World Academy of Science, Engineering and Technology
International Journal of Aerospace and Mechanical Engineering
Vol:16, No:6, 2022

Visual Odometry and Trajectory Reconstruction for UAVs

Sandro Bartolini, Alessandro Mecocci, Alessio Medaglini

Abstract—The growing popularity of systems based on Unmanned Aerial Vehicles (UAVs) is highlighting their vulnerability particularly in relation to the positioning system used. Typically, UAV architectures use the civilian GPS which is exposed to a number of different attacks, such as jamming or spoofing. This is why it is important to develop alternative methodologies to accurately estimate the actual UAV position without relying on GPS measurements only. In this paper we propose a position estimate method for UAVs based on monocular visual odometry. We have developed a flight control system capable of keeping track of the entire trajectory travelled, with a reduced dependency on the availability of GPS signal. Moreover, the simplicity of the developed solution makes it applicable to a wide range of commercial drones. The final goal is to allow for safer flights in all conditions, even under cyber-attacks trying to deceive the drone.

Keywords—Visual odometry, autonomous UAV, position measurement, autonomous outdoor flight.

I. Introduction

DURING past years, the UAV market has been mainly aimed at the military defense industry. However, the recent advancements in microelectronics, the reduction in manufacturing costs, the growth of computers' computational power, and their miniaturization, have caused an impressive spread out in the diffusion and application of this technology for civil purposes [1]. At the same time, the progress in research fields like Artificial Intelligence and Computer Vision allows the extraction of a wide range of useful information during UAV flights. This is why, nowadays, UAVs are used for: real-time monitoring and surveillance of dangerous areas [2], improvement of everyday critical situations [3], emergency rescue activity [4], and cooperation with humans for entertainment or industrial purposes [5], while many other future applications are just around the corner [6]. A particularly relevant aspect regarding the use of these technologies is related to the safety of both the UAVs and the information they gather/store. An increasingly widespread problem comes from some kind of attacks aimed at disturbing the correct functioning of a UAV to frustrate its use or steal it. Since the flight of UAVs is strictly related to the goodness of the available GPS signals, they can stop working properly due to both signal-related problems (attenuation, reflections, multi-paths, Canyon effect) and/or intentional external attacks (jamming, spoofing, replay techniques) [7]. The first case concerns a series of well-known GPS signal weaknesses, the latter, instead, pertains to a set of possible GPS attacks based on how it works. Using one of these techniques, one can force the UAV to interrupt its mission or deceive it to control its behavior. In the past 10 years, cases of GPS-based drone strikes have been increasingly reported in different countries, like Russia, the UK, the USA, and Hong Kong to name a few. Moreover, these attacks have had different targets, like airports, ships, military equipment, civil celebrations, and wildfires. For example, in Oct. 2018 in Hong Kong, attackers caused 46 professional drones, used in a light show during a local festival, to suddenly plummet into water (approximately 100,000$ of damage) [8]. In Dec. 2011, on the other hand, Iran's army managed to capture a USA Lockheed Martin RQ-170 drone aircraft with a spoofing attack [9]. In June 2017, twenty ships navigating in the Black Sea reported their navigation systems believed to be located miles away from their actual position. Later, researchers find out evidence of the incident being a GPS spoofing attack made by Russian military powers [10].

Sandro Bartolini, Alessandro Mecocci, and Alessio Medaglini are with the Department of Information Engineering and Mathematics, University of Siena, Siena, SI, 53100 Italy (e-mail: sandro.bartolini@unisi.it, alessandro.mecocci@unisi.it, alessio.medaglini@unisi.it).

II. Related Works

The emergence of GPS reliability problems has caused the development of a series of techniques aimed to detect jamming, spoofing, or other kinds of attacks and provide some possible remedies. As reported in a survey about GPS countermeasures against spoofing attacks [11], a large set of possible techniques could be used to detect and disable such attacks. Unfortunately, the countermeasures proposed do not stop this kind of attack, merely alerting the user of the GPS receiver about suspicious activities. This will decrease the odds that a spoofing attack can succeed, but it will not prevent it from happening anyway. For this reason, some different and more effective ways of managing such a problem have been presented, which are mainly related to the use of visual odometry and some other sensors, to integrate the positioning information of the UAV with visual information. In [12] the authors proposed a procedure to control all six degrees of freedom (DOF) of a UAV using the principle of optical flow together with ultrasonic, infrared, inertial, and pressure data, to enable fully autonomous flight for indoor applications. Furthermore, authors in [13] presented another approach to estimate the UAV translational velocity and position, again based on an onboard optical flow sensor, to perform real-time autonomous outdoor and indoor flight without using the GPS information. These works show how sensor data fusion can replace GPS information when it is blocked or unreliable. A later work [14] proposed a small-scale low-cost ARM-based stereo vision system, which is used not only as an onboard

sensor to continuously estimate 6-DOF UAV pose but also as an onboard preprocessor for visual information extraction. In [15] the authors presented a vision-based approach where the vehicle was localized using a downward-looking monocular camera with a visual SLAM algorithm in charge of tracking the pose of the camera and building an incremental map of the surrounding region. The objective of their work has been to develop a vision-based micro aerial vehicle (MAV) controller to be used in an unknown environment without the aid of any infrastructure-based localization system or any prior knowledge about the environment. All the previous solutions have their own drawbacks related either to the reliability of the developed system or to its practical viability (price, weight, size, limited flight environment). It is important to note that all the above-mentioned works used dedicated hardware or built the UAV from scratch using sometimes complex algorithms which require a relatively high computational cost for the hardware usually available on-board in commercial UAVs.

