The ClujUAV student competition: A corridor navigation challenge with autonomous drones ⋆

Csanád Sándor * **Szabolcs Pável** * **Erik Wieser** * **Andreea Blaga** **
Péter Boda ** **Andrea-Orsolya Fülöp** ** **Adrian Ursache** ** **Attila Zöld** **
Anikó Kopacz *** **Botond Lázár** *** **Károly Szabó** *** **Zoltán Tasnádi** ***
Botond Trinfa *** **Lehel Csató** *** **Tegzes Dan Marius** ****
Pop M. Leontin **** **Raluca Tarziu** † **Mihai Zaha** † **Sorin Grigorescu** †
Lucian Busoniu ‡ **Paula Raica** ‡ **Levente Tamas** ‡

* *Team Drone Whisperers contact: csanad.sandor@cs.ubbcluj.ro*
** *Team Drop Table contact: blaga.andre@yahoo.com*
*** *Faculty of Mathematics and Informatics, Babeş-Bolyai University, team Flying Penguins contact: lehel.csato@cs.ubbcluj.ro*
**** *Technical University of Cluj-Napoca, team Just the Drone contact: tegzes.dan97@gmail.com*
† *RoVisLab, Transylvania University of Braşov, team RoVisLab contact: Raluca.Tarziu@student.unitbv.ro*
‡ *Technical University of Cluj-Napoca, organizer contact: Levente.Tamas@aut.utcluj.ro*

Abstract: We describe a novel student contest concept in which an unmanned aerial vehicle (UAV or drone) must autonomously navigate a straight corridor using feedback from camera images. The objective of the contest is to promote engineering skills (related to sensing and control in particular) among students and young professionals, by means of an attractive robotics topic in an exciting competition format. The first edition of this contest was organized in Cluj-Napoca, Romania on October 19th 2019. Teams from industry and academia competed, with an overall positive experience. We outline the challenge and scoring rules, together with the technical solutions of the teams, and close with a summary of the results and points to improve for the next editions.

Keywords: student competition, UAV, robotics, computer vision, control, education.

1. INTRODUCTION

Unmanned aerial vehicles (UAV) have a long history in robotics (Carelli and Freire, 2003) and are currently becoming widespread in everyday life, see e.g. the autonomous aerial taxis reported recently by the BBC (2019).

For a fully autonomous behaviour, a series of building blocks have to be implemented, like (1) perceiving of the surrounding environment, (2) localizing using a map, (3) trajectory planning for navigation, and (4) low-level flight control, to enumerate just a few one. Most of these modules strongly depend on the environment: they will be different in structured, indoor versus outdoor environments, and will behave differently in full daylight vs. dimmed, cloudy conditions.

Several contests were organized in the near past for both indoor and outdoor UAV platforms, focusing on different tasks such as people tracking (NIPS, 2019), line following (Mathworks, 2019), mapping or object detection (ELROB, 2019) or safety (NASA, 2020). In October 2019, we held for the first time in Cluj-Napoca, Romania a contest involving small scale indoor UAV platforms. The challenge of the competition was to design perception and control algorithms for an UAV so as to achieve autonomous navigation in a structured, corridor-like environment.

Our main motivation for organizing the UAV flight contest was to promote the importance of STEM (Bermudez et al., 2019) for young professionals as well as special skills required for UAV perception and control tasks. All of these are excellently exemplified by the four UAV behavior features enumerated above, while perception and control take the center spot in our UAV challenge. Moreover, the robotics domain ensures a seamless vertical competence integration of the student skills earned during a bachelor degree in the computer science and control engineering field (Chen and Chiu, 2016). Finally, a key advantage of robotics and UAVs in particular is that they are attractive to students.

For our – first – 2019 call, in total five teams attended the Cluj-UAV autonomous flight contest and they represented both the academic and industrial sector. For this first edition the target was to fly through an unknown corridor(-like) environment with a Parrot Ar.Drone 2 UAV. The teams were allowed to choose their favorite method for solving the perception and control problems that arose.

⋆ This event was financially sponsored by Accenture and Elektrobit; by the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PN-III-P1-1.1-TE-2016-0670; by the European Commission via grant agreement no. 9/2018, Co4AIR - Computers, Cognition and Communication in Control: A strategIc paRtnership" project funded through the Erasmus+ KA2 scheme; and by a HAS Bolyai Scholarship. Corresponding author: Levente.Tamas@aut.utcluj.ro

In Section 2, we review the main technical work that sits behind the students' approaches, as well as related contest concepts. Following that, in Section 3 we detail the contest, and in Section 4 the teams outline their approaches, focusing on the unique points of each. Then, Section 5 summarizes the results. Section 6 concludes the paper and outlines some directions for future improvement.

