2017-09-25

Visual Multiple Target Tracking From a Descending Aerial Platform

Parker C. Lusk
Department of Electrical and Computer Engineering, *Brigham Young University*, parkerclusk@gmail.com

Randal Beard
Brigham Young University Provo, beard@byu.edu

Visual Multiple Target Tracking From a Descending Aerial Platform

Parker C. Lusk and Randal W. Beard

***Abstract*— A real-time visual multiple target tracker is demonstrated onboard a descending multirotor. Measurements of moving ground targets are generated using the Kanade-Lucas-Tomasi (KLT) tracking method. Homography-based image registration is used to align the measurements into the same coordinate frame, allowing for the detection of independently moving objects. The recently developed Recursive-RANSAC algorithm uses the visual measurements to estimate targets in clutter. Altitude-dependent tuning increases track continuity and coverage during the descent of the vehicle. The algorithm requires no operator interaction and increases the situation awareness of the unmanned aerial system. Real-time tracking efficiency is analyzed on GPUs and CPUs. Tracking results are presented and discussed using the MOTA and MOTP metrics.**

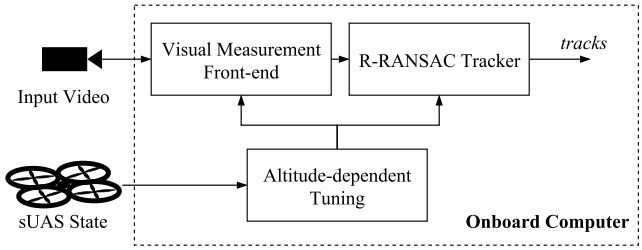


Fig. 1: Visual tracking system architecture. The altitude of the sUAS is used to tune the parameters of the Recursive-RANSAC visual tracker during a descent. Real-time tracking is performed with an onboard NVIDIA Jetson TX2.

I. INTRODUCTION

Autonomous vehicles are quickly becoming ideal platforms in research, commercial, military, and civil applications. Typical autonomous systems include self-driving cars, personal air vehicles, and small unmanned aerial systems (sUAS). As these vehicles are integrated into our society and infrastructure, an increased level of autonomy will be required – both for safety and for mission capability. This higher level of autonomy will allow vehicles to perceive their environment and act within certain parameters to achieve their goal. This construct of sensory-based decision making is modeled by *situation awareness* [1].

Situation awareness (SA) is a term that originated in the military aircraft pilot community and is defined as the human process of perceiving details in the environment, comprehending how those details affect the current goal, and projecting that comprehension to what will happen in the near future [2]. This process of SA allows human operators to make critical decisions in a timely and effective manner. Similarly, adding a sense of SA in autonomous vehicles allows them to operate effectively in dynamic environments [3]. The main contribution of this paper is in enhancing the SA perception stage of sUAS through altitude-dependent visual multiple target tracking during a descent.

Beyond visual line of sight (BVLOS) operation is an important aspect of truly autonomous vehicles. The challenge of understanding the environment and avoiding obstacles at the beginning and end of a sUAS flight has been referred to as the "first/last 50 feet problem" [4]. Specifically, difficulties arise when sUAS must descend and possibly land in an environment with obstacle uncertainty, such as in the case of forced landings or package delivery [5]. Situation awareness is crucial for these tasks to minimize the risk of damage to moving ground obstacles. In work by McAree et al. [6], the authors simulate UAS during landing at prepared sites in the presence of multiple agents and state uncertainty. The SA of the UAS is enhanced using automated dependant surveillance-broadcast (ADS-B) for cooperative sensing of other agents. The landing approach taken by Scherer et al. [7] uses lidar for non-cooperative sensing to autonomously land a full-scale helicopter at unprepared sites. Mejias et al. [8] demonstrate a camera-based landing zone detection algorithm for UAS in emergency landing scenarios. In this work, a visual multiple target tracker is used to perceive information about moving ground targets. We focus on the target tracking information needed to refine an unprepared landing site that has already been selected by other means.

