Current Robotics Reports (2021) 2:285–295
https://doi.org/10.1007/s43154-021-00060-7

AERIAL ROBOTICS (E FERON, SECTION EDITOR)

Target Tracking and Following from a Multirotor UAV

Mark Petersen[1] · Chad Samuelson[1] · Randal W. Beard[1]

Accepted: 21 May 2021 / Published online: 2 July 2021

Abstract

Purpose of Review In this paper, we present a general target tracking and following architecture for multirotor unmanned aerial vehicles (UAVs), and provide pointers to related work. We restrict our discussions to tracking ground-based objects using a monocular camera that is not gimballed.

Recent Findings Target tracking is accomplished using a novel visual front end combined with a recently developed multiple target tracking back end. The target following is accomplished using a novel, nonlinear control algorithm.

Summary We present an end-to-end target tracking and following architecture that uses a visual front end to obtain measurements and compute the homography, a tracking algorithm called Recursive Random Sample Consensus (R-RANSAC) to perform track initialization and tracking, a track selection scheme, and a target-following controller. Target tracking and following is accomplished using a monocular camera, an inertial measurement unit, an on-board computer, a flight control unit, a sensor to measure altitude, and a multirotor UAV under the assumption that the target is moving on fairly planar ground with nearly constant velocity.

Keywords Target following · Target tracking · RANSAC · R-RANSAC · Unmanned air vehicle · Autonomous vehicle · Unmanned air system

Introduction

There are numerous approaches to target tracking and following which vary in their degree of simplicity of implementation, computational expense (CPU usage), optimality, and other measures. We will focus primarily on methods appropriate for multirotor UAVs carrying a body-fixed monocular camera.

Any vision-based target-tracking method will need to extract measurements from the images. Recently, there has been extensive research in object detection and identification using deep neural networks such as YOLO [1], R-CNN [2], and others [3]. These methods achieve high accuracy, but are computationally expensive. We discuss other methods in "Visual Front End."

This article belongs to the Topical Collection on *Aerial Robotics*

✉ Randal W. Beard
beard@byu.edu

[1] Electrical and Computer Engineering Department, Brigham Young University, 450 Engineering Building, Provo, UT 84602, USA

For a good review of target-tracking algorithms, we refer the reader to [4]. There exist optimal methods such as the multiple hypothesis tracking (MHT) [5] and the probabilistic multi-hypothesis tracker (PMHT) [6]; however, they are difficult to implement and not feasible to do in real time [7]. A variant of the MHT is the track-oriented MHT (TO-MHT) [8] which can be done in real time. There are other simple, computationally efficient techniques such as the global nearest neighbor filter (GNNF) [9] and joint probabilistic data association filter (JPDAF) [10]; however, they cannot initialize tracks. A track consists of a state estimate (position, velocity, etc) and error covariance of the target. A recently developed non-optimal, target-tracking algorithm is Recursive Random Sample Consensus (R-RANSAC) which efficiently initializes and manages tracks [11–15]. We discuss this algorithm in more detail in "R-RANSAC Multiple Target Tracker."

In order to follow the detected and tracked targets, a control approach known as image-based visual servoing (IBVS) is commonly implemented [16]. IBVS in its most basic form is implemented using the proportional-integral-derivative controller (PID) [17] which is simple to implement and responsive, but causes increased error and overshoot due to UAV rotations [18]. The IBVS-Sphere (IBVS-Sph) effectively combats this problem by mapping the image plane to a virtual spherical image plane around the camera; however, this and similar spherical mapping techniques become unstable as the target moves beneath the UAV [18]. An optimal solution is to map a virtual image plane parallel to the ground directly above the detected targets. This reduces error due to UAV pose and discrepancies caused by spherical mapping. A few examples of this approach, such as the desired robust integral of the sign of error (DCRISE), are [19, 20]. The algorithm described in this paper takes advantage of this mapping of a parallel virtual image plane and will be discussed in more detail in "Target-Following Controller."

The paper is organized as follows.

In "Architecture," we present the tracking and following architecture, followed by the visual front end, R-RANSAC, and the controller in "Visual Front End," "R-RANSAC Multiple Target Tracker," and "Target-Following Controller." Finally, we discuss our results and conclusions in "Results" and "Conclusions."

Architecture

In this paper, we assume that a monocular camera is rigidly mounted on a multirotor UAS equipped with IMU, on-board computer, autopilot, and an altitude sensor. The camera sends images into the visual front-end pipeline. The visual front end is responsible for extracting point measurements of targets from the images and computing the homography and essential matrix, as shown in Fig. 1.

The visual front end produces point measurements that are processed by the tracking back end, labeled R-RANSAC in Fig. 1, which produces tracks (position, velocity, plus error covariances) of targets. Target tracking is done in the current image frame which requires transforming measurements and tracks expressed in the previous camera fame to the current frame.

Since we assume that the target is moving on a fairly planar ground, we can use the homography matrix to transform the measurements and tracks to the current image frame, as shown in Fig. 1.

The multiple target-tracking algorithm R-RANSAC sends the tracks to the track selector which determines which track to follow. The selected track is then passed to the controller which sends commands to the flight control unit (FCU) that enables target following, as shown in Fig. 1.

Visual Front End

This section describes the visual front end shown in Fig. 1.

Images from the camera along with the camera's intrinsic parameters are given to the visual front end, which is responsible for producing measurements and calculating the homography and essential matrix. The visual front end uses a variety of algorithms to extract features. Some of the algorithms we have used include image differencing, color segmentation, YOLO, and apparent feature motion methods to extract measurements. The image difference method finds the difference between two images, looks for disparities of certain shapes and sizes (blobs) which are caused by moving objects, and takes the center of the blobs as measurements. When the camera is moving, we use the homography to transform one image into the frame of the other image. The color segmentation method looks for blobs with specific color, size, and shape profile to extract point measurements. This method of course assumes that the target of interest is a unique color, and is in general not very useful except in simple controlled environments.

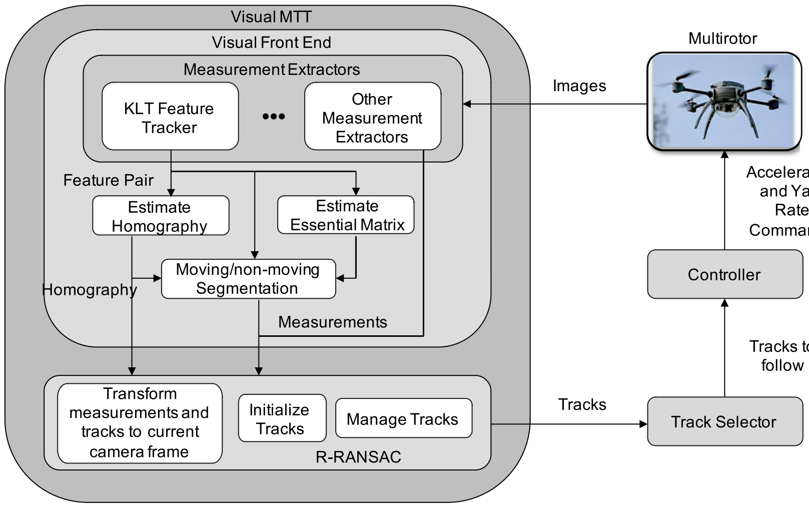


Fig. 1 Target-tracking and following architecture

The method that we use to implement the visual front end is described in the remainder of this section and is shown graphically in Fig. 1. In particular, the KLT feature tracker extracts matching points between consecutive images. Using the matching points, the homography is computed. The matching points that are outliers to the homography are considered moving. Their motion can be caused by a moving target, noise, or parallax. Motion caused by parallax is filtered out using the essential matrix.

