International Journal of GEOMATE, Aug., 2020, Vol.19, Issue 72, pp. 217 - 224
ISSN: 2186-2982 (P), 2186-2990 (O), Japan, DOI: https://doi.org/10.21660/2020.72.9181
Special Issue on Science, Engineering and Environment

ALGORITHMS DEVELOPED FOR TWO PROTOTYPES OF AIRBORNE VISION-BASED CONTROL OF GROUND ROBOTS

Ilan Ehrenfeld[1], Oleg Kupervasser[1,2]*, Hennadii Kutomanov[1], Vitalii Sarychev[2], and Roman Yavich[1]

[1] Department of Mathematics, Ariel University, Israel; [2] Transist Video Llc, Skolkovo, Russia

*Corresponding Author, Received: 15 Sept. 2019, Revised: 09 March 2020, Accepted: 16 March 2020

ABSTRACT: The paper addresses the problem of visual navigation of ground robots using a camera positioned at a certain elevation above the confined area. Unmanned autonomous robots will be widely used very soon for land use, treatment, and monitoring. Technologies, that can be used for such robots, are already described previously in "International Journal of GEOMATE" by Kupervasser et al. and Djaja et al.. Developing these technologies is continued and presented here by new patented technology of airborne vision-based control of ground robots. The main idea is that robot's "eyes" is not located on robot but are independent autonomous system. As a result, the "eyes" can go up and observe the robot from above. Algorithms, presented in this paper, are used for real simple physical prototypes of a such system. The system consists of a camera connected to the ceiling, to computer which controls a toy tank. Computer program analyses image from the camera, finds difference between real position of the tank and desirable position, and send trough Wi-Fi command to decrease this difference. Also, the next more complex prototype of a such system is very shortly described. This prototype is planned to create in framework of the KAMIN incentive program (Israel).

Keywords: Visual Navigation, Ground Robots, Tethered Platform, Airborne Control, Prototype, Vision-Based Navigation

1. INTRODUCTION

The paper addresses the problem of visual navigation of ground robots using a camera positioned at a certain elevation above the confined area. Unmanned autonomous robots will be widely used very soon for land use, treatment, and monitoring. Technologies, that can be used for such robots, are already described by Kupervasser et al., and Djaja et al. in [1,2]. The most popular outdoor robots are currently robot-lawnmowers. These robots are expensive for most home use, but not professional enough for industrial use. Many robotic lawnmowers are simple yet (like Roombas), however a new advanced generation has begun to appear for a few years. This is for example, Robomowers (Friendly Robotics, $2000) and Automower (Husqvarna, $2200).

But their consumers are mainly in Europe (Husqvarna has sold over 100,000 of their Automowers). It is because of their high price, mulching clumps, inability to handle high grass, need for staked border wires, and their random navigation methodologies. It is inconvenient, static, expensive technology [3]. The solution for the part of the problems is vision-based navigation [4-6]. Vision-based navigation of robots is like human navigation by the help of eyes vision. However, it is not necessary to build eyes into the robot. It is possible to put eyes on a top position and from the top position the robot can see itself and its motion. It means airborne terrestrial robot control (Fig. 1).

Developing these technologies is continued and presented here by new patented technology of airborne vision-based control of ground robots [7-10]. It is planned to develop a software package including approaches for comprehensive video-navigation solution of a ground robot from top position (from a tethered drone, a balloon, a tower, an antenna on a ground robot) and a physical prototype (camera on a top position, a ground robot, a ground station with computer) controlled by this software.

The described above system may be used for coordination, navigation and control of ground robots (automated transport, automated agricultural machines, municipal and aerodrome vehicles, garden lawnmowers and so on). The invention essence is the system of navigation and intercoordination. The system includes one or more roots, located on the controlled area; one or more robot tracing devices on the suspended tethered platforms; natural or artificial markings; charger; and central module for robot coordination and orientation detection. The information is transferred to the central module from all the tracing devices. The system central module is equipped with the calculation module located or on the suspended platform, or on the ground, or on the charger or on the robot. The calculation module is possible to calculate coordinates and orientation of robots and to form the control commands, based on the information received from all the above described devices.

