Safety Net Detection by Optic Flow Processing

Xavier Daini, Charles Coquet, Romain Raffin,, Thibaut Raharijaona, Franck Ruffier

Safety Net Detection by Optic Flow Processing

Xavier Daini[1], Charles Coquet[1], Romain Raffin[2], Thibaut Raharijaona[3] and Franck Ruffier[1]

***Abstract*— Drone navigation is an area of study that is receiving more and more attention. Obstacle detection techniques and autonomous guidance are continuously improving, but some types of obstacles are still very difficult to detect with current methods. Safety nets used to separate and secure 2 contiguous spaces are indeed very difficult to detect by Lidar and by image processing based on pattern recognition. The method we propose here separates the Optical Flow detections to identify the presence of a safety net: i) by using the norm of their vector, ii) by matching them to a regression defining a plane (safety net or wall). Our results show that the proposed method detects a net in front of a wall with very few false positives, thanks to a small displacement (at most 5%). Moreover, the distance estimation between the net and the wall as well as the distance between the net and the drone can be estimated with at most 20% error in the worst cases.**

I. INTRODUCTION

Considering collision avoidance, it is important for a drone to detect safety nets and to know their location and distance from the observer, especially when the drone is navigating autonomously. When the drone is flying indoor or outdoor, safety nets are a threat that is invisible to most visual detection systems. Indeed, the usual tools in computer vision to detect and map the environment (such as SLAM with Lidar), fail to detect safety nets. The human operator still needs to choose a drone-friendly environment devoid of obstacles to prevent collisions.

Indeed, most visual navigation methods are based on opaque surfaces. Recent studies are focusing on semi-transparent spaces using mostly Lidar [1] [2] [3] [4]. Other methods such as pattern recognition [5], sensor fusion (ultrasonic sensor and Lidar [6]), even image clustering method [7] or direct image processing to detect window frames [8]. Learning image processing techniques could recognize the pattern of a safety net, but providing information on the distance of this obstacle. The presence of a safety net definitely defines a dangerous obstacle for the drone.

Nevertheless, net detection is a vital point in Anti-Drone Warfare (ASW). Indeed, nets are one of the most ancient while still very performant way to destroy or put out of commission a hostile drone: projected nets are used in order to incapacitate enemy drones [9].

Several robots and drones are shown being able to navigate solely based on the optic flow [10], including to stabilize a drone without accelerometer [11], [12] and to perform minimalistic visual odometry [13].

A new trend is to use deep learning methods (CNN) to assess Optic Flow in order to perform accurate ego-motion navigation [14]. These methods are limited by the data used to train the neural network and require substantial computing power.

Previous studies rely on piece-wise planar approximation of the shape of the surrounding surfaces (series of orientated plans) using Optic Flow [15]. Low computational Optic Flow based planar detection can also lead to safe terrain-following [11].

Optic Flow-based method to detect the presence of a safety net in front of a background surface. This visual method uses successive regressions of Optic Flow data to identify planar surfaces arranged one behind the other, which indicates the presence of a non-opaque surface. We also provide a way to estimate the distance relative to the detected safety net and to what background lies behind it.
A hexarotor equipped with a low cost global-shutter camera was used to show the impact of a safety net on the Optic Flow field, measured during a flight in front of a textured background (section II). Tests were conducted with and without the safety net to evaluate the accuracy of the detection. Section II also details the material used to capture the images and the method to extract Optic Flow. Various methods exist for extracting Optic Flow vectors. Some Optic Flow extraction methods are based on a limited number of 2D-points named "corners" [16] while others focus on "dense vector field" where at each pixel is estimated a value of the Optic Flow vector-field [17], [18].

In section III, we explain how Optic Flow can be used to detect surfaces. The method using successive planar surfaces detection to detect the safety net is provided in section VI.

In section VIII, we present aerial robotic experiments where the drone was following a desired trajectory. The Optic Flow-based detection method was performed offline in these preliminary results. The method we propose detected a safety net with very low number of false positives (less than 5%) in conjunction with accurate estimations of the distances between the drone and the safety net, as well as between the drone and the background.

II. MATERIAL AND OPTIC FLOW ACQUISITION METHODS

We will describe hereafter both the UAV and the camera it carries.

[1] Aix-Marseille Université, CNRS, ISM, 13009 Marseille, France {xavier.daini, charles.coquet, franck.ruffier}@univ-amu.fr

[2]Université de Bourgogne, LIB EA 7534, 21000 Dijon, France romain.raffin@u-bourgogne.fr

[3]Université de Lorraine, Arts et Métiers Institute of Technology, LCFC, HESAM Université, F-57070 Metz, France thibaut.raharijaona@univ-lorraine.fr