KLT Feature Tracker

In order to compute the homography matrix, the essential matrix, and to calculate apparent feature motion, good features need to be tracked between consecutive frames. A common and popular method is to use "Good Features to Track" [21] to select good features from the previous image and find their corresponding features in the current image using the Kanade-Lucas-Tomasi (KLT) feature tracker [22, 23]. The combination of the two algorithms yields matching features. These algorithms can be implemented using the OpenCV functions `goodFeaturesToTrack()` and `calcOpticFlowPyrLK()` [24].

Estimate Homography

The homography describes the transformation between image frames and maps static features from one image to static features in the other image. Thus, if we map the matched features into the same image frame and compare the distance from their matched counterpart, we can identify which features correspond to static objects and moving object.

The matching features obtained from the KLT Tracker are used to estimate the homography. The relevant geometry of the Euclidian homography is shown in Fig. 2.

Suppose that $\boldsymbol{p}_f$ is a feature point that lies on a plane defined by the normal (unit) vector $\boldsymbol{n}$. Let $\boldsymbol{p}^a_{f/a}$ and $\boldsymbol{p}^b_{f/b}$ be the position of $\boldsymbol{p}_f$ relative to frames a and b, expressed in those frames respectively. Then, as shown in Fig. 2, we have

$$\boldsymbol{p}^b_{f/b} = R^b_a \boldsymbol{p}^a_{f/a} + \boldsymbol{p}^b_{a/b}.$$

Let d_a be the distance from the origin of frame a to the planar scene, and observe that

$$d_a = \boldsymbol{n}^\top \boldsymbol{p}^a_{f/a} \qquad \Longrightarrow \qquad \frac{\boldsymbol{n}^\top \boldsymbol{p}^a_{f/a}}{d_a} = 1.$$

Therefore, we get that

$$\boldsymbol{p}^b_{f/b} = \left(R^b_a + \frac{\boldsymbol{p}^b_{a/b}}{d_a} \boldsymbol{n}^\top \right) \boldsymbol{p}^a_{f/a}. \tag{1}$$

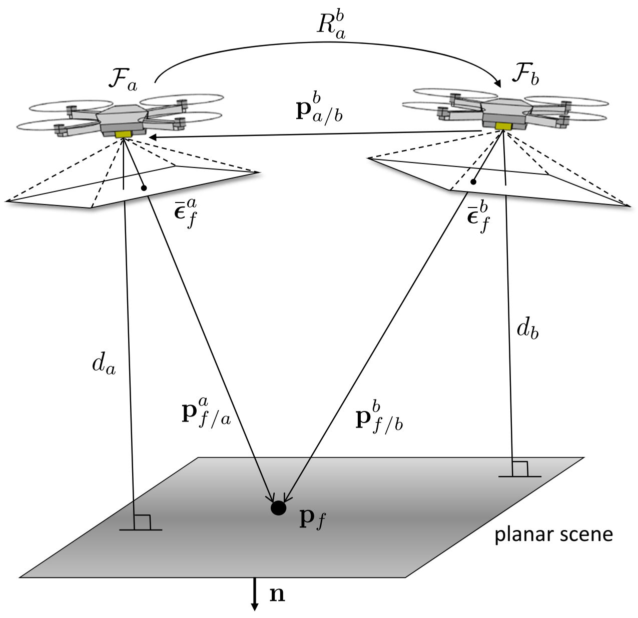


Fig. 2 The geometry for the derivation of the homography matrix between two camera poses

Let $\boldsymbol{p}^a_{f/a} = (p_{xa}, p_{ya}, p_{za})^\top$ and $\boldsymbol{p}^b_{f/b} = (p_{xb}, p_{yb}, p_{zb})^\top$, and let $\epsilon^a_{f/a} = (p_{xa}/p_{za}, p_{ya}/p_{za}, 1)^\top$ represent the normalized homogeneous coordinates of $\boldsymbol{p}^a_{f/a}$ projected onto image plane a, and similarly for $\epsilon^b_{f/b}$. Then, Eq. 1 can be written as

$$\frac{p_{zb}}{p_{za}} \epsilon^b_{f/b} = \left(R^b_a + \frac{\boldsymbol{p}^b_{a/b}}{d_a} \boldsymbol{n}^\top \right) \epsilon^a_{f/a}. \tag{2}$$

Defining the scalar $\gamma_f = p_{zb}/p_{za}$, we get

$$\gamma_f \epsilon^b_{f/b} = H^b_a \epsilon^a_{f/a}, \tag{3}$$

where

$$H^b_a \triangleq \left(R^b_a + \frac{\boldsymbol{p}^b_{a/b}}{d_a} \boldsymbol{n}^\top \right) \tag{4}$$

is called the Euclidian homography matrix between frames a and b [25]. Equation 3 demonstrates that the Euclidian homography matrix H^b_a transforms the normalized homogeneous pixel location of $\boldsymbol{p}$ in frame a into a homogeneous pixel location of $\boldsymbol{p}$ in frame b. The scaling factor γ, which is feature point dependent, is required to put ϵ^b in normalized homogeneous coordinates, where the third element is unity.

The homography can be calculated using the `openCV` function `findHomography()`, which combines a 4-point algorithm with a RANSAC [26, 27] process to find the homography matrix that best fits the data. The `findHomography()` function scales the elements of H so that the $(3, 3)$ element is equal to one. Feature pairs that do not satisfy (3) are labeled as features that are potentially moving in the environment.

Estimate Essential Matrix

The homography works well to segment between moving and non-moving features provided that the scene is planar; however, that is rarely the case due to trees, lamp posts, and other objects that stick out of the ground. The objects that do not lie on the same plane used to describe the homography will be outliers to the homography and appear as moving features even if they are static. The essential matrix provides a strategy to filter out the static features using the epipolar constraint.

Figure 3 shows the essence of epipolar geometry. Let $\boldsymbol{p}_f$ be the 3D position of a feature point in the world, and let $\boldsymbol{p}^a_{f/a}$ be the position vector of the feature point relative to frame $\mathcal{F}_a$ expressed in frame $\mathcal{F}_a$, and similarly for $\boldsymbol{p}^b_{f/b}$.

