CrossMark

Visual Based Landing for an Unmanned Quadrotor

**Francesco Cocchioni · Emanuele Frontoni · Gianluca Ippoliti ·
Sauro Longhi · Adriano Mancini · Primo Zingaretti**

Abstract One of the main challenges for Unmanned
Aerial Systems (UAVs) is to extend the endurance of
small vehicles such as multi-rotors. Actually, Li-po
batteries that guarantee a flight of about 20 min power
this type of vehicles. The endurance can be extended
by enabling vehicles to look for recharging station(s).
In this paper, we propose a vision system able to detect
and track a given pattern hosted on the target-landing
platform. The pattern is also useful to estimate the
UAV position while approaching the target or during
the hovering close to the target. The paper focuses
on an optimized adaptive thresholding technique that
manages critical situations as changes in the scene's
illumination / shadows. The developed system runs at
90 Hz for processing a 752×480 grayscale image.

Preliminary results on an NVidia Tegra Jetson K1 plat-
form are also presented to distribute the computation
between CPU and GPU.

1 Introduction

During the last years the Unmanned Aerial Vehicles
(UAVs) and in particular the multi-rotors are capturing
the interest of the research community and of many
military/civilian companies [1]. Mainly owing to their
versatility and reprogrammability, many types of mis-
sions could be performed, such as search and rescue,
building inspection, surveillance, ...

One of the main still open problems is the mission
endurance. Li-po batteries that guarantee 10–25 min
of flight power most of the multi-rotors platforms. In
this context it is necessary to look for a simple solu-
tion to recharge the battery. Many solutions have been
proposed with a dichotomy: active vs passive systems.

Active systems ensure a short stop time, but require
complex electro-mechanical mechanisms to replace
the discharged battery with a new one [19, 20]. Pas-
sive systems are simpler, but impose a stop time that
varies from 10 min to 1 h.

Another important problem is the capability to
flight from/to a docking station in a safe way ensuring
a short time to perform a landing. Many approaches

F. Cocchioni · E. Frontoni · G. Ippoliti · S. Longhi ·
A. Mancini (✉) · P. Zingaretti
Dipartimento di Ingegneria dell'Informazione,
Università Politecnica delle Marche, 60131 Ancona, Italy
e-mail: mancini@dii.univpm.it

F. Cocchioni
e-mail: francesco.cocchioni@gmail.com

E. Frontoni
e-mail: emanuele.frontoni@univpm.it

G. Ippoliti
e-mail: gianluca.ippoliti@univpm.it

S. Longhi
e-mail: sauro.longhi@univpm.it

P. Zingaretti
e-mail: zinga@dii.univpm.it

have been tested, but also here a dichotomy is present: marker-less vs marker-based. Considering the simpler case of marker-based approaches, we can find two common approaches: natural landmarks [10] vs artificial landmarks. In our case, we decided to select an artificial marker to ensure a precise and accurate alignment of the UAV respect to the landing platform.

Many markers have been proposed from QR-code[7] to structured pattern. We designed the marker starting from the design of the mechanical platform that has to host the UAV during the recharging. The concept is to use four inverted cones to passively bring the UAV to the center of each cone where a copper plate is installed.

The paper describes the vision system developed for target detection and tracking. We adopted an optimized solution to save computation time ensuring a real time image processing supported by the ROS environment. The vision system works in indoor / outdoor environments even in absence of GPS or similar localization systems [9].

The paper is structured as follows. In Section 2 the adopted UAV system configuration is introduced. In Section 3 the design and implementation of the landing platform for the UAV is presented. Section 4 focuses on the vision system for landing, based on a fast, accurate and precise detection of an artificial target. In Section 5 an example of mixed scenario, indoor / outdoor is presented. Section 6 introduces a first approach to distribute the computational load by using a powerful NVidia GPU. Section 7 outlines conclusions and future works.

2 System Configuration

The following system configuration has been developed in the context of the European project R3-COP (Resilient Reasoning Robotic Co-operating Systems) [5], involving a collaboration with an Unmanned Ground Vehicle during the mission [12, 13].

In our study, we decided to use the Asctec Pelican manufactured by the Ascending Technologies. The Pelican is able to host 650 g of payload and is equipped with a small set of sensors. The common usage is outdoor and it is ready for GPS based way-point navigation (GPS, IMU and barometric sensors). In our work, we extended the platform to perform also indoor missions by using a vision system to correctly

land and navigate. We also designed and manufactured by a 3D printer an extension of the landing feet as described in Section 3.

The Asctec Autopilot provides two micro controllers based on ARM7. The Low Level Processor (LLP) fuses the measurement data to control the roll/pitch and yaw. The source code is not available while the High Level Processor (HLP) can be freely flashed for custom code.

Our drone is equipped with the Asctec Mastermind based on the dual core Intel Core2Duo SL9400 running at 1.86 GHz. The adopted camera is the Blue-Fox MLC200wC manufactured by the Matrix Vision, capable of a color / grayscale acquisition of a 752x480 frame at 90 Hz. We selected a 2.8 mm lens to cover the largest region of interest when the drone is close to the ground. We selected the ultrasonic sensor Maxbotix XL-MaxSonar-EZ MB1320 to measure the height over ground. The Asctec Pelican is only equipped with a pressure sensor that can not be used for accurate and precise height measurement.

In Fig. 1 our quadrotor with all the installed sensors is shown.

According to the well tested workflow in the mobile robotics community we decided to develop our code in the Robot Operating System (ROS) environment [6]. The OpenCV libraries [4] have been used to process the image for target detection (during the landing phase, see Section 4) and navigation (by visual odometry) [12].

The Asctec MAV Framework [2] was used to bridge the Asctec Autopilot to ROS even if other

Fig. 1 Our customization of the Asctec Pelican

Fig. 2 *Left* 3D model of our custom landing foot. *Right* real printed landing foot derived from our 3D model that hosts the copper plate contact to enable the battery recharge
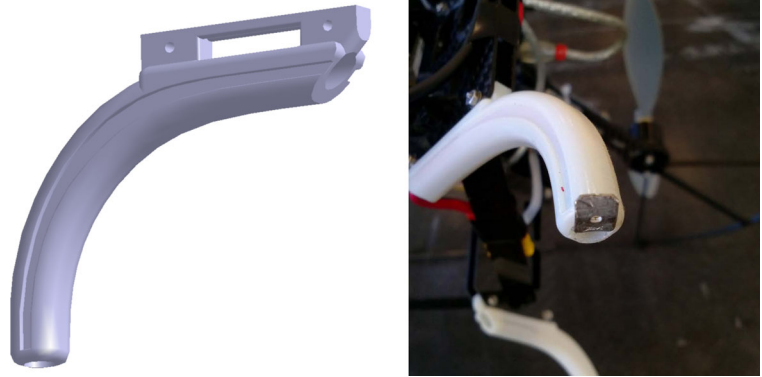
features are available. In particular, it is easy to integrate an external localization system with the Ultra Wide Band (UWB) technology, which could be integrated to compensate the odometric drift as discussed in [7].

3 Landing Platform

The reduced endurance of UAVs does not allow a real continuous autonomous fly on a wide area, limiting the set of the achievable missions. To overcome this limitation a specific landing and recharging platform has been designed.

We decided to recharge and not replacing (with actuators) a battery [19, 20], to reduce the system complexity and, consequently, development time and costs.

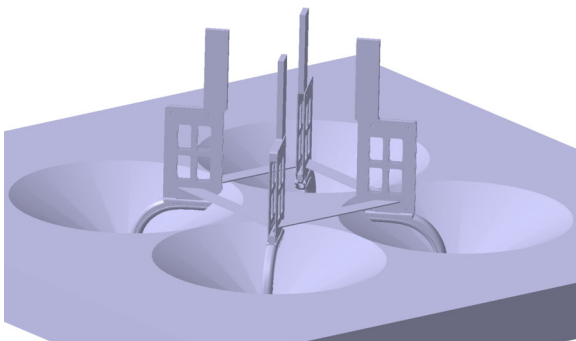The landing platform has been initially modelled in a 3D CAD environment and printed in 3D at 1:1 scale.

