Measuring drone flight-stability using computer vision

Dávid Dezső [1,*], and *Kornél* Sarvajcz [1]

[1] University of Debrecen, Faculty of Engineering, 4028 Debrecen, Hungary

Abstract. In this project we made an image processing software that we used to examine drone flight stability. One can read about the closed loop control of a quadcopter, the computer vision technics and the results of the test under real circumstances.

1 Introduction

In the last few years the use of quadcopters expanded, more people use it as a hobby and its presence can be seen in professional enviroment. Their use as a video recorder device is the most common.

We work with these kind of quadcopters in the Drone Club of the Faculty of Engineering. The size of these are much larger than a racer drone and its main attribution is stability. This means they can hold their position with the less movement possible. To reach this we need multiple sensors and their precise cooperation.

The closed-loop control needs a lot of tuning to work as precisely as it should. After an adjustment we only can be sure if the change went into the right direction when we test the position hold flight mode under normal circumstances. To see this we can only use our eyes and the impression of the pilot. To fly we need good weather conditions and this can mean that between two flights too much time goes by and we cannot be sure that the tuning is better or worse than it was last time.

We want to give a solution to this problem in this paper. To reach this goal, we have to make a testing method that gives exact data on the stability of the position holding so that multiple versions of the same quadcopter or different drones can be compared.

Our plan is to write a program that processes a recording of a drone flight using image processing algorithms. It detects a marker and shows the route of a drone in the test time. After that it makes diagrams on these we can see the results.

The full test stands of the recording, the process of the recording with the analyser software and last the evaluation of the results.

2 A QUADCOPTERS SENSORS AND CONTROL

2.1. Sensors

So that a multicopter can fly stable without special pilot abilities it needs closed-loop control. In a perfect world where all the parts the drone stands of are perfect and where there are no disturbing external forces this control system would not be needed. But in the real life there is no such thing as two perfectly similar motor or propeller. This is why we need sensors to „keep an eye” on the conditions of the quadcopter and electronics to control the actuators according the datas they get.

The most necessary sensors take place in the IMU (Inertial Measurement Unit). One of these is the gyroscope. The task of the gyroscope is to give information about the angle of the UAV (Unmanned Aerial Vehicle) compared to horizontal. For this it uses the angular momentum. In the everyday life they use it in many fields such as on airplanes in the so called artificial horizon. Two gyroscope measures the horizontal and also the vertical angles, the three axes they give information of are:

- Yaw = roll on the vertical axis
- Roll = roll on the vertical perpendicular to forward movement axis
- Pitch = roll on the vertical parallel to forward movement axis

Despite the angular momentum a drift appears in the gyroscope. This is the moving of the rotation plane. Here gets part the next part of the measurement unit, the accelerometer. It can compensate the forces that cause this drift. The principle of operation of the accelerometer is the same as the behaviour of a mass fastened to springs, the forces moves the mass so the lengths of the springs change. These two sensors are cooperating with each other and we need both to get the right datas.

Knowing the position is just as important as the angle. Primary is the height. For this a barometer is integrated to the IMU. The barometer measure the atmospheric

* Corresponding author: dezso.david15@gmail.com

pressure. The vertical position of a multicopter specifies the height of the air pillar above it and the weight of that. Naturally the pressure of the air around the drone is highly affected by the temperature. That is why we also need a temperature sensor at the side of the barometer. With the information of these two can be calculated the exact altitude of our quadcopter.
Staying at the calculation of the flight altitude a few words about a sensor that is used in the middle-class and higher priced UAV as a standard equipment in the last few years. This is the ultrasonic distance sensor. This is placed at the bottom of the multicopters vertically downwards. A sensor like that send an ultrasonic wave from the „trigger" transceiver, this wave is reflected from an object, in this case the ground, and the receiver called „echo" detects it. The time difference between the released wave and the received one shows how much time th ultrasonic wave needed to go back and forth. We can calculate the altitude if we know the speed of the wave that is 343.2 m/s in 20 degree celsius. Of course the ultrasonic wave cannot go large distances. That is why it is only useful while take off and landing or during low altitude flights as a completion of the barometer.
The fully autonomous position holding cannot be achieved without a GPS receiver. This sensor with triangulation calculates the position of the UAV using its own clock signal, the time code and orbit data received from the satellites, to do this the GPS sensor needs minimum 4 satellite signals [1].