Visual target tracking has been an active area of computer vision research [9]–[11]. Tracking can focus on single or multiple objects and can take place on a static or moving platform. Initialization of tracking typically falls in the category of tracking-by-detection or detection-free tracking. Tracking-by-detection performs frame-to-frame association of detections generated by pre-trained object classifiers [12], feature descriptors [13], or even a simple corner detector [14]. Conversely, detection-free tracking must be manually initialized, as in the work of Kalal et al. [15].

Visual tracking gives sUAS situation awareness and can enable many autonomous applications. Cameras are ideal sensor packages for sUAS because they are low-cost, small, and rich with visual information. In the work of Thomas et al. [16], a small quadrotor with a downward-facing camera is used to track and follow a single object with known geometry. Teulière et al. [17] also track and follow a single object using a template image, removing the need of prior target geometry. A color-based tracker and a particle filter are used, allowing tracking robustness in partial or full occlusion

P. C. Lusk is a M.S. student in Electrical and Computer Engineering at Brigham Young University, Provo, UT (email: parkerclusk@gmail.com).

R. W. Beard is a Professor of Electrical and Computer Engineering at Brigham Young University, Provo, UT (email: beard@byu.edu).

of the target selected by the user. Pestana et al. [18] performs visual servoing from a sUAS by tracking a single user-specified object.

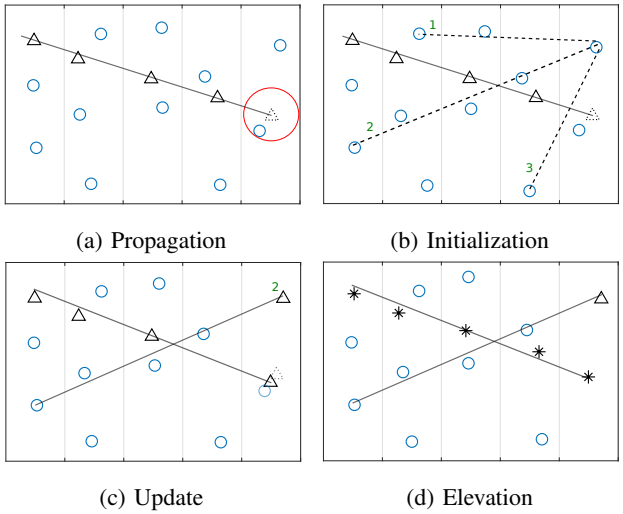
Multiple object tracking (MOT) from a moving camera is of particular interest in sUAS applications because it adds robustness to surveillance techniques where the number of targets or the nature of the camera motion is not known beforehand. Rodriguez-Canosa et al. [19] use Parallel Tracking and Mapping (PTAM) [20] for motion estimation which is then used to create an artificial optical flow field. The difference between Lucas-Kanade optical flow and the artificial flow field exposes dynamically moving objects; however, this technique requires initialization using a marker map and may struggle to detect objects moving with similar velocity to the multirotor. Jiang and Cao [21] are able to track multiple objects in post-processed aerial video using detections based on background modeling, but do not use Bayesian filtering for track management. Li et al. [22] successfully track other sUAS for sense and avoid applications in post-processed aerial footage using a similar visual tracking front-end to ours; however, they assume targets are non-deformable which limits the types of objects that can be tracked. Teutsch and Krüger [23] post-process aerial videos and demonstrate traffic surveillance and tracking from a constant altitude. Their approach is most similar to our tracking pipeline, but assumes constant altitude stable flight over structured environments, which results in smooth video input.

We perform real-time visual multiple object tracking onboard a sUAS using an algorithm known as the Recursive-RANSAC (R-RANSAC) visual tracker. In contrast to existing tracking solutions, our algorithm runs on a moving platform, requires no manual initialization, can run in real-time during a flight, and can track an arbitrary number of moving objects in clutter. In addition, we propose an altitude-dependent tuning scheme that increases track continuity during a descent, as shown in Figure 1. This coupling of aircraft state and tracker increase the situation awareness of the sUAS allowing future work to autonomously plan safe trajectories during the "last 50 feet" [4].

