J Intell Robot Syst
DOI 10.1007/s10846-014-0081-2

Joint Localization of Pursuit Quadcopters and Target Using Monocular Cues

Abdul Basit · Waqar S. Qureshi · Matthew N. Dailey · Tomáš Krajník

Received: 28 January 2014 / Accepted: 27 June 2014

Abstract Pursuit robots (autonomous robots tasked with tracking and pursuing a moving target) require accurate tracking of the target's position over time. One possibly effective pursuit platform is a quadcopter equipped with basic sensors and a monocular camera. However, the combined noise in the quadcopter's sensors causes large disturbances in the target's 3D position estimate. To solve this problem, in this paper, we propose a novel method for joint localization of a quadcopter pursuer with a monocular camera and an arbitrary target. Our method localizes both the pursuer and target with respect to a common reference frame. The joint localization method fuses the quadcopter's kinematics and the target's dynamics in a joint state space model. We show that predicting and correcting pursuer and target trajectories simultaneously produces better results than standard approaches to estimating relative target trajectories in a 3D coordinate system. Our method also comprises a computationally efficient visual tracking method capable of redetecting a temporarily lost target. The efficiency of the proposed method is demonstrated by a series of experiments with a real quadcopter pursuing a human. The results show that the visual tracker can deal effectively with target occlusions and that joint localization outperforms standard localization methods.

Keywords Quadcopters · Joint localization · Monocular cues · State estimation filters · Visual tracking · Redetection · Backprojection · Pursuit robot · AR.Drone

1 Introduction

Surveillance and monitoring using human operators can be tedious, difficult, dangerous, and error prone. Recent technological developments have enabled the use of mobile robots and vision systems for such applications. Our focus is on mobile robots that are capable of tracking and monitoring a target in scenarios such as person/child/animal monitoring or tracking a fugitive. Ground robots might be useful in some scenarios, but they are expensive and difficult to navigate,

A. Basit (✉) · W. S. Qureshi · M. N. Dailey
Asian Institute of Technology, Khlong Luang, Pathumthani 12120, Thailand
e-mail: Abdul.Basit.Khan@ait.asia; a.abdul.basit@ieee.org

W. S. Qureshi
e-mail: waqar.s.q@ieee.org,

M. N. Dailey
e-mail: mdailey@ait.asia

T. Krajník
University of Lincoln, Brayford Pool, Lincoln, UK
e-mail: tkrajnik@lincoln.ac.uk

T. Krajník
Department of Cybernetics, CTU in Prague FEE,
Prague 2, Czech Republic

Published online: 17 July 2014

as they must avoid obstacles in the field, negotiate uneven surfaces, and place sensors over a sufficient range of heights to get a good view of both the object of interest and the terrain. On the other hand, aerial robots with airborne sensors, which are capable of low-altitude flying and vertical takeoff and land (VTOL) maneuvers, would require less complex navigation and would provide a better field of view. Aerial robots are already being used for other applications such as sports assistance [18], traffic monitoring [16], fire detection [26], remote sensing [19], and precision agriculture [2, 8, 15, 17, 24].

Quadcopters are VTOL aerial vehicles with four rotary wings that can hover over a fixed location or perform quick maneuvers but may have limited on-board sensors and processing power due to low payload capacity.

We are investigating the feasibility of autonomous pursuit of targets using inexpensive quadcopters with monocular cameras. We use the Parrot AR.Drone quadcopter (shown in Fig. 1) for research purposes. The AR.Drone is an inexpensive quadcopter with a built-in control system enabling it to hover in a stationary position. For visual tracking, the quadcopter features front-facing and downward-facing cameras. Localization is aided by an accelerometer, a pressure sensor, a three-axis gyro, and a magnetometer. The robot can be teleoperated from a host machine through WiFi.

In pursuit applications where the target's position may be extremely dynamic, the UAV should be capable of fast autonomous planning and motion. It must also be able to track the target object in real time.

In this paper, we take steps toward robust UAV target pursuit. We focus on two important challenges for the monocular vision-based tracker. First, we require an appearance-based tracker that is sufficiently accurate and fast. Second, since such a tracker will necessarily be extremely noisy, we require sensor modeling and state estimation able to obtain target position estimates usable for motion planning.

In the following sections, we introduce related work in 2D visual target tracking and state estimation for quadcopters and similar UAVs, and then we outline our contributions.

1.1 Localization

A pursuit robot must keep track of its position and orientation relative to the target and surrounding environment. Localization is a form of sequential state estimation. Sequential state estimation methods are used extensively in robotics and computer vision to solve problems such as SLAM, monocular SLAM