2. RELATED WORK

Related contest concepts include both academic (Mathworks, 2019), industrial (NASA, 2020) and military (ELROB, 2019) variants. The educational impact of each contest in part is specific to its audience, however the UAV related contests are still in focus today having impact on the educational side as well (Bermudez et al., 2019).

A autonomous drone flight challenge is organized by Mathworks (2019) at the IFAC 2020 World Congress targets a line following scenario, with a full deployment chain from Matlab code to embedded hardware for two low cost drone platforms. An important educational impact is expected, specifically for students from engineering curricula. The competition organized by NASA (2020) targets more realistic scenarios, where the variability of the environment is important. The situation is similar in the case of the ELROB competition (ELROB, 2019), where the drone should be able to recognize generic targets in the wild. The most similar drone contest to ClujUAV is the one reported by Bermudez et al. (2019): they created a simulation environment in ROS/Gazebo for an ArDrone like UAV, and the teams could evaluate their algorithms within this realistic environment, for a task or navigating through floating frames. A dynamic model of the drone is available, which teams can use for low level control.

Our contest is unique among autonomous UAV competitions due to a combination of three factors: the use of real drones, the simplicity of the task, and the focus on computer vision and high-level control tasks.

On the technical side, one can distinguish two clusters of problems for the autonomous flight in corridor-like environments: the perception and the control of the drone. For both topics, significant related work exists in the robotics literature (Mathworks, 2019; NASA, 2020).

The problem of scene recognition is still an active research topic, although several prior papers are related to this problem. In the early 2000's global scene perception related questions were reported in (Oliva and Torralba, 2001; Torralba et al., 2003; Fei-Fei and Perona, 2005; Quattoni and Torralba, 2009) focusing mainly on holistic scene understanding. Later on different visual feature related solutions were proposed, including linear (Grompone von Gioi et al., 2010) or planar (Kim et al., 2018) ones. The solution described by Kim et al. (2018) tackles the navigation problem in an active exploration context with a SLAM framework.

According to the sensors used for spatial reasoning, there is also a great variety in this research field: 2D cameras are the most popular solutions (Páll et al., 2015; Shichao Yang et al., 2016; Dorbala et al., 2019) but laser (Pasteau et al., 2013; Park et al., 2015) or ultrasonic sensor based (Carelli and Freire, 2003) ones are also frequently used.

Due to the uncertainty in the detection and the disturbances caused by the moving camera, some authors propose specialised filtering algorithms both for the IMU and image readings for enhancing the quality of the perception (Páll et al., 2015; Park et al., 2015). On the control side there are also different approaches that were investigated starting from simple proportional controllers (Páll et al., 2015) up to complex Lyapunov-based variants (Carelli and Freire, 2003).

Of course, deep-learning based solutions have also been proposed for this problem, e.g. for layout estimation (Shichao Yang et al., 2016) or corridor middle point extraction (Dorbala et al., 2019). Both variants proved to be relevant also for the experiments carried out by the teams in the competition.

3. THE CLUJUAV CONCEPT AND CHALLENGE

The main challenge of the contest was to autonomously navigate a Parrot AR.Drone 2 UAV along an approximately 2m wide corridor, based solely on the information from the front camera and the IMU unit. The autonomous flying is nontrivial by the limited camera resolution and the reduced field of view, as well as the noisy IMU readings from a low-cost UAV platform.

A bird-eye view of a typical navigation scenario is presented in Fig. 1. The longitudinal navigation direction for the drone can be placed in a relationship with the *vanishing point* (Páll et al., 2015), i.e. the center of the corridor, where the lines of a perspective view intersect each other. Many of the adopted navigation solutions by the team members rely on the estimation of this point from the perspective camera, and the guidance of the UAV with respect to this reference.

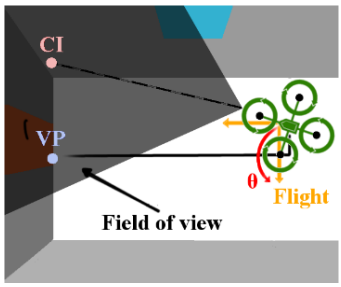

Fig. 1. Bird-eye view of the corridor with a UAV showing the field of view including the vanishing point (VP) indicating the center of the corridor and the camera image center (CI) Páll et al. (2015)

.