Fig. 3 3D model of Asctec Pelican base with landing foot installed, once landed inside the landing platform

The developed landing platform (shown in Fig. 7) includes four cones exploiting the gravity as a passive centering system. The drone has to land inside the four cones, but is necessary to take into account the size of the landing platform:

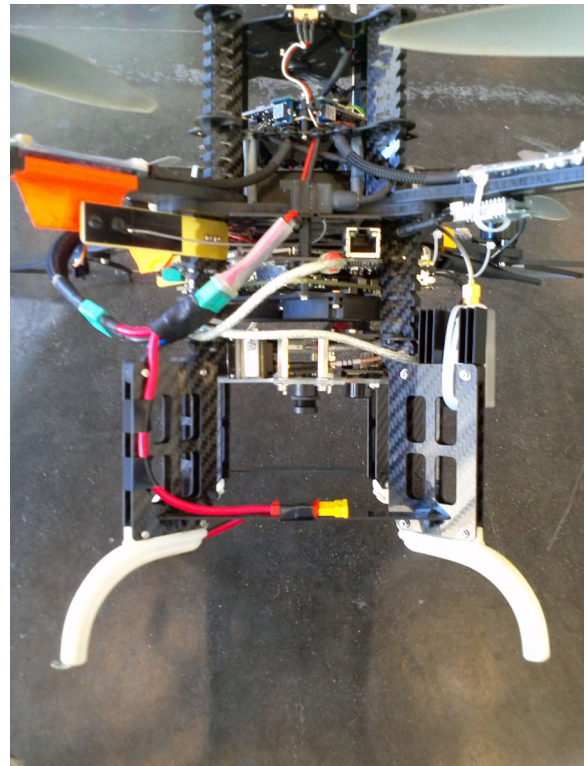– the radius of a cone determines the maximum allowable error;

Fig. 4 UAV used for the final demonstrator, with all the hardware changes
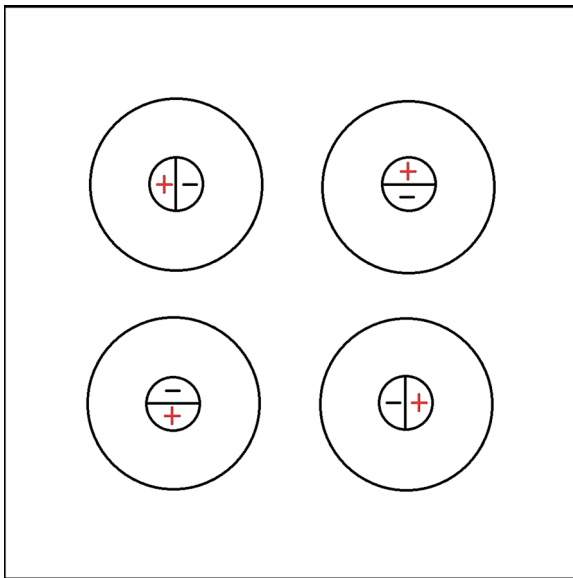
Fig. 5 Displacement and geometry of copper plates to manage the balancing

Fig. 7 Landing platform and B/W target

– the slope is directly related to the height of the platform;
– a higher slope increases the probability to passively bring the UAV to the copper plate
– the height of a cone sets a constraint on the landing foot of the UAV

By performing a series of simulations, we calculated the maximum allowable translational and heading error. An error of 0 degrees for the heading sets
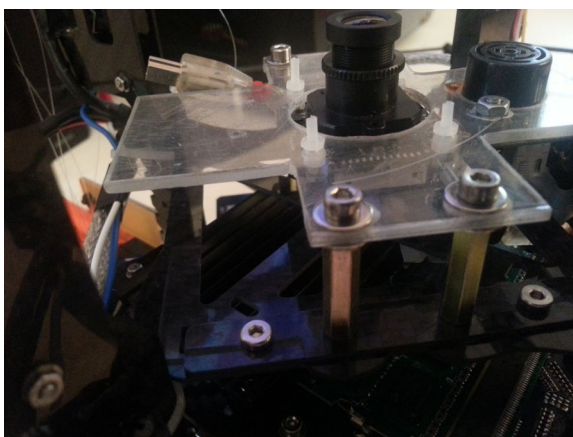

Fig. 6 The camera and ultrasound sensors installed on the *bottom* part of our UAV

the maximum translational error, which is 10 cm with a radius of 10 cm. If the heading error is equal to 40 degree, the UAV has to land exactly with zero translation error.

From these considerations, it is clear the importance of a precise and accurate landing in terms of position and orientation. This is the starting point for the vision landing system described in Section 4.

The original Asctec Pelican with foot extension does not fit with our centering system. For this reason we modelled a new landing foot to be attached on the existing carbon fiber frame, thus avoiding any dangerous change to the mechanical frame.

Figure 2 shows the designed and developed landing foot characterized by a precise height to avoid any mechanical interference with the landing platform. By using this new configuration, it is possible to extend

Table 1 Dimensions of landing target elements

Position	Ring	Triangle
Inner	R = 3 cm r = 2 cm	b = 2.73 cm
		$l_1 = l_2 = 2.03$ cm
Outer	R = 27 cm r = 25 cm	b = 14.00 cm
		$l_1 = l_2 = 10.63$ cm

Fig. 8 *Left* our algorithm. *Right*: Wellner' s algorithm

the footprint of our drone to match with the design of our platform.

The design has the following features:

– minimization of overall weight by removing all the unnecessary volumes;
– inner cavity to host the cable for battery recharging;
– damping of vibration caused by hard landing.

The analysis performed for the maximum landing error (along X and Y) that can be compensated with four cones of 10 cm radius demonstrated a particular affordable combination of the maximum translation and rotation landing error, that is, 5 cm along X and Y axis and 10 degrees with respect to North, South, East or West orientation. This is a good margin for the maximum landing error that should be achieved with the final navigation controller. A larger centering system could provide more margin, but at the same time it will require a larger landing platform and a larger landing foot. The original Asctec Pelican base with

foot extension, indeed, creates a square with side of approximately 15.5 cm, which can be easy extended to 20 cm with the non-invasive landing foot shown in Fig. 2.

The cones of 10 cm radius also allow small cone height, achieving a good sliding surface slope. With an height of 5 cm the internal surface of each cone has a slope of 27 degrees, which, in addition to a small landing foot contact surface (0.5 cm of radius), allows a perfect UAV sliding for each possible landing position inside the cones. For these reasons, the final landing foot additional height has been chosen to be the minimum one required (5 cm). The final landing foot with an overall weight of 7 g is shown in Fig. 2.

Figure 3 shows the 3D model of the final landing foot installed on the original Asctec Pelican base, together with the final landing platform of $60 \times 60 \times 7$ cm. The bottom of two upside down cones and connected to a Li-po battery charger hosts the copper plates to recharge the system. The drone is equipped

Fig. 9 *Top left* source image. *Top right* Otsu. *Bottom left* OpenCV adaptive threshold - blockSize=71 C=30, *Bottom right* Wellner - S=752/12 T=0.35
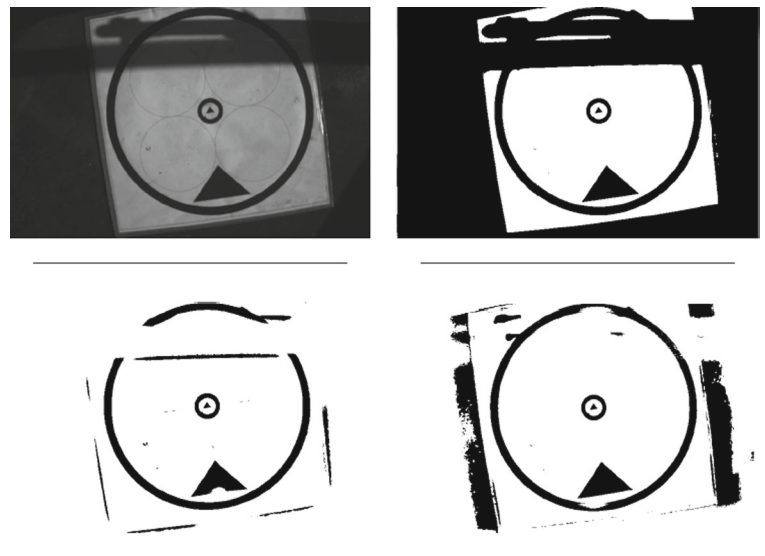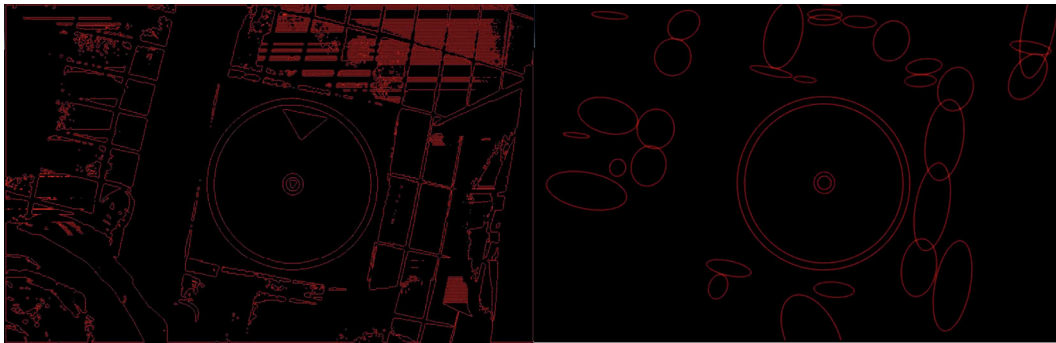
Fig. 10 *Left* extracted contours from the binary image. *Right* interpolation of ellipses on the extracted contours