The organization of this paper is as follows. Section II gives an overview of R-RANSAC and the visual measurement front-end. In Section III we present the proposed tuning scheme to increase track continuity during a descent. Section IV discusses the hardware implementation and demonstration. Tracking results are reported and discussed in Section V and future research directions are given in Section VI.

II. VISUAL TRACKING

As shown in Figure 1, the visual multiple target tracker used in this work consists of two major components: the visual measurement front-end and the Recursive-RANSAC (R-RANSAC) tracker. Using the visual tracker in conjunction with an altitude-dependent tuning scheme allows the algorithm to continuously track objects as the sUAS descends. We discuss these components in the following subsections.



(a) Propagation (b) Initialization

(c) Update (d) Elevation

Fig. 2: The four steps of R-RANSAC demonstrated on a surveillance region in $\mathbb{R}^2$. Timesteps are denoted by the grey vertical lines, the current timestep is rightmost. Measurements ($\circ$) may be clutter or from targets. (a) Hypothesis models ($\triangle$) are predicted forward in time (dotted). New measurements are associated as inliers or outliers. (b) Outliers are used to generate RANSAC hypotheses (1, 2, 3). (c) Inlier measurements are used to correct the model prediction. The RANSAC hypothesis with the most support (hypothesis 2) is stored as a model. (d) Models that have good support and have been tracked for a while without too many missed detections are elevated to a track ($*$).

A. Recursive-RANSAC Tracker

R-RANSAC is an online estimation algorithm capable of tracking an arbitrary number of objects in clutter [24]–[27]. Measurements are contained in the surveilance region $\mathcal{R}$ of the system, where $\mathcal{R} \subset \mathbb{R}^2$ in this work. Using the incoming measurements, the algorithm forms hypothesis models that fit the specified motion dynamics. At every timestep, the following four tasks are performed, as shown in Figure 2.

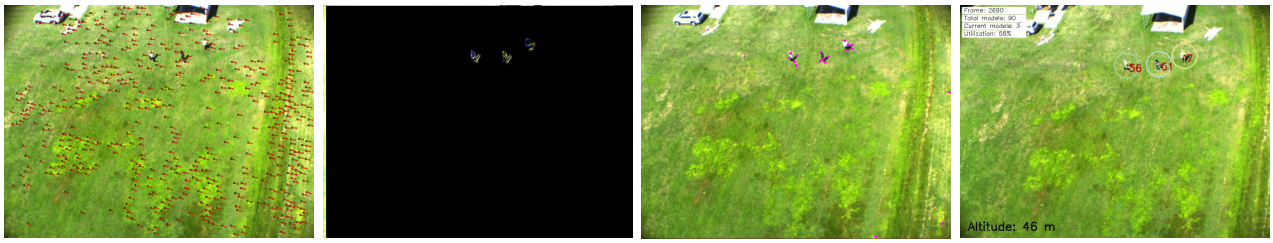
Model propagation: Existing models are propagated forward with nearly constant jerk motion dynamics. The new scan of measurements are classified as inliers or outliers to each predicted model position based on Euclidean distance. Measurements within the inlier region defined by a circle of radius τ_R are classified as inliers.

Model initialization: For each measurement that is an outlier to all models, a RANSAC-based initialization step is performed to find models that fit nearly constant velocity dynamics. Only the best RANSAC hypothesis model is kept.

Model update: At the end of each timestep, each model uses its associated inliers to perform a Kalman update.

Model elevation: A model is elevated to a track once it has survived τ_T iterations without having more than τ_{CMD} consecutive missed detections. This step is also where models with poor support are pruned and models with similar positions are merged if within τ_x and τ_y of each other.

R-RANSAC's strength lies in its ability to initialize new hypothesis models from noisy data and subsequently manage those models without operator intervention. This allows the use of computationally cheap computer vision algorithms



(a) KLT Tracking (b) Homography (c) Moving Features (d) Tracking Results

Fig. 3: The three steps (a)-(c) of the visual measurement front-end with the resulting tracks (d) from R-RANSAC. Images are taken from sequence **run2a**. Feature correspondences from (a) are used to estimate a homography. Note how the homography-compensated difference image in (b) masks out the feature motion resulting from camera motion and exposes independently moving objects.

with less precision. Additionally, R-RANSAC is not strictly a computer vision algorithm; it can filter measurements from diverse sensor modalities [28]. The work of fusing different measurement sources in R-RANSAC is currently being investigated. For more information about the R-RANSAC derivation we refer the reader to [24].

