);
axes('Position',[0 0 1 1]);
    text_str = cell(1,1);
        [m,ind] = min(ptCloud);
        nearestCar = ptCloud(ind(1),:);% #modify //distance from Lidar not
camera
        text_str = ['distance: ' num2str(nearestCar(1),'%0.2f')];% #modify
//distance from Lidar not camera
        %nearestCar(:,1) = []; % #modify //distance from Lidar not camera
        [m,ind] = max(velo_img);
        position(1,1) = m(1);
        [m,ind] = min(velo_img);
        position(1,2) = m(2);
```

Fig. 29. Function «Detect car»

```
function gp_play(in)
    tic;
    base_dir = '';
    calib_dir = '';
    if ( in == 1 )
        base_dir = 'E:\FCIS\4th\Second
Term\1GP\2011_09_26\2011_09_26_drive_0052_sync';
        calib_dir = 'E:\FCIS\4th\Second
Term\1GP\2011_09_26\2011_09_26_drive_0052_sync\2011_09_26';
    else
        base_dir = 'E:\FCIS\4th\Second
Term\1GP\2011_10_03\2011_10_03_drive_0047_sync';
        calib_dir = 'E:\FCIS\4th\Second
Term\1GP\2011_10_03\2011_10_03_drive_0047_sync\2011_10_03';
    end
    writerObj = VideoWriter('E:\FCIS\4th\Second
Term\1GP\videos\myVideo.avi');
    writerObj.FrameRate = 9.8;
    open(writerObj);
    calib =
loadCalibrationCamToCam(fullfile(calib_dir,'calib_cam_to_cam.txt'));
    Tr_velo_to_cam =
loadCalibrationRigid(fullfile(calib_dir,'calib_velo_to_cam.txt'));
    cam        = 2; % 0-based index
    R_cam_to_rect = eye(4);
    R_cam_to_rect(1:3,1:3) = calib.R_rect{1};
    P_velo_to_img = calib.P_rect{cam+1}*R_cam_to_rect*Tr_velo_to_cam;
    set(0, 'DefaultFigureVisible', 'off');
    set(0, 'DefaultAxesVisible', 'off');
    distanceIndex = 0;
    distances = 0;
    meanDistance = 0;
    for i = 0:836
        fid =
fopen(sprintf('%s/velodyne_points/data/%010d.bin',base_dir,i),'rb'); %frame i
        ptCloud = fread(fid,[4 inf],'single')';
        fclose(fid);
        ptCloud(:,4) = [];

        idx = ptCloud(:,1)<2.6;
        ptCloud(idx,:) = [];
        idx = ptCloud(:,2)<-1;
        ptCloud(idx,:) = [];
        idx = ptCloud(:,2)>2;
        ptCloud(idx,:) = [];
        idx = ptCloud(:,3)<-2;
        ptCloud(idx,:) = [];
        idx = ptCloud(:,3)>1;
        ptCloud(idx,:) = [];

        % Find the points corresponding to obstacles
        ptCloudObsatcles = gp_removeGround(ptCloud,0.2);
        [objects,objCount] = gp_cluster3d(ptCloudObsatcles);
        ptCloudObsatcles =
gp_nearestClusteredObject(objects,objCount,ptCloudObsatcles);
        [m,ind] = min(ptCloudObsatcles);
```

Fig. 30. Function «Speed limit detector»

```
function gp_play2(in)
    tic;
    cnnMatFile = 'E:\GP\Data Sets\imagenet-caffe-alex.mat';
    net = helperImportMatConvNet(cnnMatFile);
    load('E:\GP\Data Sets\classifier0.mat');
    featureLayer = 'fc7';
    writerObj = VideoWriter('C:\Users\Ismail Samir\Desktop\v\myVideo.avi');
    writerObj.FrameRate = 9.8;%9.8;
    open(writerObj);
    base_dir = '';
    calib_dir = '';
    if ( in == 1 )
        base_dir = 'E:\GP\Data Sets\2011_10_03_drive_0047_sync';
        calib_dir = 'E:\GP\Data Sets\2011_09_26_drive_0052_sync\2011_09_26';
    else
        base_dir = 'E:\FCIS\4th\Second
Term\1GP\2011_10_03\2011_10_03_drive_0047_sync';