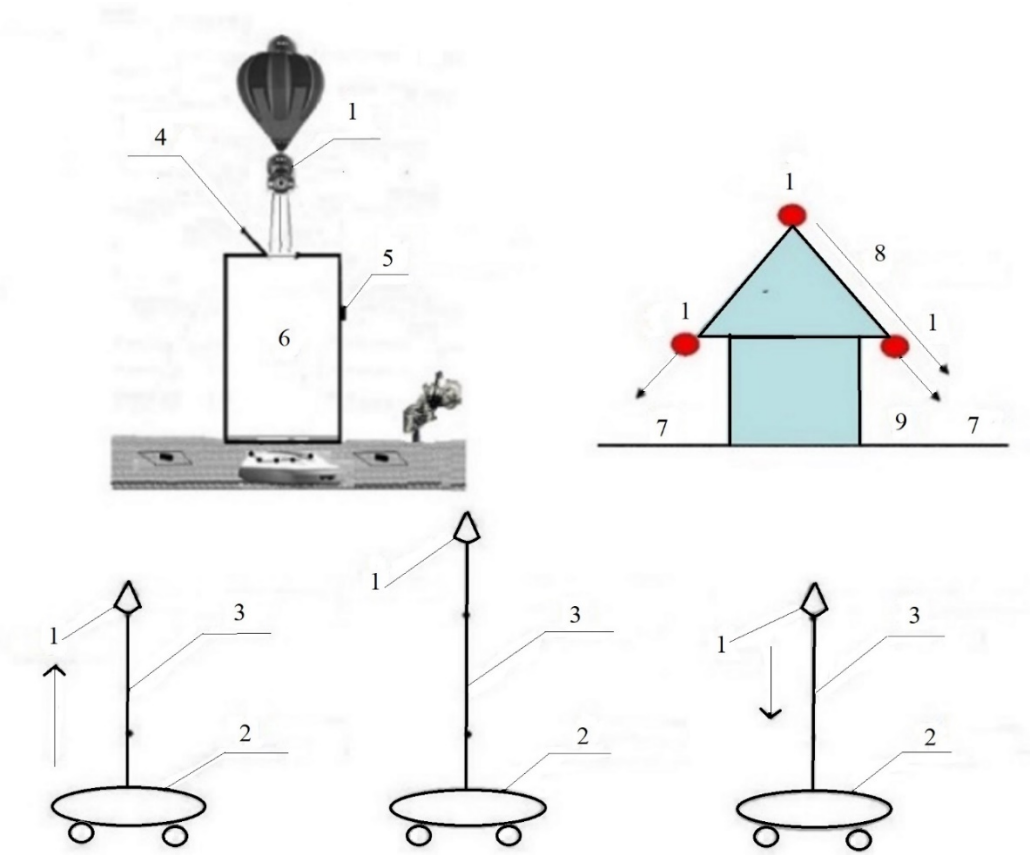

Fig. 1. Airborne terrestrial robot control, some possible camera dispositions; 1- camera, 2 – lawnmower; 3 – moving up and down long antenna; 4 – open door; 5 – press button; 6 – ground-based energy charged device (ECD); 7 – lawn; 8 – line of sight; 9 – blind spot

The technical result is the development of the robot efficient coordination using the devices located on the towers, the aerial apparatuses tracing the robots on the controlled area, supervising their environment including natural and artificial markings. One of the developments is to take into account delays in the control system. The technical result coincides with the engineering challenge.

This visual system can also be used for preventing collision of ground robot with children or animals. Also, we can use this visual system for security purposes.

The system (the proof of concept) was defined to work as following:

1. A toy tank will simulate a lawnmower.
2. The tank will be controlled by a software that will connect to a webcam that watch all the area.
3. There will be two options to drive the tank:
   a. A random movement in user predefined boundaries
   b. A user predefined route
4. For random route, the user will define on the live video by using set of boundary straight lines (polygonal chains) and squares:
   a. The external boundaries of the area (practically close to the edge of the frame) that the tank cannot go out.
   b. internal areas inside the frame that the tank cannot go to.
5. For predefine route, the user will define on the live video:
   a. A continuous and close route starts at the location of the tank and ends at the same location.
   b. The user will use route straight lines (polygonal chain) to draw the route.
6. The software will receive the life video from the webcam
7. The software will analyze the location and orientation of the tank.
8. The software will instruct the tank what to do at the next step according to its analyzing and the user definition of the desirable location and orientation.