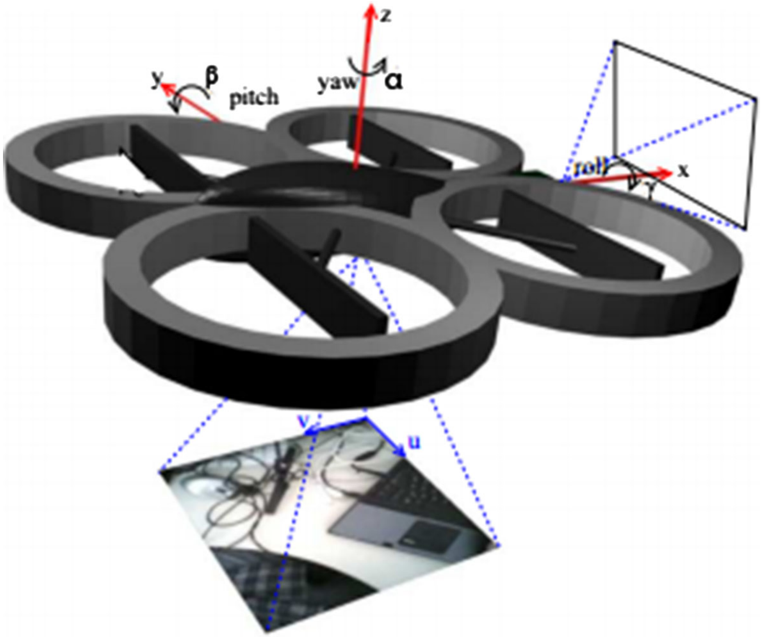


Fig. 1 Quadcopter visual sensors and coordinate system. The AR.Drone quadcopter incorporates a front-facing and downward-facing camera. Roll (γ_t^r) and pitch (β_t^r) are used to control forward-backward and left-right linear motions. Overall rotor speed is used to control upward-downward motion. Yaw control is used for change in orientation. The front-facing camera is useful in applications such as tracking and surveillance. The downward camera is useful for inspecting buildings, crops, or a field

[12], visual target tracking [25], and 3D reconstruction. Methods include Kalman filters, particle filters [29], and grid-based methods. The most widely used filtering method that supports non-linear state estimation models is the extended Kalman filter [34].

EKF models are fast and effective in any application in which the posterior state estimate is accurately represented by a Gaussian and the non-linearities in the system and sensor are moderate. EKFs have been used by several researchers to improve the performance of visual tracking methods [14, 20, 31].

Finding the position of a target relative to a UAV using a monocular camera requires extraction of depth cues from the 2D image. Monocular depth cues are extremely noisy, so relying purely on the 2D image to obtain 3D positions with depths would introduce a great deal of error. One way to improve the noisy sensor readings we would get from a 2D visual tracker is to filter sensor based estimates with a sequential state estimator.

In a previous paper [4], we propose a state estimation model based on the EKF for pursuit by SUGVs (Small Unmanned Ground Vehicles) to improve the accuracy of the estimated trajectory of both the pursuit robot and the target. The proposed method fuses robot kinematics and target dynamics to obtain superior robot and target trajectory estimates.

In this paper, based on our previous work, we propose a joint localization model that fuses quadcopter kinematics with target dynamics to improve the relative trajectory estimates of both the quadcopter and the target.

1.2 2D Object Tracking

The major methods for 2D image-based tracking use either *feature matching, optical flow, or feature histograms*. Feature matching algorithms such as SIFT [36], SURF [33], and shape matching algorithms such as contour matching [35] are too computationally expensive to be considered for real-time tracking by a small UAV with modest compute resources. Optical flow methods [13] or [30] may be within reach in terms of speed, but they do not maintain an appearance model. Histogram-based trackers, on the other hand, are not only fast, but also maintain an appearance model that is potentially useful for recovering tracking after an occlusion or reappearance in the field of view.

The Sliding window is a general approach in computer vision to look for a target object in an image. The naive sliding window approach is however, a computationally inefficient way to scan an image to search for a target. Porikli [28] propose an “integral histogram” method using integral images to speed up the process. Perreault and Hebert [27] compute histograms for median filtering efficiently by maintaining separate column-wise histograms, and, as the sliding window moves right, first updating the relevant column histogram then adding and subtracting the relevant column histograms to the histogram for the sliding window. Sizintsev et al. [32] take a similar approach to obtain histograms over sliding windows by efficiently updating the histogram using previously calculated histograms for overlapping windows.

CAMSHIFT (Continuously Adaptive Mean Shift) [1, 6] is a fast and robust feature histogram tracking algorithm potentially useful for UAVs. The method begins with manual initialization from a target image patch. It then tracks the region using a combination of color histograms, the basic mean-shift algorithm [10, 11], and an adaptive region-sizing step. It is scale and orientation invariant. Unfortunately, since the method performs a search for a local peak in the global backprojection, it is easily distracted by background objects with similar color distributions. In this paper, we incorporate an adaptive histogram similarity threshold with CAMSHIFT to help avoid tracking false targets. We also use this adaptive similarity threshold with a backprojection technique to recover the target object and reinitialize the CAMSHIFT visual tracker after an occlusion.

1.3 Monocular Visual Tracking by UAVs

Bi and Duan [5] implement a visual tracking algorithm using a low-cost quadcopter. The target is a colored landing platform that is moved manually on a cart. The quadcopter is controlled by a host machine that processes the video stream and tracks the landing platform. The host machine calculates the current position of the landing platform by using the centroid of image moments calculated for a binary image that is obtained after thresholding the green channel of the RGB image. The authors use independent controllers for pitch and roll that receive feedback from the visual tracker. The feedback input to the controllers at each frame is the current positioning error.

Kim and Shim [21] present a visual tracker and demonstrate its capabilities and usage on a tablet computer with the AR.Drone. The method uses color and image moments for the visual tracker.

All of the object tracking and navigation algorithms proposed thus for low cost UAVs use color and image moments. None of this work has demonstrated model-based object tracking that is sufficiently fast and accurate for real time control of a UAV.