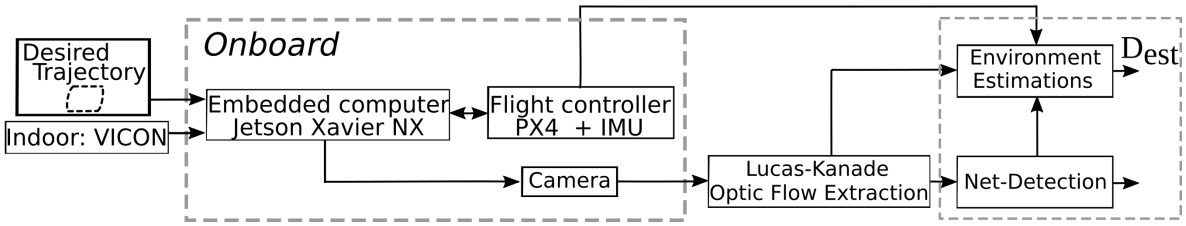

Fig. 1. The images are processed using a Lucas Kanade method to extract the Optic Flow field. This field is then processed to detect the presence of the net in viewed or not. To estimate the distance toward the safety net, the processing algorithm uses both the ground truth velocity from the MoCap system and the Optic Flow field: it allows distance estimation between the drone and the safety net D_{est}^{net} as well as between the safety net and the wall behind D_{est}^{wall}.

Fig. 2. The hexarotor flying in the Marseille's flying arena in front of the safety net. Through the safety net, we can see a textured panel considered here as a background wall

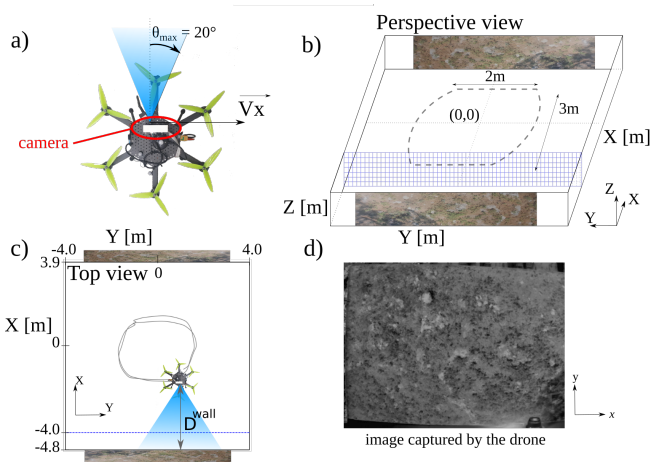

Fig. 3. The hexarotor (a) has an embedded camera. It moved perpendicularly to its optical axis and along the wall. Because the arena is very large, we only used the center of it in order to have the best tracking margin with our MoCap system (b). This leads us to focus on the two parallel plans, one in the back and one in front of the arena (b). Also, the movement is considered to be at constant z altitude (c) and so we removed the Z vertical component of the Optic Flow acquired from captured images.

Specifics	Global Shutter Camera	Used parameters
Size pixels	$3.6\mu m^2$	$3.6\mu m^2$
Number pixels	1080×720 px	640×480 px
FOV	100°	40°
Framerate	30 Hz	10 Hz

TABLE I

INITIAL PARAMETERS OF THE CAMERA (LEFT) AND THE USED ONES (RIGHT).

A. The Hexarotor

We used a hexarotor developed together with Hexadrone™ based on the PX4 low-level flight controller [19]. We also used a trajectory tracking algorithm[1] to apply an hexagonal-like route on the hexarotor. PX4 is particularly convenient, thanks to its adaptability to the nature of the drone (air-wing, quadrotor, hexarotor, *etc.*), and its reliability when the drone is associated with QGroundControl, a Ground Control Station (GCS), and the MAVLINK protocol. The position and orientation of the drone used in the drone controllers came from the MoCap system setup in the Mediterranean Flying Arena. The flying arena was equipped with 19 motion-capture cameras (VICON) inside a volume of $6 \times 8 \times 6$ meters.

The experimental setup is illustrated in Figure 1. The drone was equipped with a Jetson Xavier NX as onboard computer. The onboard computer was connected in USB to a low-cost global-shutter camera[2], as we can see in Figure 3a. Finally, the datasets including the Optic Flow measurement were recorded via the Robot Operating System (ROS) and processed at a frequency $f = 10$ Hz.

B. Optic flow acquisition

1) Drone trajectory: For our experiments, we built a hexagonal-like trajectory in order to keep the drone on a continuous circuit. This trajectory had 2 linear trajectories where the camera went along 2 different vertical textured planes, as illustrated in Figure 3b. The first vertical plan is composed of a net and a plan behind this net. We planned the trajectory to keep a minimal distance of $D^{net} = 2.7\text{m}$ between the drone and the safety net. The wall behind is placed at $D^{wall} = 3.5\text{m}$ from the drone's trajectory. The second vertical plan has no net in front of it, and was set at $D^{wall} = 2.7\text{m}$ from the drone. The interest of a hexagonal-like trajectory is that it reduces as much as possible the angle to turn before a straight line, while also reducing the movement on the borders (and so not going out of the area