In this paper, we present the results of an approach based on monocular visual odometry capable of overcoming the problems mentioned above. In particular, we are capable of reducing the influence of GPS-based attacks by limiting the drone's dependence on such data to only some sub-parts of flight. More precisely, we have developed a system that limits the use of GPS to an initial phase of alignment. During such a phase, a series of measurements is collected and then used to reconstruct the entire trajectory traveled by the UAV with reduced dependence on the availability of GPS data. In little words, the drone can fly based on visual information rarely integrated by GPS data. From the point of view of the UAV's equipment, our work requires the presence of a stabilized 3-axis gimbal camera, an Inertial Measurement Unit (IMU), and a barometric sensor. All these devices are already usually present onboard commercial drones and consequently, our research results can be applied without modifying the UAV in any way. In fact, our experimental tests are based on the use of a commercial drone available on the market, and our work does not need to make any change to the way it operates, to reach the highest possible compatibility level. The presented solution is low-cost, accurate, and simple, it can be adapted to a wide range of commercial drones without a major effort and it can obtain remarkable results with very low requirements from a hardware point of view. This will allow us to solve the GPS reliability problem in a simple way and with a wide range of possible applications.

III. Visual Odometry

In general, visual odometry systems can be divided into two main categories: mono visual odometry and stereo visual odometry, being the second typically more reliable [16]. In our case, however, we have used a monocular visual odometry system, since most commercial drones have a single camera. The mono systems only provide 2D positioning information but without any knowledge about depths, because the depths of scenes are, generally, not measurable with a single camera. As a consequence, the trajectory can be reconstructed only up to a scale factor. To estimate the scale factor, additional information is needed coming from other sensors aboard the UAV. In particular, in our work, we solved this problem using the information collected by the barometer, which allows us to reconstruct the altitude of the UAV with good precision. Moreover, barometer data are reasonably resistant to external attacks. So, once the position of the UAV has been estimated up to a scale factor, thanks to odometry techniques, it will be sufficient to replace the information about the z-axis with that coming from the barometer, to obtain the complete correct reconstruction.

The first approach we tried, and the most natural one, is to estimate an affine 3D transformation between the **ref** (GPS-based) coordinate system and the **odometry** one. To employ this estimate in the trajectory reconstruction process, the odometry-estimated z-coordinate is continuously replaced with the one estimated by the barometer: in the following this approach will be referred to as **3D + barom**. Since the altitude of the flight is considered as given, the error discussed in the following is only related to x and y coordinates. This assumption is also based on the fact that using barometer measurements to replace GPS altitude leads to an error which, at the flight altitude of a commercial drone (a few tens of meters), can be considered negligible [17]. Moreover, it is computed only for points higher than 15 meters, because it is assumed that for lower altitudes landing systems are in use. A second approach called **2D + barom** was also tried: assuming that the UAV takes off vertically (hence the orientation of the z-axis of the odometry coordinate system is substantially orthogonal to the ground), the training pairs are projected onto the ground plane and a 2D affine transformation is estimated. Then, before employing it in the trajectory estimate process, the altitude estimation is added, again given by barometer readings.

The other main problem of monocular visual-odometry systems is the drift error [18]. This problem is usually solved through local optimization (typically based on the last m camera poses), sensor fusion, or by using prior knowledge about the scene [19]. Another possible solution is to periodically compare the GPS trace and the estimated one: if no anomalies are detected, the GPS readings can be considered "safe" and taken as additional input in a new alignment procedure, performed on-the-fly. On the other hand, if the GPS readings are considered not reliable, the trajectory must be estimated with old data only. Performances can be further improved, at the cost of a heavier computational burden, by continuously realigning traces by exploiting visual cues in the scene, under the assumption that a previous reference flight has been done and such cues have been geotagged.

The odometry algorithm used in our work is based on the Semi-Direct Visual Odometry (SVO) algorithm by Forster et al. [20] whose open implementation is available [21]. This algorithm adopts a hybrid approach to visual odometry, that uses both direct methods and feature-based methods. Moreover, from the studies reported in the original paper, it requires very few computational resources when compared with most existing algorithms [20]. In fact, the SVO algorithm can reach up to 400 frames per second on an i7 processor and the time to process a single frame is only slightly higher even

on a Jetson TX1 board.

IV. Experimental Apparatus and Methodology

In our experiments, we used two commercial drones from DJI, namely: a Mavic 2 Enterprise Dual, and a Mavic Pro. Both of these UAVs are equipped with: a 3-axis gimbal-stabilized camera, a GPS module, a barometer, and an Inertial Motion Unit (IMU) to store the acceleration, velocity, and orientation of the UAV. It should be noted that the camera frequency is 30 fps while the frequency of the GPS module is only 10 Hz. This leads to a different sampling rate between images and GPS data with a ratio of three images for each GPS point. This is a relevant aspect that has been mitigated by down-sampling the video at the same frame rate as the GPS information to avoid the problems related to the correct coupling of data. Moreover, even if down-sampling is applied, no information is lost thanks to the fact that the adopted sampling frequency is relatively high compared to the drone flight speed. In fact, considering a flight altitude between 60 and 90 meters and the speed of a UAV usually around 4-8 m/s, the frames that are discarded due to the down-sampling procedure do not produce a relevant scene change for odometry calculation purposes.

