Telemetry-based search window correction for airborne tracking

Pau Climent-Pérez[1], Georgios Lazaridis[1], Georg Hummel[2], Martin Russ[2], Dorothy N. Monekosso[1], and Paolo Remagnino[1]

[1] Robot Vision Team (RoViT), Faculty of Science, Engineering and Computing
Kingston University London, Penrhyn Road Campus
KT1 2EE Kingston upon Thames, Greater London, UK
{P.Climent,D.Monekosso,P.Remagnino}@kingston.ac.uk
[2] Institute of Flight Systems, University of the Bundeswehr Munich
Werner-Heisenberg-Weg 39, 85577 Neubiberg, Germany
{georg.hummel,martin.russ}@unibw.de

Abstract. Tracking from airborne cameras is very challenging, since most assumptions made for fixed cameras do not hold. Therefore, compensation of platform ego-motion is seen as a necessary pre-processing step. Most existing methods perform image registration or matching, which involves costly image transformations, and have a restricted operational range. In this paper, a novel ego-motion compensation approach is presented, that transforms the local search window of the visual tracker. This is much more computationally efficient, and can be applied regardless of the amount of texture in the background. Experiments with ground truth and tracker output data are conducted and show the validity of the approach.

1 Introduction

Recent development in embedded technologies has lead to an increasing use of video analysis using unmanned aerial vehicles (UAVs) equipped with cameras for applications such as agriculture and natural preservation [1], traffic monitoring [2], or emergency response [3].

Using an aircraft platform introduces noise such as vibrations, rocking, locomotion leading to a highly dynamic and constantly changing image background. Therefore compensation of platform ego-motion is necessary before commencing with the actual tracking. Most existing techniques tackle the problem by applying image stabilisation [2], camera pose estimation [4] or image matching (or registration) [5, 6]. These techniques are costly, as compared to the proposed approach, and have a restricted operational range. In the following, these video-only, hybrid, and telemetry-based methods are further discussed. After that, a novel ego-motion compensation approach for local search windows is presented. To the best of our knowledge, no previous work introduces the use of telemetry and video data in combination to transform the local search window of an object tracker as a more efficient ego-motion compensation method that does not require image transformations.

1.1 Video-based homography estimation

Homography estimation has been extensively used for many different applications in the field of computer vision, and specifically, it has been used in moving vehicles, both terrestrial and aerial, as well as robots, for the compensation of the motion of the vehicle (or ego-motion), this pre-processing step allows for frame differencing to be calculated, and as such, it allows many fixed-camera methods (and assumptions) to be employed.

A very common approach among the studied works is to perform an 'image registration' or 'image correspondence', in order to calculate the homography between consecutive frames [7–9]. These ideas are based on the extraction of features from both images, via an interest point detection algorithm (e.g. Kanade-Lucas-Tomasi's 'good features', or others). Once the points have been extracted, a correspondence among them is found, and the most common step after that is to apply a random sample consensus (RANSAC) algorithm to eliminate outliers. By doing so, an homography matrix can be estimated, which encodes the translation and rotation of the aerial vehicle between two frames. In some works this idea is extended, and maps of the explored area are created, via 'image stitching' or 'mosaicing' techniques [10] or simultaneously locating the vehicle and mapping the scene (UAV SLAM) [11].

However, homography-based techniques have a major drawback, since they are based on interest points, and these can only be found in places of the image that are rich in texture (and therefore corners). An ill-posed case occurs when the only available texture is that of the moving objects (due to an homogeneous background), the homography matrix estimation procedure will take into consideration the motion of the ground object, and as such, the obtained matrix will not be representative of the vehicle's motion. Texture-poor backgrounds are common when the UAV is flying close to the ground over an individual, or when flying over a large group of people or a crowd (e.g. in a demonstration or rally) where static elements of background might not be visible, or might constitute the minority of the matched points, thus wrongly being classified as outliers.

1.2 Telemetry-based approaches

With the advent of more accurate and readily available global positioning and inertial navigation systems (GPS/INS), another body of research has pivoted in their favour for the calculation of the homography between frames, especially since all UAVs are equipped with such systems, and thus the only requirement is for their data to be available. Applications, include mapping or SLAM [12], frame differencing for background modelling [13], and target geo-location [14, 15].

In this last application, translation from the image coordinates to the geographical coordinates is thus made possible, and allows for the target to be located in a frame that is different from that of the sensor. Furthermore, it needs to be noted that the computational overhead introduced by image correspondence and homography matrix estimation is not negligible. This raises the question of the real necessity of homography estimation as a pre-requisite for further video processing.