2.2 Control system

After the introduction of the sensors used in control let's talk about the closed loop control itself. The UAV needs closed loop control to keep its state stable while flying. Stability here means that despite the environmental effects and despite the imperfection of the structural elements the drone stays in a balanced state. A system like that consists of a controller and a controlled section.

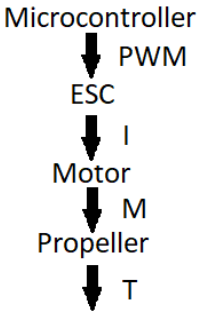

Fig. 1. : Controlled section of an UAV

On Fig. 1. the controlled section of a quadcopter can be seen. The microcontroller acts according the controller section's commands by sending Pulse Width Modulated (PWM) signals. The electronics of the motors translates the PWM signal to the current the brushless DC motor needs. This electronics called ESC (Electronic Speed Controller). The motors rotates the propellers with the momentum needed to reach the thrust force we want to archive [2].

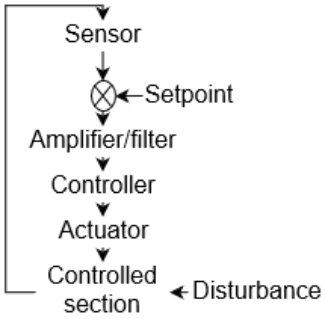

Fig. 2. : Controller section of an UAV

In quadcopters, PID (Proportional–Integral–Derivative) control units are used to create the command needed for the intervention unit, that is, the microcontroller. After the sensor signal has been merged, a differential generator compares the set signal that is sent by the autopilot or the remote control operator. After amplifying and filtering, it reaches the PID unit. Such a unit consists of three parts: The P, proportional member output signal is proportional to its input signal. The integrating I, which output signal is proportional to the integrity of the input signal. Finally, the D, differentiator part, its output signal is proportional to the change in the input signal.
Most quadcopters' central computers are pre-programmed to self-tune the closed-loop control but we may achieve better results with self-adjusting. The custom method, as well as the built-in, compensating parameter setting is done using the Ziegler-Nichols method. The loop gain is increased until a permanent swing is obtained. With drones this is a continuous tremor in the floating. Then the period of time for the critical open loop gain can be determined. These data can be used to determine the value of each of the three parts by using the given relations [1].

3 OBJECT FOLLOWING PROGRAM

3.1 Importing libraries

The list and short description of the libraries needed to operate the program:

- CV2: The OpenCV (Open Source Computer Vision) library gives thousands of algorithms for image processing and computer vision [3].
- Numpy: This is a indispensable algorithm package if you use OpenCV. Python's basic library for scientific calculations. It contains multi-dimensional array elements creating functions. Tools for integrating C / C ++ scripts. It can be used to calculate linear algebra, Fourier transformation and random number generation [4].
- Matplotlib: The Python's two-dimensional drawing library. It allows you to create diagrams for presentations and evaluations. Interactive environment for communication between platforms [5].

- Imutils: A free to use library made by Adrian Rosebrock directly for OpenCV. With it you can transform and resize images [6].