B. Visual Measurement Front-end

In this work, R-RANSAC receives data from a visual measurement front-end. The vision processing is done with a calibrated camera in a three-step pipeline in order to (i) find feature correspondences between images, (ii) compute a homography, and (iii) detect true object motion. The input video rate is controlled by the *frame stride* parameter which dictates how many frames to skip. For example, with incoming video at 30fps, $stride = 3$ results in 10Hz processing.

(i) *Feature management*: At each timestep k, features from the last image X_{k-1} are propagated forward into the current image as X^+_{k-1} using optical flow. Feature correspondences (X_{k-1}, X^+_{k-1}) are sent as input to the next step in the pipeline for further processing. A new set of features X_k are then found using the Shi-Tomasi corner detection method for the current image $\mathcal{I}_k$. These features will be propagated on the next iteration. This step is known as Kanade-Lucas-Tomasi (KLT) tracking and is depicted in Figure 3a.

(ii) *Homography generation*: Using the feature correspondences (X_{k-1}, X^+_{k-1}) from the KLT tracker, a perspective transformation H known as a homography is estimated using a RANSAC-based scheme. This step is crucial for moving platform tracking because it allows the set of features X_{k-1} and X_k to be represented in the same coordinate frame through image registration. The quality of a homography estimation between camera views can be visualized via difference imaging (see Figure 3b), defined as

$$\mathcal{D}_k = \mathcal{I}_k - \mathcal{I}^+_{k-1} = \mathcal{I}_k - H\mathcal{I}_{k-1}. \tag{1}$$

Note that the R-RANSAC visual tracker only makes use of KLT features and that the difference image $\mathcal{D}_k$ is only computed when assessing the homography estimation quality.

(iii) *Moving object detection*: Equipped with a homography and a set of feature correspondences, the velocity of each of the feature points can be calculated as

$$V = X^+_{k-1} - HX_{k-1}. \tag{2}$$

If the homography estimate is good, then the velocity of static features will be nearly zero, leaving behind the motion of independent objects only, as shown in Figure 3c. Measurements ($z_j = [x, y, v_x, v_y]^\top$) of independently moving objects are defined as feature points that have a velocity magnitude within predefined thresholds, given by

$$Z = \{(x_i, v_i) : x_i \in X^+_{k-1}, v_i \in V, \tau_{v_{\min}} \leq v_i \leq \tau_{v_{\max}}\}.$$

This scan of measurements is then given to R-RANSAC to estimate the position of targets. Figure 3d shows the tracking results.

Note that the calibration parameters of the camera are used to undistort the features extracted in step (i). Further, the camera matrix is used to project features from 2D pixel space to the *normalized image plane* in 3D space where coordinates are normalized such that the depth is unity. This results in tracker parameters that are less sensitive to differences in calibrated cameras and allows tuning to be done in more intuitive units, as described below.

III. ALTITUDE-DEPENDENT TUNING

Track continuity is an important attribute of situationally aware systems. This attribute implies that moving targets maintain a unique track ID throughout its lifetime. As the aerial vehicle changes altitude, objects will change in size with respect to the camera field of view. This can cause tracks to fragment into multiple IDs.

In order to maintain track continuity during a UAS descent, we propose using the vehicle altitude to tune parameters of the R-RANSAC visual tracking system. The relevant tuning parameters for R-RANSAC are the inlier region τ_R and the absolute difference threshold for model merging, τ_x and τ_y. The visual front-end feature velocity thresholds $\tau_{v_{\min}}$ and $\tau_{v_{\max}}$ are also tuned during flight. Denote the UAS altitude as h. The parameters are then

$$\tau_R = \frac{s}{2h}; \quad \tau_x = \tau_y = \frac{d_{\text{merge}}}{h} \tag{3}$$

$$\tau_{v_{\min}} = \frac{v_{\min}}{h}; \quad \tau_{v_{\max}} = \frac{v_{\max}}{h}, \tag{4}$$

where the tuning parameters are: s, the object size in meters; d_{merge}, the distance for model merging in meters; v_{min} and v_{max} the minimum and maximum target velocities in meters per second.