The relationship between $\boldsymbol{p}^a_{f/a}$ and $\boldsymbol{p}^b_{f/b}$ is given by

$$\boldsymbol{p}^b_{f/b} = R^b_a \boldsymbol{p}^a_{f/a} + \boldsymbol{p}^b_{a/b}. \tag{5}$$

Defining

$$\left\lfloor \begin{pmatrix} a \\ b \\ c \end{pmatrix} \right\rfloor = \begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix}$$

as the cross-product operator, then multiplying both sides of Eq. 5 on the left by $\lfloor \boldsymbol{p}^b_{a/b} \rfloor$ gives

$$\lfloor \boldsymbol{p}^b_{a/b} \rfloor \boldsymbol{p}^b_{f/b} = \lfloor \boldsymbol{p}^b_{a/b} \rfloor R^b_a \boldsymbol{p}^a_{f/a}.$$

Since $\lfloor \boldsymbol{p}^b_{a/b} \rfloor \boldsymbol{p}^b_{f/b} = \boldsymbol{p}^b_{a/b} \times \boldsymbol{p}^b_{f/b}$ must be orthogonal to $\boldsymbol{p}^b_{f/b}$ we have that

$$\boldsymbol{p}^{b\top}_{f/b} \lfloor \boldsymbol{p}^b_{a/b} \rfloor R^b_a \boldsymbol{p}^a_{f/a} = 0. \tag{6}$$

Dividing (6) by the norm of $\boldsymbol{p}^b_{a/b}$, and defining

$$\boldsymbol{t}^b_{a/b} \triangleq \frac{\boldsymbol{p}^b_{a/b}}{\left\| \boldsymbol{p}^b_{a/b} \right\|}$$

gives

$$\boldsymbol{p}^{b\top}_{f/b} \lfloor \boldsymbol{t}^b_{a/b} \rfloor R^b_a \boldsymbol{p}^a_{f/a} = 0. \tag{7}$$

The matrix

$$E^b_a = \lfloor \boldsymbol{t}^b_{a/b} \rfloor R^b_a \tag{8}$$

is called the essential matrix and is completely defined by the relative pose $(R^b_a, \boldsymbol{p}^b_{a/b})$.

Dividing (7) by the distances to the feature in each frame gives

$$\epsilon^{b\top}_{f/b} E^b_a \epsilon^a_{f/a} = 0, \tag{9}$$

where $\epsilon^a_{f/a}$ and $\epsilon^b_{f/b}$ are the normalized homogeneous image coordinates of the feature in frame a (respectively frame b). This equation is the epipolar constraint and serves as a constraint between static point correspondences.

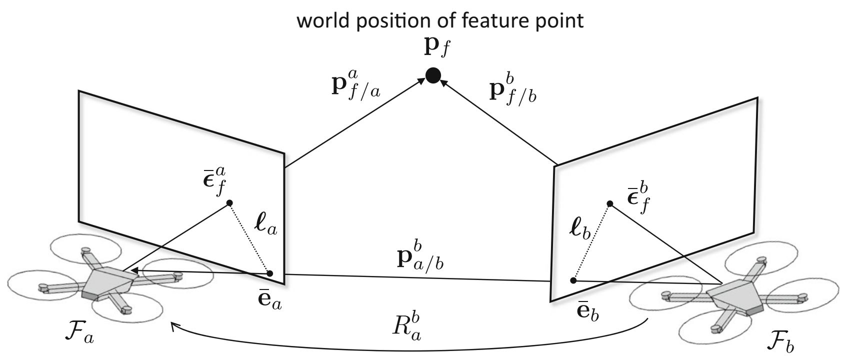
The epipoles $\bar{e}_a$ and $\bar{e}_b$ shown in Fig. 3 are the intersection of the line connecting $\mathcal{F}_a$ and $\mathcal{F}_b$ with each image plane. The epipolar lines ℓ_a and ℓ_b are the intersection of the plane $(\mathcal{F}_a, \mathcal{F}_b, P_f)$ with the image planes. The epipolar constraint with the pixel $\epsilon^a_{f/a}$ is satisfied by any pixel on the epipolar line ℓ_b. In other words, if P_f is a static feature or its motion is along the epipolar line then its point correspondence $\epsilon^a_{f/a}$ and $\epsilon^b_{f/b}$ will satisfy the epipolar constraint [28].

The essential matrix can be calculated using the `openCV` function `findEssentialMat()` which uses the five-point Nister algorithm [29] coupled with a RANSAC process.

Moving/Non-moving Segmentation

This section describes the "Moving/non-moving Segmentation" block shown in Fig. 1. The purpose of this block is to segment the tracked feature pairs into those that are stationary in the environment, and those that are moving relative to the environment. As shown in Fig. 1, the inputs to the "Moving/non-moving Segmentation" block at time k are the

Fig. 3 Epipolar geometry



homography H_{k-1}^{k}, the essential matrix E_{k-1}^{k}, and the set of matching feature points $\mathcal{M}_k = \{(\epsilon_i^k, \epsilon_i^{k-1})$ between image $\mathcal{I}_{k-1}$ and image $\mathcal{I}_k$.

When the camera is mounted on a moving UAV observing a scene where most of the objects in the scene are not moving, the homography computed from planar matching features will correspond to the motion of the UAV. As previously stated, moving objects or static objects not coplanar with the features used to compute the homography will appear to have motion when their corresponding features from the previous image are mapped to the current image. Therefore, given the set of matching feature points $\mathcal{M}_k$, we can segment $\mathcal{M}_k$ into two disjoint sets $\mathcal{M}_k^{in}$ for inliers and $\mathcal{M}_k^{out}$ for outliers where, for some small $\eta_1 > 0$

$$\mathcal{M}_k^{in} = \left\{(\epsilon_i^k, \epsilon_i^{k-1}) \in \mathcal{M}_k \mid \left\|\gamma_i \epsilon_i^k - H_{k-1}^k \epsilon_i^{k-1}\right\| \leq \eta_1\right\}$$
$$\mathcal{M}_k^{out} = \left\{(\epsilon_i^k, \epsilon_i^{k-1}) \in \mathcal{M}_k \mid \left\|\gamma_i \epsilon_i^k - H_{k-1}^k \epsilon_i^{k-1}\right\| > \eta_1\right\}.$$

Therefore, $\mathcal{M}_k^{in}$ are all matching feature pairs that are explained by the homography H_{k-1}^k, and therefore correspond to ego-motion of the UAV, and $\mathcal{M}_k^{out}$ are all matching feature pairs that are not explained by the homography H_{k-1}^k, and therefore potentially correspond to moving objects in the environment.

Figure 4 illustrates the application of this homography segmentation scheme, where feature outliers $\mathcal{M}_k^{out}$ have been retained.

The homography matrix provides good moving/non-moving segmentation either if the motion of the UAV is purely rotational, or if the surrounding environment is planar. A planar environment may be an adequate assumption for a high-flying fixed-wing vehicle moving over mostly-flat terrain. However, it is not a good assumption for multirotor UAV moving in complex 3D environments, where non-planar, stationary features will appear to be moving due to parallax. In that case, the potentially moving features $\mathcal{M}_k^{out}$ need to be further processed to discard features from the 3D scene that are not moving.

