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

Abstract

Unmanned autonomous robots will be widely used very soon for land use, treatment, and monitoring. Our and the other groups already described technologies, that can be used for such robots (Kupervasser et al., International Journal of GEOMATE, May, 2018 Vol.14, Issue 45, pp.10-16; Djaja et al., International Journal of GEOMATE, Aug, 2017, Vol.13, Issue 36, pp.31-34). We continue developing these technologies and present here 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. We present in this paper algorithms used for two real physical prototypes of a such system.

Keywords—visual navigation; ground robots; tethered platform; airborne control; prototype; vision-based navigation

INTRODUCTION

Unmanned autonomous robots will be widely used very soon for land use, treatment, and monitoring. Our and the other groups already described technologies, that can be used for such robots [1,2].

The most popular outdoor robots are currently robot-lawnmowers. Current robot-lawnmowers need for staked border wires. They have random navigation methodologies. They are inconvenient, static, expensive technology.

You can see the following citation from "THE ROBOT REPORT: TRACKING THE BUSINESS OF ROBOTICS" [3]:

"... Too expensive for most home use, not professional enough for industrial use, robotic lawnmowers have yet to become as commonplace as Roombas, But a new crop of consumer manufacturers is quietly making inroads by providing more bang for the buck."

".., Friendly Robotics and their line of Robomowers, $2,000, and Husqvarna and their Automower line, $2,200, have been around for a few years but never hit consumer traction outside of Europe (Husqvarna has sold over 100,000 of their Automowers), partly because of their high price, mulching clumps, inability to handle high grass, need for staked border wires, and their random navigation methodologies."

The solution of the problem is vision-based navigation [4-6]. Vision-based navigation of robots is similar to human navigation by the help of eyes vision. However, we do not eyes on the robot. We put eyes on a top and from the top robot can see itself and its motion. We use airborne terrestrial robots control (Figure 1).

We continue developing these technologies and present here 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 ground robot from top position (tethered drone, balloon, tower, antenna on the ground robot) and a physical prototype (camera on the top position, ground robot, ground station with computer) controlled by this software.

The system relates to the control systems of automated devices and may be used for the coordination of the ground movable automated devices (automated transport, automated agricultural machines, municipal and aerodrome vehicles, garden lawnmowers and so on), hereinafter referred to as the robots.

The invention essence is the system of navigation [1-10] and intercoordination of one or more roots located on the controlled area including one or more robot tracing devices on the suspended platforms, natural or artificial markings, central module for robot coordination and orientation detection to which the information is transferred from all the tracing devices, charger and the system is equipped with the central calculation module located on the suspended platform, on the ground, on the charger or on the robot designed with the possibility to detect the coordinates, to orient the system and to form the control commands based on the information received from all the above described devices.

The technical result is the development of the robot efficient coordination using the devices located on the towers or the aerial apparatuses tracing the robots on the controlled area and 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.

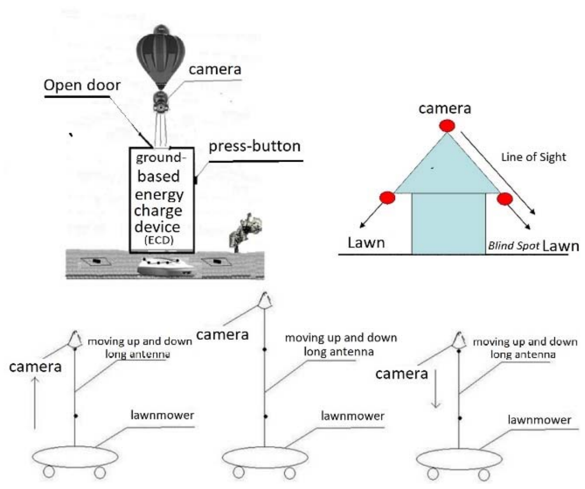
Open door
camera
press-button
ground-based energy charge device (ECD)
camera
Line of Sight
Lawn
Blind Spot Lawn
camera
moving up and down long antenna
lawnmower
camera
moving up and down long antenna
lawnmower
camera
moving up and down long antenna
lawnmower

Fig. 1. Airborne terrestrial robots control, some possible camera dispositions

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

1. A toy car will simulate the lawnmower.
2. The car 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 car:
    a. A random movement
    b. A user predefined route
4. For random route, the user will define on the live video:
    a. The edges of the area (practically close to the edge of the frame).
    b. Using lines and squares, areas in the frame that the car can't go to.
5. For predefine route, the user will define on the live video:
    a. Continues and close route that is starting at the location of the car and end about the same location.
    b. The user will use lines 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 car.
8. The software will instruct the car what to do next according to its analyzing and the user definition.