IV. HARDWARE DEMONSTRATION

The R-RANSAC visual tracker software with altitude-dependent tuning is demonstrated in real-time during four flight tests. A GPU implementation is discussed that enhances the tracking ability of the algorithm. The flight scenario is described, along with performance metrics for multiple object tracking.

A. GPU Implementation

The most computationally expensive portion of the R-RANSAC visual tracker is the KLT optical flow feature tracking. Leveraging OpenCV libraries allowed for a GPU implementation of the visual front-end. Although able to run on mobile CPU machines, using a GPU gives the visual tracker extra processing time between each frame, allowing more features to be extracted at a higher frame rate (i.e., lower $stride$). It also allows processing time for higher level functions such as mapping, localization, and path planning.

An increase of feature correspondences between frames allows for a more robust homography estimation process. A homography describes the perspective transformation of a plane from one view to another. In low-altitude scenarios there is often more depth in the scene from tall structures and trees, causing parallax and making it more difficult to find a single plane to describe the entire image. However, moving objects are most often found on local planes (e.g., streets, grass, etc). With more feature points used in the homography estimation process, there will be a higher likelihood of choosing the local plane of the targets and rejecting the parallax points as outliers. This increases the ability of the visual front-end to maintain detection of independent object motion.

Using the flight test data, we perform a comparison of real-time tracking efficiency on various CPU and GPU platforms. Efficiency is expressed in terms of *utilization*, which is measured per frame as the ratio of time spent processing to total time available ($\frac{1}{\text{fps}}$).

B. Flight Scenario

The selected sUAS for testing is the 3DR Y6 multirotor with an onboard NVIDIA Jetson TX2, as shown in Figure 4. Four flight tests were performed to demonstrate the R-RANSAC visual tracker in real-time. Each scenario lasted 2 minutes starting at an altitude of 122 meters and ending at 20 meters. The descent was controlled by four pre-programmed waypoints resulting in a 45° descent. The multirotor stops at each waypoint for 10 seconds.

During the descent, targets move at various rates in the camera field of view. Using the Robot Operating System (ROS), all stages of the visual tracking system are recorded, including vehicle state and tracking results. ROS manages the initialization and communication of the software as soon as the flight computer is powered on.

Fig. 4: 3DR Y6 multirotor used in hardware demonstration. The Pixhawk autopilot runs APM:Copter firmware. The camera has a resolution of 800×600 at 30fps and is mounted at a 45° angle.

C. Performance Metrics for Multiple Object Tracking

To measure the performance of multiple object tracking, we use the CLEAR MOT metrics, MOTP and MOTA [29]. These metrics provide an overall performance measure of tracker precision (MOTP) and accuracy (MOTA). MOTP measures the ability of the tracker to precisely estimate the true position of objects. In this work, an intersection-over-union (IOU) ratio of bounding boxes is used; thus, a MOTP score of 1.00 represents perfect precision. MOTA gives a measure of how well the tracker detects objects and their trajectories, being penalized for false positives, track mismatches, and missed detections. A given track is associated with ground truth if there is at least 10% overlap (IOU) between bounding boxes.

In addition to the CLEAR MOT metrics, we provide a measure of *track coverage* for each ground truth object in the flight test. It is defined as the ratio of total frames tracked to total frames present (and moving); thus, it is a means of breaking down the MOTA score to each object. False positive (FP) coverage is defined as the ratio of false positives to the number of frames in the sequence.

In order to calculate these metrics, ground truth is required. Flight video was manually annotated using a VATIC-inspired JavaScript implementation[1] [30].

