2D VISUAL TRACKING FOR SPORTS UAV CINEMATOGRAPHY APPLICATIONS

Orestis Zachariadis[⋆] *Vasileios Mygdalis*[⋆] *Ioannis Mademlis*[†] *Nikos Nikolaidis*[†] *Ioannis Pitas*[⋆ †]

[⋆] Department of Electrical and Electronic Engineering, University of Bristol, UK
[†] Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece

ABSTRACT

In this paper, we provide a preliminary study of basic requirements for autonomous UAV cinematography via 2D target tracking. Our contribution is two-fold. First, we develop a mathematical framework so as to determine hardware camera requirements (specifically, focal length), on a representative case study, i.e., orbiting a still or moving target. Second, we examine the on-board software requirements in order to successfully achieve autonomous target following. To this end, we evaluate the performance of state-of-the-art real-time 2D visual trackers in videos captured by commercial drones. Overall, it was found that state-of-the-art 2D visual trackers are dependable and fast enough to be used in drone cinematography, particularly when combined with periodic target re-detection. A proposed variant of the Staple tracker achieved the best balance between real-time performance and tracking accuracy, on a dataset composed of 31 sports videos recorded by commercial drones.

Index Terms— UAV cinematography, video tracking, target following, target tracking, UAV shot types

1. INTRODUCTION

Unmanned Aerial Vehicles ("UAVs", or "drones"), are a recent addition to the cinematographer's arsenal. By exploiting their agility and ability to fly, drones are potentially able to capture video streams that would be impossible otherwise. One of the most demanding media production applications is the aerial coverage of live outdoor (e.g., sports) events, since it mainly involves filming multiple moving targets (e.g., athletes, boats, cars etc.). The drones must capture stable, noncluttered and visually pleasing AV streams, which requires demanding drone piloting skills or impressive drone intelligence. Therefore, there is an increased need of defining the potentials and limitations of drone application to cinematography.

The fundamental principle for autonomous UAV moving target shooting is the drone's ability to perceive and manipulate information about the target position/orientation in the 3D space. Towards this end, advanced vision-based methods [1, 2, 3, 4, 5, 6, 7] can be employed. The overall approach is the following. First, the target is detected/tracked in a video stream captured by a drone camera, over a series of successive video frames. The relative target position in the 3D space is estimated by using the 2D coordinates of the detected/tracked moving target and, thereby, the drone makes a decision to move itself or the camera gimbal. The whole procedure should be executed fast enough in order to maintain the target in the camera field of view. Although such algorithms have already been implemented in commercial drones, their reliability is subject to constraints that have not been investigated thoroughly in the past, thus making updated theoretical analysis (e.g., [3]) necessary. Theoretical constraints that should be taken under consideration include the drone (and gimbal) maneuverability, as well as the drone's target perception capability.

This paper is structured as follows. In Section 2, we describe the related work in 2D visual tracking, as well as our proposed implementation for a UAV platform. In 3, we analyze the theoretical foundation of the orbitting movement, as a representative case study. Conducted experiments on 2D tracking are analyzed in Section 4 and finally, our conclusions are drawn in Section 5.

2. 2D TRACKING OVERVIEW

The most successful and recent 2D tracking approaches employ the tracking-by-detection approach [8, 9, 10]. That is, a discrimination model $\boldsymbol{w}$ is learned incrementally within the successive frames, and detects/recognizes the target ROI in the next frame. This discrimination model can be viewed as a Ridge Regression problem:

$$\min_{\boldsymbol{w}} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2 + \lambda \|\boldsymbol{w}\|^2, \qquad (1)$$

where $\boldsymbol{x}_i$ are the samples and y_i their regression targets, and λ a regularization parameter. This problem has the following closed-form solution [11]:

$$\boldsymbol{w} = (\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^T \boldsymbol{y}, \qquad (2)$$

where each row of $\boldsymbol{X}$ contains the samples $\boldsymbol{x}_i$ and $\boldsymbol{y}$ contains the elements y_i, and $\boldsymbol{I}$ is an identity matrix of appropriate

This project has received funding from the European Union's European Union Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE).