The paper is constructed as following:

Section 1 is the introduction and includes a description of the system. Section 2 provides a system overview. Section 3 describes the software. Section 4 describes the algorithms in use. Section 5 describes the future plans and conclusion.

2. SYSTEM OVERVIEW

The following figure describes the system components (Fig. 2):

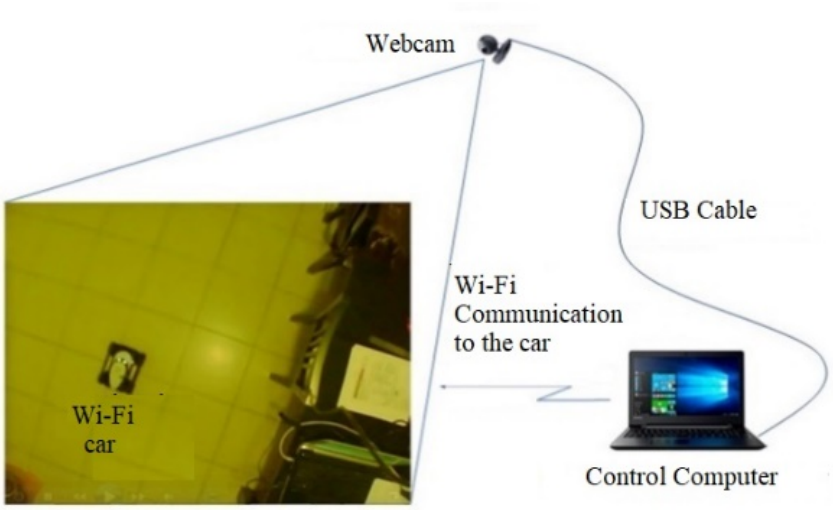

Fig. 2. The system components

The list of the components in the system is following:

- Tank is a game tank of type iSpy. This tank has Wi-Fi communication and an API to be controlled from a computer.
- Webcam is connected to the computer by USB cable.
- Control Computer is a computer running the software that we developed. This computer connected to the webcam to get a live video of the area and gives instructions to the tank using Wi-Fi communication.

3. SOFTWARE OVERVIEW

3.1 Software General Description

The software (Fig. 3) is written in Python. It consists of two files:

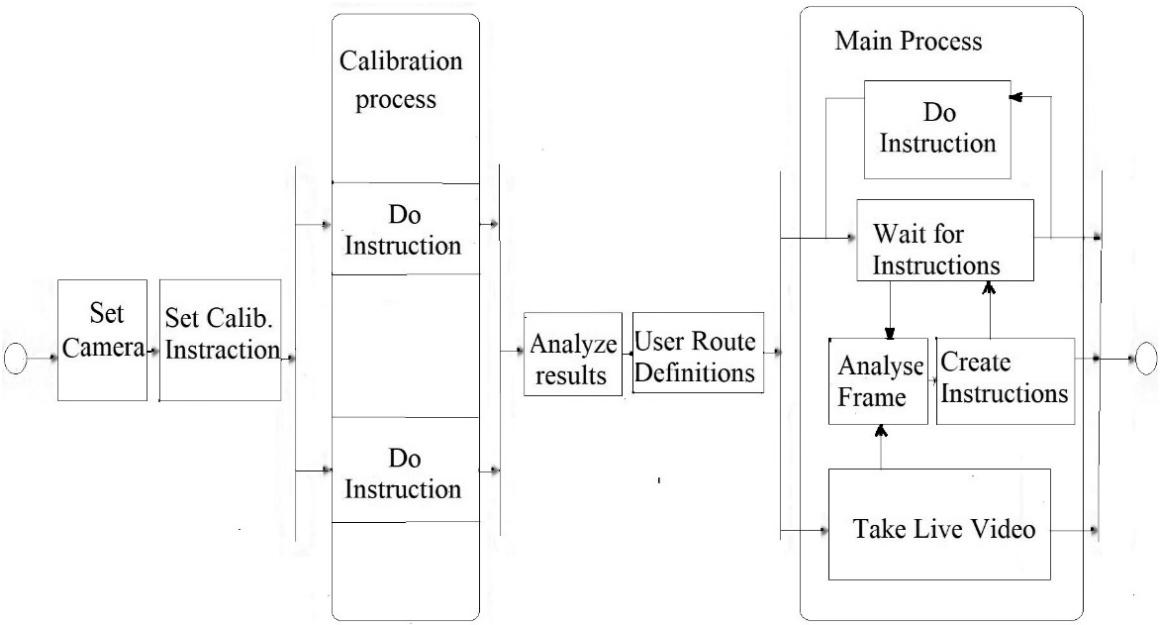

Fig. 3. State Diagram

- ColorTrack.py is a main file. This file contains all the flow control, logics, computer vision and the user interface (UI)
- carControl.py: this file is a place where the communication with the tank is made.