To apply visual odometry, some prior knowledge of the camera system is required. In this work, the pinhole camera model was adopted, with RadTan distortion for the gimbal camera of the UAV. The pinhole camera model is the most widely used in computer vision, it is rather simple but quite effective [22] as a first-order approximation of the mapping from a 3D scene to a 2D image. To perform the monocular camera calibration the `camera_calibration` ROS package was used with a standard black-white chessboard target [23]. The procedure takes as input the 3D coordinates of object points and their corresponding 2D projections and applies the algorithm proposed in [24], which is optimized to exploit multiple views of the same planar surface. The basic idea for our flight control system is to start every flight by reaching a chosen target altitude and then performing an alignment phase. The GPS measurements are acquired and assumed to be the actual current UAV position, to collect a certain number of pairs (P_{odom}, P_{GPS}). P_{GPS} is the position recovered with GPS: its coordinates are in a Cartesian coordinate system, they are called **ref** and follow ENU convention. P_{odom} represents the corresponding estimated position in the **odometry** coordinate system (which has an arbitrary scale and whose origin and orientation with respect to **ref** depends on the initial UAV pose). Then, data collected in the alignment phase are used to estimate the affine transformation, using one of the already explained modes (**2D + barom** or **3D + barom**). Lastly, the obtained affine transformation is applied to all the subsequent estimated odometry output, to get an approximation of absolute positioning, concerning the takeoff point, without relying anymore on GPS signal validity. In particular, the duration of the alignment phase was fixed at 70 seconds, which gives a good compromise between the need to collect a certain number of corresponding points (required to achieve a robust transformation), and the limited autonomy of the vehicle battery. It is important to note that the selected duration is perfectly compatible with the normal flight of a drone since generally the UAV take-off procedure, up to the target altitude, requires tens of seconds anyway. The proposed approach works under the assumption that, during the alignment phase, the GPS readings are reliable enough, which is reasonable since the alignment is performed near the takeoff point (hence, at a *safe* location). As already said, since down-sampling is applied, a perfect time matching between images and GPS data is not possible and so the pairing is not extremely precise (it is one of the most impacting sources of error). It is worth noticing that most of the other approaches assume a high frame rate sensor stream to retrieve scale information in monocular odometry via sensor fusion [25]. Furthermore, in the first offline implementation of the software, which will be illustrated in the next section, imprecise time matching is also caused by the limited reliability of the flight log: it can be noticed that the log reports the recording starting time with an unpredictable delay (in a few cases it is negligible but it can reach up to one second) with respect to the actual beginning of storing camera frames. Hence, an offset compensation procedure is performed during the hyper-parameters tuning, to obtain good quality matches.

V. Implementation

The software implementation was divided into two phases, called offline and online, explained below. This subdivision was decided in order to ensure that the developed position estimation method was safe and reliable before applying it to real flights, so as not to endanger people and devices.

A. Offline Implementation

In the first phase of our work we used an offline setup, namely, we analyzed data taken from a complete log of already ended flights. For this reason, the ROS communication system was used in this phase. In fact, in this way, we can publish the data stored in the log files into a ROS topic and then subscribe to it for retrieving the sensors' information. This structure can be easily used also in the real case, in which the messages arrive in real-time from the UAV. In fact, if we replace the source of streaming data from log files with real-time data, this will not require any changes to our code. A first Python utility takes care of transforming the GPS point coordinates, taken from the flight log, into the coordinates of a Cartesian reference system, centered on the takeoff place. To carry out the transformation from GPS to Cartesian take-off reference system, the transition from geodetic to ENU (East, North, Up) coordinates has been made [26]. Moreover, another Python script is responsible for streaming a saved video (which comes from the drone's internal storage) into a specific ROS topic. For the odometry part, as already said, the SVO [20] algorithm was used, of which a ROS implementation exists. After the flight, data were analyzed from a python script that: a) aligns the first portion of the odometry output with the GPS trace; b) applies the obtained transformation to the entire odometry output; c) computes an error metric based on the Euclidean

distance formula. In particular, the matching process takes care of an eventual time offset between the video stream's clock and the GPS clock. Then it further filters the matches according to their height and speed. In fact, better results will be obtained by keeping only the data with a height close to the target altitude, since in this way the transformation we get will be specialized to manage data obtained around that height. Furthermore, choosing data with a sufficiently low speed gives us better quality GPS measurements, thus reducing the impact of imprecise matching. Since aggressive filtering could reduce too much the number of matches used in the estimation process (negatively affecting the robustness of the transformation), adaptive tuning is applied during the alignment phase. A RANSAC-based method was used [27], taking advantage of the implementation offered by OpenCV, to estimate the affine transformation between the two (i.e. between **odometry** and **ref**). Transformation estimation is iteratively repeated in a grid search manner with tunable hyper-parameters [28]. The transformation that provides the best performance on the validation data is selected to be applied to the subsequent odometry output.

During this first offline implementation of our system, we have therefore implemented and fine-tuned the position estimation procedure. This procedure has produced interesting results and for this reason we have decided to apply it to real flights.

B. Online Implementation

The second phase, called online implementation, was about the real-time version of the offline procedure. It was decided to implement it through the remote execution of the odometry process. Such an approach requires that a third device, in addition to the drone and the remote controller, is connected to the others and takes care of calculating the odometry. The main advantage of this solution is related to the fact that the UAV does not need any change (or a new homologation procedure), and there is complete freedom regarding the choice of the external hardware. Therefore it is possible to easily tackle the problems related to the computational burden. The limitations relate instead to the reduced range of motion, due to the maximum distance between the external device and remote controller, and the latency and stability of the connection between them. If frames are lost, in the communication between the remote controller and the remote processing device, this could cause a great degradation of the accuracy of the visual odometry. For this reason, in the next future, it could be possible to proceed with an integration of the external control board into the UAV, with the first one connected to the latter as an extra payload. With this update, it will be possible to run the same software developed for this online implementation avoiding the problems of reduced range of motion and connection stability introduced by remote processing. Of course, this will impose an extra homologation procedure for the drone before it can be used or the adoption of a suitably modified payload (for the drones that can handle it). Currently, partial protection regarding the connection stability aspect can be achieved by maintaining a reduced distance between the remote controller and the remote processing device, minimizing the possibility of signal disturbance.