with two contacts installed on the bottom of two landing feet connected to a Li-po battery. The developed landing foot shown in Fig. 2 has been manufactured using a 3D printer, and then integrated into our UAV shown in Fig. 1. The two slip rings connecting battery poles have been installed in the two front landing feet, and a 20Amps fuse protects the system from any short circuit. Figure 4 shows the right side of the drone used in the demonstrator with all the described components.

The cones in the real landing platform shown in Fig. 7 have been also produced using a 3D printer. We used a modified version of the Groupner Ultramat 16S charger to charge the Li-po battery with a maximum current of 20A (3C for a 6000mAh battery).

We also studied the possibility to balance the cells during the charging by changing the geometry and displacement of the copper plates. In Fig. 5 the displacement is shown. However this solution introduces a higher complexity and it is necessary to ensure a safe contact to establish a balancing avoiding high resistance contacts.

4 Landing Vision System

Hovering and Navigation are still open problems for UAV in both indoor and outdoor environments. Actually the PX4FLOW Smart Camera [15] is a novel interesting hardware platform enabling the estimation of position by visual odometry. This platform integrates a 752480 MT9V034 image sensor. A L3GD20 3D Gyro processed by a 168 MHz Cortex M4F CPU. This device mainly outputs x/y flow measurement,

velocity, gyros rate and height. For small size UAV with strongly limited payload this devices is a perfect solution for the optical stabilization and navigation in complex environments. In our case the developed vision system is shown in Fig. 6.

Our configuration is close to the PX4FLOW Smart Camera, but our solution, presented in [12, 13], exploits the on board IMU of the UAV. Our system is based on a powerful computing unit that allows fast computation and complex image processing algorithms, among which target detection. In our case, we need to detect the target to start the landing procedure. In [13] we discuss the landing over an Unmanned Ground Vehicle (UGV).

In particular, the developed thresholding algorithm requires the full processing of the entire frame. The presence of shadows over the target is a problem that usually sticks up standard thresholding algorithms.

Table 2 Average computation time required to perform the steps of our workflow

Operation	Average time	Standard deviation
Adaptive threshold	2.156 ms	0.551 ms
Contour detection	2.402 ms	0.296 ms
Interpolation of ellipses	0.758 ms	0.122 ms
Filtering of ellipses	0.024 ms	0.005 ms
Filtering correction	0.006 ms	0.002 ms
Target validation	0.002 ms	0.001 ms
Triangle identification	0.118 ms	0.026 ms
Total	5.467 ms	0.551 ms

The developed landing platform discussed in Section 3 requires that maximum allowable translation error is 5 cm and 10 degrees for heading. These values represent the target performance for the developed landing system, which is based on fast detection and tracking of a user-defined B/W target. The target must be detectable at low and high altitudes and for this reason we included four regular shapes: two concentric annular rings and two filled triangles (see Fig. 7). Our idea is to estimate the 6DOF of the UAV. Circles provide the estimation of X, Y, Z, roll and pitch values. The triangle provides the estimation of the heading.

Table 1 summarizes the dimensions of the adopted target.

The workflow of the landing vision system is composed by the following blocks:

– Image acquisition

Fig. 11 Our algorithm with different images characterized by strong changes in illumination / shadows at *low* altitude. (S=752/24, T=0.45, Sb=50, Tb=0.25, Lens 2.8 mm
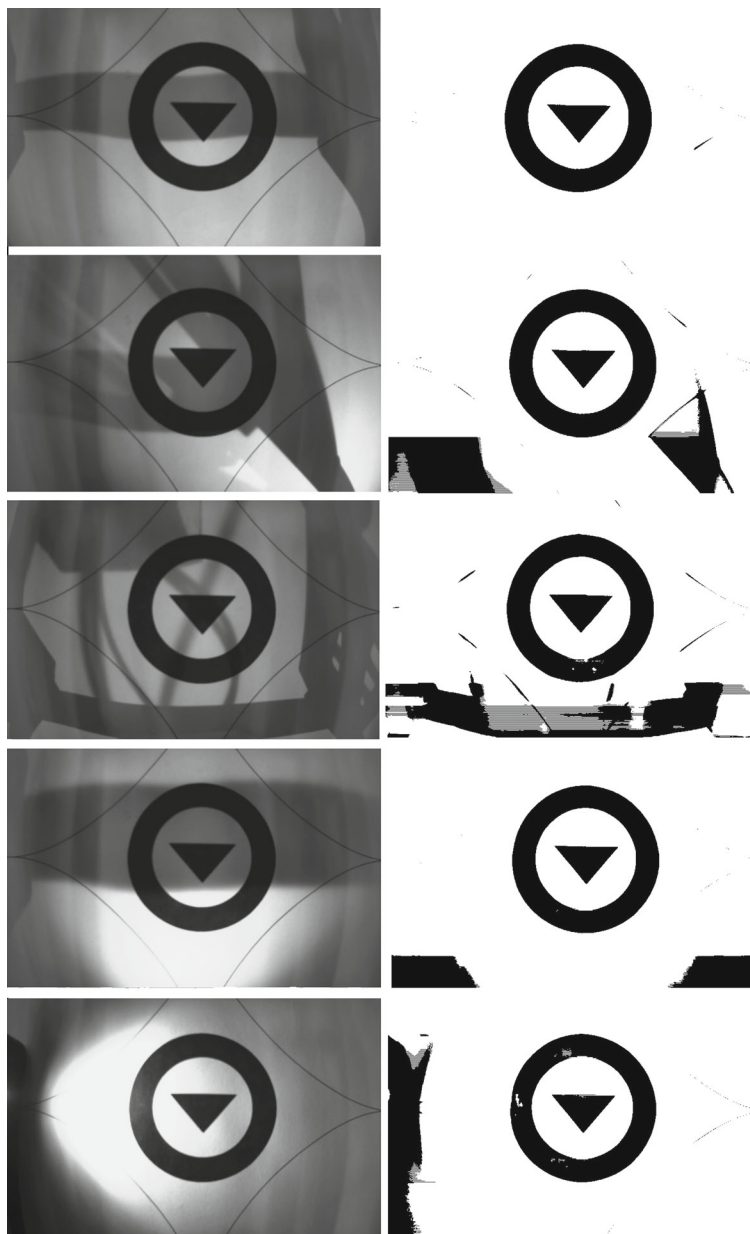
- Adaptive thresholding
- Contours detection and un-distortion
- Ellipse extraction and filtering
- Triangle detection
- Pose estimation

The OpenCV library has been used to support some blocks as discussed in the following.

The first step concerns image acquisition from the down looking camera (BlueFox MLC200wC), setup with the following parameters:

- mono8 gray-scale image (to avoid color to grayscale conversion)
- max exposure time of 1/90s
- automatic expose time

This setup allows to gather frames at the maximum allowable frame rate of 90 fps. Images are also converted in a ROS data type to be shared with other ROS nodes.

The threshold should be efficient and accurate for both outdoor and indoor environments. We designed an adaptive algorithm ensuring the robustness to shadow / illumination changes especially in outdoor scenarios. We based our work on a optimization of the Wellner algorithm [21].

To save time in this phase the undistortion is postponed and applied only to the contours extracted, significantly improving the overall performance.