dimensions. However, in order to properly train a tracker, besides using the actual target ROI in the search region, samples resulted after circular shifts of the target ROI inside the tracker search region, should be employed as well. By using a dense sampling strategy, it induces the periodic assumption of the local image patches. Therefore, when good sampling practices are followed, the data matrix $\boldsymbol{X}$ is circulant [10], and can be expressed as $\boldsymbol{X} = C(\boldsymbol{x})$. As a circulant matrix, it has the following property:

$$\boldsymbol{X} = \boldsymbol{F}^H diag(\boldsymbol{F}\boldsymbol{x})\boldsymbol{F}, \tag{3}$$

where $\boldsymbol{F}$ is the so-called DFT matrix, and $\boldsymbol{F}^H$ is its Hermitian transpose. By replacing (3) in (2), we obtain:

$$\hat{\boldsymbol{w}}^* = \frac{\hat{\boldsymbol{x}}^* \odot \hat{\boldsymbol{y}}}{\hat{\boldsymbol{x}}^* \odot \hat{\boldsymbol{x}} + \lambda}, \tag{4}$$

where $\odot$ denotes element-wise operations, $\hat{\boldsymbol{w}}$ is the FFT transform of $\boldsymbol{w}$, and $\hat{\boldsymbol{w}}^*$ is the complex-conjugate of $\hat{\boldsymbol{w}}$, and the same equivalent notation is used for $\boldsymbol{x}$ and $\boldsymbol{y}$.

Therefore, for a test feature patch $\boldsymbol{z}$, we calculate its correlation map $\hat{\boldsymbol{r}}$ in the frequency domain for calculations purposes:

$$\hat{\boldsymbol{r}} = \hat{\boldsymbol{w}} \odot \hat{\boldsymbol{z}}, \tag{5}$$

and the inverse FFT of $\hat{\boldsymbol{r}}$ is the correlation map in the spatial domain, i.e., our detected object.

In order for the correlation filter to be able to adapt to occlusions, changes to target scale/pose/angle, illumination changes etc., the filter is required to be updated, ideally in a per frame basis. As have been proposed by [12], the Minimum Output Sum of Squared Error (MOSSE) filter is updated as follows:

$$\hat{\boldsymbol{w}}_f = \frac{A_f}{B_f}, \tag{6}$$

$$A_f = (1-h)A_{f-1} + h\left(\hat{\boldsymbol{x}}_f^* \odot \hat{\boldsymbol{y}}_f\right) \tag{7}$$

$$B_f = (1-h)B_{f-1} + h\left(\hat{\boldsymbol{x}}_f^* \odot \hat{\boldsymbol{x}}_f + \lambda\right), \tag{8}$$

where $0 < h \leq 1$ is the so-called learning rate, which is in essence a balance between learning from the last frame and forgetting all the previous frames.

In general, correlation-based trackers mostly differ on the issue of representing the initial ROI, i.e., by employing different feature descriptors. fHOG features, color histograms, deep neural representations have all been tried and tested [13]. The DSST tracker [8] operates by extracting fHOG features from the initial and target ROIs. This is applied to both translation and scale tracking. The Staple tracker [9] is an extension of DSST that exploits both fHOG and color histogram features. The different descriptor types are fused in a weighted fashion, to formulate a correlation filter that considers them both during its training. Finally, our proposed Staple2 implementation combines all the above approaches and is described below.

2.1. Proposed Staple2 implementation

Since our implementation was based on the Staple tracker [9], we refer to our proposed method as Staple2, hereafter. Since we focus on UAV tracking, we have introduced elements of the FAST [1] tracker to Staple. That is, we have introduced the Peak-to-Sidelobe-Ratio metric to estimate the tracking quality (for the correlation map extracted by the Staple tracker), that is computed as follows: $PSR = \frac{max(\boldsymbol{r}) - mean(\boldsymbol{r})}{std(\boldsymbol{r})}$. Depending on the quality, we have defined 2 thresholds t_1 and t_2, where $t_1 > t_2$, leaving 3 states: a) $PSR > t_1$, the algorithm tracks and updates its discrimination model, b) $t_1 > PRS > t_2$, tracking is allowed, but the discrimination model is not updated and c) $PSR < t_2$, the target is re-detected in the entire frame with the discrimination model learnt by the tracker. Overall, Staple2 is more robust and also faster, since the model is not updated in every single frame.