1.4 Contributions

In this paper, we propose a novel joint purser and target localization model specifically for the AR.Drone that reduces position estimation error caused by monocular sensor measurements. The model maintains an estimate of the state of the target, assuming a simple linear dynamical model, as well as an estimate of the AR.Drone robot's state, assuming standard quadcopter robot kinematics [7].

We additionally propose a target tracking method suitable for quadcopters that efficiently handles both visual detection and tracking in real time. We perform target redetection when the target is occluded or leaves the field of view. We suspend tracking based on an adaptive histogram threshold. Once the object is reacquired in a redetection phase, we apply CAMSHIFT to confirm the redetected object. After CAMSHIFT validation, we reinitialize the tracker.

We thus fuse information from 2D visual tracking and the UAV's odometry measurements with knowledge of the AR.Drone's kinematics in an extended Kalman filter to obtain superior state estimation. The filter significantly improves estimation accuracy compared to standard sensor-based position estimates as well as compared to localization models not incorporating pursuit robot kinematics.

In an empirical evaluation, we show, on real-world videos, that the proposed method is a robust approach to target tracking and redetection during pursuit that is accurate, is successful at reinitialization, has very low false positive rates, and runs in real time.

2 System Design

In this section, we introduce the AR.Drone's hardware (sensors and actuators), provide details of the pursuer-pursuit-target localization method for the AR.Drone, and finally describe the visual target tracking method.

2.1 Quadcopter Overview

The AR.Drone has a carbon fiber support structure and a plastic body with removable hulls optimized for indoor and outdoor flight. The rotors are propelled by high-efficiency brushless DC motors and control circuitry. The drone is equipped with one frontal and one bottom facing camera. The control board of the AR.Drone consists of a 1-GHz ARM-Cortex-A8 32-bit processor with an 800 MHz digital signal processor and 2GB RAM. The user can control the quadcopter's roll, pitch, yaw, and vertical speed through a host machine over a WiFi link. The function of the control board is to convert user navigation commands into motor commands and to adjust the motor speeds to stabilize the drone at the required pose. Figure 2 shows how the drone communicates with the host machine. We use the video stream from the frontal camera as the main sensor for localization. The camera has a frame rate of approximately 18 fps. Estimated roll, pitch, yaw, vertical speed, and velocity in the x-y plane are estimated from the onboard accelerometer, a pressure sensor, a three-axis gyro, and a magnetometer. The estimated data are sent to the host at 200 Hz.

2.2 Mathematical Model and Algorithmic Flow

In this section, we detail the joint localization method for the quadcopter purser and target. Joint localization incorporates both target dynamics and drone kinematics to correct the sensor-based measurements of the target's and purser's state. As the main target state measurement sensor, we use the AR.Drone's front-facing monocular camera.

The tracking algorithm's flow is shown in Fig. 3. The monocular 2D visual tracker tracks the target as long as it is in the field of view. The monocular visual tracker returns a 2D tracking window indicating the target's size and position in the image. To transform 2D measurements into 3D, we take a ray from the center of the camera through the image plane at the center of the 2D target region and calculate the

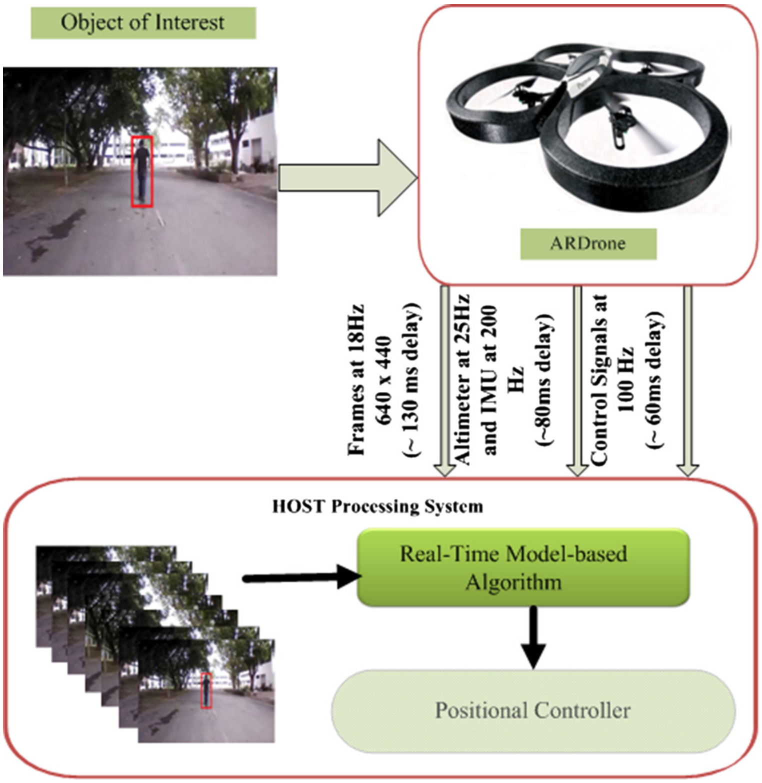

Object of Interest
ARDrone
Frames at 18Hz 640 x 440 (~ 130 ms delay)
Altimeter at 25Hz and IMU at 200 Hz (~80ms delay)
Control Signals at 100 Hz (~ 60ms delay)
HOST Processing System
Real-Time Model-based Algorithm
Positional Controller