In the preparation phase of the contest (half a year before the contest date), each team was allowed to use their favorite software for the implementation. The hardware constrains were only related to the UAV platform: the affordable, well documented ArDrone2 was considered as standard platform for all the teams.

Our scoring rules were as follows: 30% part of the score was awarded for the technical approach as well as its connection to the state of the art from the domain. The second, and largest (70%) part of the score was based on the experimental solution to the challenge.

In the morning of the contest, each team was allowed 10 minutes for setting up and possible last-minute tweaks. Before this, they never used for testing the corridor of the contest.

During the challenge, each team had two independent trials along the corridor (so they were allowed to reinitialize the drone once, after a possible completion or crash). The teams were evaluated according to the number of collisions with the walls of the corridor: a smaller penalty if the drone recovers, and larger if it crashes; as well as with a penalty related to the flight time. Teams also received a bonus when they passed half the corridor length; and when they completed the full corridor length. These scores were recorded by an independent jury composed of three members. Based on the total scores, the team received their final ranking and prizes. Monetary prizes were awarded using sponsorships from the Accenture and Elektrobit companies.

There were five teams attending the contest, 2 from industry and 3 from academia. Most of the teams were local, with one team from a different city. The total number of participants from all the teams was 17. All of the students were at the BSc. or MSc .level. In the next section, the technical solutions adopted by each team are summarized.

4. THE SOLUTIONS PROPOSED BY THE TEAMS

Next, the teams outline their approaches to the problem, in alphabetical order of the team name.

4.1 Drone Whisperers

The solution of this team to the contest had two main components: a vanishing point estimation based perception module, and a PID based control module. The solution was implemented using the ROS framework (Quigley et al., 2009), providing the toolkit for quick prototyping and debugging of the algorithms.

The different modules were implemented as nodes in the ROS framework. All computations were offloaded to a laptop device, while communication between the drone and the laptop was handled by the ArDrone-Autonomy (Monajjemi et al., 2012) ROS node, which implemented the driver and the necessary interfaces for receiving sensor data and issuing control commands. The perception module was also included in a single node, and had the front camera images as input, and the coordinates of the vanishing point as output. The control of the drone was realized using two more nodes, a PID controller taking the vanishing point coordinates as input, and outputting yaw rates. A final node implementing additional control logic took the yaw rates as inputs, and communicated with the driver node to send the final processed commands, closing the control loop. We also used the toolkit provided by the ROS framework to debug our control loop. We recorded multiple sequences and tested the perception algorithm without the need for using the drone all the time. We also recorded logs about our test flights, which helped us to discover potential corner-cases and during parameter tuning.

The perception module used the front camera of the drone to decide the optimal direction of flight. Our first attempt was based on Structure-from-Motion methods, where we used an open-source, camera- and IMU-based fusion algorithm (Qin et al., 2018), and further implementation based on an epipolar geometry (Hartley and Zisserman, 2003). In both cases we faced difficulties, the main reason being the keypoint detection and matching step used by both algorithms. We were mainly interested in keypoints situated on the wall of the corridor, but because of their uniform appearance they had very few well distinguishable corner points or blobs on them. As a consequence the number of identified keypoints was too small for our use case.

As a final solution we used a vanishing point estimation pipeline: first we rectified the image based on parameters estimated using Zhang's calibration method (Zhang, 2000), then converted it to gray-scale, followed by a Canny edge detector (Canny, 1986) and a dilation to find edges. The edge detection step is followed by a probabilistic Hough line transform (Hough, 1962) to detect straight lines. The returned lines were then filtered based on their lengths and slope: we were interested in lines corresponding to the intersection of the floor and the walls, whose slope depends on the width of the corridor and it is usually well bound. The remaining lines were assigned to the four quadrants of the image based on the position of their midpoint. We consider a line intersection a possible vanishing point only if a line from the left half of the image intersects a line from the right side. This simple heuristic filtered out most of the spurious intersections. Finally a single vanishing point was computed as the median of all line intersections on the image. While this pipeline gives a separate vanishing point per image, we also used a temporal exponential smoothing to avoid quick changes in the vanishing point, possibly resulting in sudden changes of flight direction.