The next part of the current section describes shortly the carControl.py structure. The rest of the description gives content of ColorTrack.py file. carControl.py is built as a class with regular __init__ function, where all the initializations for this class described.

There is the "run" function. This is a thread control function. This function runs in endless loop and wait for a command. The command is received to the self.move variable. When this variable is set to something that is different from "no" (no command), it will run a function according to one from the following commands:

- moveForCalib: this function is used for calibration of a straight move.
- turnForCalib: this function is used for calibration of a turn.
- goForward is a move forward function. It receives, as an input, a move distance in pixels and calculates the move time and sends by Wi-Fi the right command to the tank.
- goBackword is a move backward function. It receives, as input, a move distance in pixels and calculate the move time and send by Wi-Fi the right command to the tank.
- goLeft is a turn left function. It receives, as input, an angle in degrees for a turn and calculate the move time and sends by Wi-Fi the right command to the tank.
- goRight is a turn right function. It receives, as input, an angle in degrees for turn and calculates the move time and send by Wi-Fi the right command to the tank.

3.2 Software Flowcharts

In this paragraph we show the flowcharts of the software (Fig. 4-6). It starts with the start flowchart and then show separately the flowcharts for route mode driving or turn and for random mode driving or turn.

In the flowcharts, a rectangle represents a function in the software, the function name is written in parenthesis inside of the rectangle.

4. ALGORITHMS AND CALCULATIONS DESCRIPTION

4.1 Tracking Algorithm

The tracking algorithm is based on motion detection. It is based on OpenCV and examples from OpenCV.org for implementation of motion detection.

The following paragraph is the description of the algorithm for finding tank on the image and creating enclosing rectangle of the tank:

1. Save an image of the operation area with no moving object inside. This image will be the reference image. In our case, it is done when the user pushes <Camera Ready> button to initiate the function "onCameraReady"
2. The rest of the algorithm is implemented in the function "motionDetection". The algorithm makes the following:
   a. Take the current image
   b. Changes the image to gray level image
   c. Applies Gaussian blur to the gray level image
   d. Calculates absolute values of differences between the current image and the reference image.
   e. Applies binary threshold for the result (we will get output as the black and white image where the white is corresponding to pixels where the difference is greater than the threshold)
   f. Applies filter to reduce noises (dilate filter function from OpenCV): the result is kept in image called "mask" (the masked image)
   g. The rest of the steps in this algorithm are implemented in the function "showTrackingResults"
      i. By using OpenCV function "findContours" we find the contours of all the objects in the mask image
      ii. By using OpenCV function "max" we select the biggest object in the image
      iii. By using OpenCV function "minAreaRect" we define and save the enclosing rectangle of the tank
   h. If we have the next image go to a., else stop the algorithm.

4.2 Movement Calculations

4.2.1 General

All movement logics are based on the orientation of the tank relatively to an image's axes. We use in the software two parameters to describe the orientation:

1. carOrientation - this is the general direction of the tank. The orientation names are: NW – north west, NE – north east, SE – south east, SW – south west.
2. carAngle: this is the exact angle of the tank. We calculate it relatively to its general direction in