The approach explores images row by row the calculating the moving average of the previous S pixels explored. If a pixel is at least T percent darker than

Fig. 12 Our algorithm with different images characterized by strong change in illumination / shadows at *high* altitude. (S=752/24, T=0.45, Sb=50, Tb=0.25, Lens 6 mm
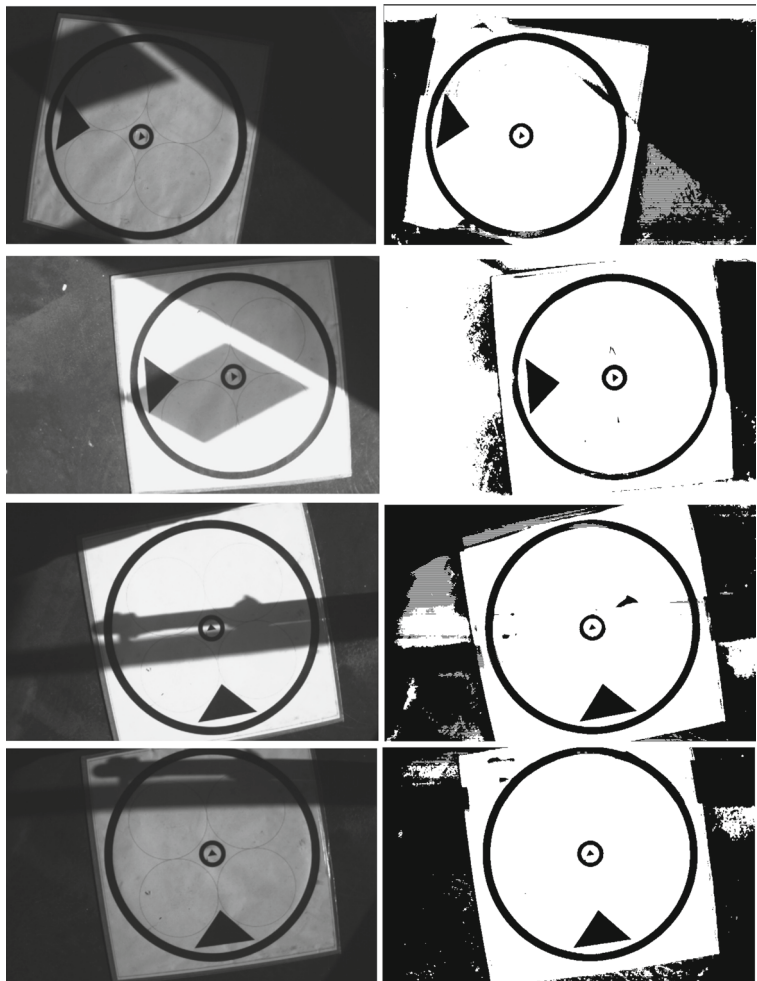
the moving average then it will be black in the binary image, otherwise white.

S and T are the only parameters to be set according to the operative scenarios.

Denoting with p_n the intensity of the n-th pixel, and with g_n the average intensity of previous S pixels, the value of the n-th pixel in the binary image O is:

$$O(n) = \begin{cases} 0 \ if \ p(n) < g(n) \cdot (1 - T) \\ \quad 1 \ otherwise \end{cases} \quad (1)$$

where 0 is a black pixel and 1 a white one.

Since the algorithm is influenced by direction of row exploration, according to the author it is necessary to alternate the image scan from left to right and from right to left. Moreover, the binarization is more accurate if the moving average of each pixel on the previous row is evaluated. $g(n)$ is replaced by $h(n)$:

$$h(n) = \frac{g(n) + g(n - w)}{2} \quad (2)$$

where w is the image width.

The moving average for the n-th pixel of a row is replaced by the average between the n-th pixel moving average and the moving average of the pixel directly above in the previous line. An iterative moving average formula is useful to reduce the computational time. The exact moving average calculation has been replaced in our algorithm with the approximation provided by the exponential moving average defined as:

$$g(n) = \alpha \cdot p(n) + (1 - \alpha) \cdot g(n - 1) \quad (3)$$

Fig. 13 *Top left* source grayscale image. *Top right* binarized image. *Middle left* extracted contours. *Middle right* filtered contours with interpolated ellipses. *Bottom left* Detected target. Bottom right: Detected triangle. *Bottom center* undistorted target. (S=752/24, T=0.45, Sb=50, Tb=0.25)
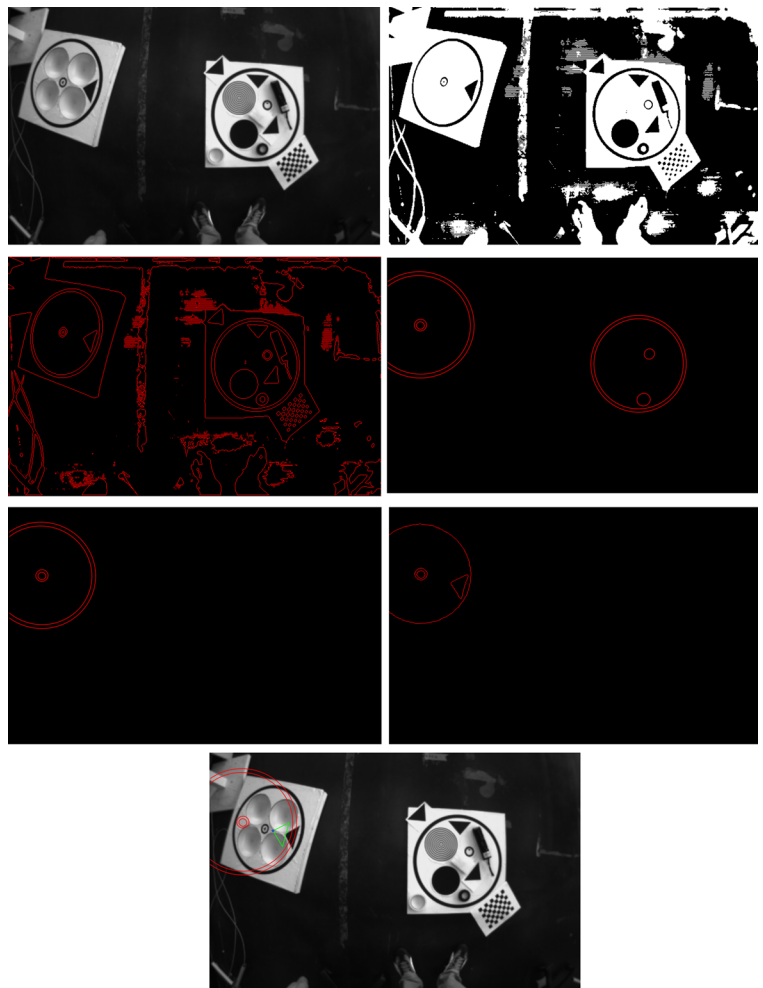
where α is a forgetting factor between 0 and 1. In our case $\alpha = 1/S$. The Wellner algorithm provides good results but the required time is too high for our application. The on-board Asctec Mastermind (Intel Core2Duo SL9400) performs the binarization within 8.971 ms, which could produce on overrun in the execution time considering the adopted workflow and the imposed deadline of 1/90sec.

Our requirement is to process the image provided by the camera (752x480 pixel) within 11.11 ms (90 Hz). We reduced operations involving floats by pre-computing, in a dedicated set of buffers, all possible operations with S and T defined a-prior considering the first decimal digital approximation starting from the uint8 pixel intensity. At the start-up, we calculate and store all the possible values.

The computation does not depend on S and T. It depends only on the size of the input image. The average required time to process an image using our algorithm is 1.997 ms.