The control module controlled the yaw rate of the drone while maintaining a constant forward velocity. After takeoff the drone was set into hover mode, and waited a few seconds until it stabilized. Then a PID controller was initiated, which took as input the horizontal coordinates of the vanishing point (which had to stay in the middle of the image), and produced the yaw rate. The parameters of the controller were tuned manually using a trial and error approach. During our experiments we mainly focused on the proportional and derivative components, which provided a stable trajectory for drone. The yaw rate and the constant forward velocity command were sent out to the drone simultaneously.

Our main problem in the competition was caused by the corridor floor: it had painted black stripes at 45 degree angle. As our perception module was based on an edge detection and vanishing point estimation, it was important to filter these lines to avoid the appearance of incorrect vanishing points. Recorded logs of the test run showed, that while these stripes were correctly filtered in most of the frames, a few failures were enough to divert the drone from the correct trajectory, resulting in collision with the walls. Fig. 2 shows correct and incorrect vanishing point detection from the records.

4.2 Drop Table

The solution uses classical image processing algorithms, aimed for detecting edges and lines. Our customised algorithm detects the two lines between the floor and the walls of the corridor. After the detection our algorithm moves the drone according to the computed line angles. Fig. 3 shows the overall algorithm.

In order to detect the lines, we first applied the Canny edge detector algorithm (Canny, 1986) to get all the edges in a frame. We fined tuned its parameters in order to detect the relevant edges. The next step was to apply Probabilistic Hough Transform (Matas et al., 2000), using OpenCV implementation. This grouped edge pixels into line segments. Again, we fine tuned its parameters to get the relevant lines. The fine tuning of

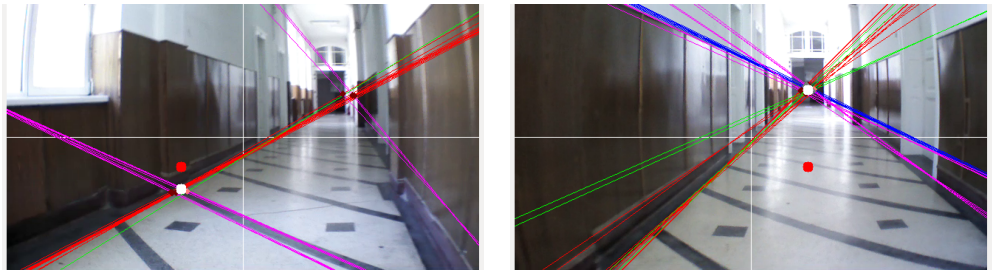

Fig. 2. Edge and vanishing point (white dot) detection by the perception node. **Left:** incorrect vanishing point position due to the black stripes on the floor. **Right:** correct vanishing point estimate.

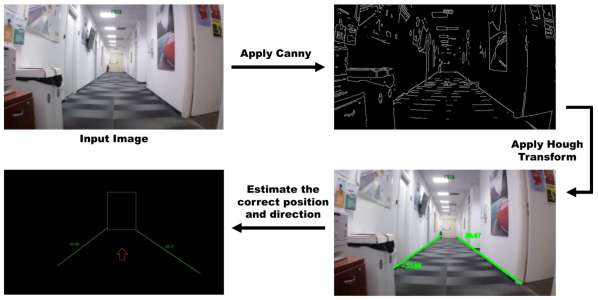

Fig. 3. Algorithm overview of the Drop Table team

the algorithms' parameters was made on the spot, after multiple tests. Afterwards, the lines were filtered according to their position in the frame. Each individual line slope was computed, then converted to angles. We are interested in two diagonal lines, one in the left half of the image (with negative angle), the other in the right half (with positive angle). We build two buffers with angles (one for each side) and average their values in order to avoid the effect of noisy detection.

After getting enough data in the buffers, the drone can move according to the angles' averages (in absolute values). If the right average is bigger than the left one, the drone steers slightly to left. If the left average is bigger, then the drone steers slightly to the right. If the averages are near the same value, the drone moves forward.

Our control method is as follows: an open loop system having to main elements: capturing the video stream, process as per algorithm, output commands. So, we capture frame by frame the video stream, then process it (Canny, Hough Transform) estimate position and direction by calculating the relevant left and right line angles. Finally, we send output the commands that need to be executed by the drone.