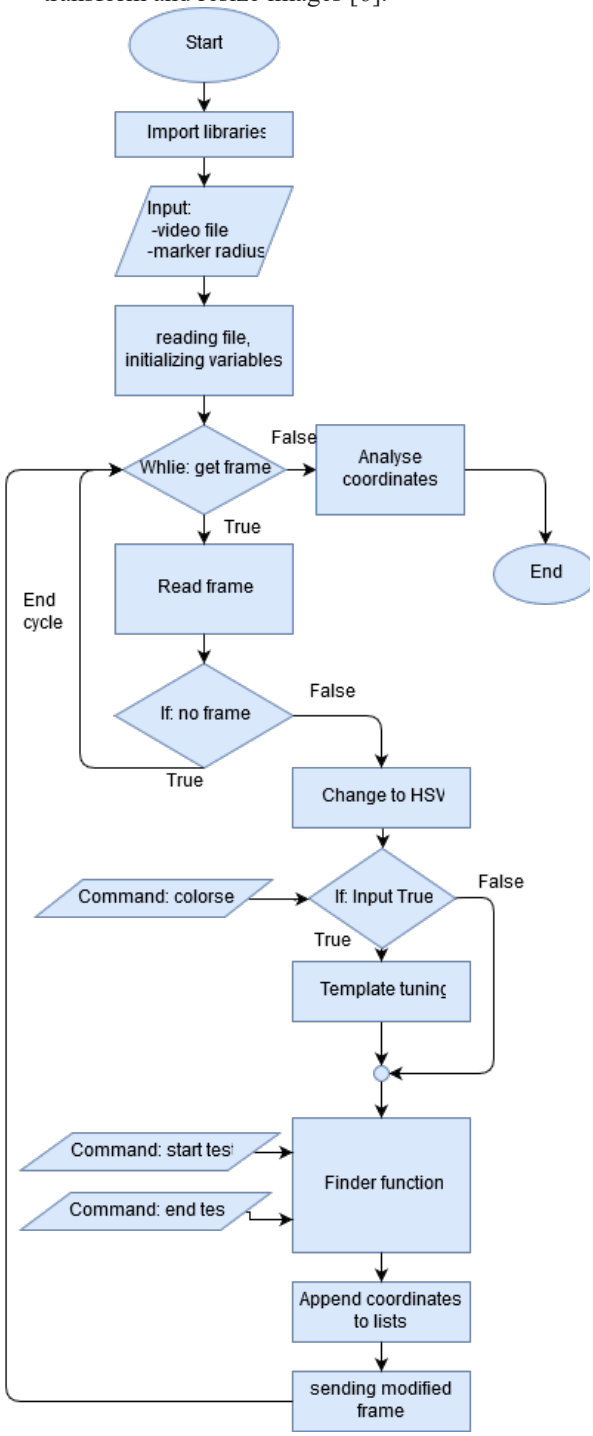

Fig. 3. : Simplified flowchart

3.2 Data input

When a Python program is started, an operation window pops up immediately. After the startup, the following message appears:
„Insert name of the video file:”
That is, the program needs the name and extension of the recording you want to test. In case the file is not in the same folder as the program, the name of the file must be entered along with its path. Since the OpenCV's primary punctuation is real-time image processing, of course this is possible. If you want to process continuous stream, the video file name must be replaced by the serial number of the connected recorder, ie. "0". After the filename is read, another initial variable is required, and this is the circle radius of the object marker in millimeters:
„Radius of the circle (in mm):”
For this input I have set a default value of 25 mm. By sending this you will also start processing and the following message will appear in the keypad window: „Press 'C' to set the colour (esc if done) S' to start test and 'Q' to quit.”
The flowchart shown in Fig. 3. has three inputs in its while loop, which are used to control the software. The message above provides information on this.

3.3 Detecting cycle

After the video file has been called in and the necessary variables and lists initialized, the program enters its main cycle. This is where the object is detected and the necessary data for motion analysis is recorded. As a first step, the next frame of the recording is called with the "getframe" function, which returns a boolean value with the frame that indicates there is another frame or not. If you have an empty image, the variable is "False". That follows the test, so if the program runs out of frames, it interrupts the cycle and jumps to the coordinate analysis part. Then the frame size is reduced by using the "imutils" library, thus accelerating the processing. Then the "cvtColor" function changes the color space. RGB encoding switches into HSV (H = hue, S = saturation, V = value) color space. With this encoding, the boundaries can be defined, which means that if only the marker remains, detection will be successful in the color search function. Values are defined in the next section. At this point, the video file is displayed from frame to frame and the program is waiting to receive one of the commands from the keypad. Pressing the "C" key sends the "colorset" command. At this point, the video stops in a given frame. New window pops up, you have three windows for the current frame, the mask, and one where six sliders represent the color boundary values. Once they appear, the program enters a while cycle. The positions on the sliders are scanned continuously. Using these, you create a new mask that is updated in the previous window. It is a good idea to start the border search by adjusting the hue, as it can only slightly change and thanks to the homogeneous marker it can be within narrow limits. After that, it is worth refining the picture by shrinking the boundaries of saturation. The final set is the brightness, in this case it is not fortunate to set it for only that frame, since the movement of the drone can change the brightness, and if it is outside the boundaries, wrong coordinates will be given. When we are done with setting the boundaries, pressing the "ESC" key accepts the current HSV values and the main cycle of the script continues. After that comes the "finder" function written by me, which detects the marker on the frame, changes the displayed image and returns the X and Y coordinates and the radius of the marker in pixels.