It was decided to perform the odometry process on a Linux device, after sending the data to it from an Android application specifically developed. This application was developed within our research group, to receive the flight data from the UAV and share them with the external device. Regarding the selected equipment it was chosen to use a UP board with x86_64 architecture, equipped with an Ubuntu 16.04 operating system. The UP board is a credit card size board with high performance and low power consumption features, it was also provided with a Wi-Fi module for connecting it to the wireless network. Then, the procedure explained in the previous section was re-implemented to estimate the position of the UAV through the use of the odometry software. This procedure was adapted to receive and process the video frame data in real-time. Such data are provided by the Android application which in turn receives it communicating with the UAV. In detail, the procedure consists in collecting and aligning a series of pairs of points, taken from the GPS and the odometry function, and then estimating the affine transformation that leads from one reference system to the other. Lastly, the obtained transformation is applied to each new point coming from the odometry function, replacing the value of the z-axis with that retrieved from the barometer, thus obtaining the estimated position of the UAV.

VI. Results and Discussion

To evaluate the efficiency and the accuracy of the proposed system, we did a certain number of flights in different contexts. For each of them, we gathered the positional data given by our visual odometry estimation and the corresponding GPS data. Then the maximum and mean errors between the GPS position and the visually estimated position during the whole flight were calculated as a measure of the quality of the visual estimations. The minimum error about the UAV position during the flight was not reported, since in some moments, e.g. immediately after the transformation estimation procedure, it is equal to zero. Since the GPS position and the odometry estimated position were reported in Cartesian coordinates, the error was calculated as the Euclidean distance between two points. This metric is sufficient to give an estimate of the error since the two positions refer to the trajectory covered by the same object. For this reason, there is no need for metrics that consider both spatial and temporal information in the trajectory, therefore the metrics needed to evaluate the trajectory distance measures can be relaxed [29]. To analyze the distribution of data, the first and third quartiles were calculated along with the standard deviation, to have a measure respectively of the interquartile range (IQR) and the variability of the data around the mean error.

A. Offline Software Result Analysis

To test the offline procedure some flights were performed: the list of flights with their properties is reported in Table I, while the results are shown in Table II. The mean and maximum errors, obtained for each flight after the alignment

procedure, are reported using both transformation methods, namely **2D + barom** and **3D + barom**.

TABLE I
OFFLINE TEST FLIGHTS

Flight	Duration [s]	Max distance [m]	Target altitude [m]
A	325	281	81
B	356	167	61
C	388	99	31
D	103	174	109
E	269	120	81
F	192	173	83
G	210	58	30

TABLE II
COMPARISON OF THE RESULTS OBTAINED FOR 8 OFFLINE TEST FLIGHTS

Flight	2D + barom		3D + barom	
	AVG err. [m]	MAX err. [m]	AVG err. [m]	MAX err. [m]
A	3.29	9.15	3.48	10.28
B	2.04	7.03	1.86	7.01
C	1.71	4.70	1.73	4.57
D	1.51	2.59	1.27	2.52
E	1.76	3.65	1.18	4.67
F	3.62	7.62	6.70	18.33
G	0.87	1.56	1.03	5.26
H	0.56	1.19	1.13	2.67

In general, the **2D + barom** and **3D + barom** modes produce similar estimations. However, when evaluating the altitude error, the **3D + barom** performs slightly better (see Figs. 1 and 2) even if its error drastically increases when the UAV altitude decreases, particularly when it is about a few tens of meters. On the other hand, the **2D + barom** mode works sufficiently well also during the descent phase.

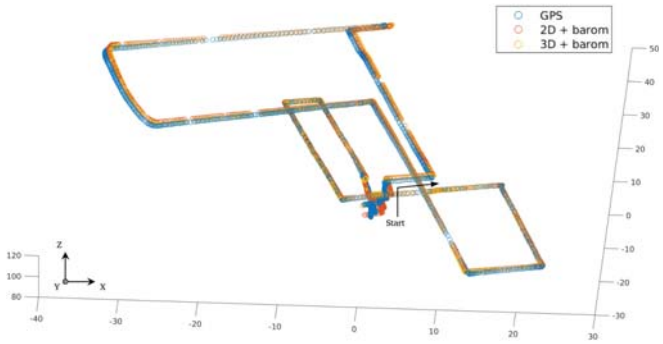

Fig. 1 Flight G: 2D vs 3D. Comparison between 2D+barom and 3d+barom modes

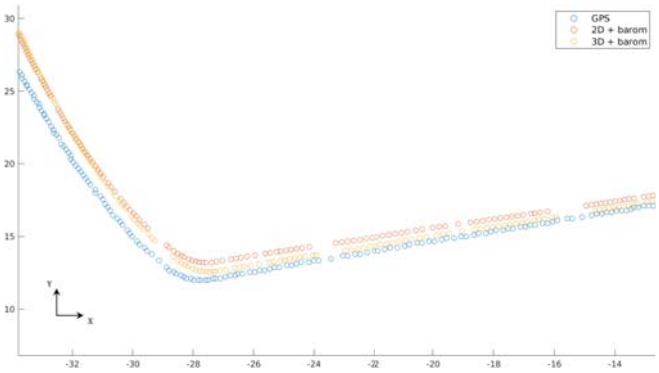

Fig. 2 Flight G: zoom. Comparison between 2D+barom and 3d+barom modes

In the following, the histograms of the errors referring to the data collected during some of the test flights are reported. In Figs. 3 and 4, the left chart refers to the whole flight and the right chart refers to the set of points that meet the condition |z - target_height| $<$ 2 meters. In this way, after setting the $target\_height$ of the flight, only those points around it are considered, hence removing the take-off and landing phases. As can be seen, in both cases the maximum error occurs only in a small number of positions, while most of the time the error is low and comparable with the typical uncertainty of the GPS receiver (usually around five meters). When only points with an altitude near to the target altitude of the flight are considered, **3D + barom** mode performs almost equally or rather slightly better.