This team also implemented a safety check so that, if detection fails for a consecutive number of frames, the drone lands. The main issue of this solution is that the video processing is not done in parallel with the drone's movement commands.

4.3 Flying Penguins

Given the relatively reduced algorithmic complexity of the challenge, we aimed for a simple algorithm and an appropriate calibration such that it performs the task in an optimal way. Our team assumed that there are no obstacles when flying the drone, i.e. we found the target point, we direct the drone such that it goes towards the point we identified. Similar to the Drone Whisperer team, we targeted the *vanishing point*.

The vanishing point was identified each processed frame from the video sequence using the following pipeline: first blur the image – a resolution reduction –; then to extract the edges using the *canny edge detection* method, followed by the *probabilistic Hough* algorithm to find the lines.

Our team tested the method to identify the vanishing point based on the algorithms provided by the *OpenCV* system, but soon we identified two problems:

(1) the vanishing point was moving way too abruptly for a controller to be reliable;
(2) due to artefacts e.g. on the floor of the corridor, the line detection algorithm identified lines that were outliers that made the OpenCV method perform extremely bad.

To counter both side effects from above, we implemented a *robust vanishing point detection* method that used as *prior* the value of the previous detection.

The algorithm: We used a *probabilistic setup* [1] that considered – for each step – an a-priori distribution for the vanishing point, modelled as a *symmetric Gaussian distribution*, parametrised by the mean position $\boldsymbol{\mu} = [\mu_x, \mu_y]$ and the scaling factor of the variance σ.

For each frame the detected lines were considered as *observations* determined the vanishing point in the following way: the vanishing point was closest to all valid lines. From each line we derived the following likelihood:

$$P(\boldsymbol{p}_v|\ \mathrm{seg}_n) \propto \exp\left(-|\mathrm{dist}_n(\boldsymbol{p}_v|\right)$$

and to find the new position of the vanishing point, we evaluated the following integral:

$$p_{post}(\boldsymbol{p}_v) \propto p_0(\boldsymbol{p}_v|\ \boldsymbol{\mu}, \sigma \boldsymbol{I}_2) \prod_{n=1}^{N} P(\boldsymbol{p}_v|\ \mathrm{seg}_n).$$

Since we used the *robust likelihood* – as given above – the a-posteriori distribution is not analytical and we employ an approximation to it using the *Gaussian quadrature*.

The Bayesian and probabilistic treatment of the vanishing point detection allowed us the mitigate the observed instabilities of the standard vanishing point detection procedure in the OpenCV system and simultaneously it allowed for a more stable estimates.

Control: We used the horizontal position of the detected vanishing point and applied a controller that corrected the displacement from the centre position in this direction. The forward speed of the drone was increased according the certainty of our

[1] We followed a quasi-Bayesian methodology to find the most probable position of the vanishing point.

estimation of the vanishing point, σ. A minimal base speed was applied in the forward direction in order to lower the instability of the drone caused by the turbulence in the narrow corridor. We had to experiment with horizontal corrections in order to avoid side collision with the corridor wall. The slopes of the detected lines on the corresponding side of the frame reflect the distance between the current position and the wall.

Lessons Learned: The benefit of our approach was to adapt to the temporal absence of valid lines to determine the vanishing point. We also had to be robust and to filter the detected lines as best as we can: e.g. the pattern on the floor that originally mislead the vanishing point detection algorithm. After correcting for the misleading floor lines (as seen in Fig.2.a) – using the assumption that lines on the floor do not contribute to the vanishing point – we were able to fly the drone safely.

Our conclusion is that the robust approach to the vanishing point detection was justified and that given the task at hand one can – and *should* – adapt the algorithm to the special features of the task.

4.4 Just the Drone

Short introduction: Our approach was to implement the idea found in (Dorbala et al., 2019), so in order to do this we used their dataset which is composed from a set of images and a specific value assigned to each image. The value represents an angular velocity that was obtained by extracting features like vanishing point (VP) and vanishing line (VL) from each image and passing them to a control law in order to obtain the velocity vector. After training an algorithm on this dataset we implemented a proportional control for the drone, on the rotation around the Z axis, based on the angular velocity value obtained from each frame.

Training the algorithm: In order to implement an algorithm on this dataset, made by 3023 images, we split it into a training set (2617 images) and a validation set (406 images), 87% training images, respectively 13% validation images. This dataset can be found online [2] .