3.4 Color finder function

The "finder" function mentioned in the previous chapter gives the root of the software. When calling, three arguments are required:

- HSV: The frame in new colorspace.
- MIN: The minimum boundaries set on the „colorset" sliders.
- MAX: The maximum boundaries set on the „colorset" sliders.

These are all needed in the first step. As in the mask setting, it also uses the "inRange" function to highlight the marker to the template image. This examines all the pixels if the color values are within the defined limits. It gives the corresponding pixels "High" value on the template and all other receives logical "0" values, so the template itself can be interpreted as a binary array. Our next task would be to find the contours, but in this case the operation of the program would not be impeccable, it would make mistakes several times, making it impossible to perform the test. This is because the edge of the mask in the template is uneven and overall too noisy from the scattered "High" pixels. The problem is solved by morphological transformations. The collective name of multiple interconnected image processing tasks in computer vision. In our case erosion and dilation are applied by the software. During erosion, it slides the template below itself and only the pixel remains "1" which a high value is found on both layers. Thus the tiny noises from the template disappears, but the contour that is important for us will also be smaller. Although the program would reduce the size of the reduced mask, the radius values collected for the later process of analysis would be altered, causing the accuracy of the test result to fall. That's why dilation is also needed. What makes the inverse process, that is, every pixel gets a high value with a value of "1" on either layers. Here you get the original size of the contour you want, but the lost noise will not come back. It can be seen that the two processes are the same, the difference lies in the fact that eroding is the logical "and" the dilatation acts as the logic "or". Subsequently, the template is now suitable for finding contours. This means that you can determine the visible shapes by searching for "0" - "1" borders. This is done by CV's "findContour" function. Creating an array of all the found contour's datas, the task is to choose only the largest of these, after all, the boundary setting was made so that only the marker would appear on it. The software runs on all frames to this point. I made it like this because these transformations are the most complex operations that require the most time. Therefore, if the test starts, there will not be a noticeable deceleration in the processing, and until the beginning of the test, the frames won't be shown too fast.