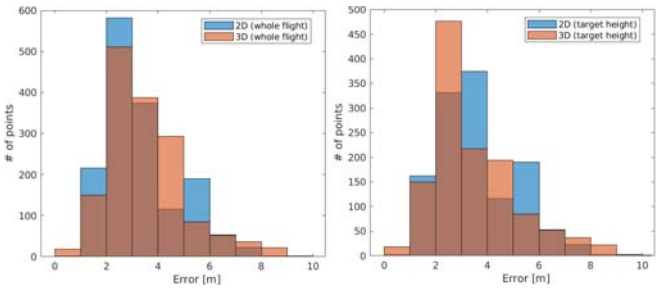

Fig. 3 Histogram of error, Flight A. Comparison between 2D+barom and 3d+barom modes, considering the whole flight or just the target height

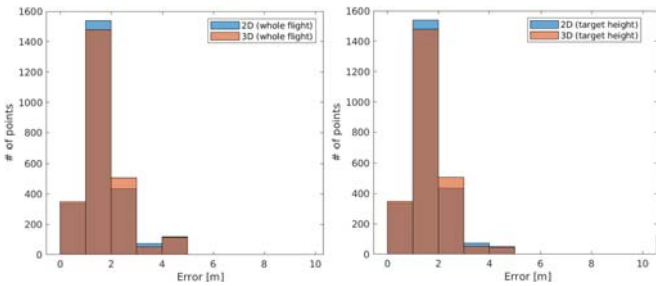

Fig. 4 Histogram of error, Flight C. Comparison between 2D + barom and 3d + barom modes considering the whole flight or just the target height

In conclusion, the preferred mode is the **3D + barom** one. In fact, the **2D + barom** mode would have an advantage only during the take-off and landing phases but, since the UAV already uses other positioning systems during such phases, it is not worth using it. Another most flexible approach, currently not employed because it would require more computations, could be the estimation of a transformation for both the modes, to switch among them in real-time according to the UAV's height evolution.

To make a comparative analysis, the value for the first and third quartiles are reported in Table III, along with the IQR and the standard deviation measures for each flight of this first offline case, related to the error made with the **3D + barom** mode since it is the selected one.

TABLE III
ERROR DISPERSION ANALYSIS FOR 8 OFFLINE TEST FLIGHTS

Flight	1st quartile [m]	3rd quartile [m]	IQR [m]	Std. deviation [m]
A	2.56	3.88	1.32	1.49
B	0.89	2.02	1.13	1.28
C	1.07	2.41	1.34	1.36
D	1.10	1.54	0.44	0.46
E	0.69	1.29	0.60	0.87
F	3.90	6.99	3.09	4.46
G	0.51	1.30	0.79	0.84
H	0.94	1.46	0.52	0.55

B. Online Software Result Analysis

After the development of the online implementation, another series of flights were performed to test its effectiveness. The setup needed for each of these test flights is composed of the UAV, the Android application, and the UP board. The latter is in charge of running the main function of the software. The application, instead, is the bridge that connects the remote controller of the UAV to the UP board via Wi-Fi, to share data and video information between them. When the UAV takes off and reaches the minimum height required, the alignment procedure starts, collecting the GPS and the visual odometry positions to find the best affine transformation between them. Once the optimal transformation was identified, the remainder of the flight was computed calculating the estimated position without using the GPS signal anymore. The results are shown in Tables IV and V.

TABLE IV
COMPARISON OF THE RESULTS OBTAINED FOR 8 ONLINE TEST FLIGHTS

Flight	Duration [s]	Max distance [m]	Target height [m]	AVG err. [m]	MAX err. [m]
I	125	67	49	5.81	12.82
J	108	82	68	5.86	17.36
K	119	91	79	8.18	19.72
L	170	142	69	6.60	19.70
M	204	186	68	9.27	19.09
N	165	196	56	9.15	22.63
O	155	162	61	5.34	12.07
P	195	158	73	5.78	15.91

TABLE V
ERROR DISPERSION ANALYSIS FOR 8 ONLINE TEST FLIGHTS

Flight	1st quartile [m]	3rd quartile [m]	IQR [m]	Std. deviation [m]
I	4.57	7.28	2.71	2.72
J	3.48	7.10	3.62	3.49
K	7.19	10.23	3.04	3.74
L	3.87	9.08	5.21	4.19
M	7.46	11.09	3.63	3.85
N	7.01	11.29	4.28	4.12
O	3.91	6.05	2.24	2.05
P	3.28	7.77	4.49	3.35

As can be seen, with similar flight properties between online and offline test flights, also the results are roughly comparable. The errors, both mean and maximum, in the online case are just slightly worse than those of the offline procedure. Furthermore, the statistical indicators reported in Table V, show a reduced variability of the data around the average error, both considering all the samples (as happens for the standard deviation) and even better by removing the outliers (as in the interquartile range case).

C. Result Evaluation

The differences between the results presented in Tables II and III and those of Tables IV and V are caused by the fact that the Wi-Fi connection between the board and the application may sometimes be lost and therefore the related data are also missing. This problem was expected since there are many devices connected via Wi-Fi or radio signals. However, the obtained results are still good and it is possible to perform flights with an acceptable level of accuracy. The effectiveness of the obtained results is also demonstrated by Figs. 5 and 6 which show the trajectory plots, containing the GPS position and the estimated one, of two flights.