After we split the data set, we took the ResNet-50 architecture (He et al., 2015), that was pretrained on ImageNet dataset, and we fine-tuned this model on our training images (resized to 224x224).

To do this we removed the last four layers from the model and added a Flatten, Dense (ReLU activation), Dropout and a final Dense layer with linear activation (Fig. 4). We added this types of layers because we needed to eliminate the specific purpose for which the ResNet was trained, so we put two regular densely-connected NN layers, but not before we flatten the input for the first Dense layer, and adding a Dropout layer after that, because this will stop the model from overfitting due to the fact that we had a small dataset. Also, for determining the gradients for backpropagation we used a Mean Squared Error (MSE) loss, defined as:

$$\text{MSE} = \frac{1}{2n}\sum_{i=0}^{n}(\hat{\omega} - \omega)^2,$$

where $\hat{\omega}$ is the ground truth velocity, and ω is the predicted velocity.

[2] `https://drive.google.com/open?id=1R4ctBxnCxHc433Ls6gYgjtnSikZsLQ0Z`

We used a batch size of $n = 32$ for training over 20 epochs, and used an Adam optimizer, an extension to stochastic gradient descent, with a learning rate of 0.0002.

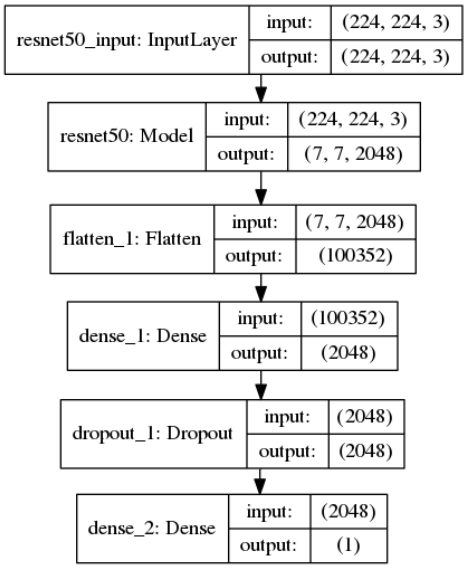

Fig. 4. Layers added to the ResNet50 model

Control: After training the Convolutional Neural Network (CNN) on our dataset we tested to see the values predicted, so that we observed that the angular velocity predicted by our model was pretty accurate, because if the drone was positioned to the left of the hallway center, it predicted negative values (right command), respectively if it was positioned to the right of the wall it predicted positive values (left command) (Fig. 5).

Fig. 5. Values predicted by the model, relative to the center of the image, marked with red

The only thing remained was to scale this values because they were too big to give them as direct commands to the drone so we divided them by 2. Now the drone is moving forward with a constant (small) speed on the X axis, and for each frame is rotating either to the left, either to the right, based on the sign of the angular velocity and just by the half of this value, around the Z axis, as you can see in the machine state diagram Fig. 6.

A problem we encountered with this approach is that when the drone camera gets too close to the wall, to the point where it's vertical on the wall, the algorithm will predict some unreliable values, which will lead to poor control. This is a worst case scenario, so in order to get past this problem we need to design a corner case on which the drone makes a quarter rotation along the Z axis.

Lessons Learned: First of all, working on this project taught us the importance of starting early and doing a lot of upfront work. Another valuable lesson we learned is that data normalization