V. TRACKING RESULTS

A. Tracking Performance

The results of the CLEAR MOT tracking analysis are found in Table I. The arrows in the heading indicate if low (↓) or high (↑) ratios are preferred. The overall results for each test flight sequence are given as well as a breakdown according to alitude ranges dictated by the four waypoints (122m, 84m, 53m, 20m). Targets 1–3 are people and target 4 is a small RC vehicle. Runs 1 and 2 are smoother and more visually consistent, while runs 3 and 4 are more difficult because of harsh lighting, strong winds, and many strong edges in the scene. This difficulty is reflected in the MOTA and MOTP scores.

[1]Source code can be found at `github.com/plusk01/vatic.js`

TABLE I: Tracking Performance[2]

Sequence	Duration [s]	Altitudes [m]	MOTA ↑	MOTP ↑	FP ↓	Target Coverage ↑			
						1	2	3	4
run1a	**123**	**122 to 20**	**0.75**	**0.52**	**0.28**	**1.00**	**0.94**	**0.87**	
	49	122 to 84	0.66	0.53	0.43	1.00	0.98	0.82	
	37	84 to 53	0.81	0.52	0.17	1.00	0.92	0.88	
	37	53 to 20	0.78	0.52	0.22	1.00	0.87	0.93	
run2a	**120**	**122 to 20**	**0.90**	**0.59**	**0.10**	**0.99**	**0.97**	**0.91**	
	49	122 to 84	0.91	0.52	0.13	1.00	0.98	0.97	
	36	84 to 53	0.89	0.67	0.13	1.00	0.94	0.91	
	35	53 to 20	0.90	0.60	0.05	0.99	1.00	0.83	
run3a	**123**	**122 to 25**	**0.47**	**0.39**	**0.27**	**0.93**	**0.55**		
	50	122 to 85	0.42	0.48	0.54	0.98	0.63		
	40	85 to 49	0.53	0.36	0.11	0.91	0.35		
	33	49 to 25	0.48	0.29	0.10	0.90	0.73		
run4a	**120**	**123 to 25**	**0.25**	**0.30**	**0.39**	**0.62**			**0.33**
	53	123 to 83	0.45	0.30	0.38	0.74			0.33
	38	83 to 49	0.10	0.28	0.40	0.33			0.14
	29	49 to 25	0.23	0.32	0.38	0.66			0.59

Note that scores tend to improve as the altitude decreases. This is expected because moving objects have more detail in the frame, allowing more features to be extracted and tracked. We expect that scores would improve further if we employed a feature parallax detection and rejection scheme.

For the smoother video in runs 1 and 2, the visual tracker performs very well. This is in contrast to the erratic, wind-induced camera motion in runs 3 and 4. We note that the most apparent difference in the two pairs of sequences is that runs 1 and 2 tended to have the targets in the middle of the frame while the targets were more often on the edge of the frame in runs 3 and 4. This caused the targets to frequently move in and out of the camera field of view, incurring track fragmentation and missed detection costs. We suggest that using an object detection method (feature descriptors, template matching, etc) or tracking measurements in the 3D world coordinate frame would mitigate these issues.

B. GPU vs CPU Utilization

Figure 5 shows the results of running the four video sequences on three different machines while holding the parameters constant. Each machine has an associated pair of bars, with the left representing CPU-only and the right GPU-enabled. The **i7Mobile** is a Gigabyte Brix computer with only a CPU. An **i7Desktop** machine is listed for comparison. Each bar breaks down how much time is spent in each part of the tracking pipline during one period ($\frac{1}{\text{fps}}$). The algorithm is running in real-time if utilization is under 100%.

Note that the **i7Mobile** computer can on average run in real-time with a frame stride of 3; however, there is almost no extra processing time for higher-level tasks. The **TX2** on the other hand performs comfortably in real-time for strides of 1, 2, and 3. We choose to process at $stride = 3$ (10Hz) so that there is more target motion between frames at higher altitudes and so there is plenty of computational resources available for future higher-level tasks.