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 describe the software.

Section 4 describes the algorithms in use.

SYSTEM OVERVIEW

System Components

The following figure describe the system components (Figure 2):

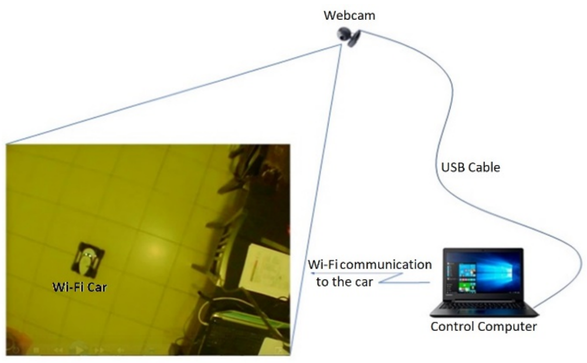

Fig. 2. The system components

SOFTWARE DESCRIPTION

Software General Description

Following is a list of the components in the system:

- Car – A game car of type iSpy. This car has Wi-Fi communication and an API to be controlled from a computer.
- Webcam – connected to the computer by USB cable.
- Control Computer – A computer running the software we developed. This computer connected to the webcam to get a live video of the area and give instructions to the car using Wi-Fi communication.

The software (Figure 3) is written in Python. Is build out of two files:

- ColorTrack.py – main file. This file contains all the flow control, logics, computer vision and the UI
- carControl.py – This file is where the communication with the car is done.

The following paragraph describe shortly the carControl.py structure. All the rest of the description in this document exist in ColorTrack.py file.

The carControl build as a class with regular __init__ function, where all the initializations for this class happened.

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

- moveForCalib – This function use for calibration of a strait move.
- turnForCalib – This function use for calibration of a turn.
- goForward – Move forward function. It receives, as input, the distance to move in pixels and calculate the time to move and send by Wi-Fi the right command to the car.
- goBackword – Move backward function. It receives, as input, the distance to move in pixels and calculate the time to move and send by Wi-Fi the right command to the car.
- goLeft – Turn left function. It receives, as input, the angle, in degrees, for turn and calculate the time to move and send by Wi-Fi the right command to the car.
- goRight – Turn right function. It receives, as input, the angle, in degrees, for turn and calculate the time to move and send by Wi-Fi the right command to the car.

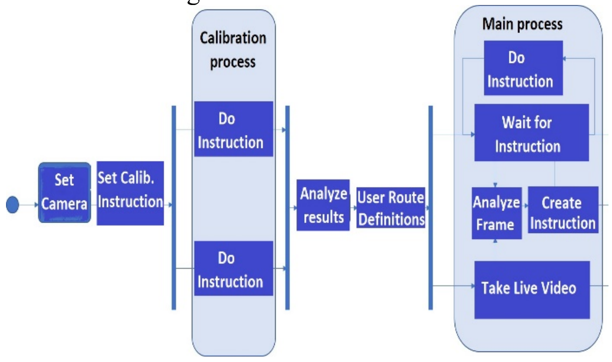

Fig. 3. State Diagram

Software Flowchart

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

In the flowcharts, a rectangle that represent a function in the software, the function name is written in red and in parenthesis in the rectangle.

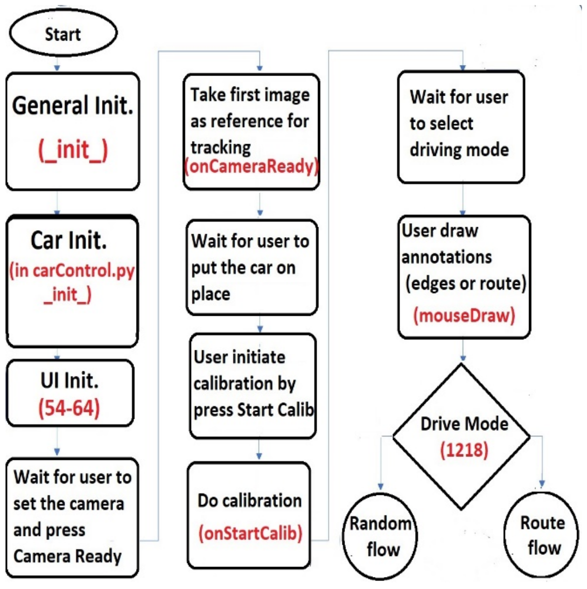
Start
General Init. (_init_)
Car Init. (in carControl.py _init_)
UI Init. (54-64)
Wait for user to set the camera and press Camera Ready
Take first image as reference for tracking (onCameraReady)
Wait for user to put the car on place
User initiate calibration by press Start Calib
Do calibration (onStartCalib)
Wait for user to select driving mode
User draw annotations (edges or route) (mouseDraw)
Drive Mode (1218)
Random flow
Route flow