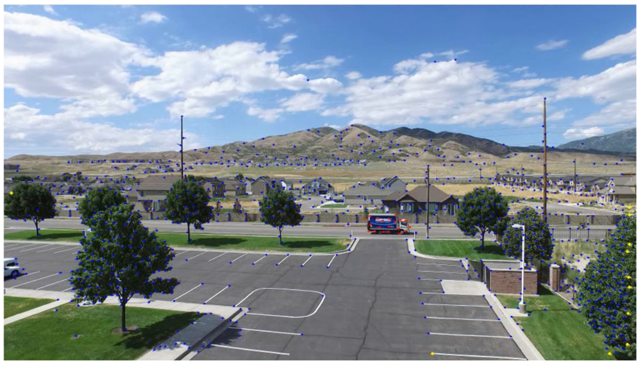


Fig. 5 Motion detection using the essential matrix. Matching pairs in $\mathcal{M}_k^{out}$ are shown in blue and red, where the red features are in $\mathcal{M}_k^{moving}$

Our approach uses the epipolar constraint given in Eq. 9 that is satisfied by stationary 3D points. Therefore, potentially moving 3D points are given by

$$\mathcal{M}_k^{moving} = \left\{(\epsilon_i^k, \epsilon_i^{k-1}) \in \mathcal{M}_k^{out} \mid \left|\epsilon_i^{k\top} E_{k-1}^k \epsilon_i^{k-1}\right| > \eta_2\right\}$$

for some small $\eta_2 > 0$.

Figure 5 illustrates the moving/non-moving segmentation scheme using video from a multirotor flying in close proximity to 3D terrain. The blue feature points correspond to features on 3D objects, which due to parallax are not discarded by the homography threshold and are therefore elements of $\mathcal{M}_k^{out}$. However, these points satisfy the epipolar constraint and therefore are not flagged as moving features. The red dots in Fig. 5 correspond to $\mathcal{M}_k^{moving}$ and are actually moving in the scene. One drawback to this approach is that features that are moving along the epipolar lines (i.e., moving in the same direction as the camera) will be filtered out. However, this can be mitigated by controlling the camera so that its motion is not aligned with the target's motion.

R-RANSAC Multiple Target Tracker

Recursive Random Sample Consensus (R-RANSAC) is a modular multiple target tracking (MTT) paradigm originally

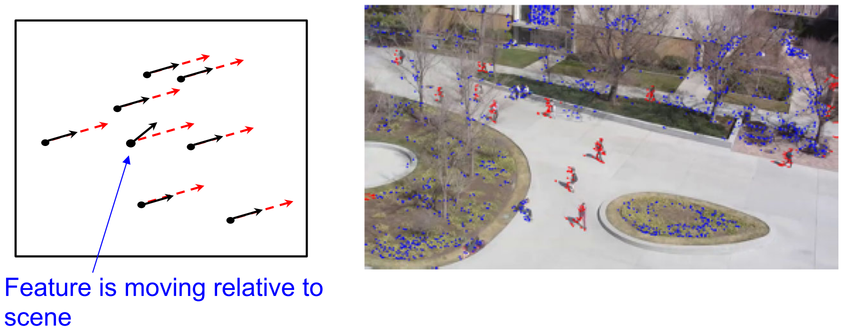


Fig. 4 Motion detection using the homography matrix. Matching features are shown in red and blue. The set $\mathcal{M}_k^{in}$ is shown in blue, and the set $\mathcal{M}_k^{out}$ is shown in red

developed in [11–15] and extended by various others [30–37]. The novel aspects of R-RANSAC include feature (measurement) and track propagation, track initialization, and track management. R-RANSAC tracks objects in the current camera frame. Since the camera frame moves as the UAV moves, features and tracks need to be transformed to the current camera frame. As new measurements are received, tracks are initialized, updated, and managed.

Transform Measurements and Tracks

This section describes the "Transform measurements and tracks to current camera frame" block shown in Fig. 1 which transform all measurements and tracks from the previous image frame to the current image frame.

We have shown how uncalibrated pixels are transformed between frames by the homography matrix as

$$\gamma_f \epsilon^b_{f/b} = H^b_a \epsilon^a_{f/b}.$$

The visual multiple target tracking algorithm produces pixel velocities, pixel accelerations, and 2×2 covariance matrices associated with each of these quantities. In this section, we show how to transform pixel velocities, accelerations, and covariances using the homography matrix.

Throughout this section, we will use the following notation. The homography matrix will be decomposed into block elements as

$$H^b_a \triangleq \begin{pmatrix} H_1 & \boldsymbol{h}_2 \\ \boldsymbol{h}_3^\top & h_4 \end{pmatrix},$$

and the homogeneous image coordinates are decomposed as $\epsilon \triangleq \begin{pmatrix} \hat{\epsilon}^\top & 1 \end{pmatrix}^\top$.

Given the relationship

$$\gamma_f \begin{pmatrix} \hat{\epsilon}^b_{f/b} \\ 1 \end{pmatrix} = \begin{pmatrix} H_1 & \boldsymbol{h}_2 \\ \boldsymbol{h}_3^\top & h_4 \end{pmatrix} \begin{pmatrix} \hat{\epsilon}^a_{f/a} \\ 1 \end{pmatrix},$$

$$\iff \begin{pmatrix} \gamma_f \hat{\epsilon}^b_{f/b} \\ \gamma_f \end{pmatrix} = \begin{pmatrix} H_1 \hat{\epsilon}^a + \boldsymbol{h}_2 \\ \boldsymbol{h}_3^\top \hat{\epsilon}^a_{f/a} + h_4 \end{pmatrix},$$

which implies that

$$\hat{\epsilon}^b_{f/b} = \frac{H_1 \hat{\epsilon}^a_{f/a} + \boldsymbol{h}_2}{\boldsymbol{h}_3^\top \hat{\epsilon}^a_{f/a} + h_4} \quad \text{and} \quad \gamma_f = \boldsymbol{h}_3^\top \hat{\epsilon}^a_{f/a} + h_4.$$

Defining the function

$$g(\hat{\epsilon}, H) \triangleq \frac{H_1 \hat{\epsilon} + \boldsymbol{h}_2}{\boldsymbol{h}_3^\top \hat{\epsilon} + h_4}, \tag{10}$$

we have that 2D pixels are transformed between frames as $\hat{\epsilon}^b_{f/b} = g(\hat{\epsilon}^a_{f/a}, H^b_a)$. Therefore, the 2D pixel velocity is transformed as

$$\dot{\hat{\epsilon}}^b_{f/b} = \frac{\partial g}{\partial \hat{\epsilon}}\Big|_{\hat{\epsilon}=\hat{\epsilon}^a_{f/a}} \dot{\hat{\epsilon}}^a_{f/a} = G(\hat{\epsilon}^a_{f/a}, H^b_a) \dot{\hat{\epsilon}}^a_{f/a}, \tag{11}$$

where

$$G(\hat{\epsilon}, H) = \frac{(\boldsymbol{h}_3^\top \hat{\epsilon} + h_4) H_1 - (H_1 \hat{\epsilon} + \boldsymbol{h}_2) \boldsymbol{h}_3^\top}{(\boldsymbol{h}_3^\top \hat{\epsilon} + h_4)^2}. \tag{12}$$

The next lemma shows how position and velocity covariances are transformed between images.