[1]https://github.com/gipsa-lab-uav/trajectory_control

[2]ELP USBGS720P02-L36 with a 3.6 mm objective

covered by our MoCap system). Furthermore, this trajectory can be continuous, without pauses to stabilize as most of the rotation inducing perturbation will be corrected during the "sides movement" which is not covered by this study. This continuous rotation on the sides is what causes us to call it hexagonal-like, as the shape will not follow a strict hexagon, as you can see on Figure 3bc. Finally, a short pause of 2 s when the drone finished to go along a wall was necessary to catch up a potential delay between the setpoint and the real trajectory.
During the trajectories, and more precisely during the parallel translation along the planes, we had $Vx_d = 0.25\mathrm{m}.s^{-1}$ and $Vy_d = 0$ while having a strong control on the yaw of the drone (to be as much as possible parallel even in case of noises or perturbations).
Finally, on its onboard computer, the hexarotor saves the images captured by the camera as well as its position (red on the MoCap) at the same time in a single file.

2) Saved images: We can see in Table I that the images read are cropped (from 1080×720 px to 640×480 px) and the Field-of-View (FOV) is smaller (from 100° to 40°). This is due to our choice to reduce as much as possible the computations and the data flow through our onboard computer. The process, we used for the flight, was a series of waypoints to follow. In order to achieve this, the MoCap system refreshed the drone position at 100Hz, leaving few times for images acquisition. Images acquisition were made every 100ms as a trade-off.

After the experiment, we obtained a set of images as shown in Figure 3d, with corresponding attitude from the drone. The images were then pre-processed in order to extract their 60-pixel wide central band. We decreased the number of pixels to be processed, reducing the search area for Optical Flow method. This reduces the computational cost, and sets the study in a 2D case.

3) Lucas Kanade Method for Optic Flow Extraction: We used a Lucas Kanade [16] algorithm in conjunction with a Shi-Thomasi [20] corner detector, in order to get the Optic Flow vector field. This method selects some points on one image (also called corners) and tries to follow them in the next one. The process is highly configurable: we can set the number of searched point, their quality as well as the number of filter's levels, which tune the final Optic Flow measurement but require more computations.
Other methods exist, such as the Gunnar Farnesback method [17] or [18] which give a dense Optic Flow, very interesting in principle for our study. But its main drawback is that they smooth the Optic Flow measurements, removing the discontinuities that we try here to detect (like in Figure 4d). As a consequence, these alternative Optic Flow extraction methods would prevent the detection of two parallel planes (one in front of the other), as we try here to separate them here.

C. Backprojection

At this step, the Optic Flow measurement was planar, being taken from a 2D image. It is composed of 2D vectors following detected corners between two images. However, the most useful representation of the Optic Flow is the spherical one. The only difference is the fact that the planar flow is projected on a sphere, in order to get an optic flow magnitude in $[rad.s^{-1}]$.
To do this, a matrix expression of the retro-projection, taking into considerations the camera intrinsic parameters (size of pixels, focal and resolution) had to be used. It was applied to all the Optic Flow vectors detected before and put from a 2D planar representation to a 3D spherical one.
We shall not go deep in the computations here, however it is an important step that the reader can find in [19] and the camera parameters, depending on the type of camera used, are given in [21]. The idea is to compute the corresponding position (X, Y, Z) on a sphere or radius f (f being the focal distance of our lens) of each point (x, y) on the receptor plane. This computation is made by a matrix and by knowing all the positions (X, Y, Z) we can compute the 3D Optic Flow vector, also named spherical flow.

III. Ground-truth Optic Flow expression

It has been shown in [22] that the expression of the ground-truth spherical Optic Flow at step i and angle observed θ: $\overrightarrow{\omega_i}(\theta_i)$ can be computed via the knowledge of the desired point to which we compute the ground truth Optic Flow, with respect to the observer (distance: σ_i and direction: $\overrightarrow{d_i}$), and the motion of the observer ($\overrightarrow{T}$).
First, we assumed that the drone trajectory was linear (without rotation) meaning $\overrightarrow{R} = \overrightarrow{0}$. Therefore, we canceled the rotational term of the general ground-truth Optic Flow equation. The Optic Flow perceived by the drone can then be defined as follows :

$$\overrightarrow{\omega_i}(\theta_i) = -\frac{1}{\sigma_i(\theta_i)} * \left(\overrightarrow{T_i} - (\overrightarrow{T_i}.\overrightarrow{d_i}(\theta_i)).\overrightarrow{d_i}(\theta_i)\right) \quad (1)$$

By defining the angle of observation (θ_i) with respect to the main optical axis of the camera, we obtain the following expression:

$$\begin{aligned} |\omega_i(\theta_i)_x| &= \frac{cos(\theta_i)}{D} * (V_x - V_x * \sin(\theta_i) * \sin(\theta_i)) \\ &= \frac{\cos(\theta_i)}{D} * V_x * (1 - \sin^2(\theta_i) \\ &= \frac{V_x}{D} * \cos^3(\theta_i) \end{aligned} \quad (2)$$

Where:

- D is the distance from the orthogonal point of view, it can either be D^{wall} or D^{net}
- V_x is the component of $\overrightarrow{T_i}$ in the direction parallel to the plan
- θ_i is the angle at which the considered point is observed