        calib_dir = 'E:\FCIS\4th\Second
Term\1GP\2011_10_03\2011_10_03_drive_0047_sync\2011_10_03';
    end
    calib =
loadCalibrationCamToCam(fullfile(calib_dir,'calib_cam_to_cam.txt'));
    Tr_velo_to_cam =
loadCalibrationRigid(fullfile(calib_dir,'calib_velo_to_cam.txt'));
    set(0, 'DefaultFigureVisible', 'off');
    set(0, 'DefaultAxesVisible', 'off');
    cam       = 2; % 0-based index
    R_cam_to_rect = eye(4);
    R_cam_to_rect(1:3,1:3) = calib.R_rect{1};
    P_velo_to_img = calib.P_rect{cam+1}*R_cam_to_rect*Tr_velo_to_cam;
    for frameNumber = 0:836
        fid =
fopen(sprintf('%s/velodyne_points/data/%010d.bin',base_dir,frameNumber),'rb')
; %frame i
        ptCloud = fread(fid,[4 inf],'single')';
        fclose(fid);
        ptCloud(:,4) = [];
        idx = ptCloud(:,1)<2.6;
        ptCloud(idx,:) = [];
        idx = ptCloud(:,2)<-5;
        ptCloud(idx,:) = [];
        idx = ptCloud(:,2)>10;
        ptCloud(idx,:) = [];
        idx = ptCloud(:,3)<-2;
        ptCloud(idx,:) = [];
        idx = ptCloud(:,3)>3;
        ptCloud(idx,:) = [];

        [ptCloud,inliers] = gp_removeGround(ptCloud,0.2);
        [objects,objCount] = gp_cluster3d(ptCloud);
        objectsClustered = cell(1,objCount);
        for k = 1:size(objects,1)
            if ( isempty(objectsClustered{objects(k)}) )
                objectsClustered{objects(k)} = k;
            else
                objectsClustered{objects(k)} =
[objectsClustered{objects(k)},k];
```

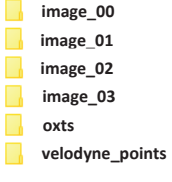
Fig. 31. Function «Vehicle recognition»

10. 2. System Testing and Deployment

The used datasets are acquired from the following link:
http://www.cvlibs.net/datasets/kitti/raw_data.php
Select [synced rectified data] option (Fig. 32).

Fig. 32. KITTI used Dataset [14]

Each dataset contains 6 folders:



Here:

– 'image_00': left rectified gray-scale image sequence;

– 'image_01': right rectified gray-scale image sequence;

– 'image_02': left rectified color image sequence;

– 'image_03': right rectified color image sequence;

– 'velodyne_points': Lidar 3d data points;

– 'oxts': GPS/IMU data etc...

Steps:

1. Consider «gp_play2» function;

2. Set base_dir = the folder which contains the dataset;

3. Set cnnMatFile = the alexNet mat file;

4. Set the SVM classifier path in the load function;

5. Set the video path of your choice;
6. Run «gp_play2» function;
7. It will go through each frame in the dataset you have downloaded;
8. Wait until the program finishes executing;
9. The result video is in the path you specified.

10. 3. Future Work

In this paper, we implemented a real-time image/Lidar processing.

At the beginning, we presented a real-time system which is composed of comprehensive modules, these modules are 3d object detection, object clustering and search, ground removal, deep learning using convolutional neural networks. Starting with nearest vehicle module our target is to find the nearest ahead car and consider it as our primary obstacle (Fig. 33).

Detecting all objects in the scene after removing ground presented in Fig. 34.

Recognizing only vehicles (filtering out non-vehicles) presented in Fig. 35, 36.