Fig. 2 System architecture and data flow. The base station receives sensor data (images and odometry) at the given rates and latencies. The frequency of the odometry data (200 Hz) is high compared to the image transfer rate (18 fps), so it is straightforward to accumulate odometry data between images and perform a model update with the arrival of each image

depth of the target along that ray using the assumed target height in an absolute frame. Meanwhile, we acquire AR.Drone odometry data in the form, at time t, of $(\gamma_t^r, \beta_t^r, \alpha_t^r, h_t^r, \dot{x}_t^r, \dot{y}_t^r, \dot{z}_t^r)$, where $(\gamma_t^r, \beta_t^r, \alpha_t^r)$ describes the roll, pitch and yaw of the quadcopter, h_t^r is the measured altitude, and $(\dot{x}_t^r, \dot{y}_t^r, \dot{z}_t^r)$ is the linear velocity of the quadcopter in the robot coordinate system. See Fig. 4 for a visualization of the coordinate system. On the arrival of each image frame, the target tracking sensor measurement and odometry data are fed to the proposed joint localization model. The model predicts the pursuer's and target's new state based on the previous state and corrects (updates) the estimate based on the newly acquired sensor measurement. If the target is occluded or leaves the camera field of view, we stop 2D tracking and state correction, and we run a redetection algorithm until the target reappears, at which time we restart the normal flow of the algorithm.

We briefly outline the visual tracking method below and provide more details on the 3D state estimate model for the AR.Drone in the next section. Refer to Basit et al. [4] for more details on the 2D tracking algorithm, which proceeds as follows:

1. Initialize CAMSHIFT with initial image I_0 and bounding box (x_c, y_c, w, h).
2. Continue CAMSHIFT tracking, updating adaptive histogram threshold.
3. Suspend tracking when similarity between region returned by the tracker and appearance model is too low.
4. Run redetection.
5. Suspend redectection when candidate target most similar to the appearance model is similar and large enough.
6. Run CAMSHIFT to confirm redetected region. If confirmed, reinitialize tracking; otherwise, return to redetection.

2.3 Joint Estimation of AR.Drone and Target State

In this section we describe the proposed model for obtaining smooth pursuer and target trajectories in the 3D world coordinate frame.

