Федеральное государственное автономное образовательное учреждение

высшего образования

«Национальный исследовательский университет

«Высшая школа экономики»

Факультет компьютерных наук

ООП «Прикладная математика и информатика»

Отчёт о прохождении практики

Студент:	Ренева Юлия Денисовна
Группа:	155
Вид практики:	Учебная

Организация: НИУ ВШЭ

Руководитель от департамента /кафедры/организации

Bruno Bauwens, assistant professor

Bauwens

<степень, звание, ФИО>

(подпись руководителя практики об ознакомлении с отчетом)

Москва, 2017

Contents:

Introduction

Nowadays, with the growing popularity of robots and quadcopters, people clearly understand, that it is very useful, to make the drone work autonomously. The field of application, where it can be used, is colossal and expanding rapidly.

The main goal of the practice was to take the software, which was developed by last year's students, and upgrade the target recognition, tracking and following, which were implemented in it.

The problem of drone automatization is a fairly complex and relevant one. The fact, that the drone can find the target, follow it, change its position depending on the position of the target brings us to the conclusion that it can implement more complicated tasks. Using these basic functions, we can make the mission more sophisticated, so one day our drone will be able to find the person in the crowd without any human help.

The drone is a low-cost quadcopter Parrot AR.Drone 2.0 of the French company Parrot. It is equipped with 2 cameras that can send an image to the computer using a Wi-Fi signal. The idea was to handle the camera stream from the drone, using a computer vision algorithm, then calculate the velocity and send it back. To control the position of the drone it was decided to use the bottom camera.

There were two main fields to improve: the CV algorithm and the controller that calculates velocity and manages the drone.

The work was carried out jointly with Georgy Kozhevnikov, 2 BAMI course student.

The project can be found in its repository in GitHub: https://github.com/Barahlush/ardrone_autopilot

Implementation

ROS

There are 3 most conductive and suitable tools to deal with the drone:

1. ROS (Robot Operating System) – open source collection of software frameworks for robot software development that provides a lot of services to work with robots and other machines. Can be managed with C++ or Python.
2. NodeCopter – module for NodeJS that allows to control the drone using high-level commands, wrote in JavaScript.
3. Parrot SDK 2.0 – official software development kit for the AR.Drone 2 that uses C as main language and specialized in mobile devices development.

Previously it was said, that the work was started by an year ahead students. The implementation tools were chosen by them, so we just decided to develop their ROS-direction in our own way.

It consists of the main framework (ROS itself) and packages that can be added or removed. Every user can implement his own package that other users can install and use.

The special characteristic of the ROS is that it contains of the special files, calls nodes, where you write different parts of code. The main feature of nodes is the possibility to transfer the data between each other, what makes a project easier to debug and develop. And of course, it gives us an ability to add different programming languages to the project.

Generally, ROS is quite suitable tool for the project and there are some worked out packages to work with the AR.Drone 2.0. Some of them are involved in the project:

- ardrone_autonomy – package-driver that allows receiving information and image from the drone and sending commands to the drone.
- tum_simulator – package, that simulates the drone in the Gazebo simulator. The package was widely used in the process of debugging the project. Not used in the final version.

But ROS has some weaknesses in its commands, so it was found some other libraries and tools, which further developers can use in their project. It will be commented later.

The project consists of the 3 main nodes: image processing node, controller node and interface node. The main principle of ROS working is a loop that repeats very fast (50 Hz) and runs the nodes. Every node checks incoming information, does it works and sends information further.

Interface

The code was written by the previous student, but we've improved it. It is written in Python using the QT library and there is an implementation of some user interface to control the drone.

Were implemented:

- Window that shows an image from the camera
- Keyboard control for the drone

Added:

- Switching cameras by pressing a key
- Enabling and disabling of the image processing by pressing a key
- Enabling and disabling of the autopilot controller by pressing a key
- Changing P-controller parameters by pressing keys

Computer vision (CV)

To recognize the target was considered to process the image using open source OpenCV library. It provides a lot of tools for image handling, objects detecting and tracking.

In the previous version of project was used rather slow algorithm based on the features point detection. It was necessary to load a target image to the project before running it. Then the CV algorithm tried to find something similar to the uploaded picture in the images from drone's camera.

The algorithm is not very stable and vulnerable to the camera rotation and image resolution. Furthermore, it was written in Python, that is not as fast as C++. The target was a complex image consists of a lot of small corners, circles and other geometric figures. Consequently, it was decided to change the way of the target detection.

The first thing was done is changing the target. Now the target is a simple image that consists of 3 big bright circles which the camera can clearly see.

Next, CV algorithm was changed and rewritten in C++. Current algorithm depends on the target's color. It extracts the part of the image with the target's color and determines the shape of the extracted object. It is possible to change the object's color making changes in special parameters in the code. In addition, there is a possibility to track multiply colors.