The original algorithm has been modified to increase the accuracy and precision of the binarized image. Equation 1 has been modified so that also the moving average of the last S_b pixel defined as black is taken into account, as:

$$O(n) = \begin{cases} 0 \;\; if \; p(n) < g(n) \cdot (1 - T) \\ or \; if \; p(n) < j(n) \cdot (1 + T_b) \\ \qquad 1 \; otherwise \end{cases} \quad (4)$$

where $j(n)$ is the exponential moving average for the last S_b pixel defined as black, while T_b is a value between 0 and 1. Pixels in the input image that are at

Fig. 14 *Top left* source grayscale image. *Top right* binarized image. *Middle left* extracted contours. *Middle right* filtered contours with interpolated ellipses. *Bottom left* Detected target. *Bottom right* Detected shaped and triangle. *Bottom center* undistorted target. (S=752/24, T=0.45, Sb=50, Tb=0.25)
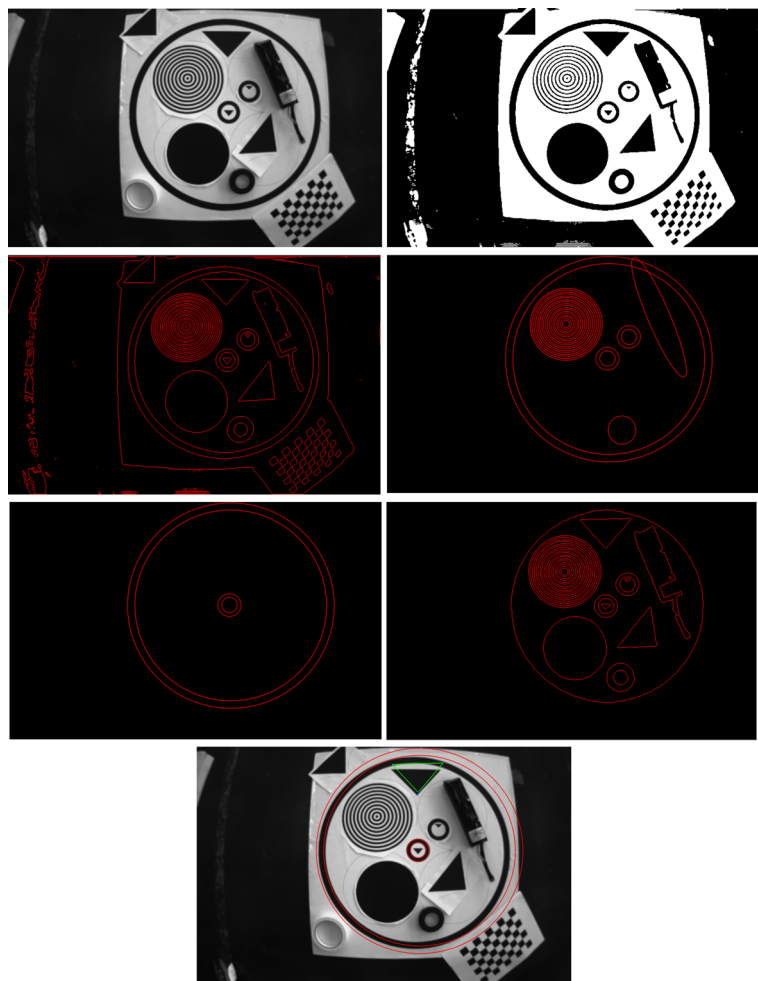
least the T_b percent less dark than the average intensity of last S_b pixel defined as black, are classified as black on the binary image.

This trick increases the accuracy of segmented image especially in regions with a large set of adjacent black pixels. This adaptation requires two additional buffers with performance of 2.16 ms on average.

The exact moving average could be calculate in a fast way applying the following equation:

$$y(n) = \frac{1}{N} \sum_{i=0}^{N-1} x(n-i) \\ y(n+1) = \frac{1}{N} \sum_{i=0}^{N-1} x(n+1-i) \tag{5}$$

and manipulating (5) it is possible to calculate the average according to:

$$y(n+1) = y(n) + \frac{(x(n+1) - x(n-N))}{N} \tag{6}$$

The ratio between the difference of actual pixel and the $x(n-N)$ is simplified by pre-calculating the result on a dedicated table.

In Fig. 8 a comparison between approximated average and exact one is shown. The computational time by applying (6) is 2.18 ms on overage.

The binarized images with our adaptive threshold algorithm have a better performance than OpenCV algorithms in terms of computational time and quality of results. The computational time is close to Otsu global threshold [16], which in our system takes on average 0.991 ms, while regarding quality our algorithm provides better results in presence of different light conditions and shadows.

In the OpenCV adaptive threshold function the chosen parameter directly defines the computational time required to process an image. In our solution, it depends only on image size. At the same time the parameter used for an image acquired at low altitude does not work well for an image acquired at higher altitudes in OpenCV adaptive threshold. Our solution with a fixed set of parameters for thresholding (T, S, T_b, S_b) correctly processes images taken at different altitudes with different light and shadow conditions. The computational performances of the OpenCV adaptive threshold are also worst, requiring on average in the faster implementation (with a simple mean over a small window of pixels) more than 6 ms in our system.

In Fig. 9 a comparison between Otsu, OpenCV adaptive threshold and Wellner algorithm is shown.

The second step is dedicated to contour extraction by applying the Suzuki algorithm [18], requiring 2.4 ms on average. Considering the geometry of our target we remove extracted contours that have no parents (we have concentric circles/ellipses) or with size/dimension not compatible with our target.

To remove the distortion of 2.8 mm lens, by applying the Zhang and Bouguet approach described in [22], the undistort method is applied only to the features of interest extracted.

This approach requires on average 0.76 ms. It saves a lot of time considering that the undistortion of a single frame requires on average 5.5 ms.

Fig. 15 The adopted reference systems
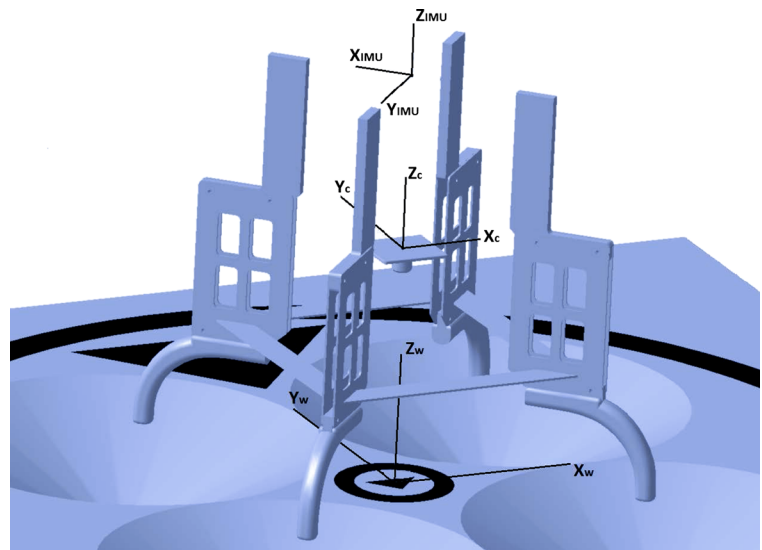
In the following phase undistorted contours are interpolated using the OpenCV function for extracting a set of ellipses, as shown in Fig. 10.

The sub-step applied to ellipses extracted is the detection of outliers and their removal. The basic idea is to cluster the center of the ellipses in different sets.

We select the set with the largest number of ellipses considering the working scenario and selected target. This procedure requires on average 0.04 ms.