Fig. 4. Start Flowchart

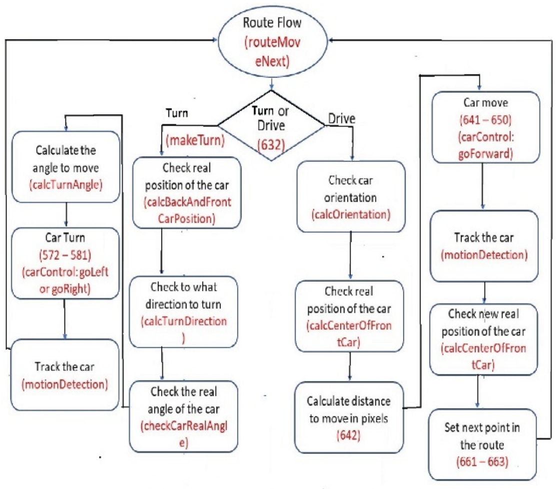
Route Flow (routeMoveNext)
Turn or Drive (632)
Turn
(makeTurn)
Drive
Calculate the angle to move (calcTurnAngle)
Car Turn (572 – 581) (carControl: goLeft or goRight)
Track the car (motionDetection)
Check real position of the car (calcBackAndFrontCarPosition)
Check to what direction to turn (calcTurnDirection)
Check the real angle of the car (checkCarRealAngle)
Check car orientation (calcOrientation)
Check real position of the car (calcCenterOfFrontCar)
Calculate distance to move in pixels (642)
Car move (641 – 650) (carControl: goForward)
Track the car (motionDetection)
Check new real position of the car (calcCenterOfFrontCar)
Set next point in the route (661 – 663)

Fig. 5. Route Mode Drive Flowchart

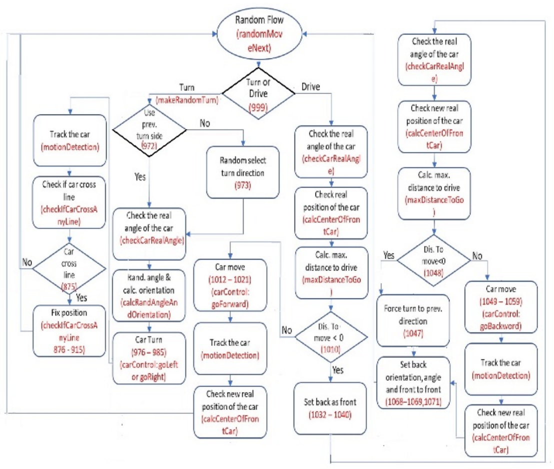
Random Flow (randomMoveNext)
Turn or Drive (999)
Turn
(makeRandomTurn)
Drive
Use prev. turn side (972)
No
Yes
Random select turn direction (973)
Track the car (motionDetection)
Check if car cross line (checkIfCarCrossAnyLine)
Car cross line (875)
No
Yes
Fix position (checkIfCarCrossAnyLine 876 - 915)
Check the real angle of the car (checkCarRealAngle)
Rand. angle & calc. orientation (calcRandAngleAndOrientation)
Car Turn (976 – 985) (carControl:goLeft or goRight)
Car move (1012 – 1021) (carControl: goForward)
Track the car (motionDetection)
Check new real position of the car (calcCenterOfFrontCar)
Check the real angle of the car (checkCarRealAngle)
Check real position of the car (calcCenterOfFrontCar)
Calc. max. distance to drive (maxDistanceToGo)
Dis. To move < 0 (1010)
No
Yes
Set back as front (1032 – 1040)
Check the real angle of the car (checkCarRealAngle)
Check new real position of the car (calcCenterOfFrontCar)
Calc. max. distance to drive (maxDistanceToGo)
Dis. To move<0 (1048)
Yes
No
Force turn to prev. direction (1047)
Set back orientation, angle and front to front (1068–1069,1071)
Car move (1049 – 1059) (carControl: goBackword)
Track the car (motionDetection)
Check new real position of the car (calcCenterOfFrontCar)