Several of the studied works make an assumption about the orthogonality of the camera's axis to the ground plane, where the UAV is hovering over the plane, and thus the roll (ψ) and pitch (θ) angles are near-to-zero all the time [3, 16]; as well as the assumption that at enough distance from the ground, the surface inside the field of view (FOV) of the camera, is planar, i.e. 'planarity assumption' [9, 12, 13, 17]. This allows the $3D$ problem to be constrained to a plane ($2D$). By doing so, the calculations can be simplified. Besides, in these works the camera is assumed to be co-axial with the center of mass of the UAV, and the offset in position between the GPS/INS devices and the camera is negligible, and the FOV angles are known.

The rest of this paper is organised as follows: in Section 2, the proposed method for correcting the search window is presented; Section 3 explains the experiments conducted to validate the approach; in Section 4 results are presented, conclusions on the presented method are drawn, and ideas for future work are presented.

2 Methodology

This paper presents a novel search window correction method to facilitate tracking from UAVs, based on the motion of the aerial vehicle. Homography estimation is shown unnecessary, because of its computational overhead. Our method applies transformation operations on the 'search window' from one frame to another.

Since the UAV is assumed to be an octo-copter (see Section 3 for details), the planarity and orthogonality assumptions can easily be made. This, along with knowledge of camera parameters (such as FOV angles, see Fig. 1), constrains the problem to a $2D$ plane which facilitates the math.

The visual tracker employed, is also assumed to require the use of a local search window. Once the UAV is on flight, a human operator can see the camera output, and decide on the rectangle of interest (ROI) enclosing a target of choice. The local search window (win) is then defined as a rectangle, around the ROI, with an allowance or margin where the target might be re-detected in subsequent frames (to see specifics about how the margin is set, refer to Sec. 3). At this point, the presented method recalculates the position of the local search window in the next frames, based on the movement of the camera mounted on the UAV. The proposed algorithm first obtains all current camera (vehicle) pose parameters, in the form of a tuple:

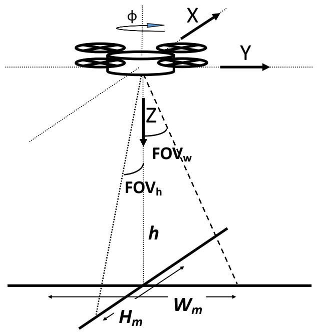

Fig. 1. The employed UAV octo-copter platform (left); and a schematic of the UAV hovering over the ground plane. The two FOV angles are depicted, as well as the UAV coordinate system, and W_m and H_m ground distances (right).

$$d_{telemetry} = [(\varphi, \lambda),\ (\psi, \theta, \phi),\ h]\ , \tag{1}$$

where the pair (φ, λ) represents the latitude and longitude in degrees from Equator and Greenwich meridian, respectively; ψ, θ, and ϕ represent the roll, pitch and yaw angles in degrees, respectively; and h represents the current altitude (in meters) from the ground and is calculated as:

$$h = h_{(\mathrm{ASL},t)} - h_{(\mathrm{ASL},0)}\ , \tag{2}$$

where $h_{(\mathrm{ASL},0)}$ is the initial ground altitude above the sea level (ASL) of the UAV before take-off, and $h_{(\mathrm{ASL},t)}$ is the current ASL from the telemetry reads. For every frame in the video feed, the tracker provides a ROI enclosing the tracked person, and a wider local search window (win) is calculated around it. The search window is expressed in pixels, with a coordinate pair that represents its upper-left corner (win_x, win_y), and its size (win_w, win_h). Once this information has been gathered, it is important to analyse which changes in the pose of the camera influence the most in the apparent motion of the search window, taking into consideration the platform type (copter). Three different aspects are found to affect the window's apparent motion the most:

– The translation of the UAV along the X and Y axes ($\propto (\varphi, \lambda)$, because of the assumption of orthogonality introduced earlier),
– the translation of the UAV along the Z axis (changes in its height, or h), and
– the rotation of the UAV about the Z axis (changes in its yaw, or ϕ).

Here the X, Y, Z axes are in the vehicle frame, that is, the X axis crosses the UAV from back to front, the Y axis crosses the UAV from left to right, and the Z axis crosses the vehicle from top to bottom and across the camera centre. As it can be observed, the roll and pitch angles (ψ, θ) are not employed, because of the assumptions introduced earlier. It can also be observed that there is a discrepancy both in units (degrees, meters, pixels) as well as in coordinate systems (GPS, UAV and image) employed. For this reason, we introduce a common framework, and express all geo-location data in meters, and the yaw (φ) is now expressed in radians.