3. CASE STUDY: ORBITTING A TARGET

A very common scenario in UAV cinematography is analyzed below as a representative case study, i.e., orbiting a still or moving target (ORBIT). In this case, the camera gimbal is slowly rotating, so as to always keep the still or linearly moving target properly framed, while the UAV (semi-)circles around the target and, simultaneously, follows the latter's linear trajectory (if any) [14, 15]. Given a camera frame-rate of T, time t is discrete, non-negative and proceeds in steps of $\frac{1}{T}$ seconds. $t = 0$ indicates the start of an ORBIT shooting session. At each time instance t, the 3D positions of the UAV ($\tilde{\mathbf{X}}_t = [\tilde{x_{t1}}, \tilde{x_{t2}}, \tilde{x_{t3}}]^T$) and the target ($\tilde{\mathbf{P}}_t = [\tilde{p_{t1}}, \tilde{p_{t2}}, \tilde{p_{t3}}]^T$), as well as an estimated 3D target velocity vector ($\tilde{\mathbf{U}}_t = [\tilde{u_{t1}}, \tilde{u_{t2}}, \tilde{u_{t3}}]^T$), are known in a fixed, orthonormal, right-handed World Coordinate System (WCS), with its z-axis vertical to a local tangent plane, parallel to the sea level (hereafter called "ground plane").

Additionally, at each time instance t, a current, orthonormal, right-handed target-centered coordinate system (TCS) is defined. Its origin lies on the current target position, its z-axis is vertical to the ground plane and its x-axis is the $\mathcal{L}_2$-normalized projection of the current target velocity vector onto the ground plane. In case of a still target, the TCS x-axis is defined as parallel to the projection of the vector $\tilde{\mathbf{P}}_0 - \tilde{\mathbf{X}}_0$ onto the ground plane. In both coordinate systems, the $x - y$ plane is parallel to the ground plane and the z-component is called "altitude". Below, vectors expressed in TCS are denoted without the tilde symbol (e.g., $\mathbf{X}_t$, $\mathbf{P}_t$ and $\mathbf{U}_t$). Due to the way TCS is defined, it holds that $u_{t2} = 0$. The 3D scene point at which the camera looks at time instance t is denoted by $\mathbf{L}_t$ (in TCS), thus defining the "LookAt vector" $\mathbf{O}_t = \mathbf{L}_t - \mathbf{X}_t$. $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are the TCS axis unit vectors (corresponding to the x-axis, the y-axis and the z-axis, respectively).

During shooting an ORBIT, a current, orthonormal, right-handed target-centered coordinate system (TCS) is defined, in which the UAV altitude remains constant, but may vary in the World Coordinate System (WCS). The parameters that must be specified are the desired 3D Euclidean distance $\lambda_{3D} = \|\tilde{\mathbf{X}}_t - \tilde{\mathbf{P}}_t\|_2 = \|\mathbf{X}_t\|_2$ (constant over time), the angle of the entire rotation to be performed around the target (θ) and the desired UAV angular velocity ω. Additionally, we can easily derive the starting angle θ_0 formed by the TCS x-axis (of time instance 0) and the projection of the known initial position $\mathbf{X}_0$ onto the TCS $x-y$-plane. Then, ORBIT may be described in TCS using a planar circular motion:

$$t \in [0, \frac{T\theta}{\omega}] \tag{9}$$

$$\theta_0 = arctan\left(\frac{x_{02}}{x_{01}}\right) \tag{10}$$

$$x_{t3} = x_{03}, \forall t \tag{11}$$

$$\lambda = \sqrt{\lambda_{3D}^2 - x_{t3}^2} \tag{12}$$

$$\mathbf{X}_t = [\lambda \cos(t\frac{\omega}{T} + \theta_0), \lambda \sin(t\frac{\omega}{T} + \theta_0), x_{t3}]^T \tag{13}$$

$$\mathbf{L}_t = \mathbf{P}_t. \tag{14}$$

Based on the above abstract description we would like to specify the maximum allowable camera focal length f (determining zoom level), so that 2D visual tracking is not lost in-between successive video frames due to rapid object motion. Given that tracker behavior varies per algorithm, we simply assume a maximum search radius R_{max} (in pixels) defining the video frame area within which the tracked object ROI of time instance $t+1$ must lie, relatively to the ROI position of time instance t, in order to permit successful tracking. This is true of all trackers. Thus, a distance R_t between two temporally successive target ROIs, where $R_t > R_{max}$, implies tracking failure. The case where $R_t = R_{max}$ marks the limit scenario where the tracker marginally succeeds.