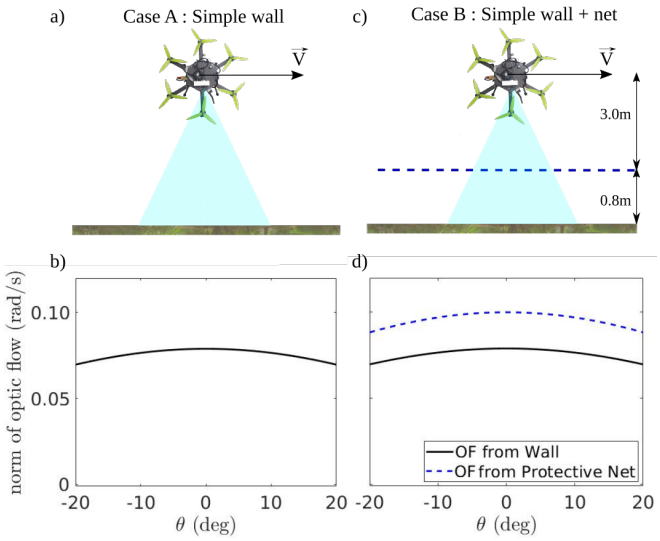

Fig. 4. The value of the Optic Flow magnitude seen along the $x-axis$: $|\omega_i(\theta_i)_x|$ vary according to the angle considered with $V_x = 0.25m/s$. In Case (A), (a) the drone moves parallel to a single textured wall. The Optic Flow has a cos^3 type of curve as shown in (b). In Case (B) there is a safety net between the drone and the wall (c). Some corners detected by the Lucas & Kanade method would then belong to the safety net, leading to two curves: one from the wall (in black) and one from the net (in blue) (d).

IV. Simplified Case Study

In this first study, to simplify, the drone moves in translation only parallel to the plane to be detected (Figure 4ac). We measured the Optical Flow in the direction of movement (along the x-axis of the image) during a movement parallel to the background plan (Figure 4ac).

We tested our net detection algorithm in 2 cases:

- Case A: there is nothing between the drone and the plane, so the Optical Flow amplitude measurements should be compact around a single Optical Flow magnitude curve (Figure 4b).
- Case B: there is a net between the drone and the plane, so the Optical Flow amplitude measurements should be more dispersed and possibly distributed around 2 different Optical Flow magnitude curves (Figure 4d).

V. Ground-truth optic flow curve defining a plane

In presence of a plane, the Optic Flow magnitude measurements can be fitted to a ground truth Optic Flow curve defining a plane: $A_\omega * \cos^3(k * \theta)$ (equation 2) with A_ω and k being the parameters of our ground truth equation model. k is introduced here to compensate for the approximation of the ground truth Optic Flow equation, for the measurement errors and for the uncertainties of the drone trajectory. Our assumptions of a perfectly linear translation along the walls is an approximation, as the drone will have some small variations on its pitch and roll. The addition of the k parameter helps to reduce the fitting error, while being simple and computationally efficient.

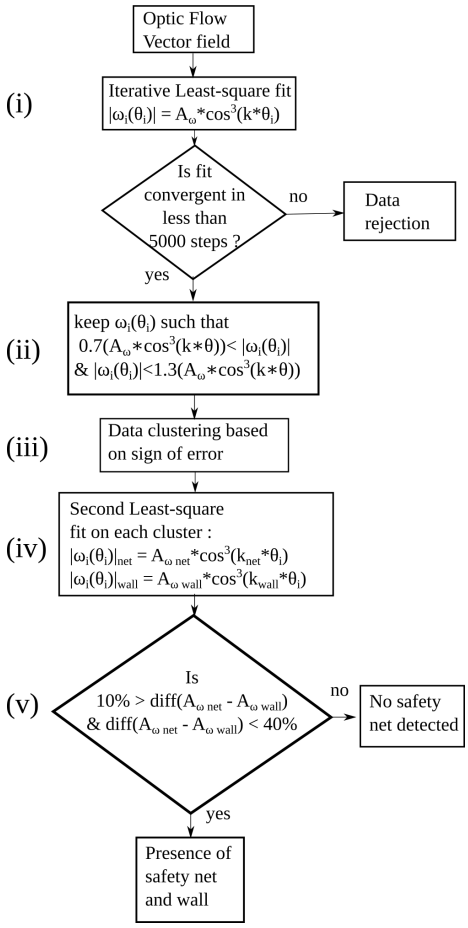

Fig. 5. Flowchart of the process used in order to detect the presence of a safety net. First, the magnitude of the flow vectors are fitted using an iterative least-square method. Then clusters are made from the initial data and the fitted curve. Finally, a second fit is realized on each cluster. If the two new fitted curves are different enough, then we concluded to a presence of a safety net on the original image, if not, then we only concluded that no net were detected.

Different forms of regression curves to match the data points could also be envisioned.