For the conversions of the latitude and longitude data (WGS-84 standard) provided by the inertial measurement unit (IMU), a simplified version of the Universal Transverse Mercator (UTM) conformal projection coordinate system is used. UTM is a cylindrical projection separating the surface of the earth in 6 degree wide zones. Therefore a position of an object is given in a zone, a band (or an hemisphere), the *northing*, and the *easting* value. Within a zone, a Cartesian coordinate system is used with with the northing and easting values expressed in meters. The easting is given from the zone's initial easting and the northing is given from the Equator.

To convert the pixel positions, the information about the vehicle's height and its camera's FOV are employed (see Fig. 1). This allows the width and height (in meters) of the area covered by the camera to be estimated with trigonometric rules. Furthermore, a ratio r_c can establish the conversion between pixels and meters:

$$\begin{gathered}(W_m,\ H_m) = (2h \cdot \tan \mathrm{FOV}_w,\ 2h \cdot \tan \mathrm{FOV}_h)\ [\mathrm{m}]\ , \\ r_c = W_m / W_{img}\ \ [\mathrm{m/pixels}]\ \ . \end{gathered} \tag{3}$$

Correction due to XY translation. Using all the above data, three different operations can be performed, in order to obtain the corrected search window (win'). First, the translation of the UAV over the surface of the world is calculated as the difference in northing and easting values:

$$\Delta_N = N_t - N_{t-1} \quad \text{and} \quad \Delta_E = E_t - E_{t-1}\ . \tag{4}$$

These values are then used for the correction of the rectangle in the image space, the corrections needed are calculated as:

$$\begin{aligned} win'_x &:= win_x + \delta x \quad \text{where} \quad \delta x = \Delta_E \cdot r_c^{-1}\ , \\ win'_y &:= win_y + \delta y \quad \text{where} \quad \delta y = -\Delta_N \cdot r_c^{-1}\ . \end{aligned} \tag{5}$$

Please note the difference in the calculation of δx and δy, due to the change in coordinate systems.

Correction due to altitude changes. The second operation that is performed on the local search window, is related to its size. Due to the changes in altitude (h) of the UAV between frames, the apparent size of the target in the image changes, and as such, the local search window around the target must grow or shrink accordingly, so that an optimal size is maintained. To proceed, a ratio (r_h)

among the altitude (h) values in the current and previous frame is calculated, and used as a factor to resize the local search window:

$$(win'_w,\ win'_h) := (win_w \cdot r_h,\ win_h \cdot r_h) \quad \text{where} \quad r_h = \frac{h_{t-1}}{h_t}\ . \tag{6}$$

Correction due to the yaw (ϕ) changes. In this last operation, the local search window is corrected to compensate for variations of the rotation on the Z axis of the UAV coordinate system (yaw or ϕ). Changes in yaw occur when the vehicle steers either when hovering over an area, or in conjunction with a translation in the XY axes. To apply this correction, the position of the new window is calculated based on the difference between the current and previous yaw values ($\varDelta_\phi$). First, the central position of the window is needed ($w_x,\ w_y$); then, this point is expressed a vector from the center of the screen ($c_x,\ c_y$); after that, the rotation over $\varDelta_\phi$ is applied over that vector, to counteract the rotation undergone by the vehicle; finally, the coordinates are translated back to have their reference back to the top-left corner of the image, as originally:

$$\begin{aligned}
(w_x,\ w_y) &= (win_x + win_w/2,\ win_y + win_h/2)\\
(c_x,\ c_y) &= (w_x - W_{img}/2,\ w_y - H_{img}/2)\\
(c'_x,\ c'_y) &= (-c_x \cdot \cos\varDelta_\phi + c_y \cdot \sin\varDelta_\phi,\ -c_y \cdot \sin\varDelta_\phi - c_y \cdot \cos\varDelta_\phi)\\
(w'_x, w'_y) &= \left(W_{img}/2 - c'_x,\ H_{img}/2 - c'_y\right)\\
(win_x,\ win_y) &= \left(w'_x - win_w/2,\ w'_y - win_h/2\right)
\end{aligned} \tag{7}$$

Please note the inverted symbols in the third line in (7), since what is intended is to revert or counteract the effect of the ego-motion, and $\varDelta_\phi$ represents its magnitude.