On the selected working set of concentric ellipses the ratio among various real dimensions of the target is calculated as summarized in Table 1.
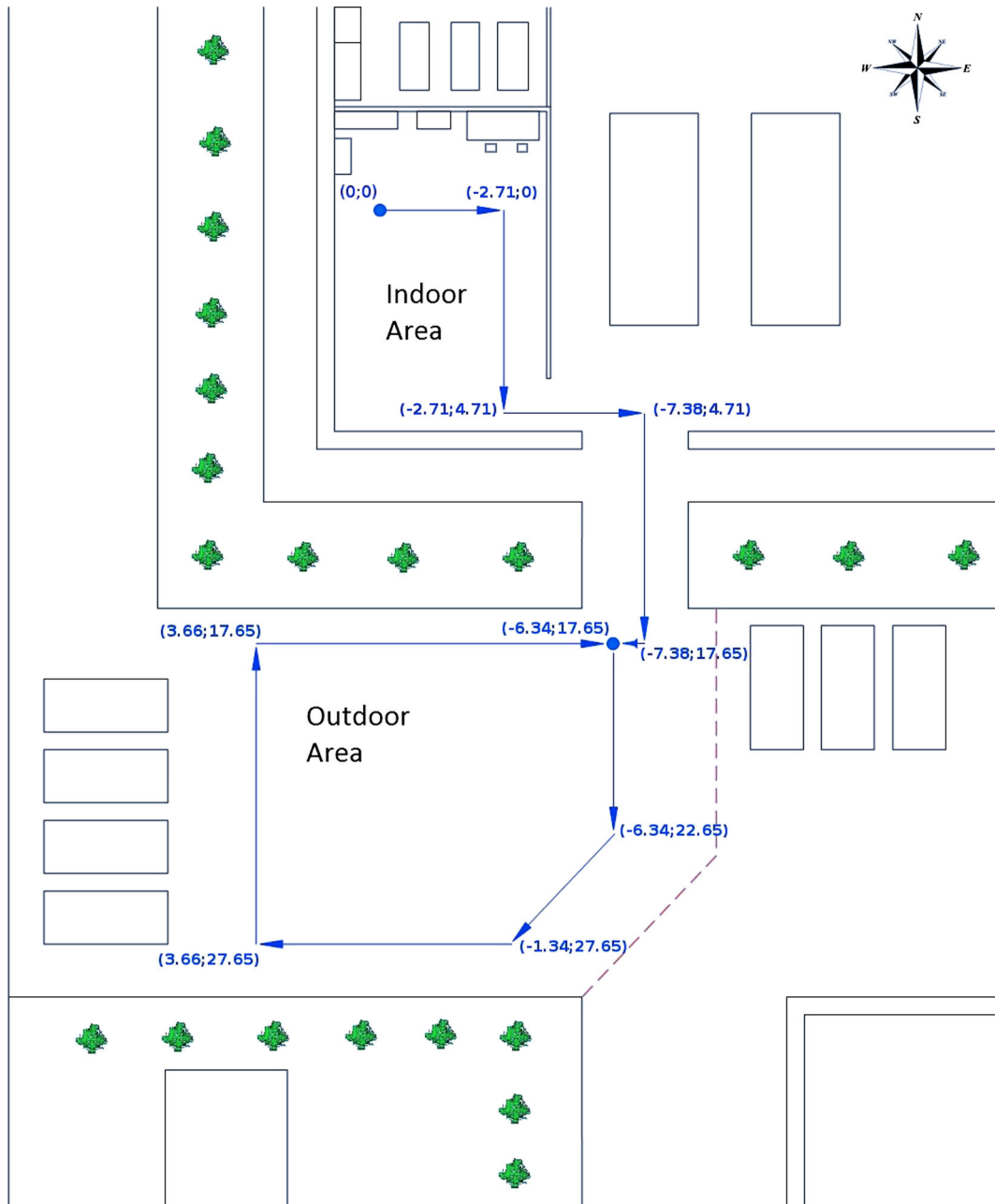

Fig. 16 The selected path with mixed indoor and outdoor areas

The ratio validates or not if the working set of ellipses is associated to the target, also detecting if the target is partially or fully visible.

In case of failure due to not valid proportion we selected - if exists - the second largest working set reiterating the filtering procedure discussed above. The required time is neglectable.

After the concentric ellipses detection, we look for the triangle by reusing the contour hierarchy.

A candidate triangle is identified by contours (approximated with 3 points by the Douglas-Peucker algorithm [14]) that have to respect the dimensions reported in Table 1. The required time is 0.12 ms on average.

In the worst case, the vision system requires 5.5 ms to completely process a frame: Table 2 lists the computational times of our workflow. It is important to consider that if a faster camera is available (e.g. 120 Hz), we are still able to respect the deadline. The most expensive process is the adaptive thresholding.

Figures 11 and 12 show the robustness of our algorithm to external disturbance. Figures 13 and 14 show the accuracy of our adaptive threshold algorithm that works fine even in presence of shadows and different illumination with a static set of parameters.

Using the available OpenCV methods it is not possible to use a single set of parameters, but it is necessary to change it according to the working scenario, which, however, is usually unpredictable.

Figure 14 shows how the filtering algorithm correctly discriminates the contours associated to the target even in presence of some noisy element with the same shape and size of the target. It clearly shows the correction of the distortion introduced by the 2.8 mm lens.

The 6DOF of the camera with respect to the landing platform are estimated from the largest visible circle and triangle. The estimation of drone pose (with exception of yaw angle) is based on the work proposed by Chen [11] starting from the undistorted ellipses.

The roll and pitch could be calculated by using the fused data of the IMU or by the perspective projection of ellipses [12].

The yaw angle is calculated from the largest visible triangle calculating the angle generated by its unique top vertex. We not used the magnetometer and/or fused yaw from IMU due to large variance in the estimation process. The landing platform sets our constraints as discussed in Section 3 and for this reason we not considered the yaw estimation provided by the IMU.
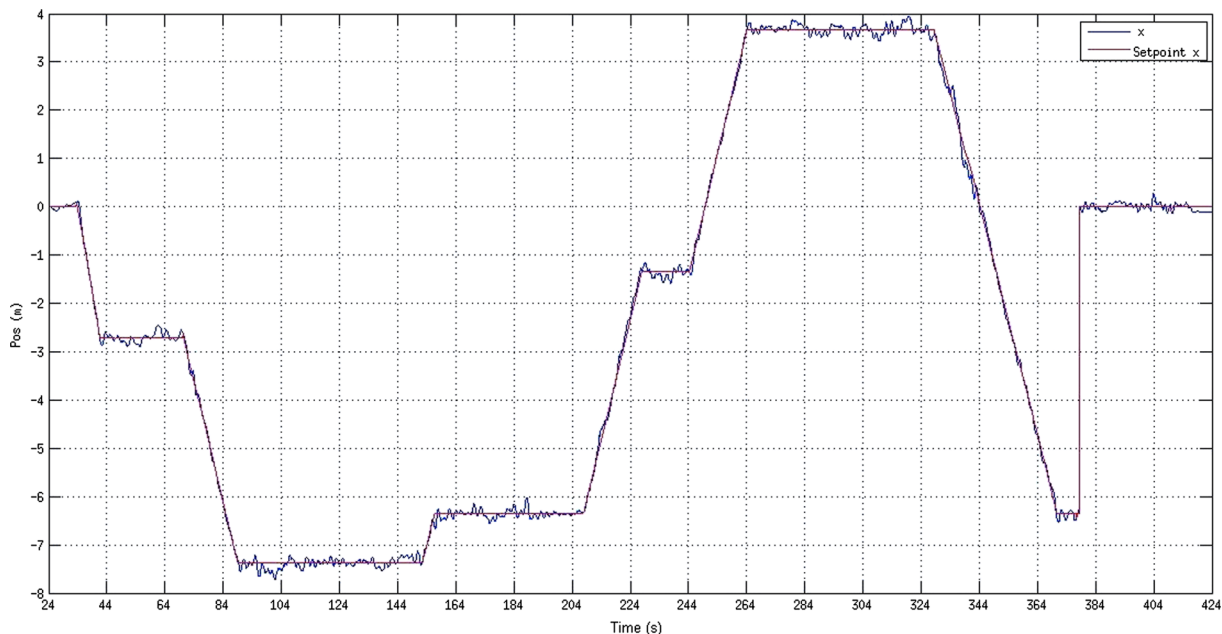

Fig. 17 A typical executed path on X axis

By taking into account the estimated roll, pitch and yaw it is possible to calculate the rotation matrix from the camera to the landing reference system (see Fig. 15).

The results obtained demonstrate that the error increases with the distance due to the loss of accuracy for each pixel in the landing target caused by:

– wide angle lens (2.8 mm);
– low resolution camera (752 × 480).

The height estimation is affected by the largest estimation error leading to an error of 5.9 cm at 200 cm of height. The planar estimation is more accurate and precise and it is bounded to 2 cm at 200 cm of height.

The attitude estimation by using the vision system is bounded to 2 degrees in the worst scenario. We used the IMU data as ground truth in quasi-static tests.

5 Indoor / Outdoor Flight Tests

In this sub-section we propose the result of flights in a mixed scenario. The first part of a flight is indoor and the remaining outdoor. In this test we use only the vision system without any external aid (e.g. GPS or UWB real time location system). The selected path is shown in Fig. 16.

In Figs. 17, 18 and 19 the executed path, respectively, along the X, Y and Z axis is shown.

The error during the path execution was bounded to 0.5m. The tests demonstrated the robustness of the developed vision system. To avoid odometric drift it is possible to use the GPS or any other localization system (e.g. UWB) especially where optical flow has a low number of features to track.

The test was critical mainly to the change in illumination and to the presence of shadows. The high accuracy and precision is ensured by the developed algorithm that thresholds the image.