We know from [22] and our assumptions that the Optic Flow magnitude measurements follows a $\cos^3$ function in the presence of a parallel plane. Hence, in the present case, the least-square method will efficiently fit the $\cos^3$ function to our optic measurement to assess A_ω and k. Alternatively, a Gaussian fit might have been more effective in the case of a broad Field-of-View instance or when the drone's movement is more complex than a line parallel to a plan.

VI. Net Detector

At this point, we had a series of Optic Flow vectors. We had then two cases to treat:

- either we have one plane, and so we should have only one curve (Figure 4ab)
- or we see both a safety net and a plane on the camera. Hence, two curves of Optic Flow magnitude will be the signature of the presence of a safety net (Figure 4cd)

As we assumed that the plane and the net were parallel to each other (Figure 4c), they had the same type of Optic Flow norm distribution (equation 2) but with a different amplitude A_ω (Figure 4d). As a result, the following algorithm was designed to determine whether one or two Optic Flow curves are present:

(i) We fit our Optic Flow magnitudes using an iterative least square method with limit on the number of iteration (as shown in Figure 5). The least square method optimizes the curve to minimize the sum of squared error to each point.

(ii) The net-detector then selects all the Optic Flow magnitudes belonging to a band around the fitted curve. This band is made between 0.7 times and 1.3 times of the initial fitted curve. This allows us to remove outliers without making many assumptions on the Optic Flow magnitude.

(iii) If the initial fit is within the confidence interval, the net-detector divides the Optic Flow magnitude measurements in two sets (or clusters) defined by their positive and negative errors with respect to the first fitting curve.

(iv) Then, these two clusters of Optic Flow magnitude are respectively fitted again to the approximated ground-truth Optic Flow equation. The net-detector uses the distance between two fitting curves to detect or not if one or two planes are seen by the camera. The algorithm will always provide a solution whether the errors are important or small.

(v) After having fitted each cluster with a $|\omega_i(\theta_i)| = A_\omega * \cos^3(k * \theta_i)$ curve, we compare the difference between $A_{\omega\, net}$ and $A_{\omega\, wall}$. If the difference is small (less than a 10% difference between them), then it means that the distribution is quite centered, we can assume a "one curve" set of Optic Flow magnitude which implies that no safety net were detected (Case A, Figure 4ab). However, if the clusters are relatively far apart and without too much disparity (more than a 10% difference but not more than 40% – more than 40% would mean a wrong fit–), the net-detector consider one curve for each cluster resulting in two planes (Case B, Figure 4cd). As it is impossible to see through an opaque plane, one of them must be semi-transparent or a safety net: the plane belonging to the safety net is defined by the largest A_ω (which will correspond to $A_{\omega\, net}$) which is the closest to the camera (Figure 4bd).

VII. Method of the Safety net distance estimation

Another aspect of this study is also to estimate the distance to the detected safety net.

By using the equation (2) and having the knowledge of our movement with respect to the observed plane –the amplitude of velocity vector–, the net-detector can directly deduce the distance drone/plane. This is achieved using the $cosine$ coefficient output by the fitting method A_ω.

$$|\omega_i(\theta_i)_x| = \frac{V_x}{D} * \cos^3(\theta_i) = A_\omega * \cos^3(\theta_i)$$

Using this equation, we can directly obtain D, which is the distance we defined as the orthogonal distance drone/plane (Figure 3c).

$$D^{wall} = \frac{V_x}{A_\omega} \tag{3}$$

The drone trajectory can generate noise since the drone's movement cannot be perfectly parallel to the plane, and some rotations are also possible. To improve the estimation, the D_{est} estimated at $10Hz$ is averaged along each trajectory.
In such preliminary experiment, the drone velocity will be given by the MoCap system in our flight arena.

VIII. Experimental results

A. Quality analysis of net detection with a textured background

We conducted two experiments using the same texture. The first one was set up without using a net (Case A), while the other experiment was done using a net (Case B). We used the same texture in both experiments to make the experiments comparable.

As we can observe, there are some images that were not usable (Table II lines 3 and 4). This is attributed to two primary factors:

- Errors in the iterative least squares fitting method resulting from excessive computation (e.g run-time error);
- Non-parallel movement of the drone with respect to the safety net.

The algorithm detects the presence of a safety nets within two successive images, with a low percentage of missed detection (less than 5%) and a low percentage of false alarms (at most 10%). Each case of determining the net detector –A or B– is the result of measurements made from the set of images taken during a single flight of the drone in front of the safety net and where all images are processes. However, a case determination based on only one image can be problematic because it is likely to be affected by factors such as noise or other kind of errors. We then decided, to choose a minimum of 70% of data pointing toward the same case in order to conclude. This 70% threshold allows discriminating, if the algorithm detects the presence of a safety net in 70% of the images or not, in function of Case A or B depicted in Figure 4.
This simple threshold filter leads to an interesting classification in either Case A or Case B situations (last line of table II). Another type of filter that could be used is a "two-consecutive measurement" trigger. This would imply to have the memory of the precedent detection result, but could also lead to an even better estimation overall.