Fig. 6. Random Mode Drive

ALGORITHMS AND CALCULATIONS DESCRIPTION

Tracking Algorithm

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

Following is description of the algorithm:

1. Save an image of the area with no moving object in it – this image will be the reference image.
    a. In our case that's done when the user is press <Camera Ready> in function "onCameraReady"
2. The rest of the algorithm is implemented in the function "motionDetection". For each image the algorithm does:
    a. Change the image to gray level image
    b. Apply a Gaussian Blur on the gray level image
    c. Calculate absolute differentiation from the reference image
    d. Apply binary threshold for the result (will get black and white image where the different between the images is greater than threshold)
    e. Apply filter to reduce noises (dilate filter function from OpenCV) – the result is kept in image called "mask" (the masked image)
    f. The rest of the steps in this algorithm are implemented in function "showTrackingResults"
        i. Using OpenCV function "findContours" we find the contours of all the objects in the mask image
        ii. Using OpenCV function "max" we select the biggest object in the image
        iii. Using OpenCV function "minAreaRect" we define and save the enclosing rectangle of the car

Movement Calculations

General

All movement logics are based on the orientation of the car relatively to the images axes. We use in the software two parameters to describe the orientation:

1. carOrientation: this is the general direction of the car. The orientation names are:
    a. NW – north west
    b. NE – north east
    c. SE – south east
    d. SW – south west
2. carAngle: this is the exact angle of the car. We calculate it relatively to its general direction in the image. In other words, the angle of the car will always be in the range of 0-90 degrees.

The axes, the general directions and their names and the angle location for each direction is shown in figure 7.

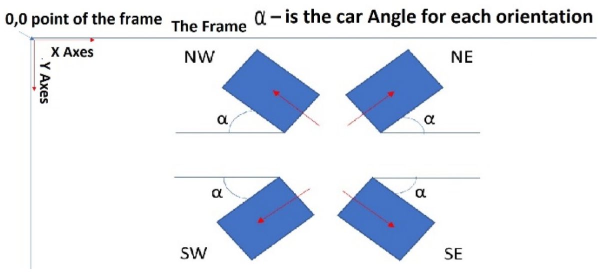

Fig. 7. The axes, the general directions and their names and the angle location for each direction

The frame of the car, which hold in array variable carBox in 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 there the other points are kept clockwise as shown in figure 8.

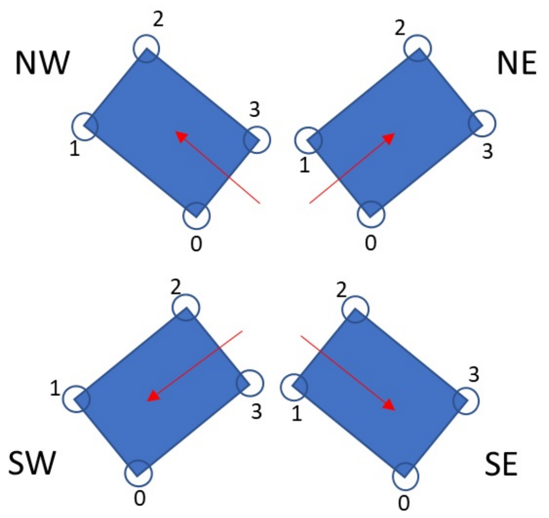

Fig. 8. The frame of the car is defined by 4 points

General Help Calculations

Based on the above explanation here are explanations on some help calculations done during the the flow of the software:

1) CALCULATING THE POSITION OF THE CAR

The position of the car is defined as the center of the front line of the car:

1. According to the orientation of the car (NW, NE, SE or SW) select the two points represent the front of the car:
   a. NW – points 1 and 2
   b. NE – points 2 and 3
   c. SE – points 3 and 0
   d. SW – points 0 and 1
2. Calculate the middle of this segment

2) CALCULATE THE REAL ANGLE OF THE CAR

1. According to the orientation of the car choose the two points along the car
2. Calculate the line angle = atan(dy/dx)

3) INTERSECTION BETWEEN TWO SEGMENTS

This calculation is implemented in function "find_intersection". It is based on an algorithm in Andre LeMothe's. The implementation was taken from [11].