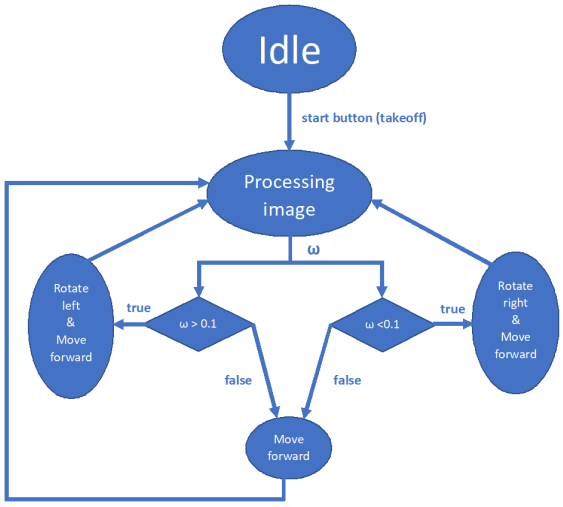

Fig. 6. Machine state diagram of the drone control

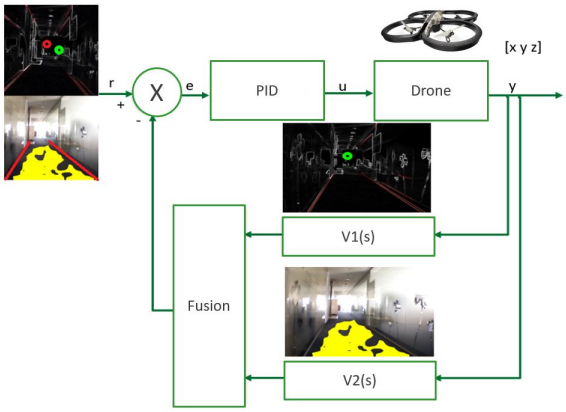

Fig. 7. Drone control diagram

and cleaning is mandatory. Often the data you have and with which you have to model a process will be noisy and inaccurate, so cleaning it is the first step.

4.5 RoVisLab

The method we adopted is a combination of multiple image based detection algorithms such as vanishing point detection and artificial intelligence, which together with a PID controller realize the autonomous flight of the drone. The solution was implemented in C++ using the OpenCV library and a neural network based on the ResNet model.

In order to detect the hallway we first applied a Canny edge detector (Canny, 1986) whose thresholds we adjusted depending on the structure of the corridor. Afterwards we used a Hough line transform (Matas et al., 2000) to extract the straight lines from the pre-processed image and keep the ones that have a slope in a specific range. The remaining lines were then intersected obtaining the vanishing point of the hall. This point is then inserted in a PID controller (Hari Om Bansal, 2012), which gives as output the rotation velocity of the drone so that it always faces the end of the hall.

For a better precision of the detection we also used a neural network, which detects the floor of the corridor in order to be able to place the drone in the center of the hallway. For this movement we are controlling the velocity of the drone on the y axis through a PID controller. After combining these two methods for the detection we apply a constant velocity on the x axis.

Table 1. Times obtained by each team

Team	DroneW.	DropT.	FlyingP.	JusttheD.	RoVisLab
Time(s)	58	64	75	49	53

As mentioned before, we utilized a PID controller whose parameters we tuned through experimentation. First we set a value for the proportional component and obtained an oscillation movement on the y axis around the desired point. Next we added a small derivative gain, which we increased until we considered that the flight is stable enough. We determined that these two components were the ones that gave us the smoothest and most accurate flight. Some of the problems we encountered were caused by sudden changes in the light brightness, which alter the detection, also the excessive number on lines detected by the algorithm because of the patterns of the corridor, for example the slightly sloped stripes on the floor.

In conclusion, the corridor navigation was quite a challenge and working with an UAV was not an easy task, but it taught us how it works like a team and that there is always place for improvement.

5. SUMMARY OF CONTEST RESULTS

In the morning set-up trials, results were excellent: nearly all the teams reached the end of the hallway, and there were only a few crashes. In the actual challenge however, which took place at noon, only one team reached the end of the hallway, and two more navigated along three quarters of the corridor. Our podium consisted of these three teams: Just the Drone in the first place, RoVisLab in the second, and Drop Table in the third. The detailed run-times are visible in the Table 5: for each team we considered the flight time over the corridor, even if they were not able to finish the route. For reaching the end of the corridor extra points were earned.

We hypothesize that the lower performance in the afternoon was largely due to lighting conditions. Most of the drones crashed when passing through a very large-contrast area from dark to direct sunlight, see Fig. 8. Moreover, the floor of the corridor chosen had inclined dark stripes, which confused vanishing point detection algorithms (since most of them rely on line detection). In addition, performance suffered due to disturbances such as air currents and, in one case, a clump of dust blocking one of the propellers which immediately crashed the drone. It is interesting – although not statistically significant – to note that the winning team was the one where machine learning was the most important component of the control. A video of this winning run is available at `http://rocon.utcluj.ro/files/clujuav_justthedrone.mov`.

6. CONCLUSIONS AND FUTURE IMPROVEMENTS

After the contest, the organizers ran a survey among the teams. Out of the five teams, four responded, and all of the respondents reported a good overall positive experience, with three stating their willingness to attend a second edition of the contest. Points of improvement noted were the fact that the rules of the challenge were finalized quite late; that the lists of teams and participants weren't made public prior to the contest; and that a clearer schedule and timing would be needed.

Fig. 8. High-contrast region along the corridor.