B. Distance estimations

In order to validate our assumptions about distance estimation, we computed the estimates of D^{wall} and D^{net} and their standard-deviation (Std) as well as the accuracy (%accuracy)

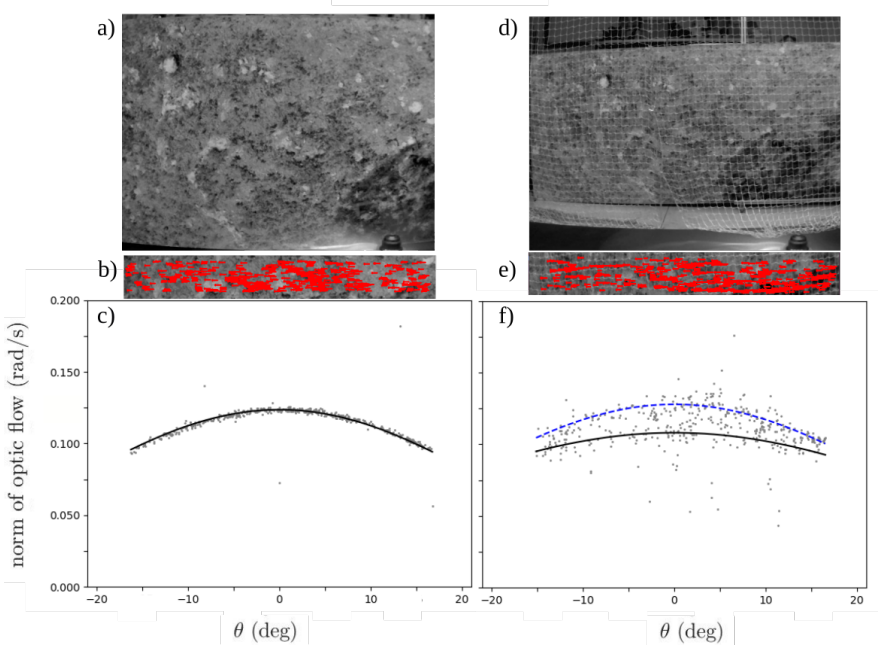

Fig. 6. The images (a) and (d) have given an Optic Flow field along their X axis, shown in (b) and (d) respectively. In case (a) we have $D^{wall} = 2.3m$ and in case (d) we have $D^{wall} = 3.5m$ and $D^{net} = 2.7m$. The Optic Flow measured is represented in red in (b) and (e) for both cases. We can see that the distribution of the norms (c) is way more diverse than in (f) but the algorithm is able to detect 2 clusters who each follows the distribution of a single plane. The presence of both clusters is the signature of two surfaces detected, one in front of the other: the cluster with the largest optic flow magnitude is the closer surface from the drone (eq. 3) and the cluster below with the smallest optic flow magnitude is the farthest surface (f).

Trajectory number :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Case A: simple wall Case B: wall + safety net	A	A	A	A	A	A	B	B	B	B	B	B	B	B	B
Nb of images per set	30	30	30	30	30	30	20	20	20	30	30	30	30	30	30
Nb usable image per sets	25	30	26	28	25	28	18	20	16	28	27	29	28	26	26
Nb net detected per set	0	3	2	3	1	1	17	20	16	28	27	29	27	26	26
% of accurate estimations	100	90	93	89	96	96	94	100	100	100	100	100	97	100	100
Case determination (threshold = 70%)	A	A	A	A	A	A	B	B	B	B	B	B	B	B	B

TABLE II

DETECTION OF SAFETY NET IN CASE (A) AND (B), PERCENTAGES OF GOOD RESULTS AND DETERMINATION OF SAFETY NET SEEN DURING TRAJECTORY.

Traj. number	$mean(D^{wall})$ [m]	$mean(D^{wall}_{est})$ [m]	Std [m]	% accuracy
1	3.29	3.24	0.50	86.9
2	2.74	2.66	0.56	83.5
3	2.81	2.96	0.42	86.7
4	3.15	3.11	0.47	88.8
5	2.68	2.52	0.47	83.7
6	2.48	2.34	0.28	91.5

TABLE III

ESTIMATED DISTANCE D^{wall} VIA NET-DETECTION ALGORITHM DURING 6 TRAJECTORIES WITHOUT NET IN SETUP (CASE A)

with respect to the ground truth (Table IV).
Furthermore, we also have run our algorithm without the presence of net, in order to assess its "general" reliability and the impact of a net on the quality of the measurement (Table III).

As we can see in Table II, the experiment shows that we are able to detect the presence of a safety-net with a very high confidence (two last lines). Furthermore, some errors persist, mainly due to incertitude with the camera optics.

Moreover, the quality of the distance estimation do not change if there is a safety net or not, with in average an accuracy of 86.9% in Case (A) (Table III) while having 86.4% for D^{net} and 87.2% for D^{wall} in Case (B) (Table IV). Furthermore, the errors are centered as shown in Figure 7.
We conclude that our method is able to detect the presence of a safety net between the drone and the wall. Our detection algorithm is also able to estimate concomitantly:

- the relative distance of the safety net with a good precision (std 0.38m)
- distance relative to the background wall behind the net (std 0.50m).