Theorem 1 *Suppose that H^b_a is the homography matrix between frames a and b and that $\hat{\epsilon}^a_{f/a}$ and $\dot{\hat{\epsilon}}^a_{f/a}$ are random vectors representing pixel location and velocity of feature f in frame a with mean $\hat{\boldsymbol{\mu}}^a_{f/a}$, and $\dot{\hat{\boldsymbol{\mu}}}^a_{f/a}$, respectively, and covariances Σ^a_p and Σ^a_v respectively. Suppose that $\hat{\epsilon}^a_{f/a}$ is transformed according to $\hat{\epsilon}^b_{f/a} = g(\hat{\epsilon}^a_{f/a}, H^b_a)$ where g is defined in Eq. 10, then the mean and covariance of $\hat{\epsilon}^b_{f/b}$ and $\dot{\hat{\epsilon}}^b_{f/b}$ are given by*

$$\begin{aligned} \hat{\boldsymbol{\mu}}^b &= g(\hat{\boldsymbol{\mu}}^a, H^b_a) \\ \dot{\hat{\boldsymbol{\mu}}}^b &= G(\hat{\boldsymbol{\mu}}^a, H^b_a) \dot{\hat{\boldsymbol{\mu}}}^a \\ \Sigma^b_p &= G(\hat{\boldsymbol{\mu}}^a, H^b_a) \Sigma^a_p G^\top(\hat{\boldsymbol{\mu}}^a, H^b_a) \\ \Sigma^b_v &= G(\hat{\boldsymbol{\mu}}^a, H^b_a) \Sigma^a_v G^\top(\hat{\boldsymbol{\mu}}^a, H^b_a) \end{aligned}$$

where G is defined in Eq. 12.

Track Initialization

Given that the measurements and tracks are expressed with respect to the same coordinate frame, we use the new measurements that do not belong to any existing track to initialize new tracks.

For simplicity, suppose that we have two observable targets whose motion can be described by a linear time-invariant model where both targets are in the camera field-of-view. Some of the camera's measurements correspond to a target while others are spurious false measurements. Since there are multiple targets and false measurements, we need a way to associate measurements to their respective targets or noise. We do this using the standard RANSAC algorithm.

Suppose that we currently have one target in the field-of-view of the camera and a batch of measurements as depicted in Fig. 6.

We take a minimum subset of measurements such that the target's trajectory can be reconstructed by the measurements in the subset and so that at least one of the measurements is from the latest time step. One particular minimum subset is depicted in Fig. 6 using red circles.

Using the minimum subset, a trajectory hypothesis is generated. The trajectory hypothesis is used to identify other measurement inliers (i.e., measurements that are close to the trajectory hypothesis). The trajectory hypothesis is then scored using the number of inliers. An example of a trajectory hypothesis is depicted in Fig. 6 by the red line.

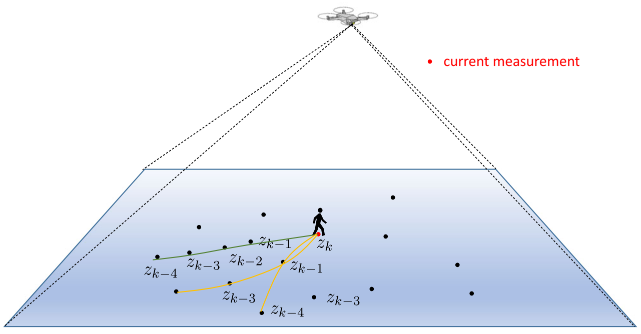


Fig. 6 Black dots indicate measurements, and the current batch of measurements are denoted with z_*. A particular minimum subset is denoted with red circle, including the current measurement z_k. A track hypothesis generated from a minimum subset of measurements, depicted with the red curve. Alternate trajectory hypotheses that are not selected are shown in yellow

This process is repeated up to a predetermined number of times. The trajectory hypothesis that has the most number of inliers is then filtered (e.g., using an EKF) to produce a new current track estimate. An example of track initialization with multiple targets is shown in Fig. 7. Alternate trajectory hypotheses that were not selected during initialization are shown in yellow in Fig. 7.

Track Management

When new measurements are received, they are associated to either an existing track or are used to initialize a new track. The measurements that are associated to a track are used to update the track. The modular design of R-RANSAC allows us to use various techniques to associate measurements and update the tracks. Some popular methods include the global nearest neighbor filter [38, 39], probabilistic data association filter [40], and joint probabilistic data association filter [40, 41]. Other possibilities include algorithms in [7].

R-RANSAC maintains a bank of M tracks. As the track initializer generates new tracks, tracks are pruned to keep the number of tracks at or below M. Every track is rated by the number of inliers it has and its lifetime. When there are more than M tracks, tracks with the lowest ratings are pruned until there are only M tracks.

As tracks are propagated and updated, they may leave the field-of-view of the camera, they may coalesce, or they may stop receiving measurements. To handle these situations, we remove tracks that have not received a measurement for a predetermined period of time, and we merge similar tracks.

Good tracks, i.e., tracks that have a high inlier ratio, are given a unique numerical track ID. The good tracks passed to the track selection algorithm at every time step.

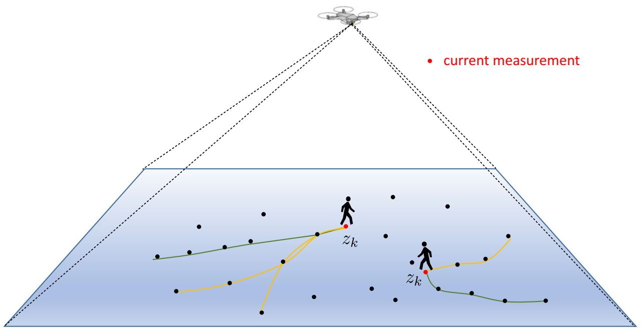


Fig. 7 Track initialization for multiple targets

Track Selection

R-RANSAC passes good tracks to the track selector which chooses a track to follow. In this section, we list several possible options for target selection.

Target Closest to the Image Center One option is to follow the track that is closest to the image center. If visual-MTT returns a set of normalized image coordinates ϵ_i for the tracks, then select the track that minimizes $\|\epsilon_i\|$.

Target Recognition A common automatic method for track selection is target recognition using visual information. This method compares the tracks to a visual profile of the target of interest. If a track matches the visual profile, then it is followed. A downside of this method requires the visual profile to be built previously. For visual target recognition algorithms, see [42–44].