3 Experimentation

A self-constructed vertical take-off and landing (VTOL) platform with eight electric propelled motors [18] was employed to conduct the experiments (see Fig. 1). The OctoXL is based on a construction Kit from HiSystems GmbH. It is equipped with an embedded computer board with Intel Core i7 processor, a SSD Harddrive and a VRMagic camera with a resolution of 752×480 pixel at 15 fps. The utilised Lensagon lens has 3.5mm focal length. The UAV flies at an altitude of 10 to 15 m above ground. The aircraft is equipped with a Xsens MTI-G IMU with 2.5 m position accuracy and $0.5°$ angular accuracy. In the Scenario the aircraft surveys an area populated with seven people to simulate regular urban pedestrian movement. The pilot operating the UAV creates ego-motion in lateral and angular directions by trying to cover the search area. The computer board on-board records images and telemetry data in synch to provide them for later processing using a distributed data system (DDS) [19].

Definition of the dataset. The recorded data, acquired with the OctoXL, has been divided into several sequences to form a dataset. From the original capture, 6144 frames long, several sequences have been selected, most around 300 frames (see Table 1 for details or Fig. 2 for sample frames). Sequences have then been named after the most prominent colour of the clothing of the person to track. Manual annotations on the position of that person are given for all frames in all sequences.

Conducted experiments. The presented approach is validated by three different experiments using the previously recorded data. The first one is a validation method that uses ground truth data from the tracking bounding boxes in order to determine the overlap of local search windows with the actual tracking target at any given time. The second experiment tests the presented approach in conjunction with a visual tracker (Covariance Tracker [20, 21]). Finally, the third experiment is conducted using the tracker without any correction for comparison purposes (i.e. *baseline* results). In all three cases, the search window sides are set to be twice as big as those of the tracked object ROI.

Validation using Ground truth. In the first experiment, the aim is to test how the local search window correction method performs by itself. To do so, ground truth data is used. The ground truth has been manually annotated for all sequences in the dataset, and is used to provide the 'real' ROI ($win_g t$) for any frame, its corresponding window is then calculated to compare against the window estimated by the correction algorithm (win_e). After that, the estimated window (for frame t) and the window generated from the ground truth (frame $t+1$) are compared using the overlap measure or Jaccard index:

$$J = \frac{win_e \cap win_{gt}}{win_e \cup win_{gt}} \ . \tag{8}$$

A novel measure, named the *C-measure*, is also employed. This measure is similar to the overlap, but as opposed to it, it is used to tell how well contained (therefore the C) within the local search window is the tracked object ROI. Its definition is as follows:

$$C = \frac{\text{ROI} \cap win_e}{\text{ROI}} \ . \tag{9}$$

The logic behind the C-measure is that if the ROI is fully contained within the local search window, the tracker will have the chance to find it much better than if it is partially outside its scope (the local search window). The top part of the fraction will be the full size of the box if the box is fully contained within the window; the bottom part makes the measure be in the range of $[0, 1]$.

Tracking with correction. In the second experiment, the goal is to test the proposed method in a real situation. For this, a covariance tracker with a local search window is employed. The search window is corrected at each frame using the proposed method. In this case, the overlap measure between the detected object and the ground truth is estimated, and used as a measure of tracking quality.

4 Results and Conclusions

Table 1 shows quantitative results from the experiments introduced in Section 3. The overlap measure is given for all experiments. For the validation with ground truth data, the C-measure scores are presented alongside. In the two experiments where a tracker is used, the Pascal overlap criterion [22] is employed. With this criterion, a match is said to be such only if the overlap is greater than 50%. The presented results are an average over the whole sequence of the accomplishment of this criterion at each frame. Figure 2 shows some qualitative results.

Table 1. Results ($\bar{x} \pm \sigma$) for the conducted experiments.