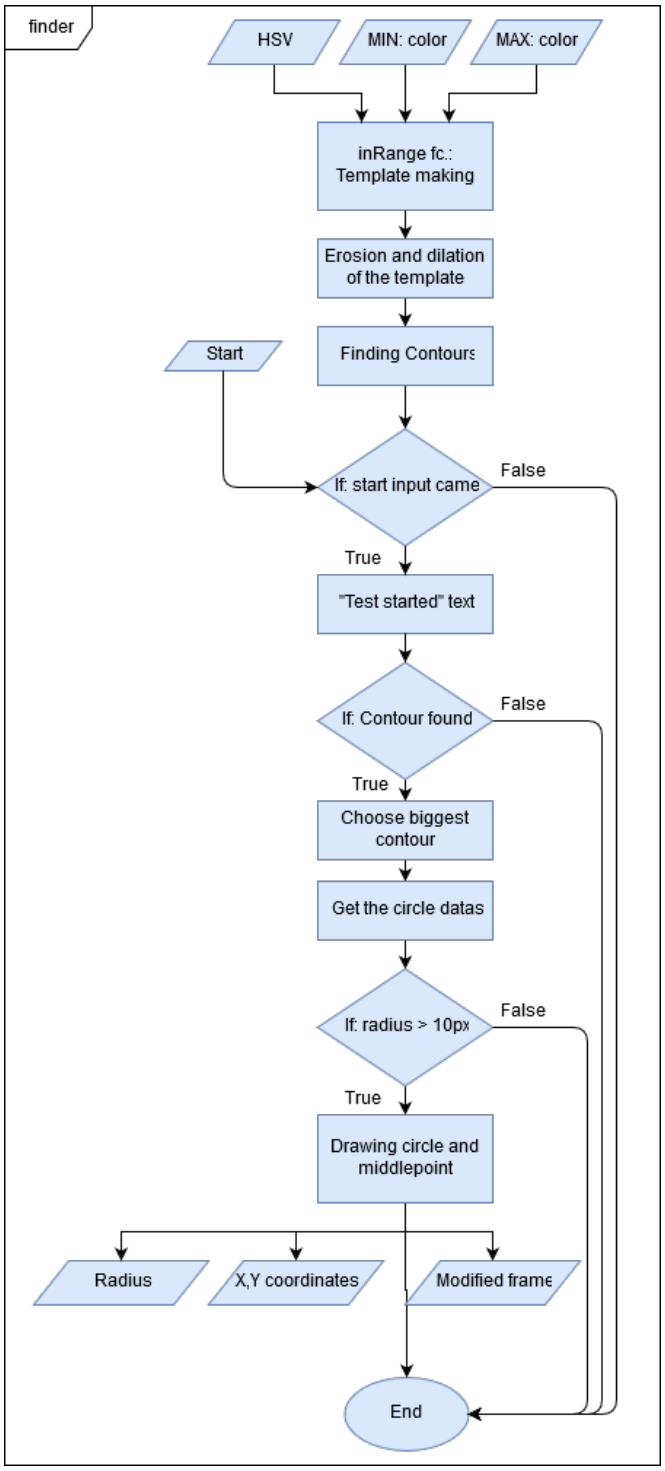

Fig. 4. : Color-finder function

The start of the test begins with the sending of the start signal as it is seen in the flowchart (Fig. 4), as it was explained in the explanatory message mentioned above, by pressing the "S" key. As a primary feedback, a yellow "TEST STARTED" text appears in the upper left corner of the window. It will then be checked if it has found a contour. If yes, it uses the OpenCV's "minEnclosingCircle" algorithm. Entering the largest contour as an argument returns three values, all of which can get from the smallest writable circle around the contour. They are in the order: the X coordinate of the circle's center, the Y coordinate, and finally the radius. If

the radius exceeds a minimum value, the circle and the center are displayed as feedback on the image. This is the end of the function. Returns the coordinate and radius values as a closing.

3.5 Analyzing coordinates

The main cycle produces three lists of Y, X coordinates and radiuses. They contain an element for each frame, before the test is started "0" values are given, so it is necessary to filter them. For this task, I defined a simple function to shorten the script. The original list contains the image coordinates, those are the positions of the object on each frame. As a first conversion, a relative movement list is developed. The first element of the filtered coordinate lists is named for the origo, then it replaces all the others with the difference between them and the origo value. This is still only suitable for comparison of percent deviation, the general use requires real movement determination. This is achieved by the list of radiuses. Because the software recognizes the true radius of the marker, since the user specified it at the beginning of processing. Calculates the actual movement of the visible axes in the frame with the following equation:

$$\frac{Radius\ in\ pixel}{real\ radius} = X \quad (1)$$

$$X = Millimeter\ pixel\ coefficient \quad (2)$$

$$\frac{Movement\ in\ pixels}{X} = Real\ movement \quad (3)$$

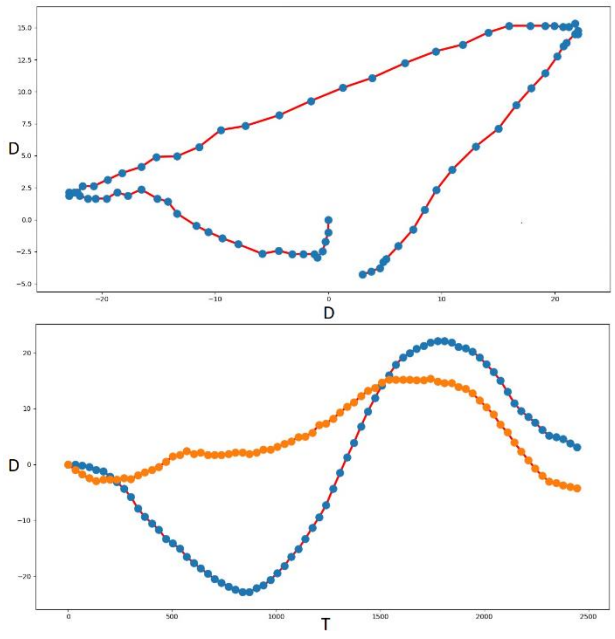

Fig. 5. : End result diagram

The Fig. 5. above shows the first result of the image processing. The X and Y axes return the full motion route. Both axes are in millimetres. This diagram shows rather the success of tracking than it is needed for evaluation. Two functions are displayed on the next diagram. With those we are capable to evaluate the motion. The vertical axis shows "d" movement in millimetres, and the horizontal "t" axis is the time. When the program generates a chart that can be magnified, it can be read up to 10 millimetres and milliseconds. .

4 Position hold test

4.1 Phantom 1 test

The test was originally designed for laboratory conditions, that is, adequate GPS signals, good lighting, and windless environment. They were not available, but it is possible to observe the effects of lack of these. At the time of this test, there was a constant 7 km / h wind and heavy occasional gusts of 15 km / h during the flight while the air temperature was 12 ° C. The drone was unable to hold its position, but apparently worked against the wind, without the control even the droneframe's horizontal close angle would not be possible. A persistent, slightly accelerating sliding can be observed. Within this 3.5 seconds, there have been some outbreaks in this measurement, but they can still be evaluated. After that, the functions are not as smooth as before, there is an explanation for this, the quadcopter has been moving away more than five meters from the camera. I limited the motion evaluation to the first three seconds.

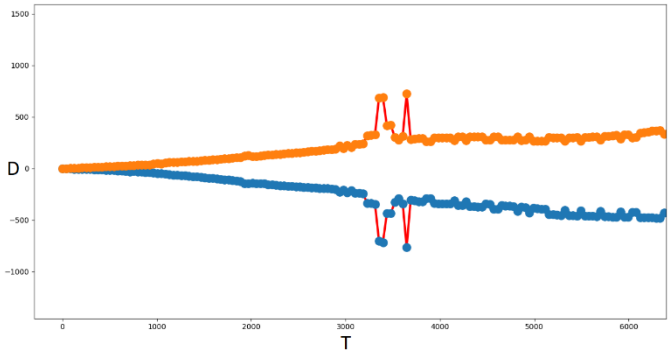

Fig. 6. : Phantom test

Table 1. Phantom test

t [s]	0,5	1	1,5	2	2,5	3
d_X [mm]	-17,4	-49,9	-86,8	-136,7	-178	-208,3
v_X [m/s]	-0,0348	-0,065	-0,0738	-0,0998	-0,0826	-0,0606
a_X[m/s^2]	0,0696	0,0604	0,0176	0,05	-0,0344	-0,044
d_Y [mm]	21,7	49,9	78,1	117,2	154,1	200
v_Y [m/s]	0,0434	0,0564	0,0564	0,0782	0,0738	0,0918
a_Y [m/s^2]	0,0868	0,026	0	0,0436	-0,0088	0,036

The horizontal motion on the "X" axis is negative, meaning that it is floating towards the left edge of the frame, which is to the right of the drone's point of view, the same as the wind direction. The speed accelerated to varying degrees for two seconds, then began to slow slightly, but the entire diagram showed that the direction of the speed did not change.
Movement along the vertical axis shows a steady increase which, with the exception of two small overthrow, is also steadily accelerating. On both examined axes, the

movement of the drone was approximately 20 centimetres from the initial.