[2]Video results can be found at `https://youtu.be/UIlvXSdVvqA`

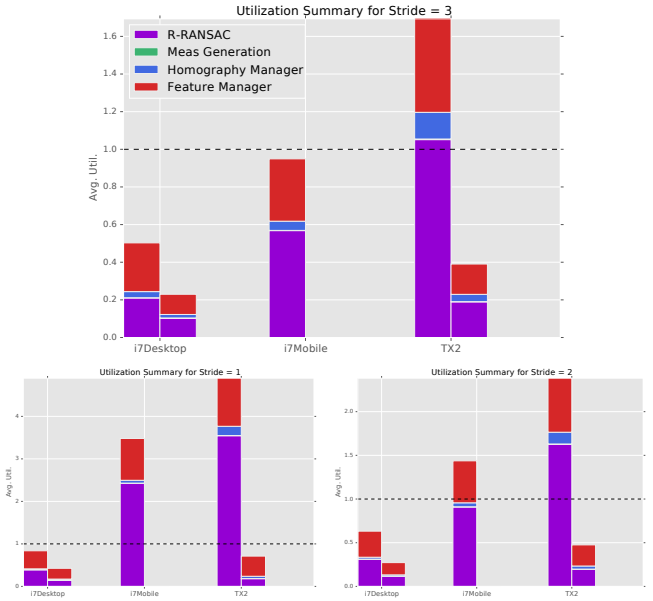


Fig. 5: R-RANSAC visual tracker utilization for frame strides of 1, 2, and 3. Each of the three bar pairs represent a single machine. The left bar in each pair shows the utilization on CPU-only tracking and the right bar shows the utilization with the GPU (if available) enabled. The tracker is running in real-time if utilization is below the dotted line (100%).

VI. CONCLUSION

In this paper we have demonstrated a visual multiple target tracker running in real-time onboard a descending multirotor. With a robust target tracking solution, the situation awareness of autonomous vehicles is increased. A utilization analysis shows that extra processing time is available for tracking improvements and higher-level tasks such as path planning and control. Future work will include parallax compensation, world frame tracking, and trajectory optimization for ground target avoidance during landing.

ACKNOWLEDGMENT

The authors thank the NASA Langley Systems Analysis & Concepts Directorate (SACD) and the NASA UAS Traffic Management (UTM) project for funding this work and providing flight test support at the NASA Langley sUAS testing site. This research was additionally supported by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation-sponsored industry/university cooperative research center (I/UCRC) under NSF Award No. IIP-1161036 along with significant contributions from C-UAS industry members, and in part by AFRL grant FA8651-13-1-0005.

REFERENCES

[1] M. R. Endsley, "Toward a Theory of Situation Awareness in Dynamic Systems," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 37, no. 1, pp. 32–64, 1995.

[2] M. R. Endsley, "Automation and Situation Awareness," *Automation and Human Performance: Theory and Applications*, pp. 163–181, 1996.

[3] J. A. Adams, "Unmanned Vehicle Situation Awareness: A Path Forward," *Proceedings of the 2007 Human Systems Integration Symposium*, no. 615, 2007.

[4] P. Kopardekar, J. Rios, T. Prevot, M. Johnson, J. Jung, and J. E. Robinson III, "UAS Traffic Management (UTM) Concept of Operations to Safely Enable Low Altitude Flight Operations," *AIAA Aviation Technology, Integration, and Operations Conference*, no. June, pp. 1–16, 2016.

[5] Amazon.com Inc., "Determining Safe Access with a Best-Equipped, Best-Served Model for Small Unmanned Aircraft Systems," *NASA UTM 2015: The Next Era of Aviation*, July 2015.

[6] O. McAree and W. H. Chen, "Artificial situation awareness for increased autonomy of unmanned aerial systems in the terminal area," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 70, no. 1-4, pp. 545–555, 2013.

[7] S. Scherer, L. Chamberlain, and S. Singh, "Autonomous landing at unprepared sites by a full-scale helicopter," *Robotics and Autonomous Systems*, vol. 60, no. 12, pp. 1545–1562, 2012.