6 GPU Boosting

In this section we present results on the implementation of the thresholding algorithm on the NVidia Jetson TK1 platform [3]. This platform is equipped with the GPU NVidia Kepler "GK20a" with 192 SM3.2 CUDA cores (over 300 GFLOPS). This platform is suitable to boost the on board image processing on a UAV platform [17].
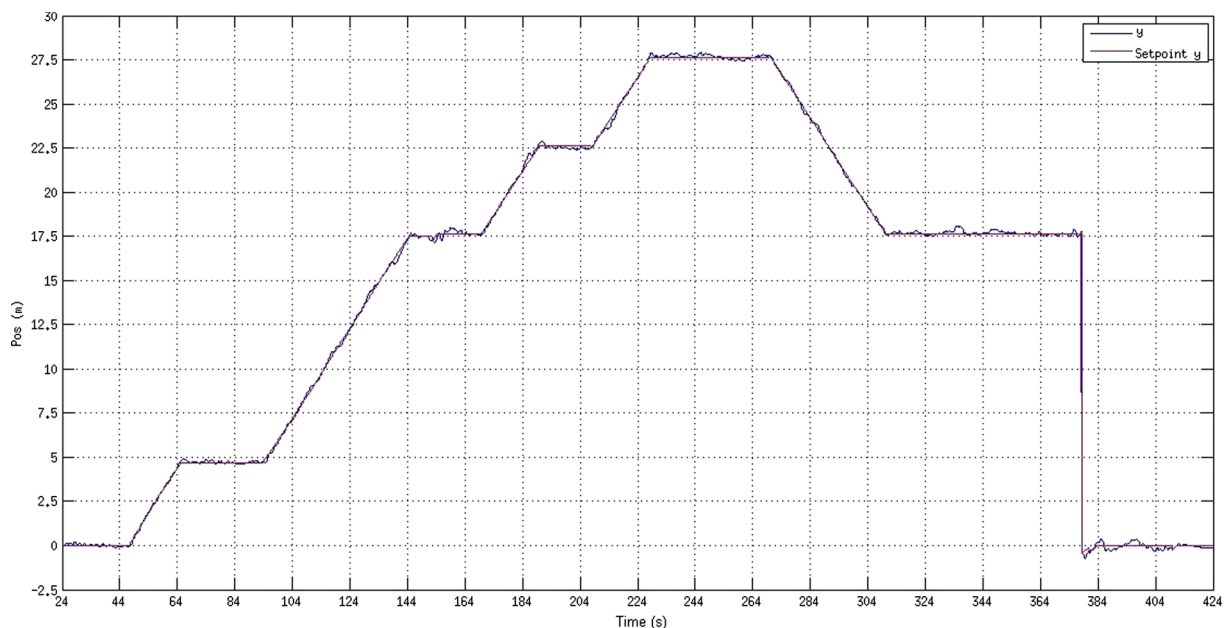

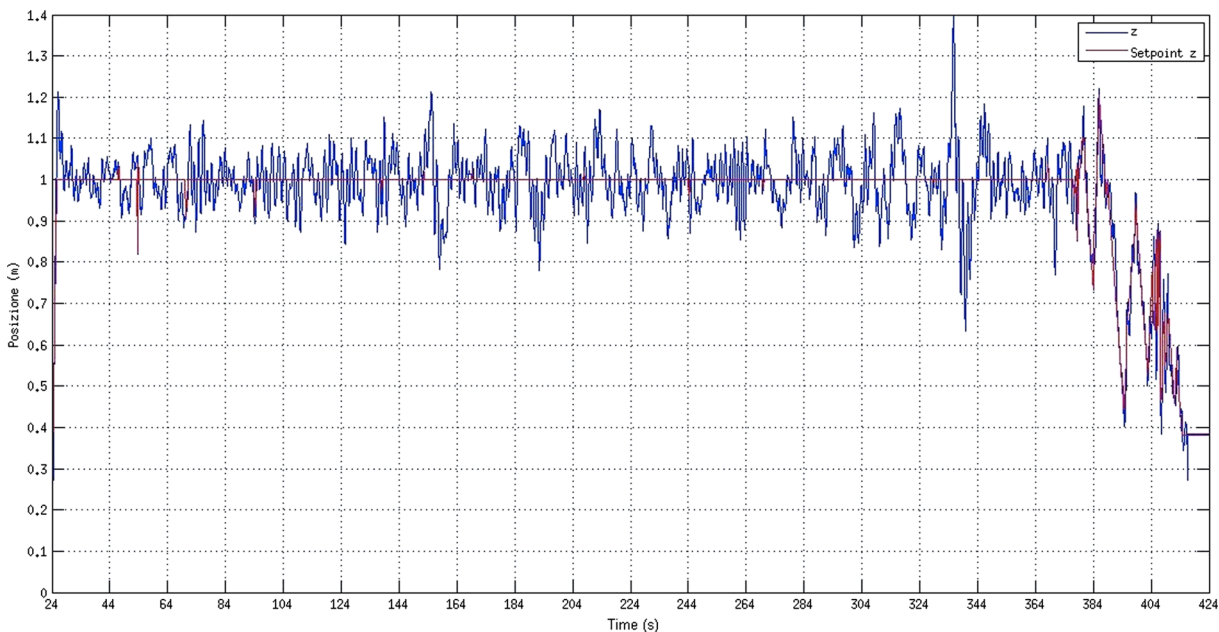**Fig. 18** A typical executed path on Y axis

Fig. 19 A typical executed path on Z axis

The main features of Jetson TK1 GK20A are listed in Table 3.

In particular, we focus on the following operations:

– Thresholding
– Image resizing

The thresholding algorithm described in Section 4 has been rewritten to be GPU ready.

The first step of adaption is the computation of pixel average. The pixel average can be easily computed by applying integral images to the reshaped image matrix from a 2D matrix to 1D array. In this way given the integral image $I(n)$ where n varies from 0 to $(W \cdot H) - 1$, the average for the n-th pixel, considering a window of N pixels, can be calculated as:

$$g(n) = (I(n) - I(n - N)) \cdot \frac{1}{N} \qquad (7)$$

The term $1/N$ could be also pre-calculated to avoid useless operations.

It is also possible for a pixel to consider the average of previous raw according to Eq. 2 in the following way:

$$\tilde{g}(n) = ((I(n) - I(n - N)) + (I(n - W)$$
$$- I(n - N - W))) \cdot \frac{1}{2 \cdot N} \qquad (8)$$

By applying this approach Eq. 8 considers only left-to-right row exploration. The approach can also be modified changing the calculation of the integral image.

The 1-D integral image is $I(n) = I(n - 1) + p(n)$ for left to right row pixel exploration while $I(n+1) = I(n) + p(n + W - 1)$ in case of right to left.

This approach is a cumulated sum that considers also the row change to implement left-to-right and right-to-left approach and is similar to the well-known prefix scan algorithm [8].

Table 3 Main features of the Jetson TK1 GK20A

Property Name	Value
Compute	3.2
Number of multiprocessors	1
Cores	192
Base clock rate	852 MHz
Total global memory	1746 MB
Total shared memory per block	48 KB
Total constant memory	64 KB
Memory clock rate	924 MHz
Memory bus width	64-bit
Total registers per block	32768
Warp size	32

Fig. 20 Example of adaptive thresholding by using the GPU

A simpler approach is to consider a not-separable kernel to calculate the average. The kernel is the following:

$$\frac{1}{2N}\begin{bmatrix} 1 \ldots 1 & 0 \ldots 0 \\ 0 \ldots 0 & 1 \ldots 1 \end{bmatrix} \qquad (9)$$

The asymmetric kernel is formed by a block of N ones and N zeros and it simulates the left to right and right to left scan of the image. The operations required are $2N \cdot W \cdot H$, but it is possible to compensate the complexity by using the GPU. In Fig. 20 the result of thresholding using the GPU is shown. The correction of Eq. 4, which considers pixels in the input image that are at least the T_b percent less dark than the average intensity of last S_b pixel defined as black, is not applied in the adopted GPU kernel.

For tests, we adopted the following configuration: CUDA v6.0 and OpenCV2.4.8 for Tegra with built-in support for GPU computing.

The image resizing exploits the built-in GPU functions of OpenCV for L4T(Linux for Tegra). The comparison between CPU and GPU on frames by 752x480 pixels shows that for a small image most time (1 ms on average) is spent to transfer data from host to device and vice-versa.

7 Conclusions and Future Works

In this work, we presented an integrated approach to deal with the problem of autonomous quadrotor landing over a platform.