4.2 Inspire 1 test

I tried to reverse the use of program with an inverted approach during the test of the DJI Inspire 1 drone. In this case, the marker is not attached to the droneframe, but to a stable point, specifically on the ground. Thus, the program captures the relative motion of the video maker to a reference point.

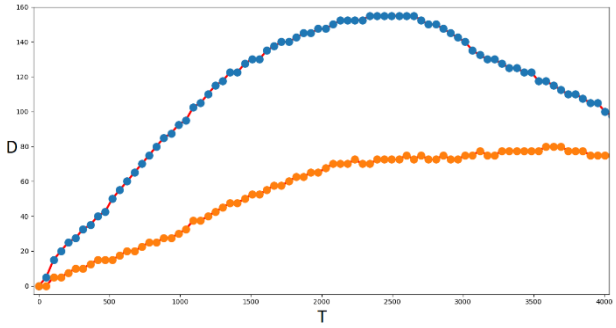

Fig. 7. : Inspire test

Table 2. Inspire test

t[ms]	400	800	1200	1600	2000	2400	2800	3200	3600	4000
d_X [mm]	39	77,2	110,5	133,8	147,5	155	150	130	117	100
v_X [m/s]	0,195	0,191	0,167	0,117	0,069	0,038	-0,03	-0,1	-0,07	-0,09
a_X [m/s^2]	0,98	-0,02	-0,12	-0,25	-0,24	-0,16	-0,31	-0,38	0,18	-0,1
d_Y [mm]	14,5	25,2	40,3	54,6	66,3	72,6	72,6	75	80	75
v_Y [m/s]	0,073	0,054	0,076	0,072	0,059	0,032	0	0,012	0,025	-0,03
a_Y [m/s^2]	0,363	-0,095	0,11	-0,02	-0,07	-0,14	-0,16	0,06	0,065	-0,25

During the test, it was windless and about 12°C. One possible mistake was that it was surrounded by a densely built environment that could disturb GPS signals. There is a slow slide here, but it is also captured by these sensors. Axis "X" shows the movement perpendicular to the direction of motion. Beginning at the starting point to the right at a speed of 0.195 m / s. The control reacted to that and slowed down smoothly, and by the time it reached the maximum 15.5 cm shift the drone started moving backwards. There is a very similar tendency along the "Y" axis. It moves forward at a slower starting speed than on the other axis. The response speed declaration is also smaller and even though it stops for 2.8 seconds, it reaches its maximum movement after 3.6 seconds and then corrects itself back.

5 Summary

During this project we were able to create a software that uses color detection method to follow a marked object on the video file of its movements. We then completed it with properties that make it suitable for use in a measurement to determine the stability of a quadcopter during the position hold mode. These properties include customizing the boundaries of the searched color, starting and ending the analysis of the recording at a command, determining the true motion and displaying the result of the test in an evaluable form. Once all these have been completed, the test of real conditions has taken place. Measurements were made using two commercially available UAVs. In addition to the appropriate conditions, the software can be used during the development of quadcopters to analyse the effects of changes in hardware or hardware.

6 Acknowledgement

The work/publication is supported by the EFOP-3.6.1-16-2016-00022 project. The project is co-financed by the European Union and the European Social Fund.

References

1. „hobbielektronika.hu,”[Online]. Available: https://www.hobbielektronika.hu/cikkek/multicopterek_nullarol_az_uav-kig_i.html?pg=1. (2015)
2. Phd. J. Tóth , Author, Measurement and Control Technology I. . [Performance]. University of Debrecen, Faculty of Engineering Debrecen, Hungary (2015)
3. „Opencv introduction,” OpenCV, [Online]. Available: https://opencv.org/about.html (2017)
4. „Numpy,” Numpy, [Online]. Available: https://www.numpy.org/ (2017)
5. „Matplotlib,” Matplotlib,. [Online]. Available: https://matplotlib.org/ (2015)
6. A. Rosebrock, „Pyimagesearch,” [Online]. Available: https://www.pyimagesearch.com/ (2017)
7. Phd. J. Tóth , Author, Measurement and Control Technology II. . [Performance]. University of Debrecen, Faculty of Engineering Debrecen, Hungary (2015)