Sequence	**Length**	**Validation with GT**		**Tracker w/ correction**		**Baseline (no correction)**	
		Overlap$^{\%}$	**C-measure**$^{\%}$	**Overlap**$^{\%}$	**Pascal**$^{\%}$	**Overlap**$^{\%}$	**Pascal**$^{\%}$
white	668	85.1 ± 7.7	99.3 ± 3.2	48.2 ± 15.4	$68.3 \pm 17.1^*$	15.8 ± 12.9	20.6 ± 23.8
white_s	161	77.1 ± 7.9	98.7 ± 3.5	61.6 ± 11.8	$82.1 \pm 24.7^*$	21.1 ± 16.3	27.2 ± 20.4
black	391	82.9 ± 8.8	99.8 ± 1.6	64.6 ± 8.4	$88.8 \pm 24.5^*$	53.1 ± 17.0	72.4 ± 20.1
red	390	88.9 ± 6.4	100.0 ± 0.5	44.7 ± 5.8	18.7 ± 24.4	44.8 ± 5.8	$19.7 \pm 24.1^*$
blue	303	86.6 ± 7.3	99.9 ± 1.3	41.4 ± 8.6	14.5 ± 25.0	44.2 ± 8.1	$26.6 \pm 20.7^*$
white1	365	84.6 ± 4.8	100.0 ± 0.4	29.6 ± 10.4	9.6 ± 23.8	18.6 ± 12.8	$9.8 \pm 24.0^*$
black1	102	89.5 ± 4.6	99.9 ± 0.7	61.4 ± 7.4	80.6 ± 24.3	61.6 ± 7.6	$85.4 \pm 25.1^*$
blue1	316	85.5 ± 6.6	100.0 ± 0.4	27.1 ± 9.5	$10.4 \pm 24.2^*$	27.7 ± 9.3	10.4 ± 24.2
blue2	354	87.5 ± 7.6	100.0 ± 0.8	63.0 ± 8.9	$80.6 \pm 24.2^*$	47.6 ± 14.9	65.1 ± 14.4
mean	338.8	85.3 ± 6.8	99.7 ± 1.4	49.1 ± 9.6	50.4 ± 23.6	37.2 ± 11.6	37.5 ± 21.9

$^{\%}$ denotes values are expressed in percent.
* denotes best Pascal value.

Analysing the results from the first experiment, it can be seen that the C-measure is next to a 100% in most cases, with very low deviation. This means that the proposed method successfully keeps the object within the local search window, and therefore it fulfils its main goal, regardless of the tracking method.

With regards to the comparison between the baseline and the proposed correction method, several aspects need to be noted. First, the generally low values for the overlap are due to the strictness of this measure, which heavily penalises false negatives and false positives. This is a problem for the covariance tracker, since its authors used a much more relaxed measure for its evaluation (any detection within a 9×9 window of the ground truth centroid was considered as a *match* [20, 21]). Therefore, such low values should not be discouraging.

From the 'Tracker with correction' experiment, it can be observed that in general terms, the performance is better when the correction is used. In the best cases, the improvement factor is greater than 3 (3.31 for the `white` sequence, for instance). There are also some other sequences where the baseline tracking performs on a par with the corrected tracking (factor is ≈ 1; for instance in the `white1`, `red`, or `blue1` sequences). The reason for this can be explained by the

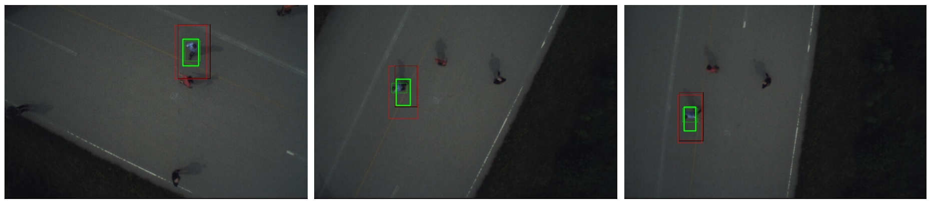

Fig. 2. Frames 2220, 2240, and 2260 from the '`blue`' sequence. In 40 frames, the UAV undergoes a severe rotation about the Z axis. The proposed method successfully tracks the target (green) with a reduced search window size (red).

nature of the sequences, where the UAV's movements are smoother or slower than in other videos. Finally, there are some sequences where the proposed correction method actually disadvantages the tracker with respect to the baseline. The causes for this require further investigation, and is left as future work. Search window size variation requires further analysis as well.

To summarise, in this work, a novel method for the correction of a local search window has been proposed. Validation with ground truth data showed the validity of the method. However, when using a tracker other factors need to be taken into account, such as its accuracy when calculating the size of the detected object, or the loss of track due to sudden changes in the appearance model.

Acknowledgment This work has been supported by the European Commission's Seventh Framework Programme (FP7-SEC-2011-1) under grant agreement № 285320 (PROACTIVE project).