Our work started from the design of an artificial B/W target that enables the estimation of 6DOF of the UAV.

The developed platform hosts the UAV also for recharging the battery owing to copper plated contacts installed on the bottom part of cones. This feature opens new frontiers to extend the endurance of UAV quadrotors, especially in structured environments where it is possible to install landing platforms over the region of interest. The landing time obviously depends on the environmental conditions, but is limited to 1–2 min.

The developed vision system is able to run at the maximum frame rate of our camera (90 Hz), but it is possible to go faster due to the optimization performed in each step of our workflow.

We introduced also the use of the Jetson TK1 processing unit to boost some procedures as the thresholding and the optical flow for vision-based navigation could benefit of the GPU. The results obtained show that the GPU boosting frees the CPU for other tasks (e.g., related to the application).

Future works will be steered to adapt the landing platform to balance the Li-po cells and to continue the optimization process by extensively using the GPU.

References

1. Dod unmanned systems integrated roadmap 2013: (2013–2038) (2013). http://www.defense.gov/pubs/DOD-USRM-2013.pdf
2. Asctec MAV Framework (2015). http://wiki.ros.org/asctec_mav_framework
3. Nvidia jetson TK1 (2015). https://developer.nvidia.com/jetson-tk1
4. OpenCV (2015). http://opencv.org/
5. R3COP (2015). http://www.r3-cop.eu
6. ROS (2015). http://wiki.ros.org
7. Benini, A., Mancini, A., Longhi, S.: An imu/uwb/vision-based extended kalman filter for mini-uav localization in indoor environment using 802.15.4a wireless sensor network. J. Intell. Robot. Syst. **70**(1–4), 461–476 (2013). doi:10.1007/s10846-012-9742-1
8. Blelloch, G.E.: Prefix sums and their applications. Tech. Rep. CMU-CS-90-190, School of Computer Science. Carnegie Mellon University (1990)

9. Cesetti, A., Frontoni, E., Mancini, A., Ascani, A., Zingaretti, P., Longhi, S.: A visual global positioning system for unmanned aerial vehicles used in photogrammetric applications. J. Intell. Robot. Syst. **61**(1–4), 157–168 (2011). doi:10.1007/s10846-010-9489-5

10. Cesetti, A., Frontoni, E., Mancini, A., Zingaretti, P., Longhi, S.: Vision-based autonomous navigation and landing of an unmanned aerial vehicle using natural landmarks. In: 17th Mediterranean Conference on Control and Automation, 2009. MED '09, pp. 910–915 (2009)

11. Chen, Q., Wu, H., Wada, T.: Camera calibration with two arbitrary coplanar circles. In: Computer Vision - ECCV 2004, *Lecture Notes in Computer Science*, vol. 3023, pp. 521–532. Springer, Berlin (2004)

12. Cocchioni, F., Mancini, A., Longhi, S.: Autonomous navigation, landing and recharge of a quadrotor using artificial vision. In: 2014 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 418–429 (2014)

13. Cocchioni, F., Pierfelice, V., Benini, A., Mancini, A., Frontoni, E., Zingaretti, P., Ippoliti, G., Longhi, S.: Unmanned ground and aerial vehicles in extended range indoor and outdoor missions. In: 2014 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 374–382 (2014)

14. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The International Journal for Geographic Information and Geovisualization **10**(2), 112–122 (1973). doi:10.3138/FM57-6770-U75U-7727

15. Honegger, D., Meier, L., Tanskanen, P., Pollefeys, M.: An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 1736–1741 (2013)

16. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. **9**(1), 62–66 (1979). doi:10.1109/TSMC.1979.4310076

17. Rud, M.N., Pantiykchin, A.R.: Development of gpu-accelerated localization system for autonomous mobile robot. In: 2014 International Conference on Mechanical Engineering, Automation and Control Systems (MEACS), pp. 1–4 (2014)

18. Suzuki, S., Be, K.: Topological structural analysis of digitized binary images by border following. Comp. Vision Graph. Image Process. **30**(1), 32–46 (1985). doi:10.1016/0734-189X(85)90016-7

19. Swieringa, K., Hanson, C., Richardson, J., White, J., Hasan, Z., Qian, E., Girard, A.: Autonomous battery swapping system for small-scale helicopters. 2010 IEEE International Conference on Robotics and Automation (ICRA), pp. 3335–3340 (2010)

20. Toksoz, T., Redding, J., Michini, M., Michini, B., How, J.P., Vavrina, M., Vian, J.: Automated Battery Swap and Recharge to Enable Persistent UAV Missions. AIAA Infotech@Aerospace Conference (2011)

21. Wellner, P.D.: Adaptive Thresholding For The DigitalDesk. Tech. Rep. EPC1993-110, Xerox (1993)

22. Zhang, Z.: A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. **22**(11), 1330–1334 (2000). doi:10.1109/34.888718

Francesco Cocchioni received the M.S. degree in IT and Automation Engineering in 2010 from the Università Politecnica delle Marche, Ancona, Italy. His main research interests include image processing, computer vision, embedded systems and Unmanned Aerial Systems.

Emanuele Frontoni joined the Dept. of Information Engineering (DII) at the Università a Politecnica delle Marche, as a Ph.D. student in "Intelligent Artificial Systems". He obtained his PhD in 2006 discussing a thesis on Vision Based Robotics. At present he has an Assistant Professor position in the same department. His research focuses on artificial intelligence and computer vision techniques applied to robotics, internet of things, ehealth and AAL. He is a member of IEEE, ASME MESA TC, GIRPR and AI*IA.

Gianluca Ippoliti received his master's degree in Electronic Engineering and his Ph.D. in Intelligent Artificial Systems from the Universita' Politecnica delle Marche (formerly University of Ancona), Ancona, Italy, in 1996 and 2002, respectively. From 1997 to 1998, he was with ISERM Unit 103, Montpellier, France and then with the University of Montpellier I, Montpellier, France, in the framework of the European research projects CAMARN and MOBINET. From 2002 to 2005, he held a postdoctoral position at the Dipartimento di Ingegneria Informatica, Gestionale e dell'Automazione, Universita' Politecnica delle Marche. Since March 2005, he is an Assistant Professor at the Universita' Politecnica delle Marche. He has published more than 140 papers on international journals, books, and refereed conferences. His main research interests include switched systems and supervisory control, neural-network based system identification and control, modeling identification and control of robotic, electrical and electromechanical systems, and mobile robot control and localization.

Sauro Longhi (SM'04) received the Ph.D. degree in electronic engineering from the University of Ancona, Ancona, Italy, in 1979 and the Postgraduate Diploma in automatic control from the University of Rome "La Sapienza," Rome, Italy, in 1984. From 1980 to 1981, he held a fellowship with the University of Ancona. From 1981 to 1983, he was with the R&D Laboratory, Telettra S.p.A., Chieti, Italy, mainly involved in activities of research and electronic design of modulation and demodulation systems for spread-spectrum numerical transmission systems. Since 1983, he has been with the Dipartimento di Elettronica e Automatica, University of Ancona, which is now the Department of Information Engineering, Università Politecnica delle Marche, where he is currently a Full Professor of control systems technologies. His research interests include modeling, identification, and control of linear systems; control of mobile base robots and underwater vehicles; service robots for assistive applications; web technology in process control and remote control laboratories; power management in hybrid cars; and cooperative control of autonomous agents. He has authored or coauthored over 300 papers in international journals and conference proceedings in these fields.

Adriano Mancini joined the Dept. of Information Engineering (DII) at the Università Politecnica delle Marche, as a Ph.D. student in "Intelligent Artificial Systems". He obtained his PhD in 2010 discussing a thesis titled "A new methodological framework for land use/land cover mapping and change detection". At present, he has a Postdoc position in the same department. His research focuses on Mobile Robotics also for assisted living, Machine Learning, Image Processing and Geographical Information Systems. He is a member of the IEEE.

Primo Zingaretti is Professor of Computer Science at the Università Politecnica delle Marche, Italy. His main research interests are in artificial intelligence, robotics, intelligent mechatronic systems, computer vision, pattern recognition, image understanding and retrieval, information systems and e-government. Robotics vision (aerial, ground and underwater autonomous systems) and geographical information systems (GIS) have been the main application areas, with great attention directed to the technological transfer of research results. He his author of over 150 scientific research papers in English, Senior Member of IEEE, Member of ASME, Co-founder of AI*IA and Member of GIRPR-IAPR.