Fig. 33. Result. Implementation of a real-time image/Lidar processing (1)

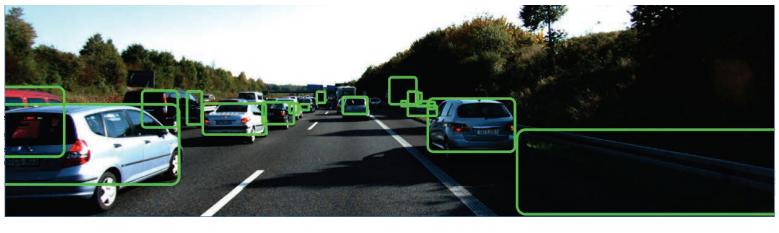


Fig. 34. Result. Detecting all objects in the scene after removing ground (2)

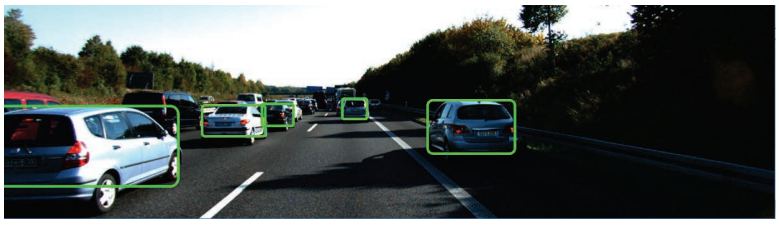


Fig. 35. Result. Recognizing only vehicles (filtering out non-vehicles) (3)

Fig. 36. Result. Calculating distances from each vehicle (4)

11. Conclusions

In this research paper, we implemented a real-time image/Lidar processing. At the beginning, we presented a real-time system which is composed of comprehensive modules, these modules are 3d object detection, object clustering and search, ground removal, deep learning using convolutional neural networks. Starting with nearest vehicle module our target is to find the nearest ahead car and consider it as our primary obstacle.

Presented an Adaptive cruise pre-crash system and vehicle recognition. The Adaptive cruise pre-crash system module depends on Deep Learning and LiDAR sensor data, which meant to control the driver reckless behavior on the road by adjusting the vehicle speed to maintain a safe distance from objects ahead (such as cars, humans, bicycle or whatever the object) when the driver tries to raise speed. At the very moment the vehicle recognition module, detects and recognizes the vehicles surrounding to the car.

The effectiveness of the results obtained is confirmed for the following test cases:

– detecting all objects in the scene after removing groun;

– recognizing only vehicles (filtering out non-vehicles);

– calculating distances from each vehicle.

Acknowledgments

Thanks to student Yasmine Bakr and her colleague, our graduate students for some dataset programming which are supervised by Dr. Bassant Elbagoury at Faculty of Computer and information sciences, ain shams university, cairo, Egypt.

References

1. Verification of On-Line Vehicle Collision Avoidance Warning System using DSRC / Hsu C. W., Liang C. N., Ke L. Y., Huang F. Y. // World Academy of Science, Engineering and Technology. 2009. Vol. 3, Issue 7. P. 808–814.
2. Chang B. R., Tsai H. F., Young C.-P. Intelligent data fusion system for predicting vehicle collision warning using vision/GPS sensing // Expert Systems with Applications. 2010. Vol. 37, Issue 3. P. 2439–2450. doi: https://doi.org/10.1016/j.eswa.2009.07.036
3. Köhler M. Accurate PreCrash Detection. IBEO Automobile Sensor GmbH, System Development. Hamburg.
4. Schouten N. Pre-Crash Testing in the VeHIL Facility. TNO Automotive, Integrated Safety Department. 2008. 28 p.
5. Automotive Embedded Systems Handbook / N. Navet, F. Simonot-Lion (Eds.). CRC Press, 2009. doi: https://doi.org/10.1201/9780849380273
6. Jansson J., Johansson J., Gustafsson F. Decision Making for Collision Avoidance Systems // SAE Technical Paper Series. 2002. doi: https://doi.org/10.4271/2002-01-0403
7. Evans C. Notes on the open surf library. University of Bristol, Tech. Rep. CSTR-09-001, 2009.

8. Popirlan C., Dupac M. An Optimal Path Algorithm for Autonomous Searching Robots //Annals of University of Craiova, Math. Comp. Sci. Ser. 2009. Vol. 36, Issue 1. P. 37–48.
9. Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments / Dolgov D., Thrun S., Montemerlo M., Diebel J. // The International Journal of Robotics Research. 2010. Vol. 29, Issue 5. P. 485–501. doi: https://doi.org/10.1177/0278364909359210
10. The 2005 DARPA Grand Challenge: The Great Robot Race / M. Buehler, K. Iagnemma, S. Singh (Eds.). Springer, 2007. doi: https://doi.org/10.1007/978-3-540-73429-1
11. Urban challenge rules, revision. DARPA. 2007. 28 p. http://archive.darpa.mil/grandchallenge/docs/Urban_Challenge_Rules_102707.pdf
12. Urmson C. Tartan Racing: A Multi-Modal Approach to the DARPA Urban Challenge. 2007.
13. Tavel P. Modeling and Simulation Design. AK Peters Ltd., 2007.
14. Welcome to the KITTI Vision Benchmark Suite! URL: http://www.cvlibs.net/datasets/kitti/
15. Pandey G., McBride J. R., Eustice R. M. Ford Campus Vision and Lidar Data Set. URL: http://robots.engin.umich.edu/publications/gpandey-2010b.pdf
16. Robotics: Estimation and Learning. URL: https://www.coursera.org/learn/robotics-learning

Запропоновано спосіб підвищення завадостійкості детектора фазоманіпульованих (ФМ) сигналів на основі пристрою фазового автопідстроювання частоти (ФАПЧ) шляхом використання його модифікації.

Задача підвищення завадостійкості систем зв'язку до цих пір йшла в протиріччі із завданням досягнення високих динамічних показників пристрою для ефективної та коректної обробки ФМ-сигналів з великим індексом модуляції. Покращення завадостійкості системи означало погіршення її динамічної поведінки і навпаки. Запропонований спосіб дає можливість знизити шумовий поріг пристрою, не погіршуючи при цьому його динамічних властивостей.

Імітаційне моделювання граничної завадостійкості класичного та модифікованого пристроїв проводилось для двох критеріїв зриву синхронізації. В обох випадках завадостійкість модифікованого пристрою є кращою. Результати імітаційного моделювання показують, що аномальні стрибки фази опорного генератора модифікованого пристрою за короткий час спостерігаються для більших рівнів шуму, ніж в класичному пристрої (на 1,5–4 дБ залежно від параметрів пристрою).

Обидва варіанти пристроїв були фізично реалізовані на базі програмованої логічної інтегральної схеми (ПЛІС) з метою проведення експериментальних досліджень завадостійкості цих пристроїв та перевірки результатів імітаційного моделювання. Експериментальні дослідження якісно підтвердили результати моделювання та показують, що використання модифікованого фазового детектора дає виграш у завадостійкості на 1–2,5 дБ залежно від параметрів пристрою. Динамічні властивості модифікованого пристрою при цьому не погіршуються.

Наведені результати демонструють неабиякі перспективи використання пристроїв ФАПЧ з підвищеною завадостійкістю у системах зв'язку різноманітного призначення, що працюють в складній завадовій обстановці

Ключові слова: пристрій фазового автопідстроювання частоти (ФАПЧ), модифікований фазовий детектор (ФД), вузькосмуговий фільтр (ВСФ)

UDC 621.372
DOI: 10.15587/1729-4061.2018.143178

FIRMWARE IMPLEMENTATION AND EXPERIMENTAL RESEARCH OF THE PHASE-LOCKED LOOP WITH IMPROVED NOISE IMMUNITY

A. Bondariev
Doctor of Technical Sciences, Professor*
E-mail: bondap@ukr.net
S. Altunin
Postgraduate student*
E-mail: serg.alt.i@gmail.com
I. Horbatyi
Doctor of Technical Sciences, Associate Professor*
E-mail: giv@polynet.lviv.ua
I. Maksymiv
PhD, Assistant*
E-mail: Ivan.P.Maksymiv@lpnu.ua
*Department of theoretical radio engineering and radio measurement
Lviv Polytechnic National University
S. Bandery str., 12, Lviv, Ukraine, 79013

1. Introduction

Today, it is impossible to imagine a modern society without such innovations as the Internet, digital TV, mobile phones, GPS, etc. The emergence and development of these technologies would be impossible without progress in the field of radio engineering, microelectronics and digital circuitry. However, despite their complexity, these devices