References

1. Turner, D., Lucieer, A., Watson, C.: An automated technique for generating georectified mosaics from ultra-high resolution unmanned aerial vehicle (uav) imagery, based on structure from motion (SfM) point clouds. Remote Sensing **4** (2012) 1392–1410
2. Shastry, A., Schowengerdt, R.: Airborne video registration and traffic-flow parameter estimation. Intelligent Transportation Systems, IEEE Transactions on **6** (2005) 391–405
3. Andriluka, M., Schnitzspan, P., Meyer, J., Kohlbrecher, S., Petersen, K., Von Stryk, O., Roth, S., Schiele, B.: Vision based victim detection from unmanned aerial vehicles. In: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on. (2010) 1740–1747
4. Sharp, C., Shakernia, O., Sastry, S.: A vision system for landing an unmanned aerial vehicle. In: Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on. Volume 2. (2001) 1720–1727
5. Reinartz, P., Lachaise, M., Schmeer, E., Krauss, T., Runge, H.: Traffic monitoring with serial images from airborne cameras. ISPRS Journal of Photogrammetry and Remote Sensing **61** (2006) 149–158

6. Hummel, G., Kovács, L., Stütz, P., Szirányi, T.: Data Simulation and Testing of Visual Algorithms in Synthetic Environments for Security Sensor Networks. In Aschenbruck, N., Martini, P., Meier, M., Tölle, J., eds.: Future Security. Volume 318 of Communications in Computer and Information Science., Springer Berlin Heidelberg (2012) 212–215
7. Remagnino, P., Brand, P., Mohr, R.: Correlation techniques in adaptive template matching with uncalibrated cameras. Volume 2356. (1995) 252–263
8. Yuan, C., Recktenwald, F., Mallot, H.: Visual steering of uav in unknown environments. In: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on. (2009) 3906–3911
9. Plinval, H., Morin, P., Mouyon, P., Hamel, T.: Visual servoing for underactuated vtol uavs: a linear, homography-based framework. International Journal of Robust and Nonlinear Control (2013) 1–24
10. Nemra, A., Aouf, N.: Robust feature extraction and correspondence for uav map building. In: Control and Automation, 2009. MED '09. 17th Mediterranean Conference on. (2009) 922–927
11. Caballero, F., Merino, L., Ferruz, J., Ollero, A.: Unmanned Aerial Vehicle Localization Based on Monocular Vision and Online Mosaicking. Journal of Intelligent and Robotic Systems **55** (2009) 323–343
12. Heredia, G., Caballero, F., Maza, I., Merino, L., Viguria, A., Ollero, A.: Multi-unmanned aerial vehicle (uav) cooperative fault detection employing differential global positioning (dgps), inertial and vision sensors. Sensors **9** (2009) 7566–7579
13. Zhang, S.: Object tracking in unmanned aerial vehicle (uav) videos using a combined approach. In: Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on. Volume 2. (2005) 681–684
14. Barber, D., Redding, J., McLain, T., Beard, R., Taylor, C.: Vision-based target geo-location using a fixed-wing miniature air vehicle. Journal of Intelligent and Robotic Systems **47** (2006) 361–382
15. Dobrokhodov, V., Kaminer, I., Jones, K., Ghabcheloo, R.: Vision-based tracking and motion estimation for moving targets using small uavs. In: American Control Conference, 2006. (2006) 1428–1433
16. Teutsch, M., Kruger, W.: Detection, Segmentation, and Tracking of Moving Objects in UAV Videos. 2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance (2012) 313–318
17. Caballero, F., Merino, L., Ferruz, J., Ollero, A.: Improving vision-based planar motion estimation for unmanned aerial vehicles through online mosaicing. In: Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on. (2006) 2860–2865
18. Russ, M., Schmitt, M., Hellert, C., Stütz, P.: Airborne sensor and perception management: Experiments and Results for surveillance UAS. AIAA Infotech@Aerospace (I@A) Conference (2013) 1–16
19. Böhm, F., Schulte, A.: UAV Autonomy ResearchChallenges and advantages of a fully distributed system architecture. ITC 2012, International Telemetering Conference (2012) 1–10
20. Tuzel, O., Porikli, F., Meer, P.: Region covariance: A fast descriptor for detection and classification. In Leonardis, A., Bischof, H., Pinz, A., eds.: Computer Vision ECCV 2006. Volume 3952 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2006) 589–600
21. Porikli, F., Tuzel, O., Meer, P.: Covariance tracking using model update based on lie algebra. In: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on. Volume 1., IEEE (2006) 728–735

22. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. International journal of computer vision **88** (2010) 303–338