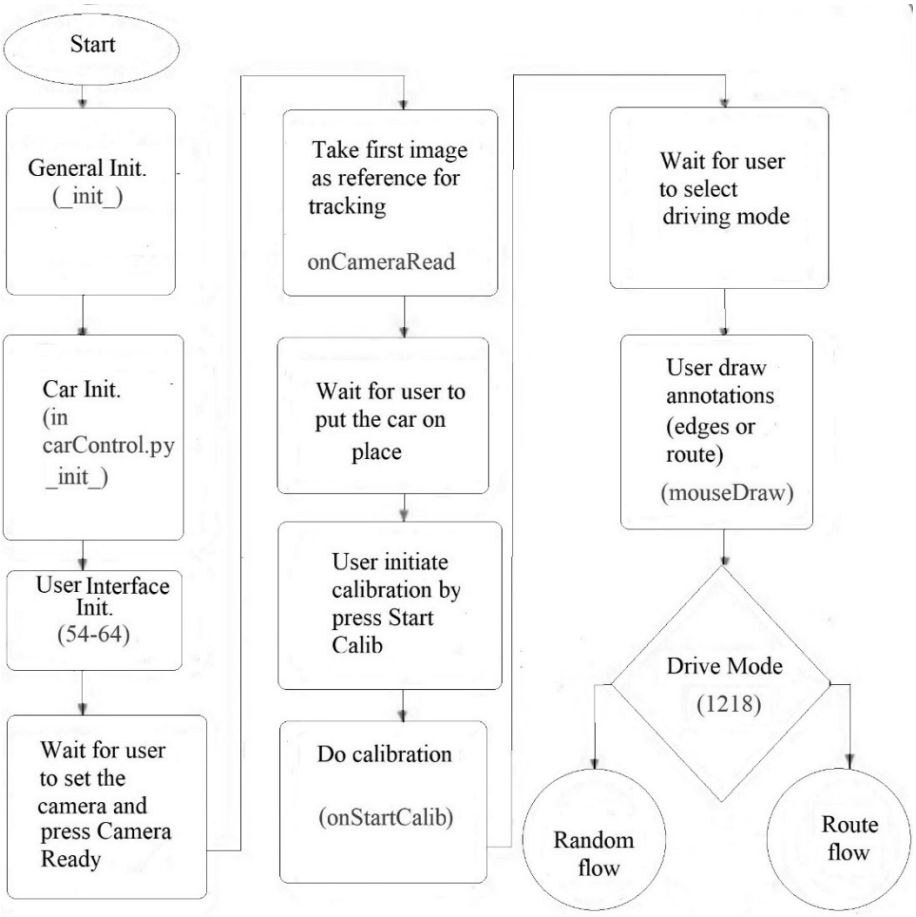

Fig. 4. Start Flowchart

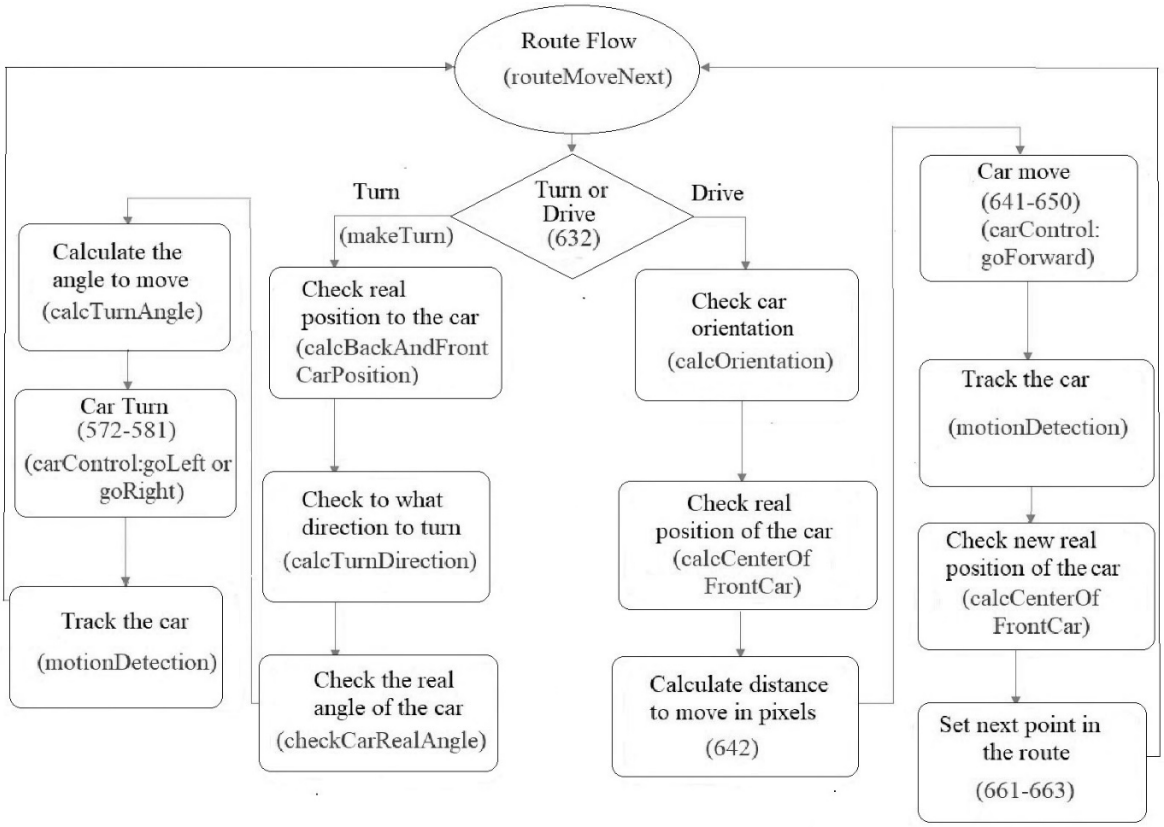

Fig. 5. Route Mode Turn or Drive Flowchart

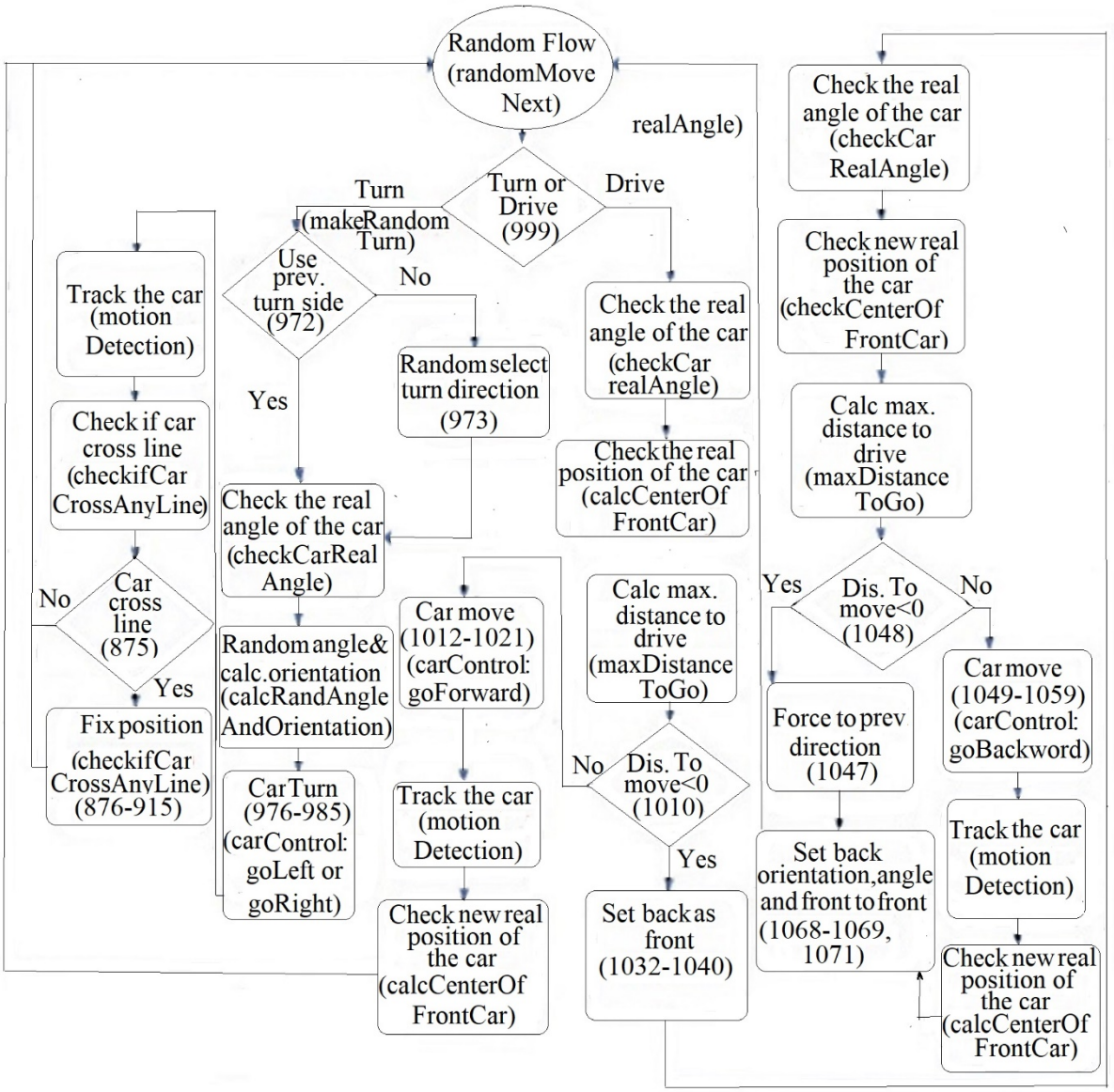

Fig. 6. Random Mode Turn and Drive Flowchart

the image. In other words, the angle of the tank will always be in the range of 0-90 degrees

The axes, the general directions, their names, and the angle location for each direction are shown at Fig. 7.

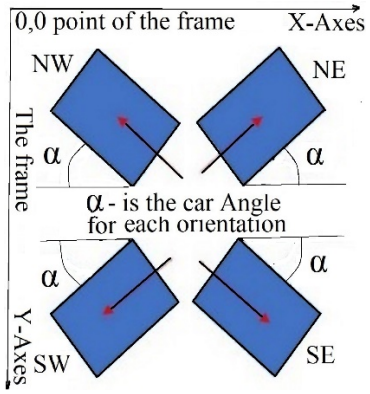

Fig. 7. The axes, the general directions, their names, and the angle location for each direction

The frame of the tank, which is described by array variable carBox of the software, is defined by 4 points. The first point (located in place 0 in the carBox array) will always be at the lowest point. From this point, the other points are defined clockwise as shown at Fig. 8.

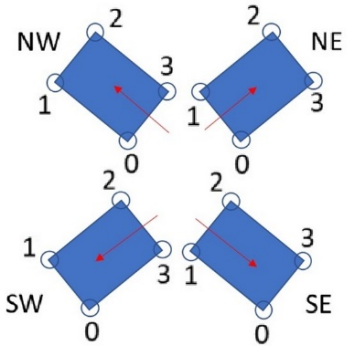

Fig. 8. The frame of the tank is defined by 4 points

4.2.2 General supporting Calculations

Based on the given above explanation, here are description for some supporting calculations, which are carried out during the flow of the software:

1) Calculating the position of the tank. The position of the tank is defined as the center of the front line of the tank:
1. According to the orientation of the tank (NW, NE, SE or SW) select the two points represent the front of the tank: NW – points 1 and 2, NE – points 2 and 3, SE – points 3 and 0, SW – points 0 and 1.
2. Calculating the middle of this segment
2) Calculating the real angle of the tank:
1. According to the orientation of the tank we choose the two points along the tank
2. Calculating the found line angle = atan(dy/dx)
3) Calculating intersection between two line segments. This calculation is implemented in function "find_intersection". It is based on Andre LeMothe's algorithm. The implementation was taken from work [11]. There, you can also find a detailed explanation of the algorithm.

4.2.3 Route Mode Supporting Calculations

Now we have the described above supporting calculations. According to the described above explanations, the orientation calculations such as: which side to turn to, what is an angle of the turn become straightforward.

To calculate the distance for driving to the next point, we need to calculate the Euclidean distance between the current location point to the destination location point.

4.2.4 Random Mode Supporting Calculations

1) Calculation of the maximum distance that the tank can move is following:
This calculation is implemented in function "maxDistanceToGo".
1. Check the shortest distance to the edge of the frame
2. Keep this distance in minDis variable
3. For the current straight line of the polygonal chains that the user created on the video:
a. Create a vector at each side of the tank:
i. Example of a side: if the tank orientation is NW one side will be points 2 and 3 and the other side will be 1 and 0
ii. The vector goes from the back part of the tank to its front part until the vector achieves the edge of the frame. For NW orientation, for example, the vector direction will be from point 3 to point 2 and the other vector will be from point 0 to point 1.
4. For each line segment (defined by these vectors) we need to check intersection with the sets of boundary straight lines (polygonal chains)
5. If the intersection exists, calculate the distance form center of the tank to the intersected boundary line.
6. If the distance is smaller than minDis, keep this distance as the new minDis.
7. If not all boundary straight lines of the polygonal chains that the user created on the video was considered go to 3, else stop algorithm.
2) Checking if the tank is on one of the boundary lines
1. Go over all set of boundary straight lines (polygonal chains) that a user created on the video
2. For each line:
a. Each two adjacent points of the tank (sides) are determined as a line segment.
b. For the each such segment (side), we need to check intersection with the boundary lines (polygonal chains).