We also assume that, during time instance t, the target ROI was at the center of the video frame. However, since $\tilde{\mathbf{P}}_{t+1}$ itself is in fact estimated using tracking information, it is assumed initially unknown at time instance $t+1$. Therefore, $\tilde{\mathbf{L}}_{t+1}$ still coincides with the known, previous 3D target position and the actual target ROI of time $t+1$ must be located via tracking, so that $\tilde{\mathbf{P}}_{t+1}$ gets estimated. Due to this uncertainty, below, the TCS is fixed to that of time instance 0. Without loss of generality, we assume $t = 0$ and examine an entire ORBIT shooting session as repeated transitions between the first ($t = 0$) and the second video frame (by proper manipulation of θ_0).

Moreover, we can assume that the difference in target/UAV altitude between successive time instances is negligible in most realistic scenarios. Thus, the following hold:

$$\mathbf{X}_t = [\lambda \cos(\theta_0), \lambda \sin(\theta_0), x_{t3}]^T \tag{15}$$

$$\mathbf{X}_{t+1} = [\lambda \cos(\frac{\omega}{T} + \theta_0) + \frac{u_1}{T}, \lambda \sin(t\frac{\omega}{T} + \theta_0), x_{t3}]^T \tag{16}$$

$$\mathbf{P}_t = [0, 0, 0]^T \tag{17}$$

$$\mathbf{P}_{t+1} = [\frac{u_1}{T}, 0, 0]^T \tag{18}$$

Based on the above and the equations linking 3D world coordinates with pixel coordinates [16], the following hold:

$$x_d(t+1) = o_x - \frac{f}{s_x}\frac{\mathbf{R}_1^T(\mathbf{P}_{t+1} - \mathbf{X}_{t+1})}{\mathbf{R}_3^T(\mathbf{P}_{t+1} - \mathbf{X}_{t+1})} \tag{19}$$

$$y_d(t+1) = o_y - \frac{f}{s_y}\frac{\mathbf{R}_2^T(\mathbf{P}_{t+1} - \mathbf{X}_{t+1})}{\mathbf{R}_3^T(\mathbf{P}_{t+1} - \mathbf{X}_{t+1})} \tag{20}$$

where o_x, o_y define the image center in pixel coordinates, s_x, s_y denote the pixel size (in mm) along the horizontal and vertical directions, and $x_d(t+1), y_d(t+1)$ are the target center pixel coordinates at time instance $t+1$. $\mathbf{R}_1$, $\mathbf{R}_2$ and $\mathbf{R}_3$ refer, respectively, to the first, second and third row of the rotation matrix $\mathbf{R}$ that orients the camera gimbal according to the LookAt vector and the current camera/UAV position. $\mathbf{R}$ is easily derived as a change-of-basis matrix, transforming from world to camera coordinates [17].

Using the above formulas, the limit constraint previously mentioned and the assumption that the target ROI was at the center of the t-th video frame, we arrive at the following equation:

$$R_{max} = \sqrt{(x_d(t+1) - o_x)^2 + (y_d(t+1) - o_y)^2}. \tag{21}$$

By substituting and solving for f, the following expression is derived from Eq. (21):

$$f = \frac{R_{max} s_x s_y \left|T\left(x_{t3}^2 + \lambda^2\right) + u_{t1}\lambda\cos\left(\theta_0 + \frac{\omega}{T}\right)\right|}{u_{t1}\sqrt{\frac{s_x^2 x_{t3}^2\left(\lambda T\cos\left(\theta_0 + \frac{\omega}{T}\right) + u_{t1}\right)^2 + \lambda^4 s_y^2 T^2 \sin^4\left(\theta_0 + \frac{\omega}{T}\right) + D}{T^2\left(\lambda^2 + x_{t3}^2\right) + 2\lambda T u_{t1}\cos\left(\theta_0 + \frac{\omega}{T}\right) + u_{t1}^2}}}, \tag{22}$$

where:

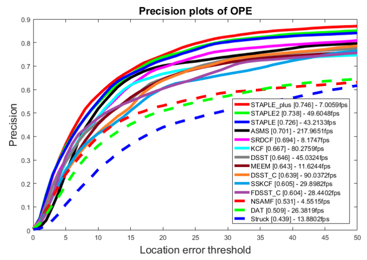
$$D = \lambda^2 s_y^2 \sin^2\left(\theta_0 + \frac{\omega}{T}\right)\left(T^2 x_{t3}^2 + \lambda T\cos\left(\theta_0 + \frac{\omega}{T}\right)E + u_{t1}^2\right), \tag{23}$$

$$E = \left(\lambda T\cos\left(\theta_0 + \frac{\omega}{T}\right) + 2u_{t1}\right). \tag{24}$$

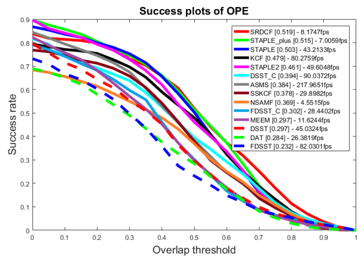
Eq. (22) provides the maximum focal length f as a function of the current target velocity and the current UAV position, relatively to the target. It can be evaluated at each video frame of an ORBIT shooting session, so that the maximum zoom level can be determined. Other hardware specifications can be derived by following a similar analysis.

4. EMPIRICAL EVALUATION AND CUDA ACCELERATION

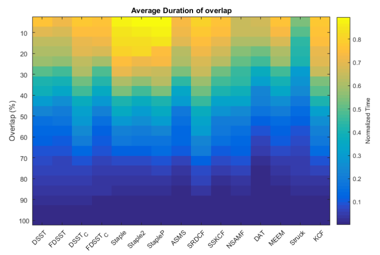
In this section, we describe the experiments conducted in order to evaluate the performance of state-of-the-art 2D trackers, for the outdoor sports tracking scenario. To this end, we have collected 31 sports videos recorded by commercial drones, including long term tracking examples (e.g.,



(a) Precision Plots



(b) Success Rates (One-pass evaluation)



(c) Average duration of overlap

Fig. 1: Tracking Evaluation Results

4min), depicting boat/bike/rowing races, football games, wake-boarding races. Our implementation included the 13 of the best performing trackers from both the VOT 2016 challenge [18] and the UAV tracking benchmark [19], using a similar procedure, along with our proposed Staple2 tracker. We have used a CPU C++ serial implementation for all trackers. The limitations of our evaluation platform include that a) the CPU of this platform is stronger than a common drone processing unit and b) no parallel implementation potentials could be evaluated thoroughly. Thus, we partially parallelized the most appropriate tracker using CUDA.

The 2D tracking evaluation results are summarized in Figure 1. Figure 1a depicts the best 10 tracker Precision plots, i.e., the ratio of successful frames whose tracker output is within the given threshold (x-axis of the plot, in pixels) from the ground-truth, measured by the center distance between bounding boxes. Figure 1b depicts the 10 best Success Rates, i.e., the ratio of the frames whose tracked box has more overlap with the ground-truth box than the threshold. The values in the brackets in the figures are the AUC (area under curve), each of which is the average of all success rates at different thresholds when the thresholds are evenly distributed. Finally, Figure 1c depicts the percentage of the total duration of the sequence (averaged over all sequences), on which the the trackers managed to keep a $p\%$ precision threshold, in 5% increments, which is in essence, how dependable a tracker is for target following purposes.

As it can be seen, the best tracking precision is obtained by Staple plus, which is an implementation of classic Staple [9] with enhanced features [18], having a speed of about 7 fps. In the UAV target following scenario, even with an i7 CPU on-board, this tracker would definitely miss some frames, a fact that limits the maximum supported target speed, as it has been noted in the literature [20]. The second best implementation is the proposed, much faster Staple2 (49 fps). Finally, as it can be seen in Figure 1c, Staple2 maintains at least 10% overlap with the ground truth for the entire video duration, making it the most dependable choice for UAV target following applications among the tested algorithms.