Thereafter, it draws ellipses around determined objects and writes coordinates of their centers. In this step, the node extracts information about the ellipses (coordinates and sizes) and sends it to the controller.

Algorithm is tied to the object’s color, therefore, it’s necessary to use some unusual and rare colors, which don’t occur in the background. Otherwise, it can detect some false ellipses in background’s objects, that will lead to the wrong motions of the drone.

Drone controller

One of the most challenging part of the practice was the drone controller. Previously it was implemented simple controller in Python, which worked only with the old version of CV algorithm.

The controller should measure parameters and send the relevant command. It was decided to write a new one in C++, to make the algorithm work faster.

The first point was to detect if the drone now exactly above the target and should we change the position, or not. It was implemented in the section of the controller.

It was decided to draw a rectangle in the center of the image, which we get from the camera of the drone. If all the circles are in this box, we don’t need to correct the drone’s position, because it is already right. If one or more targets are out the rectangle, we need to move the drone.

The next part of the controller is the motion part. It calculates the velocity and then gives the command to the drone.

After searching the information about the controllers, it was made a decision to implement a prevailing controller variation, named Proportional. With the accurate implementation, it allows the drone move to the target, not to lose it and doesn’t contribute any extraordinary intellectual expenses, but at the same time correctly execute all necessary commands.

The meaning of the controller is to calculate necessary velocity, using a difference in coordinates of the drone and the target, we call it error in the project, and send it to the drone. The bigger the error, the bigger the sending velocity.

The proportional one multiplies the error to the predetermined coefficient and the adds it to the velocity. Sometimes the drone will overfly the target, but you can easily fix it, by changing the parameters. There is a simple P-controller realization in the project.

And so we came to the most difficult part of coding the controller – tuning. It is the selection of the coefficients for which the drone performs the task the most stably. It is a hard and risky process, that can be dangerous for the drone and for the person tuning the controller.

To tune the drone, it was figured out that the driver (ardrone_autonomy) sets quadrotor's tilt and not velocity. It makes sense, because tilt is responsible for acceleration and not for the velocity. And changing position by piloting the acceleration is much more inaccurate than by piloting the velocity, because the position is the first derivative of the velocity and the second of acceleration. Therefore, there is a bigger mistake in position calculated as the second derivative than as the first.

It is expressed by excess movements and instability of the drone that sometimes led to loss of the target. Authors found 3 ways to fix this issue:

1. To implement a double-PID that will convert necessary velocity to the necessary tilt. It is a common approach and the examples of it can be found. But it can be hard to tune this double-PID system.
2. To use NodeCopter (was mentioned above) instead of ROS. It allows to control exactly velocity of the copter and not tilt. The conversion is already from velocity into tilt is already done in this library.
3. To use tum_ardrone ROS package. It also has an implemented "convertor" from the velocity to the tilt. It wasn't used in the project due to its incompatibility with the current versions of the ROS. Using an elder version can fix the problem.

Results

As a product of the practice we have a project, that supports:

1. The control of the drone by the keyboard;
2. Detection the target of some predefined color, using computer and drone cameras;
3. An autopilot that makes the drone follow the recognized target. (The drone is trying to keep the target right below its camera and flying for the target if it moves.)

Unfortunately, there are still some undecided problems:

- The drone can't stable hover above the target, it's always moving to correct the position. It also happens due to the airflows that the drone makes by its motors. Reflected by floor and walls, the flow returns to the copter and brakes its stability. It can be fixed by improving current controller (double-PID) or using another one.
- Movements are not very accurate. Quadrotor can overfly the target repeatedly before the stabilization.

Implementing the practice, we've captured a lot of new skills and knowledges:

- How to use OpenCV and how works the target detection and tracking. What are the algorithms of computer vision and usual approaches to solve problems in this area.
- How to deal with a ROS project at all and how to build it using Catkin build software.
- How the P-controller works and a lot of information about drones.
- Improved skills in working with Linux and GitHub.

The profit of the practice for me is that we have received some basic skills, how to work with drones, read a lot of information about the recent technologies, programming decisions, CV-algorithms, controllers, competitions and the tasks for competitions and also the most difficult and interesting problems in modern drone programming. Now I clearly understand the benefits of using drones in our lives.

List of used sources

- Wikipedia ------------------------------ https://www.wikipedia.org/
- ROS wiki and tutorials --------------- http://wiki.ros.org/
- OpenCV documentation ------------- http://docs.opencv.org/
- C++ documentation ------------------- http://en.cppreference.com/w/
- Python documentation ---------------- https://docs.python.org/3/
- Ardrone_autonomy documentation - https://ardrone-autonomy.readthedocs.io/en/latest/
- PID controller--------------------------- https://www.csimn.com/CSI_pages/PIDforDummies.html