There, you can also find a detailed explanation of the algorithm.

Route Mode Help Calculations

Now as we have the above help calculations and according to the above explanation on the orientations calculations such as: what side to turn, what is the angle of the turn are become straightforward.

To calculate the distance to drive to the next point is the Euclidean distance between the current location point to the destination location point.

Random Mode Help Calculations

1) TO CHECK IF THE CAR IS ON ONE OF THE LINES

1. Go over all the lines the user created on the video
2. For each line:
   a. Each two adjacent points, of the car, determined as a segment
   b. For each segment check intersection with the line

2) CALCULATE THE MAXIMUM DISTANCE THE CAR CAN MOVE

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. Go over all the lines the user created on the video
4. For each line:
   a. Create a vector for each side of the car:
      i. Example of a side: if the car 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 of the car to the front and on until 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.
   b. For each segment check intersection with the line
   c. If there is intersection, calculate the distance to the line
   d. If the distance is smaller than minDis, keep this distance as the new minDis.

FUTURE PLANS AND CONCLUSION

We plan to create much more complicated prototype using Kamin grant (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.

References

1. Oleg Kupervasser, Vitalii Sarychev, Alexander Rubinstein and Roman Yavich, ROBUST POSITIONING OF DRONES FOR LAND USE MONITORING IN STRONG TERRAIN RELIEF USING VISION-BASED NAVIGATION, International Journal of GEOMATE, May, 2018 Vol.14, Issue 45, pp.10-15:
2. Djaja et al., International Journal of GEOMATE, Aug, 2017, Vol.13, Issue 36, pp.31-34
3. "THE ROBOT REPORT" TRACKING THE BUSINESS OF ROBOTICS, 2012 http://www.therobotreport.com/news/robot-lawnmowers-still-a-work-in-progress
4. Oleg Kupervasser, Ronen Lerner, Ehud Rivlin and Hector Rotstein Error Analysis for a Navigation Algorithm based on Optical-Flow and a Digital Terrain Map In the Proceedings of the 2008 IEEE/ION Position, Location and Navigation Symposium, P.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 Terrarium Map "Positioning" Vol.4 No.1, February 2013
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 It is registered in the register of the computer programs of Federal service on intellectual property, patents and trade marks, Russia, on March, 24, 2016 http://www1.fips.ru/Archive/EVM/2016/2016.04.20/DOC/RUNW/000/002/016/613/305/document.pdf http://www1.fips.ru/Archive/EVM/2016/2016.04.20/DOC/RUNW/000/002/016/613/306/document.pdf
7. Ilan Ehrenfeld, Max Kogan, Oleg Kupervasser, Vitalii Sarychev, Irina Volinsky, Roman Yavich, Bar Zangbi, "Visual navigation for airborne control of ground robots from tethered platform: creation of the first prototype", Proceeding of the IEEE International Conference on New Trends in Engineering and Technology, September 2018, GRTIET, Tirupathi, Chennai, Tamil Nadu, India, http://grt.edu.in/news/international-conference-on-new-trends-in-engineering-and-technology-icntet-2018/
8. Kupervasser O.Yu., Kupervasser Yu.I., Rubinstein A.A., Russian Utility model: Apparatus for Coordinating Automated Devices, Patent № 131276, Russia, on Nov, 12th, 2012
9. Kupervasser O.Yu., Kupervasser Yu.I., Rubinstein A.A., German Utility model: Vorrichtng fur Koordinierung automatisierter Vorrichtungen. Patent Nr. 21 2013 000 225. It is registered in Germany Jul, 10, 2015
10. Kupervasser O.Yu., Franch Utility model application: Coordination System for Ground Movable Automated Devices. Utility model, publication number 3037157, Jun, 04, 2016, http://bases-brevets.inpi.fr/en/document-en/FR3037157.html?s=1482862654519&p=5&cHash=1f6f99c78cfb2124b5accb92be338388
11. "How do you detect where two line segments intersect?",https://stackoverflow.com/questions/563198/whats-the-most-efficent-way-to-calculate-where-two-line-segments-intersect