User Input A manual method for track selection is to query a user about which track should be followed. After the user has been queried, a profile of the target using gathered data can be made to recognize the track in the future. One example of this is [45] which uses a DNN to build the visual profile online.

The selected track is communicated to the target following controller.

Target-Following Controller

This section overviews one possible target-following controller as shown in Fig. 1. The controller consists of three parts: (1) a PID strategy that uses a height-above-ground sensor to maintain a constant pre-specified height above the ground, (2) a position controller that follows the target based on the track information, and (3) a heading controller that aligns the UAV's heading with the target's heading. In this section, we describe the position and heading controllers in detail.

The provided track contains the state estimate of the target in normalized image coordinates. Image coordinates are not invariant to the roll and pitch of the UAV; therefore, we design the controller in the normalized virtual image plane.

Let $\boldsymbol{p}^c_{t/c}$ denote the position of the target relative to the camera expressed in the camera frame; the track produced by R-RANSAC is in normalized image coordinates and is given by

$$\epsilon^c_{t/c} = \frac{K_c^{-1}\boldsymbol{p}^c_{t/c}}{\boldsymbol{e}_3^\top K_c^{-1}\boldsymbol{p}^c_{t/c}}$$

where K_c is the camera intrinsic parameter [46]. The target's velocity is given by $\dot{\epsilon}^c_{t/c}$. Note that the third element of $\epsilon^c_{t/c}$ is 1, and the third element of $\dot{\epsilon}^c_{t/c}$ is 0.

The coordinate axes in the camera frame are defined so that the z-axis points along the optical axis, the x-axis points to the right when looking at the image from the optical center in the direction of the optical axis, and the y-axis points down in the image, to form a right-handed coordinate system. Alternatively, the virtual camera frame is defined so that the z-axis points down toward the ground, i.e., is equal to $\boldsymbol{e}_3$, and the x and y axes are projections of the camera x and y axis onto the plane orthogonal to $\boldsymbol{e}_3$. A notional depiction of the camera and virtual camera frame is shown in Fig. 8.

The virtual camera frame is obtained from the camera frame through a rotation that aligns the optical axis with the down vector $\boldsymbol{e}_3$. The rotation, denoted R^v_c, is a function of the roll and pitch angles of the multirotor, as well as the geometry of how the camera is mounted to the vehicle.

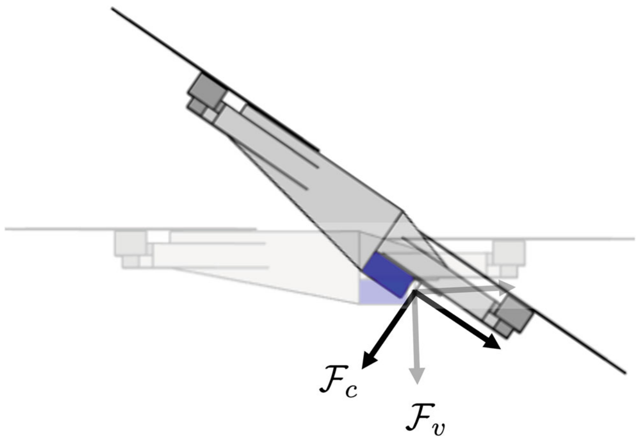


Fig. 8 A notional depiction of the camera frame and the virtual camera frame. The optical axis of the virtual camera frame is the projection of the optical axis of the camera frame onto the down vector $\boldsymbol{e}_3$

Therefore, the normalized virtual image coordinates of the track in the virtual camera frame are given by

$$\epsilon^v_{t/c} = \frac{R^v_c \epsilon^c_{t/c}}{\boldsymbol{e}_3^\top R^v_c \epsilon^c_{t/c}}. \tag{13}$$

Similarly, the pixel velocity in normalized virtual image coordinates is given by

$$\dot{\epsilon}^v_{t/c} = \frac{1}{(\boldsymbol{e}_3^\top R^v_c \epsilon^c_{t/c})^2}\left((\boldsymbol{e}_3^\top R^v_c \epsilon^c_{t/c})I - R^v_c \epsilon^c_{t/c}\boldsymbol{e}_3^\top\right) \cdot \left(R^v_c \lfloor \boldsymbol{\omega}^c_{c/v} \rfloor \epsilon^c_{t/c} + R^v_c \dot{\epsilon}^c_{t/c}\right). \tag{14}$$

Equations 13 and 14 are computed by vision data using the R-RANSAC tracker described in the previous section. We also note that $\epsilon^v_{t/c}$ is simply the normalized line-of-sight vector expressed in the virtual camera frame, i.e.,

$$\epsilon^v_{t/c} = \frac{\boldsymbol{p}^v_{t/c}}{\boldsymbol{e}_3^T \boldsymbol{p}^v_{t/c}} = \lambda \boldsymbol{p}^v_{t/c},$$

where $\lambda = 1/(\boldsymbol{e}_3^T \boldsymbol{p}^v_{t/c})$ is the constant height-above-ground. In addition, we have that

$$\ddot{\epsilon}^v_{t/c} = \lambda \ddot{\boldsymbol{p}}^v_{t/c} = \lambda\left(\ddot{\boldsymbol{p}}^v_{t/i} - \ddot{\boldsymbol{p}}^v_{c/i}\right),$$

where $\dot{\boldsymbol{p}}^v_{t/i}$ and $\dot{\boldsymbol{p}}^v_{c/i}$ are the inertial velocities of the target and camera, and $\ddot{\boldsymbol{p}}^v_{t/i}$ and $\ddot{\boldsymbol{p}}^v_{c/i}$ are the inertial accelerations of the target and camera, all expressed in the virtual camera frame.

If we assume that the inertial acceleration of the target is 0, and that the center of the camera frame is the center of the multirotor body frame, then

$$\ddot{\epsilon}^v_{t/c} = -\lambda \boldsymbol{a}^v,$$

where $\boldsymbol{a}^v = \ddot{\boldsymbol{p}}^v_{b/i} = \ddot{\boldsymbol{p}}^v_{c/i}$ is the commanded acceleration of the multirotor.

We now have the following theorem.