[4]The boxplot command is provided by the Matlab function *boxplot*: https://fr.mathworks.com/help/stats/boxplot.html

Traj. number	$mean(D^{wall})$ [m]	$mean(D^{wall}_{est})$ [m]	Std [m]	% accuracy	$mean(D^{net})$ [m]	$mean(D^{net}_{est})$ [m]	Std [m]	% accuracy
7	3.15	2.95	0.44	88.7	2.35	2.44	0.37	86.0
8	2.94	2.83	0.21	94.0	2.15	2.29	0.21	90.2
9	2.77	2.70	0.45	86.0	1.97	2.19	0.40	80.5
10	3.15	3.07	0.51	88.0	2.35	2.41	0.38	86.2
11	2.94	2.69	0.48	84.0	2.14	2.09	0.31	86.5
12	2.76	2.73	0.45	85.9	1.96	2.06	0.29	87.0
13	3.62	3.42	0.63	86.6	2.82	2.861	0.51	84.8
14	3.18	2.97	0.41	88.1	2.38	2.36	0.28	90.1
15	3.25	3.26	0.64	83.7	2.45	2.58	0.45	86.0

TABLE IV

ESTIMATED DISTANCE D^{wall} AND D^{net} VIA NET-DETECTION ALGORITHM DURING 9 TRAJECTORIES WITH A SAFETY-NET IN SETUP (CASE B)

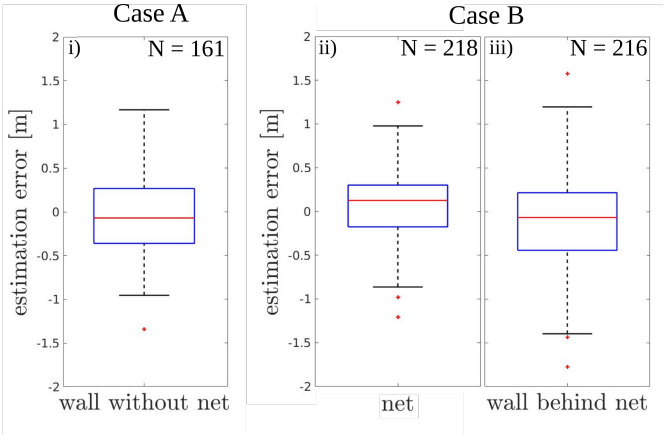

Fig. 7. Boxplot[4]of estimation distance error for three pooled data: i) D^{wall} without any net (left) (Case A), ii) D^{net} (center)(Case B), iii) and D^{wall} with the safety net in between (right) (Case B). As in Tables IV and III, we observed a relatively high Std but low mean error: i) std = 0.47m, error = 0.05m for D^{wall}_{est} without net, ii) std = 0.38m, error = 0.07m for D^{net}_{est} and iii) std = 0.50m, error = 0.13m for D^{wall}_{est} with the safety net in between. In addition, the distance estimation errors are all centered.

IX. CONCLUSION

The detection of a safety net by a drone has been made possible using Optic Flow cues taken from a low-cost global-shutter camera with a small resolution. This could lead to a better and safer flight for a drone, without the need of an operator checking the area for safety net beforehand. Our method could be added in such a workflow in order to avoid safety net and non-opaque obstacles, as already suggested for regular Optic Flow based wall avoidance. Furthermore, with more than 90% confidence on direct measures and 100% in the overall trajectory estimation, we can assume that any net would be detected in a few steps with a limited number of images.

In this study, we have also shown that the observation of the Optic Flow can help the drone determining the shape of their environment, by detecting and extracting information, not only of the presence of a background wall but also of a safety net between the drone and the background.

This development is interesting as the Optic Flow cues provide detection that other visual methods could not.

In a future study, we will show how the Optic Flow based safety net detection algorithm can also be used when the drone is moving using any successive translation and in more complex environment when the surfaces are not parallel. Such detection method could allow UAV to avoid hitting such safety nets, as well as making a coarse visually cartography of the surrounding.

Detection of (underwater) net is also very important for sea-surface vehicle (fishing vessel, sailing, ...) as well as for submarines infiltration and Anti-Submarine Warfare [23], [24].

ACKNOWLEDGMENT

We thank J.M. Ingargiola as well as Amaury Negre (from Gipsa-Lab, Grenoble) for their kind help with the drone software and trajectory generation, as well as helping to improve our piloting skills.

Financial support was provided via a ProxiLearn project grant (ANR-19-ASTR-0009) to F.R. and via SpotReturn project grant (ANR-21-ASRO-0001-02) to T.R. and F.R. from the ANR (Astrid Program).

The participation of X.D. in this research was made possible by the joint PhD grant from the Agence Innovation Défense (AID) and Aix Marseille University. X.D., C.C., T.R. and F.R. were also supported by Aix Marseille University and the CNRS (Life Science, Information Science and Engineering and Science & technology Institutes). The equipments for the experimental tests have been mainly supported by ROBOTEX 2.0 (Grants ROBOTEX ANR-10-EQPX-44-01 and TIRREX ANR-21-ESRE-0015).