5. FUTURE PLANS AND CONCLUSION

We plan to create much more complicated prototype in framework of the KAMIN incentive program (Israel).

The plane consists of the next stages:

1. Creating programs of video-navigation for airborne control from towers of ground robots on flat ground surface
2. Design and integration of robotic system: observation towers, controlling center, and ground robots
3. Testing robotic system and computer program, errors correction of robotic system and computer program
4. Rewriting programs for video-navigation than airborne control is made from tethered platform (drone or balloon) and ground robots on not-flat surface
5. Modernization of robotic system: creation of observation tethered platform (drone or balloon) and modernization of ground robots for not-flat surface
6. Testing robotic system and computer program, errors correction of robotic system and computer program

This system may be used for a wide class of robots: automated lawnmowers, robots for cleaning the rooms, tractors, snow-removal, garbage disposal and flushing vehicles, vehicles for people and goods transportation, agricultural and municipal vehicles, transport and so on. This system may be used for extra-terrestrial robots on other planets, for instance, for Mars rovers.

The system easily stays within the frames of the "smart home" or even "smart city" enabling to coordinate simultaneously a lot of actions, robots and other control objects and to solve many tasks - for instance, not only navigation but detection.

6. REFERENCES

[1] Kupervasser O., Sarychev V., Rubinstein A. and Yavich R., Robust positioning of drones for land use monitoring in strong terrain relief using vision-

based navigation, International Journal of GEOMATE, Vol.14, Issue 45, 2018, pp. 10-15

[2] Djaja K., Putera R., Rohman R A. F., Sondang I., Nanditho G. and Suyanti E., The Integration of Geography Information System (GIS) and Global Navigation Satelite System-Real Time Kinematic (GNSS-RTK) for Land use Monitoring, International Journal of GEOMATE, Vol.13, Issue 36, 2017, pp. 31-34

[3] The Robot Report: tracking the business of robotics,2012

[4] Kupervasser O., Lerner R., Rivlin E. and Rotstein H., Error Analysis for a Navigation Algorithm based on Optical-Flow and a Digital Terrain Map, in Proceedings of the 2008 IEEE/ION Position, Location and Navigation Symposium, 2008, pp.1203-1212

[5] Kupervasser O. Yu., Rubinshtein A. A., Correction of Inertial Navigation System's Errors by the Help of Video-Based Navigator Based on Digital Terrain Map, Positioning, Vol.4, No.1, 2013, pp. 89-108

[6] Kupervasser O.Yu., Computer programs: Video-navigation of UAV over relief, Part 1, Part 2, the certificates on the state registration of the computer programs № 2016613306, 2016613305, registered in the register of the computer programs of Federal service on intellectual property, patents and trade marks, 2016.

[7] Ehrenfeld I., Kogan M., Kupervasser O., Sarychev V., Volinsky I., Yavich R., Zangbi B., Visual navigation for airborne control of ground robots from tethered platform: creation of the first prototype, in Proceeding of the IEEE International Conference on New Trends in Engineering and Technology, GRTIET, Tirupathi, Chennai, Tamil Nadu, India, 2018, pp. 96

[8] Kupervasser O. Yu., Kupervasser Yu. I., Rubinstein A. A., Russian Utility model: Apparatus for Coordinating Automated Devices, Russia, Patent № 131276, 2012

[9] Kupervasser O.Yu., Kupervasser Yu.I., Rubinstein A.A., German Utility model: Vorrichtng fur Koordinierung automatisierter Vorrichtungen. Germany,Patent Nr. 21 2013 000 225, 2015

[10] Kupervasser O. Yu., Franch Utility model application: Coordination System for Ground Movable Automated Devices. Utility model, France,publication number 3037157, 2016,

[11] How do you detect where two line segments intersect - How do I determine whether or not two lines intersect, and if they do, at what x,y point?,