Theorem 2 *Assume that the inertial acceleration of the target is 0, and that the height-above-ground is constant and known. Let $\epsilon^v_{d/c}$ be the desired constant normalized line-of-sight vector to the target, and let*

$$\boldsymbol{a}^v = \frac{1}{\lambda}\left((k_1+k_2)\dot{\epsilon}^v_{t/c} + k_1 k_2\left(\epsilon^v_{t/c} - \epsilon^v_{d/c}\right)\right), \tag{15}$$

where $k_1 > 0$ and $k_2 > 0$ are control gains, then

$$\epsilon^v_{t/c} \to \epsilon^v_{d/c}.$$

The desired attitude is selected to align with the target's velocity vector $\dot{\boldsymbol{p}}^n_{t/i}$ as follows:

$$R^i_d = \left(\boldsymbol{r}_1\ \boldsymbol{r}_2\ \boldsymbol{r}_3\right) \tag{16}$$

$$\boldsymbol{r}_1 = \frac{(I - \boldsymbol{e}_3\boldsymbol{e}_3^\top)\dot{\boldsymbol{p}}^v_{t/i}}{\left\|(I - \boldsymbol{e}_3\boldsymbol{e}_3^\top)\dot{\boldsymbol{p}}^v_{t/i}\right\|} \tag{17}$$

$$\boldsymbol{r}_2 = \boldsymbol{r}_1 \times \boldsymbol{e}_3 \tag{18}$$

$$\boldsymbol{r}_3 = \boldsymbol{e}_3. \tag{19}$$

Therefore, the x-axis of the desired frame points in the direction of the desired velocity vector, and the attitude is otherwise aligned with the body-level frame. The attitude control scheme is derived using the technique given in [47].

Following Multiple Targets

We briefly mention two approaches to the following multiple targets. If the targets are clustered together, then the following can be achieved by aligning their average position with the camera's optical center using a technique similar to the one presented in this paper. A more realistic and common approach is a decentralized multiple target tracking scheme that uses a fleet of UAVs to cooperatively track targets in their respective surveillance region and share their information via a communication network [48].

Results

We implemented the target tracking and following pipeline in simulation using PX4 software-in-the-loop with Gazebo and ROS [49]. We used the IRIS multirotor model with a camera pitched down by 45° provided by the PX4. We used default simulated noise values. We had a single target move in a square upon command. For simplicity, we had the UAV find the target using visual MTT before telling the target to move. Once the target began moving, the UAV followed it fairly well in the normalized virtual image plane. Figure 9 shows the error plots. Notice that the yaw angle has large increases in error at several points. This is when the target is turning 90°. These turns also impact the error in the northeast plane. The results show the effectiveness of the complete pipeline and its robustness to target modeling errors.

A video of the simulation is at https://youtu.be/C6JWr1dGsBQ.

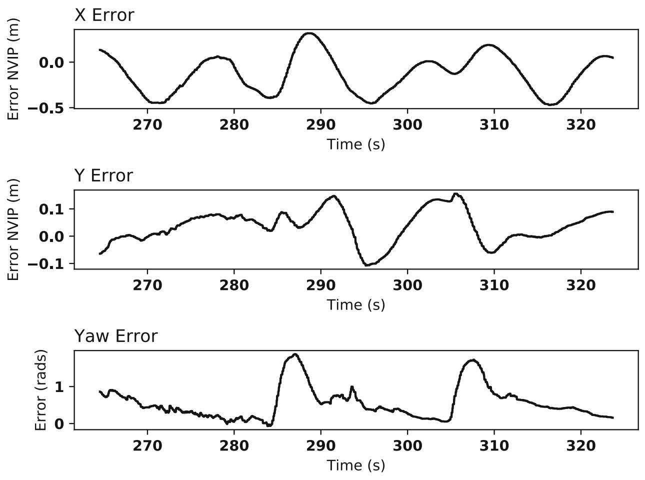


Fig. 9 The X and Y errors are in the normalized virtual image plane in units of meters and the yaw error is in units of radians

Conclusions

We have presented a review of a complete pipeline for tracking and following a target using a fixed monocular camera on a multirotor UAV. In future work, we plan to improve the controller to track multiple targets simultaneously, and incorporate target recognition for when tracks leave the camera field-of-view.

Funding This work has been funded by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation Industry/University Cooperative Research Center (I/UCRC) under NSF award Numbers IIP-1161036 and CNS-1650547, along with significant contributions from C-UAS industry members.

Compliance with Ethical Standards

Conflict of Interest Mr. Petersen has nothing to disclose. Mr. Samuelson has nothing to disclose. Dr. Beard reports grants from National Science Foundation, during the conduct of the study; In addition, Dr. Beard has a patent 10,339,387 issued.

Human and Animal Rights and Informed Consent This article does not contain any studies with human or animal subjects performed by any of the authors.

References

1. Redmon J, Farhadi A. YOLO9000: Better, faster, stronger. Arxiv:1612.08242. 2016.
2. Girshick R, et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pat- tern Recognition; 2014. p. 580–587. issn: 10636919. https://doi.org/10.1109/CVPR.2014.81. arXiv:1311.2524.
3. Zhao ZQ, et al. Object detection with deep learning: a review. IEEE Trans Neural Netw Learn Syst. 2019;30.11:3212–3232. issn: 21622388. https://doi.org/10.1109/TNNLS.2018.2876865. arXiv:1807.05511.
4. Pulford GW. Taxonomy of multiple target tracking methods. IEE Proc-Radar Sonar Navigat. 2005;152.4:291–304. issn: <null>. https://doi.org/10.1049/ip-rsn. http://arxiv.org/abs/1409.7618.
5. Blackman SS. Multiple hypothesis tracking for multiple target tracking. IEEE Aerosp Electron Syst Mag. 2004;19.1:5–18.