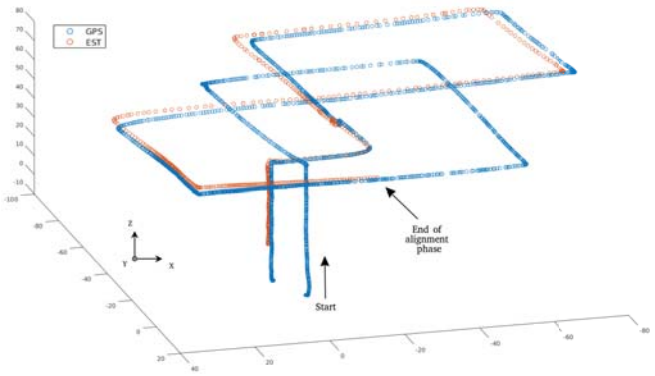

Fig. 5 Flight L. Comparison between GPS trajectory and estimated one

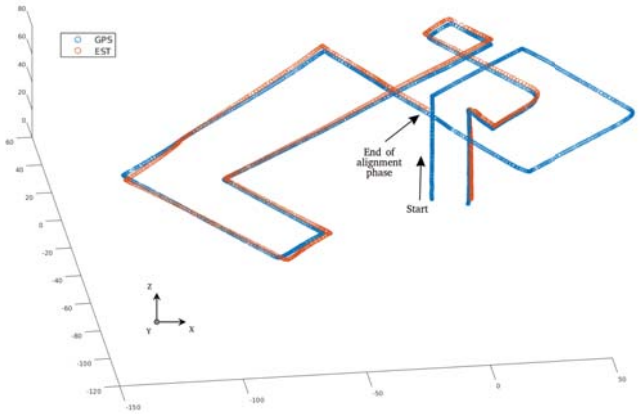

Fig. 6 Flight P. Comparison between GPS trajectory and estimated one

The occurrence of the no-signal problem just discussed can be observed in Fig. 5, where especially the orange points related to the estimated position (but also the blue ones about the GPS position) are very sparse, evidence that signal reception was not ideal. To conclude, in the following, two box-and-whisker diagrams are reported to graphically show the variation of the statistical population about the error committed during some flights.

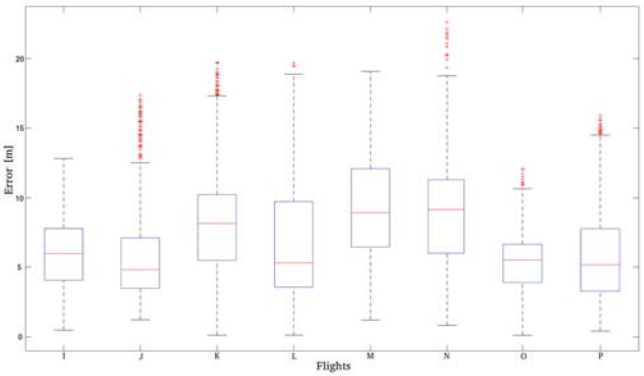

Fig. 7 Box and whisker diagrams about error for all the performed flights

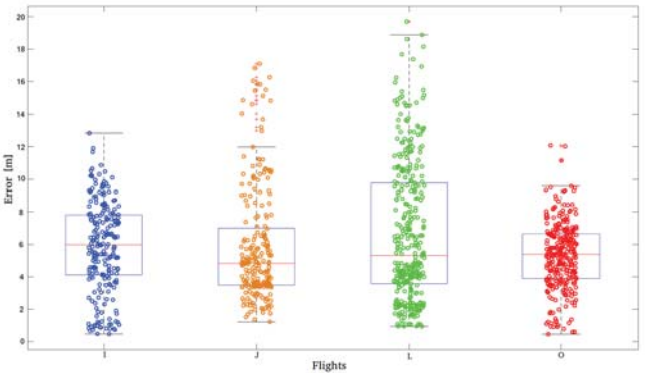

Fig. 8 Box and whisker diagrams overlapped with scatter plot about error

In particular, in Fig. 8, the box-and-whisker diagram of some flights has been overlapped with a scatter plot about the error committed in the corresponding flight. It shows the data related to flights I, J, L, and O, to have a visualization of the effective distribution of the errors. To facilitate the readability of the graph, a uniform subsampling of the error vectors was carried out to reduce the amount of data reported to one-fifth of the total. As can be seen in the case of flights J, N, K and P there is a non-negligible number of outliers but this is due to the loss of communication that sometimes occurs. Moreover, during the take-off and landing phases, the error increases significantly, since we are using an affine transformation estimated around the target altitude of the flight. This increases the dispersion of the collected dataset, as reported in Fig. 7 where for some flights the upper whisker (related to the right tail of the distribution) may appear surprisingly long. Such extension only indicates a wider error distribution but does not affect the position accuracy of the flight, since other positioning systems are in use during these phases. In fact, in almost all the cases the whisker on the lower end of the box is shorter than the other, and this indicates that the distributions are generally positively skewed. This means that most values are clustered around the left tail of the distribution, that is the one closer to the null error value. A further demonstration is shown in Fig. 8, where it can be seen that the greatest concentration of values is focused around the middle value of the dataset and below it.

Lastly, in Fig. 9 a comparison between the GPS trajectory and the estimated one is shown. The GPS uncertainty is reported as a circumference centered in the coordinates of the GPS point and with a radius of 2.5 meters (since the GPS uncertainty is usually around 5 meters). As can be seen most of the time the error is low and the estimated position falls within the circumference, therefore both points belong to the same equivalence class. In conclusion, the estimated position is comparable with the GPS receiver's uncertainty.