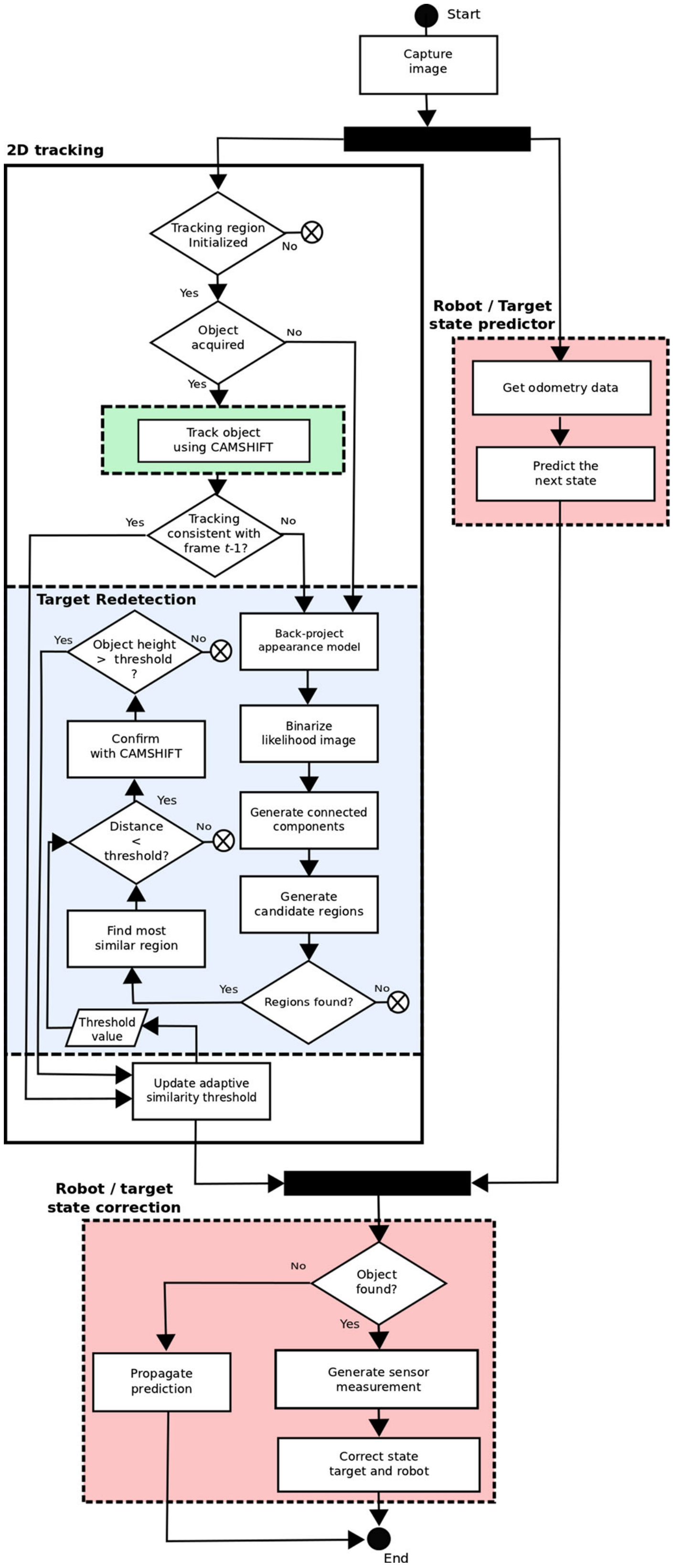

Start
Capture image
2D tracking
Tracking region Initialized
No
Yes
Object acquired
No
Yes
Track object using CAMSHIFT
Yes
Tracking consistent with frame t-1?
No
Target Redetection
Yes
Object height > threshold ?
No
Back-project appearance model
Confirm with CAMSHIFT
Binarize likelihood image
Yes
Distance < threshold?
No
Generate connected components
Find most similar region
Generate candidate regions
Yes
Regions found?
No
Threshold value
Update adaptive similarity threshold
Robot / Target state predictor
Get odometry data
Predict the next state
Robot / target state correction
No
Object found?
Yes
Propagate prediction
Generate sensor measurement
Correct state target and robot
End

Fig. 3 Algorithm flow. The green box contains the visual target tracking phase of the algorithm. The blue box contains the processing occurring in the redetection phase, when the target is not in the scene. The red box contains localization for fusing odometery and target tracking data

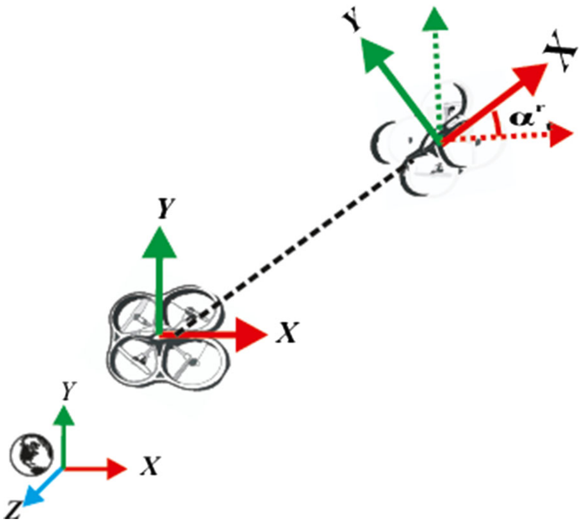


Fig. 4 Bird's-eye view of quadcopter motion in 3D world coordinate frame. The motion depends on changes of roll, pitch and altitude controls, that in turn produce 3D linear and angular motion. Linear velocity measurements $(\dot{x}_t^r, \dot{y}_t^r, \dot{z}_t^r)$ are provided by three sensors. The yaw (α_t^r) angle is used to change the orientation of the quadcopter

2.3.1 System State

The system state expresses the UAV's position and the target's position and velocity in the world coordinate frame. We define the system state at time t to be

$$\mathbf{x_t} = \left[x_t, y_t, z_t, \dot{x}_t, \dot{y}_t, \dot{z}_t, x_t^r, y_t^r, z_t^r, \gamma_t^r, \beta_t^r, \alpha_t^r\right]^T, \tag{1}$$

where (x_t, y_t, z_t) is the target's position, $(\dot{x}_t, \dot{y}_t, \dot{z}_t)$ is the target's velocity, (x_t^r, y_t^r, z_t^r) is the pursuer's position, and $(\gamma_t^r, \beta_t^r, \alpha_t^r)$ is the pursuer's 3D orientation (roll, pitch and, yaw) respectively. The positions and orientations are expressed in the world coordinate frame. The state transition model is defined as

$$\mathbf{x_{t+1}} = \mathbf{f}(\mathbf{x_t}, \mathbf{u_t}) + \nu_\mathbf{t}, \tag{2}$$

where $\nu_\mathbf{t} \sim \mathcal{N}(\mathbf{0}, \mathrm{Q}_t)$. $\mathbf{f}(\mathbf{x_t}, \mathbf{u_t})$ has two components. The first component models the AR.Drone's kinematics, assuming constant linear and angular velocity over short time periods (acceleration is modeled as noise). The second component is a first order linear dynamical system for the target's motion. We describe each component in turn. The odometry control vector is

$$\mathbf{u_t} = \left[\delta\gamma_t^r \;\; \delta\beta_t^r \;\; \delta\alpha_t^r \;\; \delta h_t^r\right]^T, \tag{3}$$

defining the change in the roll (γ_t^r), pitch (β_t^r), yaw (α_t^r), and altitude (h_t^r) of the quadcopter at time t.