6. Cho S, et al. A vision-based detection and tracking of airborne obstacles in cluttered environment. In: Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS). Philidelphia; 2012. p. 475–488. https://doi.org/10.1007/s10846-012-9702-9.
7. Bar-Shalom Y, Willett P, Tian X. Tracking and data fusion: a handbook of algorithms. YBS Publishing; 2011. isbn: 9780964831278.
8. Kurien T. Issues in the design of practical multitarget tracking algorithms. In: Multitarget-multisensor tracking: advanced applications; 1990. p. 43–83.
9. Neira J, Tardos JD. Data association in stochastic mapping using the joint compatibility test. IEEE Trans Robot Autom. 2001;17.6:890–897.
10. Fortmann TE, Bar-Shalom Y, Scheffe M. Multi-target tracking using joint probabilistic data association. In: IEEE Conference on Decision and Control including the Symposium on Adaptive Processes; 1980. p. 807–812.
11. Niedfeldt PC, Beard RW. Recursive RANSAC: Multiple signal estimation with outliers. Vol. 9. PART 1. IFAC; 2013. p. 430–435. isbn: 9783902823472. https://doi.org/10.3182/20130904-3-FR-2041.00213.
12. Niedfeldt PC, Beard RW. Multiple target tracking using recursive RANSAC. In: Proceedings of the American Control Conference; 2014. p. 3393–3398. issn: 07431619. https://doi.org/10.1109/ACC.2014.6859273.
13. Niedfeldt PC. Recursive-RANSAC: a novel algorithm for tracking multiple targets in clutter. In: All Theses and Dissertations; 2014, Paper 4195. http://scholarsarchive.byu.edu/etd/4195.
14. Niedfeldt PC, Beard RW. Convergence and complexity analysis of recursive- RANSAC: a new multiple target tracking algorithm. IEEE Trans Autom Control. 2016;61.2:456–461. issn: 00189286. https://doi.org/10.1109/TAC.2015.2437518.
15. Niedfeldt PC, Ingersoll K, Beard RW. Comparison and analysis of recursive- RANSAC for multiple target tracking. In: IEEE Trans Aerosp Electron Syst. 2017;53.1. This article compares recursive- RANSAC with other multiple target tracking methods and gives a brief tutorial on Recrusive-RANSAC., p. 461–476. issn: 00189251. https://doi.org/10.1109/TAES.2017.2650818.
16. Hutchinson S, Hager GD, Corke PI. A tutorial on visual servo control. IEEE Trans Robot Autom. 1996;12.5:651–670. issn: 1042296X. https://doi.org/10.1109/70.538972.
17. Pebrianti D, et al. Intelligent control for visual servoing system. Ind J Electr Eng Comput Sci. 2017;6.1:72–79. issn: 25024760. https://doi.org/10.11591/ijeecs.v6.i1.pp72-79.
18. Corke PI. Spherical image-based visual servo and structure estimation. In: Proceedings - IEEE International Conference on Robotics and Automation; 2010. p. 5550–5555. issn: 10504729. https://doi.org/10.1109/ROBOT.2010.5509199.
19. Liu N, Shao X. Desired compensation RISE-based IBVS control of quadrotors for tracking a moving target. Nonlinear Dyn. 2019;95.4:2605–2624. issn: 1573269X. https://doi.org/10.1007/s11071-018-4700-5.
20. Xie H, Lynch A. Dynamic image-based visual servoing for unmanned aerial vehicles with bounded inputs. In: Canadian Conference on Electrical and Computer Engineering; 2016. p. 1–5. issn: 08407789. https://doi.org/10.1109/CCECE.2016.7726618.
21. Shi J, Tomasi C. Good features to track. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94. IEEE; 1994. p. 593–600.
22. Lucas BD, Kanade T. An iterative image registration technique with an application to stereo vision. In: Proceedings of the Imaging Understanding Workshop; 1981. p. 121–130.
23. Tomasi C, Kanade T. Detection and tracking of point features. In: Carnegie Mellon University technical report CMU-CS-91-132. 1991.
24. Bradski G. The openCV Library. In: Dr. Dobb's journal of software tools. 2000.
25. Kaiser MK, Gans NR, Dixon WE. Vision- based estimation for guidance, navigation, and control of an aerial vehicle. IEEE Trans Aerosp Electron Syst. 2010;46.3:1064–1077.
26. Fischler MA, Bolles RC. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM. 1981;24.6:381–395.
27. Choi S, Kim T, Yu W. Performance evaluation of RANSAC family. In: British Machine Vision Conference, BMVC 2009 - Proceedings; 2009. https://doi.org/10.5244/C.23.81.
28. Ma Y, et al. An invitation to 3-D vision from images to geometric models: Springer; 2010.
29. Nister D. An efficient solution to the five-point relative pose problem. IEEE Trans Pattern Anal Mach Intell. 2004;26.6:756–770.
30. DeFranco PC. Detecting and tracking moving objects from a small unmanned air vehicle: Thesis Brigham Young University, MA. 2015.
31. Ingersoll K, Niedfeldt PC, Beard RW. Multiple target tracking and stationary object detection in video with recursive-RANSAC and tracker-sensor feedback. In: 2015 Interna- tional Conference on Unmanned Aircraft Systems, ICUAS 2015; 2015. p. 1320-1329. https://doi.org/10.1109/ ICUAS.2015.7152426.
32. Ingersoll K. Vision based multiple target tracking using recursive RANSAC: Phd thesis Brigham Young University; 2015.
33. Millard J. Multiple target tracking in realistic environments using recursive-RANSAC in a data fusion framework: PhD thesis. Brigham Young Universityl; 2017. p. 82. http://hdl.lib.byu.edu/1877/etd9640.
34. Wikle JK. Integration of a complete detect and avoid system for small unmanned aircraft systems. In: All Theses and Dissertations; 2017. This paper presents important improvements to recursive RANSAC, such as track initialization optimization, and extending R-RANSAC to nonlinear systems.
35. White J. Real-time visual multi-target tracking: PhD thesis. Brigham Young University; 2019. isbn: 9788578110796. https://doi.org/10.1017/CBO9781107415324.004. arXiv:1011.1669v3.
36. Yang F, Tang W, Lan H. A density-based recursive RANSAC algorithm for unmanned aerial vehicle multi-target tracking in dense clutter. In: IEEE International Confer- ence on Control and Automation, ICCA k 1; 2017, p. 23–27. issn: 19483457. https://doi.org/10.1109/ICCA. 2017.8003029.
37. Yang F, Tang W, Liang Y. A novel track initialization algorithm based on random sample consensus in dense clutter. Int J Adv Robot Syst. 2018;15.6:1–11. issn: 17298814. https://doi.org/10.1177/1729881418812632.
38. Bhatia N, Vandana. Survey of nearest neighbor techniques. Int J Comput Sci Inf Secur. 2010;8.2:302–305. 1007.0085.
39. Konstantinova P, Udvarev A, Semerdjiev T. A study of a target tracking method using Global Nearest Neighbor algorithm. In: International Conference on Computer Systems and Technologies; 2003. issn: 0042-8469.
40. Bar-Shalom Y, Daum F, Huang J. The probabilistic data association filter. In: IEEE Control systems 29.6. 2009.
41. Rezatofighi S, et al. Joint probabilistic data association revisited. In: IEEE International conference on computer vision (ICCV); 2015. https://doi.org/10.1109/icr.1996.574488.
42. Zou Z, et al. Object detection in 20 years: A Survey; 2019. 1905.05055.

43. Jia L, et al. A survey of deep learningbased object detection. IEEE Access. 2019;7:128837–128868. issn: 21693536. https://doi.org/10.1109/ACCESS.2019.2939201.
44. Liu L, et al. Deep learning for generic object detection: a survey. Int J Comput Vis. 2020;128.2:261–318. issn: 15731405. https://doi.org/10.1007/s11263-019-01247-4. arXiv:1809.02165.
45. Teng E, Huang R, Iannucci B. ClickBAIT-v2: training an object detector in real-time; 2018. 1803.10358.
46. Hartley R, Zisserman A. Multiple view geometry in computer vision: Cambridge University Press; 2003.
47. Lee T, Leok M, McClamroch NH. Geometric tracking control of a Quadrotor UAV on SE(3). In: Proceedings of the IEEE Conference on Decision and Control; 2010. p. 5420–5425.
48. Farmani N, Sun L, Pack D. Tracking multiple mobile targets using cooperative unmanned aerial vehicles. In: 2015 Inter- national Conference on Unmanned Aircraft Systems, ICUAS 2015; 2015. p. 395–400. https://doi.org/10.1109/ICUAS.2015.7152315.
49. Meier L, Honegger D, Pollefeys M. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In: 2015 IEEE International Conference on Robotics and Automation (ICRA); 2015. p. 6235–6240. https://doi.org/10.1109/ICRA.2015.7140074.