REFERENCES

[1] H. Tibebu, J. Roche, V. De Silva, and A. Kondoz, "Lidar-based glass detection for improved occupancy grid mapping," *Sensors*, vol. 21, no. 7, p. 2263, 2021.

[2] X. Zhao, Z. Yang, and S. Schwertfeger, "Mapping with reflection-detection and utilization of reflection in 3D LiDAR scans," in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2020, pp. 27–33.

[3] L. Weerakoon, G. S. Herr, J. Blunt, M. Yu, and N. Chopra, "Cartographer_glass: 2D graph SLAM framework using LiDAR for glass environments," *arXiv preprint arXiv:2212.08633*, 2022.

[4] R. Wang, J. Bach, and F. P. Ferrie, "Window detection from mobile LiDAR data," in *2011 IEEE Workshop on Applications of Computer Vision (WACV)*. IEEE, 2011, pp. 58–65.

[5] H. Ali, C. Seifert, N. Jindal, L. Paletta, and G. Paar, "Window detection in facades," in *14th International Conference on Image Analysis and Processing (ICIAP 2007)*. IEEE, 2007, pp. 837–842.

[6] H. Wei, X.-e. Li, Y. Shi, B. You, and Y. Xu, "Multi-sensor fusion glass detection for robot navigation and mapping," in *2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA)*. IEEE, 2018, pp. 184–188.

[7] M. Recky and F. Leberl, "Windows detection using k-means in cie-lab color space," in *2010 20th International Conference on Pattern Recognition*. IEEE, 2010, pp. 356–359.

[8] H. Mei, X. Yang, Y. Wang, Y. Liu, S. He, Q. Zhang, X. Wei, and R. W. Lau, "Don't hit me! glass detection in real-world scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3687–3696.

[9] M. Kustra and H. Nowakowski, "Counteracting unmanned aerial systems in the operational area of airports," *Journal of KONBiN*, vol. 50, no. 2, pp. 285–293, 2020.

[10] J. R. Serres and F. Ruffier, "Optic flow-based collision-free strategies: From insects to robots," *Arthropod structure & development*, vol. 46, no. 5, pp. 703–717, 2017.

[11] F. Expert and F. Ruffier, "Flying over uneven moving terrain based on optic-flow cues without any need for reference frames or accelerometers," *Bioinspiration & Biomimetics*, vol. 10, no. 2, p. 026003, feb 2015.

[12] G. C. De Croon, J. J. Dupeyroux, C. De Wagter, A. Chatterjee, D. A. Olejnik, and F. Ruffier, "Accommodating unobservability to control flight attitude with optic flow," *Nature*, vol. 610, no. 7932, pp. 485–490, 2022.

[13] L. Bergantin, C. Coquet, J. Dumon, A. Negre, T. Raharijaona, N. Marchand, and F. Ruffier, "Indoor and outdoor in-flight odometry based solely on optic flows with oscillatory trajectories," *International Journal of Micro Air Vehicles*, vol. 15, p. 17568293221148380, 2023.

[14] F. Mumuni, A. Mumuni, and C. K. Amuzuvi, "Deep learning of monocular depth, optical flow and ego-motion with geometric guidance for uav navigation in dynamic environments," *Machine Learning with Applications*, vol. 10, p. 100416, 2022.

[15] C. Vogel, S. Roth, and K. Schindler, "Piecewise rigid scene flow," in *International Conference on Computer Vision*. United States: Institute of Electrical and Electronics Engineers, 2013.

[16] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI'81: 7th international joint conference on Artificial intelligence*, vol. 2, 1981, pp. 674–679.

[17] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis: 13th Scandinavian Conference, SCIA 2003 Halmstad, Sweden, June 29–July 2, 2003 Proceedings 13*. Springer, 2003, pp. 363–370.

[18] J. D. Adarve and R. Mahony, "A filter formulation for computing real time optical flow," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1192–1199, 2016.

[19] R. Vassallo, J. Santos-Victor, and H. Schneebeli, "A general approach for egomotion estimation with omnidirectional images," in *Proceedings of the IEEE Workshop on Omnidirectional Vision 2002. Held in conjunction with ECCV'02*, 2002, pp. 97–103.

[20] S. Jianbo and C. Tomasi, "Good features to track," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[21] P. Corke, *Robotics, Vision and Control*. Springer, 2017.

[22] J. J. Koenderink and A. J. van Doorn, "Facts on optic flow," *Biological cybernetics*, vol. 56, no. 4, pp. 247–254, 1987.

[23] J. C. Pittman, "Zone defense–anti-submarine warfare strategy in the age of littoral warfare," Army Command and General Staff Coll Fort Leavenworth KS, Tech. Rep., 2008.

[24] J. S. Breemer, "Anti-submarine warfare: a strategy primer." Naval Postgraduate School Monterey CA, 1988.