The state transition for the AR.Drone is then assumed to be

$$\begin{aligned}
x_{t+1}^r &= x_t^r + \left(\dot{x}_t^r \cos\alpha_t^r - \dot{y}_t^r \sin\alpha_t^r\right) \cdot \Delta_t \\
y_{t+1}^r &= y_t^r + \left(\dot{x}_t^r \sin\alpha_t^r + \dot{y}_t^r \cos\alpha_t^r\right) \cdot \Delta_t \\
z_{t+1}^r &= z_t^r + \delta h_t^r \\
\gamma_{t+1}^r &= \gamma_t^r + \delta\gamma_t^r \\
\beta_{t+1}^r &= \beta_t^r + \delta\beta_t^r \\
\alpha_{t+1}^r &= \alpha_t^r + \delta\alpha_t^r,
\end{aligned} \tag{4}$$

where $(\dot{x}_t^r, \dot{y}_t^r)$ is the linear velocity of the quadcopter in the (x, y) plane. This velocity depends on the quadcopter's roll and pitch in the world coordinate frame. See Fig. 5. We define the dynamics

$$\begin{aligned}
\dot{x}_{t+1}^r &= \mathrm{C}_1\beta_t^r \\
\dot{y}_{t+1}^r &= \mathrm{C}_2\gamma_t^r
\end{aligned} \tag{5}$$

To transform the linear velocities from the UAV frame to the world coordinate frame, we have

$$\mathbf{v}_\mathbf{t}^\mathbf{w} = \mathrm{R}_t\dot{\mathbf{v}}_t^r, \tag{6}$$

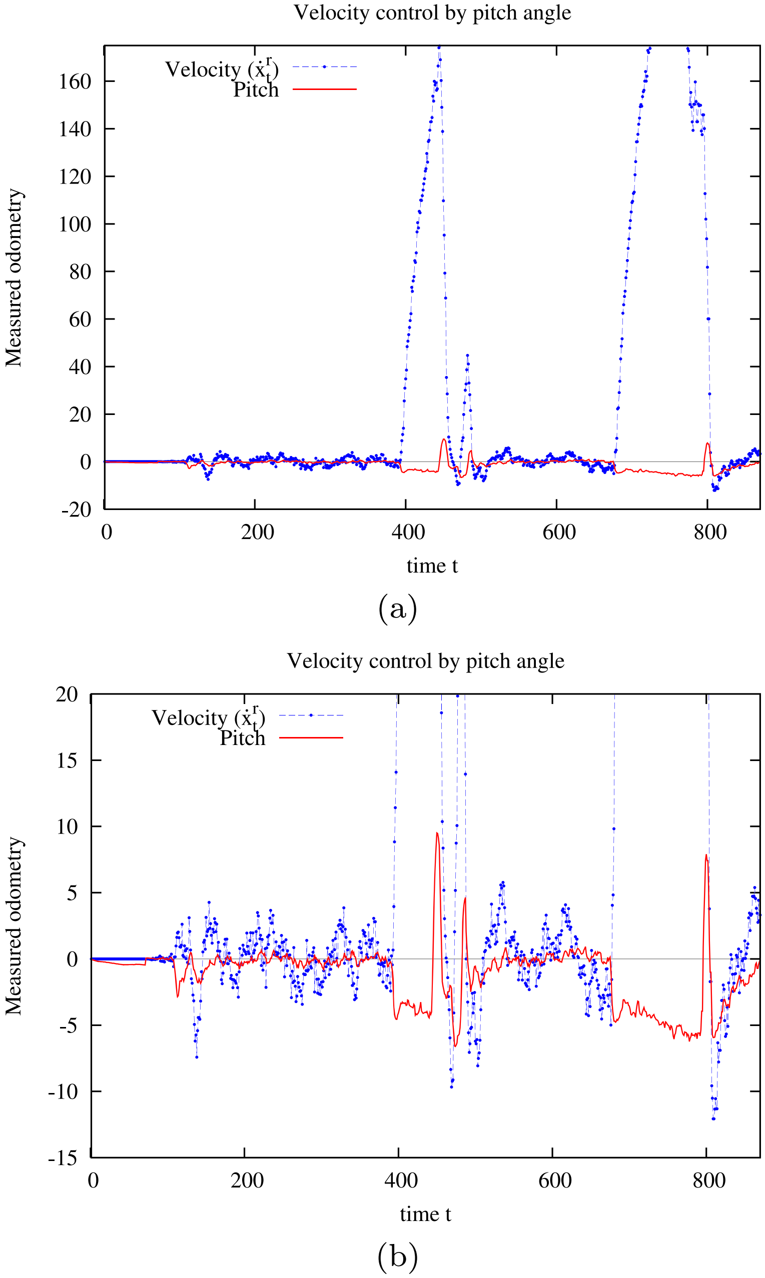
where $\dot{\mathbf{v}}_t^r$ is simply

$$\dot{\mathbf{v}}_t^r = \left[\dot{x}_t^r \;\; \dot{y}_t^r \;\; \dot{z}_t^r\right]^T, \tag{7}$$

and R_t is

$$\mathrm{R}_t = \begin{bmatrix} \mathrm{c}_{\alpha_t^r}\mathrm{c}_{\beta_t^r} & \mathrm{c}_{\alpha_t^r}\mathrm{s}_{\beta_t^r}\mathrm{s}_{\gamma_t^r} - \mathrm{s}_{\alpha_t^r}\mathrm{c}_{\gamma_t^r} & \mathrm{c}_{\alpha_t^r}\mathrm{s}_{\beta_t^r}\mathrm{c}_{\gamma_t^r} + \mathrm{s}_{\alpha_t^r}\mathrm{s}_{\gamma_t^r} \\ \mathrm{s}_{\alpha_t^r}\mathrm{c}_{\beta_t^r} & \mathrm{s}_{\alpha_t^r}\mathrm{s}_{\beta_t^r}\mathrm{s}_{\gamma_t^r} + \mathrm{c}_{\alpha_t^r}\mathrm{c}_{\gamma_t^r} & \mathrm{s}_{\alpha_t^r}\mathrm{s}_{\beta_t^r}\mathrm{c}_{\gamma_t^r} - \mathrm{c}_{\alpha_t^r}\mathrm{s}_{\gamma_t^r} \\ -\mathrm{s}_{\beta_t^r} & \mathrm{c}_{\beta_t^r}\mathrm{s}_{\gamma_t^r} & \mathrm{c}_{\beta_t^r}\mathrm{c}_{\gamma_t^r} \end{bmatrix},$$

where c. and s. are shorthand for the cosine and sine functions.



