УДК 004.855.5

MAKING USE OF COMPUTER VISION APPROACH FOR IMPLEMENTATION OF SELF-DRIVING CAR

Putilina Daria, Medvediev Maxim
Scientific advisor: prof. Svetlana Antoshchuk
OdessaNationalPolytechnicUniversity, Ukraine

ABSTRACT. This article presents an approach of steering a self-driving car on the road constructed from traffic cones using Computer Vision. Our algorithm gets an image in real-time from an onboard-camera and then, depending on the results of the analysis, it calculates a steering angle. After training, our model could forecast how and when to steer the self-driving car in real time. As a result, we reached the accuracy of steering about 95%.

Introduction. In the context of international project on algorithms andoperation of autonomous methods vehicles development, was carried out project "Neurorace" by a winter scientific school. Within the framework of this school, the task was set to teach an autonomous car to drive along the road using various approaches. Our team solved this task with the help of computer visionmethods.The autonomous machine is based on the platform NVidia Jetson TX1 and is equipped with an attached camera.The track was organized as follows: the boundaries of the road were made from markers, in the role of which were usual plastic cups, the left border was marked by yellow cups, and the right-hand border was by blue ones.

Aim. The main goal was to increase an accuracy of markers detection and car steering. Due to training of the classifier we achieved an accuracy about 99% of cup recognition in normal lighting conditions and about 92% in difficult one. While the part of false negative errors was about 2% and part of false positive about 6%. After we increased a traffic cones detection precision we could also raise an accuracy of car steering, it was about 95%.

Implementation. In the framework of this school, was supposed the car moving along the road without time; the main condition was to keep the car within the boundaries of the route.As an ideal trajectory we considered car passing through the center of the route. For successful driving our car should find the cups on each side, get the coordinates from the image and turn them into "real" coordinates. With the help of these coordinates, it will be possible to calculate the right and left boundaries of the track. Then the line of motion is calculated. It is transferred to the controller, which will receive from it the angle of rotation and the speed of movement.

Proceeding from this, we identified six main tasks that needed to be solved in order to fulfill the set goal: Read the data with the attached camera and get a static image for further processing; Image processing and obtaining contours of markers (plastic cups); Finding the bounding box of markers and getting their centroids; Training classifier to increase cones recognition accuracy; Obtaining the boundaries of the track and constructing an ideal trajectory; Calculation of the required steering angle and supplying him to the controller of the machine.

Implementation/Readout. To obtain a training sample as follows: a gamepad was connected to the machine and the car was being driven several circles under operator control.We recorded data during several hours using different configurations of tracks, mix colors of cones and changing directions. In addition, to raise our successful result we decided also collect data outdoor. That allowed us to train classifier taking into account also bad light conditions.The resulting video was divided on many static images.

Implementation/Contour recognition. For the beginning of contour definition, we executed a color segmentation in OpenCV. The simplest way to perform a division of the segment was to use a color space HSV that we made. Constructing histograms, we were able to define the most appropriate thresholds for segmentation.

Second step we started from contour analysis. Contours can be explained simply as a curve that joins all the continuous points along the boundary, having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition. After that we find contours in our binary mask.We filled the contours with white color, to eliminate holes in the blob that are too large for the Morphological Closing to deal with. Then we filled the contour with black color to completely remove the objects from the mask.[1]

Implementation/Obtaining bounding box markers.Contour Moments help us to calculate some features like center of mass of the object, area of the object, so we extracted useful data like area and centroid forfurther calculating.The easiest thing we decided to do with a found contour is to calculate a bounding box that encompasses all the points that describe cup's shape.

Implementation/Classifier/Prepare data for training. To increase traffic cones recognition accuracy we decided to use a classifier. But the first step of working with classifier we need to prepare data for training.

To train the cascade first of all, it was required to collect a good training sample. In our case, we did this by taking a glass from different directions and under different lighting conditions. About 2000 positive and about 3000 negative. Accordingly, positive images must contain the object itself, and negative images contain an environment, but do not contain the object itself. In addition, positive images relieved of noise objects and cut off.

Implementation/Classifier/Selection and training of the classifier. After preparing data, we turn to Cascade Training. We decided to use Viola-Jones method. Such classifier trains very slow, but the results of searching for a traffic cone are very fast, that is why we chose this method of recognizing cones on the image.

In addition, this detector has an extremely low probability of a false cone view. This method in general looks for cups on the principle of the scanning window.

Implementation/Transforming Coordinates. Before drawing the borderlines and trajectory, we should transform coordinates with which help we could lay the way for our car. In order to get the coordinates from the camera to the cups found, the centroid of each contour was calculated. This midpoint was used as coordinate representing one cup. As transformation of the coordinates in the picture, to coordinates representing the real world, the perspective transform of OpenCV was used. To use this transformation, is needed a matrix calculated before. For this matrix, you have to take a picture of some cups. The coordinates of the cups in the picture and the coordinates in the real world has to be measured. With these, the needed matrix for the transformation can be calculated.

Implementation/Borderlines recognizing and drawing, trajectory drawing, track finding.

When we completed all needed stages to recognize traffic cones and transforming coordinates, our next step was trajectory drawing [2].As we worked on Python, we used the Numpy library to calculate our track. Due to this, we could fit a parabola. The transformed coordinates we used to calculate the track borders. As result, we fit the parabola through the cups and get the left and right track borders. With these two parabolas, that in our situation are borderlines of our track (in the form $y = a * x2 + b * x + c$) we can calculate the new parabola which represents further trajectory. Track parabola (two track borders found)(1)):

$$atrack = \frac{aright + aleft}{k} \quad (1) \quad btrack = \frac{bright + bleft}{k} \quad ctrack = 0$$

As k can be chosen freely, we used $k = 4{,}5$. Radius of parabola is (2): $rp = -\frac{1}{2a}$ (2)

The lines on the present picture are calculated back to pixel coordinates, the drawing on the right is done with the "real world" coordinates.

Implementation/Calculating steering angle and car steering. Therefore, we finished all stages that were set us to start car steering. Now we need to implement the last step: calculating steering angle. [3] To get the right track in the next step it is needed to swap some coordinates:

$x - coordinateinthepicture = y - coordinateinrealworld$ (the distance straight from the camera); $y - coordinateinthepicture = x - coordinateinrealworld$ (left and right from the camera).

The camera is the point (0, 0) in real world coordinates, in pixel coordinates this point is in the left upper corner. It is the best to build the matrix with these rules.

Conclusion. A sample of objects was formed for learning the size of 5000. Then it was divided into test and training samples so that the test portion was 20% of the total number of objects. Objects for this sample were selected at random and removed from the general array. The rest formed a training sample.

After passing all six main steps that were solved in order to fulfill the set goal, as a result, we have a self-driving car, that can move according to builded track, not ideal now, but it can complete in itself a simple track like a circle. Thus, the set task of this project was achieved.

In the immediate future we will improve our formulas and correspondingly we are working with trajectory calculating. So in the near future, we want to present our improved method of steering self-driving car.

We consider exploring the possibilities of presented algorithm in various other subject areas.For example, we offer to use our approach with goal to minimize quantity of traffic road's accidents, victims and amount of deceased person because of mistaken traffic cone recognition by drivers.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. AlexanderMordvintsev&Abid K- Режим доступу: URL: https://media.readthedocs.org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf - [OpenCV-PythonTutorialsDocumentation]
2. Prof.ThorstenScholer, Formulasfortrajectorydrawing, Augsburg 2018
3. Prof.ThorstenScholer, Anapproachofsteeringthecar, Augsburg 2018