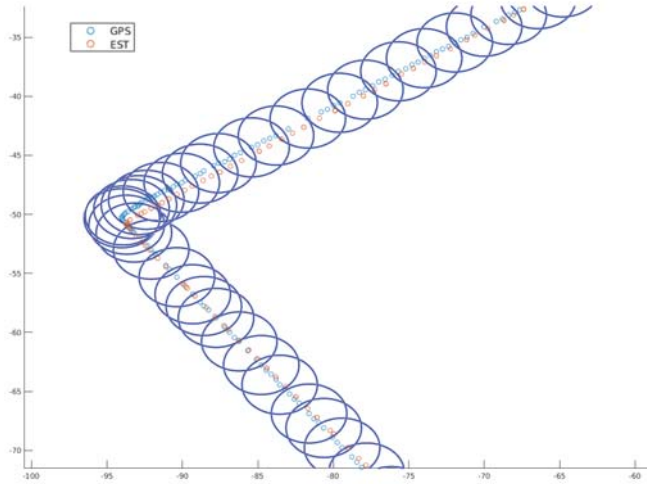

Fig. 9 GPS uncertainty in comparison between GPS trajectory and estimated one

VII. Conclusion and Future Works

This work aimed to develop a flight system based on a monocular visual odometry technique, to overcome some limitations related to the use of GPS signals. In fact, since nowadays the issues related to GPS reliability are more and more numerous, this topic is becoming increasingly relevant every day. All the experiments have been done using two commercial drones from the DJI brand and relying on a visual odometry algorithm, adapted from that proposed by Forster et al. [20], called Semi-Direct Visual Odometry. During the development phase, this algorithm was integrated initially within an offline procedure, in which the information collected during the flight is batch-processed at a later time, using the GPS only during an initial alignment procedure to then calculate the estimated position without using it anymore. Then, in a subsequent phase, this procedure was transformed into a real-time version of it. Moreover, a custom application was developed to make communication between the remote controller of the UAV and the board which performs the remote processing possible.

From the performed test flights and the data analyzed, it emerged that it was possible to create a reliable flight system that retrieves the position of the UAV using only the camera images, taken during the flight, without relying anymore on the GPS information. Moreover, external processing is a viable solution that enables us to apply our work to any commercial UAV without requiring additional equipment or heavy computational burdens. This was the answer we were looking for to be able to improve the security of the UAVs during their flights in a simple but effective way. Nevertheless, some aspects can still be improved, among which the main one is the accuracy level. This issue is closely linked to another important problem which is the limited range of coverage due to the type of connection used. Therefore, even if the reached results have shown that our solution is feasible, some aspects need to be fixed before actually using it in the field. To be able to overcome such limitations another procedure was studied and, although the idea is not yet implemented, based on the analysis carried out, it seems to be able to solve or at least limit the negative effects highlighted so far. Since the main source of issues is the possible loss of connection, which reduces the accuracy of the estimated position and also limits the UAV's range of action, it is necessary to reduce the distance between the devices. To manage these problems, it could be possible to proceed in the next future with the use of an external board connected as an additional payload to the UAV. However, there are some problems related to the weight and energy consumption that the UAV should bear. These problems lead to the necessity to look for a board that is, at the same time, able to deliver the computational power for running the software in real-time and able to keep energy consumption as low as possible in order not to significantly burden the UAV battery. A possible choice could be the NVIDIA Jetson Nano board: it is an embedded system-on-module that can run the Android O.S. and which can be connected to the UAV either via a Wi-Fi module or using a wired connection. Moreover, the specifications of this board satisfy the computational capability required by the SVO algorithm. In fact, the typical RAM usage by the odometry algorithm is about ≈700 MB, and another ≈2 GB are required from the idle Android O.S., so RAM requirements are widely met by the proposed board. About the CPU requirements, in the paper by Forster et al. [20] an extensive benchmark is reported. They used an Intel Core i7-2760QM which, compared with the ARM A57 quad-core on the NVIDIA board results to have some extra high-level features and a slightly higher clock frequency. From a) publicly available benchmarks on these exact machines and which involve similar operations [30], and b) validated through performance comparisons on available machines in our lab, we deduce that the performance degradation can be conservatively assumed not to exceed 75%. This translates into a performance degradation factor of ≈ 0.29x in the SVO algorithm, resulting in $0.29 * 400 \approx [100 - 120]$ fps, which still abundantly satisfies the real-time processing requirements of the considered reference scenario. To conclude, therefore, after only a first preliminary analysis and prototyping, this new procedure promises to be an interesting development that is worth to be further explored in the future. Besides this new possible implementation, other improvements can be carried out to improve the performance and the usability of the flight system software. For instance, the duration of the initial alignment phase could be reduced using information about previous flights. To this aim, a dataset about the previously overflown areas might be stored, performing the geotagging of some significant key points detected. In this way, the UAV would have in-memory information about the areas already seen, which could accelerate the initial phase of each flight. Moreover, these geo-tagged points can also be used to perform a realignment procedure during the flight: when a known point of interest (previously geotagged) appears in the framed scene it can be taken as a reference to refine the estimated position and to increase the overall accuracy.