(a)

(b)

Fig. 5 Sample change in linear velocity $\dot{x}_t^r$ with respect to change in pitch angle β_t^r, for quadcopter in outdoor environment with pitch changed manually. Velocity and pitch are shown as reported by the onboard sensors. Small changes in pitch at times 400 and 700 result in changes in linear velocity. Velocity $\dot{x}_t^r$ is in cm/s, pitch is shown in degrees, and time interval is approximately 20 ms. **a** Full range of the data. **b** Zoomed view of the graph in (**a**), obtained by setting the y-range to $[-15 \cdots 20]$

Target Motion We assume a simple linear dynamics

$$
\begin{aligned}
x_{t+1} &= x_t + \Delta_t \dot{x}_t \\
y_{t+1} &= y_t + \Delta_t \dot{y}_t \\
z_{t+1} &= z_t + \Delta_t \dot{z}_t \\
\dot{x}_{t+1} &= \dot{x}_t \\
\dot{y}_{t+1} &= \dot{y}_t \\
\dot{z}_{t+1} &= \dot{z}_t
\end{aligned} \tag{8}
$$

for the target object's state.

Linearization Since $\mathbf{f}(\mathbf{x_t}, \mathbf{u_t})$ is nonlinear and we will be using an extended Kalman filter, we must approximate the system described in Eq. 2 by linearizing around an arbitrary point $\hat{\mathbf{x}}_t$. We write

$$\mathbf{f}(\mathbf{x_t}, \mathbf{u_t}) \approx \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u_t}) + \mathrm{J}_{f_t}(\mathbf{x_t} - \hat{\mathbf{x}}_t), \tag{9}$$

where $\mathtt{J}_{f_t}$ is the Jacobian

$$\mathtt{J}_{f_t} = \left[\frac{\partial \mathbf{f}(\mathbf{x_t}, \mathbf{u_t})}{\partial \mathbf{x_t}} \right] \tag{10}$$

evaluated at $\hat{\mathbf{x}}_t$. We omit the detailed Jacobian calculations.

2.3.2 Sensor Model

We assume the quadcopter's target tracking camera is mounted in a fixed position near the UAV's center of rotation with roll (rotation around the principal axis) close to 0. We also incorporate a 2D visual tracking algorithm capable of producing an estimate of the 2D position of the object's projection into the image plane at time t. We assume the operator initially defines the target object to pursue by specifying a bounding box around it in the first frame. We use visual tracker based on CAMSHIFT to track the object from frame to frame.

The measurement from the algorithm is thus simply a bounding box:

$$\mathbf{z_t} = \left[u_t, v_t, w_t^{img}, h_t^{img} \right]^T, \tag{11}$$

where (u_t, v_t) is the center and w_t^{img} and h_t^{img} are the width and height of the bounding box in the image. We model the sensor with a function $h(\cdot)$ mapping the system state $\mathbf{x_t}$ to the corresponding sensor measurement

$$\mathbf{z_t} = \mathbf{h}(\mathbf{x_t}) + \zeta_\mathbf{t}, \tag{12}$$

with $\zeta \sim \mathcal{N}(\mathbf{0}, \mathtt{S}_t)$.

For a pinhole camera with focal length f and principal point (c_x, c_y), ignoring the negligible in-plane rotation of the cylindrical object, we can write the target's center (u_t, v_t) and size (w_t^{img}, h_t^{img}) as

$$\begin{aligned} u_t &= (f x_t^{cam} + c_x)/z_t^{cam} \\ v_t &= (f y_t^{cam} + c_y)/z_t^{cam} \\ w_t^{img} &= f w_0 / z_t^{cam} \\ h_t^{img} &= f h_0 / z_t^{cam}, \end{aligned} \tag{13}$$

where (h_0, w_0) is the assumed target's height and width, $\mathbf{x}_t^{cam} = (x_t^{cam}, y_t^{cam}, z_t^{cam}, 1)^T$, is the homogeneous representation of the center of the target in the camera coordinate system, calculated as

$$\mathbf{x_t^{cam}} = \mathtt{T}_t^{W/C} \begin{bmatrix} x_t \\ y_t \\ z_t \\ 1 \end{bmatrix}, \tag{14}$$

where the transformation $\mathtt{T}_t^{W/C}$ is defined as

$$\mathtt{T}_t^{W/C} = \mathtt{T}^{R/C} \mathtt{T}_t^{W/R}, \tag{15}$$

where $\mathtt{T}_t^{W/R}$ is the rigid transformation from the world coordinate system to the robot coordinate at time t, and $\mathtt{T}^{R/C}$ is the (fixed) transformation from the robot coordinate system to the camera coordinate system. Expressing the AR.Drone orientation by a rotation matrix $\mathtt{R}^T$, at time t, we can write the transformation matrix $\mathtt{T}^{W/R}$

$$\mathtt{T}_t^{W/R} = \begin{bmatrix} \mathtt{R}_t^T & -\mathtt{R}_t^T \mathbf{x_t^r} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{16}$$

where $\mathbf{x}_t^r = \left[x_t^r \;\; y_t^r \;\; z_t^r \right]$.