[8] L. Mejias and D. Fitzgerald, "A Multi-layered Approach for Site Detection in UAS Emergency Landing Scenarios using Geometry-Based Image Segmentation," in *Int. Conf. Unmanned Aircraft Syst.*, pp. 366–372, 2013.

[9] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao, and T.-K. Kim, "Multiple Object Tracking: A Literature Review," pp. 1–18, 2014.

[10] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1442–1468, 2014.

[11] L. Leal-Taixé, A. Milan, K. Schindler, D. Cremers, I. Reid, and S. Roth, "Tracking the Trackers: An Analysis of the State of the Art in Multiple Object Tracking," no. March, 2017.

[12] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008.

[13] A. Pieropan, M. Björkman, N. Bergström, and D. Kragic, "Feature Descriptors for Tracking by Detection: a Benchmark," 2016.

[14] J. Shi and C. Tomasi, "Good Features to Track," in *IEEE Conference on Computer Vision and Pattern Recognition*, no. June, (Seattle), 1994.

[15] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 1, pp. 1409–1422, 2010.

[16] J. Thomas, J. Welde, G. Loianno, K. Daniilidis, and V. Kumar, "Autonomous Flight for Detection, Localization, and Tracking of Moving Targets With a Small Quadrotor," *IEEE Robotics and Automation Letters*, vol. 2, pp. 1762–1769, jul 2017.

[17] C. Teulière, L. Eck, and E. Marchand, "Chasing a moving target from a flying UAV," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4929–4934, 2011.

[18] J. Pestana, J. L. Sanchez-Lopez, P. Campoy, and S. Saripalli, "Vision based GPS-denied Object Tracking and following for unmanned aerial vehicles," *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2013*, 2013.

[19] G. R. Rodríguez-Canosa, S. Thomas, J. del Cerro, A. Barrientos, and B. MacDonald, "A Real-Time Method to Detect and Track Moving Objects (DATMO) from Unmanned Aerial Vehicles (UAVs) Using a Single Camera," *Remote Sensing*, vol. 4, no. 4, pp. 1090–1111, 2012.

[20] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR*, 2007.

[21] X. Jiang and X. Cao, "Surveillance from above: A detection-and-prediction based multiple target tracking method on aerial videos," in *2016 Integrated Communications Navigation and Surveillance (ICNS)*, pp. 4D2–1–4D2–13, IEEE, apr 2016.

[22] J. Li, D. H. Ye, T. Chung, M. Kolsch, J. Wachs, and C. Bouman, "Multi-target detection and tracking from a single camera in Unmanned Aerial Vehicles (UAVs)," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4992–4997, 2016.

[23] M. Teutsch and W. Krüger, "Detection, segmentation, and tracking of moving objects in UAV videos," *Advanced Video and Signal-Based Surveillance (AVSS), 2012 IEEE Ninth International Conference on*, pp. 313–318, 2012.

[24] P. C. Niedfeldt and R. W. Beard, "Multiple target tracking using recursive RANSAC," *2014 American Control Conference*, pp. 3393–3398, 2014.

[25] P. C. Niedfeldt, "Recursive-RANSAC: A Novel Algorithm for Tracking Multiple Targets in Clutter," *All Theses and Dissertations*, no. July, p. Paper 4195, 2014.

[26] K. Ingersoll, P. C. Niedfeldt, and R. W. Beard, "Multiple target tracking and stationary object detection in video with Recursive-RANSAC and tracker-sensor feedback," *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015*, pp. 1320–1329, 2015.

[27] P. C. Niedfeldt, K. Ingersoll, and R. W. Beard, "Comparison and Analysis of Recursive-RANSAC for Multiple Target Tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 53, pp. 461–476, feb 2017.

[28] E. B. Quist, P. C. Niedfeldt, and R. W. Beard, "Radar odometry with recursive-RANSAC," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 4, pp. 1618–1630, 2016.

[29] K. Bernardin and R. Stiefelhagen, "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, pp. 1–10, 2008.

[30] C. Vondrick, D. Patterson, and D. Ramanan, "Efficiently scaling up crowdsourced video annotation," *International Journal of Computer Vision*, vol. 101, no. 1, pp. 184–204, 2012.