References

[1] R. Altawy, A. M. Youssef, *Security, Privacy, and Safety Aspects of Civilian Drones: A Survey*, https://doi.org/10.1145/3001836, 2016

[2] J. R. Reinhardt, J. E. James, E. M. Flannagan, *Future employment of UAVS: issues of jointness*, https://apps.dtic.mil/sti/pdfs/ADA525691.pdf,1999

[3] A. Puri, K. P. Valavanis, M. Kontitsis, *Statistical profile generation for traffic monitoring using real-time UAV based video data*, https://ieeexplore.ieee.org/abstract/document/4433658?section=abstract, 2007

[4] M. Kontitsis; K .P. Valavanis; N. Tsourveloudis, *A UAV vision system for airborne surveillance*, https://ieeexplore.ieee.org/document/1307132,2004

[5] I. K. Nikolos, N. C. Tsourveloudis, K. P. Valavanis,*Evolutionary Algorithm Based Path Planning for Multiple UAV Cooperation*, https://doi.org/10.1007/978-1-4020-6114-1_10, 2007

[6] K. Dalamagkidis, K. P. Valavanis, L. A. Piegl, *Current Status and Future Perspectives for Unmanned Aircraft System Operations in the US*, https://doi.org/10.1007/s10846-008-9213-x, 2008

[7] J. A. Volpe, *Vulnerability Assessment Of The Transportation Infrastructure Relying On The Global Positioning System, Final Report.*, https://www.navcen.uscg.gov/pdf/vulnerability_assess_2001.pdf, 2001

[8] South China Morning Post, *$1 million in damage caused by GPS jamming that caused 46 drones to plummet during Hong Kong show*, https://www.scmp.com/news/hong-kong/law-and-crime/article/2170669/hk13-million-damage-caused-gps-jamming-caused-46-drones, 2018

[9] CSMonitor, *Exclusive: Iran hijacked US drone, says Iranian engineer*, https://www.csmonitor.com/World/Middle-East/2011/1215/Exclusive-Iran-hijacked-US-drone-says-Iranian-engineer, 2011

[10] NRKbeta, *GPS freaking out? Maybe you're too close to Putin*, https://nrkbeta.no/2017/09/18/gps-freaking-out-maybe-youre-too-close-to-putin/, 2017
[11] J. S. Warner, R. G. Johnston, *GPS Spoofing Countermeasures*, http://the-eye.unblocksite.ch/public/Books/Electronic%20Archive/GPS-Spoofing-Countermeasures.pdf, 2003
[12] N. Gageik, M. Strohmeier, S. Montenegro, *An Autonomous UAV with an Optical Flow Sensor for Positioning and Navigation*, https://doi.org/10.5772/56813, 2013
[13] H. Romero, S. Salazar, O. Santos, R. Lozano, *Visual odometry for autonomous outdoor flight of a quadrotor UAV*, https://doi.org/10.1109/ICUAS.2013.6564748, 2013
[14] C. Fu, A. Carrio, P. Campoy,*Efficient visual odometry and mapping for Unmanned Aerial Vehicle using ARM-based stereo vision pre-processing system*, https://doi.org/10.1109/ICUAS.2015.7152384, 2015
[15] M. Blösch, S. Weiss, D. Scaramuzza, R. Siegwart, *Vision based MAV navigation in unknown and unstructured environments*, https://doi.org/10.1109/ROBOT.2010.5509920, 2010
[16] D. Scaramuzza, F. Fraundorfer, *Visual Odometry [Tutorial]*, https://doi.org/10.1109/MRA.2011.943233, 2011
[17] M. Albéri, M. Baldoncini, C. Bottardi, et al., *Accuracy of Flight Altitude Measured with Low-Cost GNSS, Radar and Barometer Sensors: Implications for Airborne Radiometric Surveys*, https://www.mdpi.com/1424-8220/17/8/1889, 2017
[18] H. Strasdat, J. M. M. Montiel, A. J. Davison, *Scale drift-aware large scale monocular SLAM*, https://doi.org/10.15607/RSS.2010.VI.010, 2010
[19] P. V. Gakne, K. O'Keefe, *Tackling The Scale Factor Issue In A Monocular Visual Odometry Using A 3D City Model*, https://hal-enac.archives-ouvertes.fr/hal-01942257, 2018
[20] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, D. Scaramuzza, *SVO: Semi-Direct Visual Odometry for Monocular and Multi-Camera Systems*, http://rpg.ifi.uzh.ch/docs/TRO17_Forster-SVO.pdf, 2017
[21] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, D. Scaramuzza, *rpg_svo*, https://github.com/uzh-rpg/rpg_svo, 2017
[22] P. F. Sturm, *Pinhole Camera Model*, Computer Vision, A Reference Guide, 2014
[23] Open Robotics, *How to Calibrate a Monocular Camera*, http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration, 2019
[24] Z. Zhang, *A flexible new technique for camera calibration*, IEEE Transactions on Pattern Analysis and Machine Intelligence, volume 22, number 11, pages 1330-1334, 2000
[25] G. Nutzi, S. Weiss, D. Scaramuzza, R. Siegwart, *Fusion of IMU and Vision for Absolute Scale Estimation in Monocular SLAM*, https://www.ifi.uzh.ch/dam/jcr:e885ca94-b971-4bcb-be00-c04b67bebfaa/UAV10_nuetzi.pdf, 2011
[26] S. P. Drake, *Converting GPS coordinates [phi, lambda, h] to navigation coordinates (ENU)*, https://apps.dtic.mil/dtic/tr/fulltext/u2/a404846.pdf, 2002
[27] M. A. Fischler, R. C. Bolles, *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*, https://doi.org/10.1145/358669.358692, 1981
[28] I. Syarif, A. Prugel-Bennett, G. Wills, *SVM Parameter Optimization Using Grid Search and Genetic Algorithm to Improve Classification Performance*, https://core.ac.uk/download/pdf/295538475.pdf, 2016
[29] N. Magdy, M. A. Sakr, T. Mostafa, K. El-Bahnasy, *Review on trajectory similarity measures*, https://doi.org/10.1109/IntelCIS.2015.7397286, 2015
[30] Passmark Software, *Intel Core i7-2760QM @ 2.40GHz vs ARM Cortex-A57 4 Core 1479 MHz*, https://www.cpubenchmark.net/compare/Intel-i7-2760QM-vs-ARM-Cortex-A57-4-Core-1479-MHz/884vs3914, 2021