Since Staple2 was found to be the most appropriate tracker for UAV target following applications, we have profiled its code and found the hot spots. The most taxing spot was a function that resizes and extracts the fHOG features, since multiple various-scaled windows are extracted are analysed to detect the scale changes. Thus, we decided to accelerate this part of the code using CUDA, having two benefits: a) The previous implementation was optimized with x86 SIMD extensions which is incompatible with ARM architectures, like e.g., platform Nvidia Jetson TX2, b) many other trackers (including Staple, FAST, DSST) employ the same function for scale estimation. Evaluation was performed on a desktop PC with an NVIDIA QUADRO K620 GPU. The corresponding execution time of the function that resizes and extracts the fHOG features for the various scales was measured both on CPU and GPU. This is an important measure, since it represents 45% of tracking execution time according to our profiling. Using CUDA, above $\times 3$ acceleration was obtained compared to the CPU version (runtime of 3ms on CPU, 0.9ms on GPU).

5. CONCLUSION

In this paper, we provide a theoretical treatment of the representative UAV orbit case. Our findings can be exploited for specifying UAV hardware and, moreover, as a heuristic for determining the possible UAV target-following scenarios under specific drone-camera configurations. Additionally, we have also considered the relevant software requirements for 2D visual target tracking and conducted corresponding experiments in aerial sports videos. We have determined the most appropriate 2D visual tracking implementation by combining findings from several state-of-the-art methods. 2D tracking can be dependably employed for target following, if combined with occasional whole frame detection. Currently, it is typical for tracking scale to be addressed as a single optimization problem, while, in essence, it is two separate problems that can be solved separately. Staple showcases the advantages of such an approach, which should be more thoroughly investigated in the future.

6. REFERENCES

[1] R. Li, M. Pang, C. Zhao, G. Zhou, and L. Fang, “Monocular long-term target following on uavs,” in *Pro-*

ceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2016, pp. 29–37.

[2] P. Theodorakopoulos and S. Lacroix, “A strategy for tracking a ground target with a uav,” in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 1254–1259.

[3] Vladimir N Dobrokhodov, Isaac I Kaminer, Kevin D Jones, and Reza Ghabcheloo, “Vision-based tracking and motion estimation for moving targets using unmanned air vehicles,” *Journal of guidance, control, and dynamics*, vol. 31, no. 4, pp. 907–917, 2008.

[4] C. Teuliere, L. Eck, and E. Marchand, “Chasing a moving target from a flying uav,” in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 4929–4934.

[5] V. Mygdalis, A. Iosifidis, A. Tefas, and I. Pitas, “Graph embedded one-class classifiers for media data classification,” *Pattern Recognition*, vol. 60, pp. 585 – 595, 2016.

[6] A. Iosifidis, V. Mygdalis, A. Tefas, and I. Pitas, “One-class classification based on extreme learning and geometric class information,” *Neural Processing Letters*, pp. 1–16, 2016.

[7] I. Mademlis, A. Iosifidis, A. Tefas, N. Nikolaidis, and I. Pitas, “Exploiting stereoscopic disparity for augmenting human activity recognition performance,” *Multimedia Tools and Applications*, vol. 75, no. 19, pp. 11641–11660, 2016.

[8] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg, “Discriminative scale space tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[9] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P.H.S. Torr, “Staple: Complementary learners for real-time tracking,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1401–1409.

[10] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[11] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. .L Hicks, and P. H.S. Torr, “Struck: Structured output tracking with kernels,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2096–2109, 2016.

[12] D. S. Bolme, J. R. Beveridge, B. A Draper, and Y. M. Lui, “Visual object tracking using adaptive correlation filters,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2544–2550.

[13] N. Wang and D.-Y. Yeung, “Learning a deep compact image representation for visual tracking,” in *Advances in Neural Information Processing Systems 26*, pp. 809–817. Curran Associates, Inc., 2013.

[14] E. Cheng, *Aerial Photography and Videography Using Drones*, Peachpit Press, 2016.

[15] C. Smith, *The Photographers Guide to Drones*, Rocky Nook, 2016.

[16] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, 1998.

[17] E. Angel and D. Shreiner, *Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL*, Pearson, 6th edition, 2011.

[18] Kristan M. et al., *The Visual Object Tracking VOT2016 Challenge Results*, pp. 777–823, Springer International Publishing, Cham, 2016.

[19] Matthias Mueller, Neil Smith, and Bernard Ghanem, “A benchmark and simulator for uav tracking,” in *European Conference on Computer Vision*. Springer, 2016, pp. 445–461.

[20] H. K. Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey, “Need for speed: A benchmark for higher frame rate object tracking,” *arXiv preprint arXiv:1703.05884*, 2017.