We plan to address all these points for the next edition. Moreover, to improve sensing and control reliability, three solutions are possible: (i) changing from AR.Drone to a other platforms, (ii) changing to a controlled-lighting location, or (iii) suggesting a significantly more challenging array of testing scenarios for the team, so that their solution is more robust. Nevertheless, overall we believe that ClujUAV had a significant positive impact and is a useful contest concept, which we definitely plan to refine and reiterate in the coming years.

REFERENCES

BBC (2019). Flying taxi makes test flight over Singapore. https://www.bbc.com/news/topics/cljev44x05lt/drones. [Online; accessed 19-Oct-2019].

Bermudez, A., Casado, R., Fernandez, G., Guijarro, M., and Olivas, P. (2019). Drone challenge: A platform for promoting programming and robotics skills in k-12 education. *International Journal of Advanced Robotic Systems*, 16(1).

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6), 679–698.

Carelli, R. and Freire, E.O. (2003). Corridor navigation and wall-following stable control for sonar-based mobile robots. *Robotics and Autonomous Systems*, 45(3), 235 – 247.

Chen, C.H. and Chiu, C.H. (2016). Employing intergroup competition in multitouch design-based learning to foster student engagement, learning achievement, and creativity. *Computers & Education*, 103, 99 – 113.

Dorbala, V.S., Hafez, A.H.A., and Jawahar, C.V. (2019). A deep learning approach for robust corridor following from an arbitrary pose. In *2019 27th Signal Processing and Communications Applications Conference (SIU)*, 1–4.

ELROB (2019). The European Land Robot Trial. https://www.elrob.org/. [Online; accessed 19-Oct-2019].

Fei-Fei, L. and Perona, P. (2005). A bayesian hierarchical model for learning natural scene categories. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, 524–531 vol. 2.

Grompone von Gioi, R., Jakubowicz, J., Morel, J., and Randall, G. (2010). Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4), 722–732.

Hari Om Bansal, Rajamayyoor Sharma, P.R.S. (2012). Pid controller tuning techniques: A review. *Journal of Control Engineering and Technology*, 2, 168–176.

Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

He, K., Zhang, X., Ren, S., and Jian Sun, C. (2015). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Hough, P.V. (1962). Method and means for recognizing complex patterns. US Patent 3,069,654.

Kim, S., Manduchi, R., and Qin, S. (2018). Multi-planar monocular reconstruction of manhattan indoor scenes. In *2018 International Conference on 3D Vision (3DV)*, 616–624.

Matas, J., Galambos, C., and Kittler, J. (2000). Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding*, 78(1), 119–137.

Mathworks (2019). MathWorks Minidrone Competition. https://de.mathworks.com/academia/student-competitions/minidrones/ifac-2020.html. [Online; accessed 19-Oct-2019].

Monajjemi, M. et al. (2012). ardrone autonomy: A ros driver for ardrone 1.0 & 2.0.

NASA (2020). Safeguard with Autonomous Navigation Demonstration. https://www.nasa.gov/san. [Online; accessed 19-Oct-2019].

NIPS (2019). Game of Drones – Competition at NeurIPS 2019. https://www.microsoft.com/en-us/research/academic-program/game-of-drones-competition-at-neurips/. [Online; accessed 19-Oct-2019].

Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3), 145–175.

Páll, E., Tamás, L., and Buşoniu, L. (2015). Vision-based quadcopter navigation in structured environments. In *Handling Uncertainty and Networked Structure in Robot Control*, 265–290. Springer, Cham.

Park, J., Kim, T., and Park, T. (2015). Autonomous navigation system for a mobile robot using a laser scanner in a corridor environment. In *2015 IEEE/SICE International Symposium on System Integration (SII)*, 512–516.

Pasteau, F., Babel, M., and Sekkal, R. (2013). Corridor following wheelchair by visual servoing. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 590–595.

Qin, T., Li, P., and Shen, S. (2018). Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4), 1004–1020.

Quattoni, A. and Torralba, A. (2009). Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 413–420.

Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A.Y. (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Shichao Yang, Maturana, D., and Scherer, S. (2016). Real-time 3d scene layout from a single image using convolutional neural networks. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2183–2189.

Torralba, Murphy, Freeman, and Rubin (2003). Context-based vision system for place and object recognition. In *Proceedings Ninth IEEE International Conference on Computer Vision*, volume 1, 273–280.

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22.