As with the transition model, to linearize $\mathbf{h}(\mathbf{x_t})$ around an arbitrary point $\hat{\mathbf{x}}_t$, we require the Jacobian

$$\mathtt{J}_{h_t} = \left[\frac{\partial \mathbf{h}(\mathbf{x_t})}{\partial \mathbf{x_t}} \right]. \tag{17}$$

evaluated at an arbitrary point $\hat{\mathbf{x}}_t$.

2.3.3 Initialization

We require an a-priori state vector $\mathbf{x_0}$ to initialize the system. As explained before, we assume that the quadcopter is at the origin of the world coordinate system or an alternative initial position of the robot is given. We do not assume any knowledge of the target's initial trajectory. We can therefore treat the user-provided initial target bounding box as a first sensor measurement $\mathbf{z_0}$ and initialing the system as

$$\hat{\mathbf{x}}_0 = [x_0, y_0, z_0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T = \mathbf{h}^{inv}(\mathbf{z_0}). \tag{18}$$

To obtain (x_0, y_0, z_0), we first calculate an initial target position in the camera-coordinate frame $\mathbf{x_0^{cam}}$ then, noting that the robot frame at time 0 is also the world frame, we can map to the world coordinate frame by

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} = \left(\mathtt{T}^{R/C}\right)^{-1} \begin{bmatrix} x_0^{cam} \\ y_0^{cam} \\ z_0^{cam} \\ 1 \end{bmatrix}. \tag{19}$$

Inspecting the system in Eq. 13, we can find x_0^{cam} and y_0^{cam} given u_t and v_t if z_0^{cam} is known. We can obtain z_0^{cam} from w_0^{img} or h_0^{img}. We use $z_0^{img} = f \cdot h / h_0^{img}$, on the assumption that the user-specified bounding box is more accurate vertically than horizontally.

2.3.4 Noise Parameters

The sensor noise is given by the matrix $\mathtt{S}_t$. We assume that the measurement noise for both the bounding box center and the bounding box size are a fraction of the target's width and height in the image:

$$\mathtt{S}_t = \lambda^2 \begin{bmatrix} (w_t^{img})^2 & 0 & 0 & 0 \\ 0 & (h_t^{img})^2 & 0 & 0 \\ 0 & 0 & (w_t^{img})^2 & 0 \\ 0 & 0 & 0 & (h_t^{img})^2 \end{bmatrix}. \tag{20}$$

We use $\lambda = 0.1$ in our simulation, corresponding to 10 % of the width and height of the target's 2D bounding box in the image, determined through a series of experiments with the CAMSHIFT visual tracker in real environments. For the initial state error denoted by $\mathtt{P}_0$, we propagate the measurement error for $\mathbf{z_0}$ through $\mathbf{h}^{inv}(\mathbf{z}_0)$ and take into account the initial uncertainty about the target's velocity:

$$\mathtt{P}_0 = \mathtt{J}_{h^{inv}} \mathtt{S}_0 \mathtt{J}_{h^{inv}} + \mathrm{diag}(0, 0, 0, \eta, \eta, \eta, 0, 0, 0, 0, 0, 0). \tag{21}$$

η is a constant and $\mathtt{J}_{h^{inv}}$ is the Jacobian of $\mathbf{h}^{inv}(\cdot)$ evaluated at $\mathtt{z}_0$.

We assume that the state transition noise covariance $\mathtt{Q}_t$ is diagonal for simplicity. We tie the covariance $\mathtt{Q}_t$ to the target linear velocities $(\dot{x}_t, \dot{y}_t, \dot{z}_t)$, AR.Drone linear velocities $(\dot{x}_t^r, \dot{y}_t^r, \dot{z}_t^r)$ and orientation $(\gamma_t^r, \beta_t^r, \alpha_t^r)$ odometry reading. We let

$$\begin{bmatrix} v_t \\ s_t \end{bmatrix} = \begin{bmatrix} \sqrt{\dot{x}_t^2 + \dot{y}_t^2 + \dot{z}_t^2} \\ \sqrt{(\gamma_t^r)^2 + (\beta_t^r)^2 + (\delta\alpha_t^r)^2} \end{bmatrix}. \tag{22}$$

We let the entries of $\mathtt{Q}_t$ corresponding to the target position be $\Delta_t^2(\rho_1 v_t^2 + \rho_2)$ and the entries of $\mathtt{Q}_t$ corresponding to the target velocity be $\Delta_t^2(\rho_3 v_t^2 + \rho_4)$. We let the entries of $\mathtt{Q}_t$ corresponding to the AR.Drone's position be $\Delta_t^2(\rho_5 s_t^2 + \rho_6)$, and we let the entries of $\mathtt{Q}_t$ corresponding to the robot's orientation be $\Delta_t^2(\rho_7 s_t^2 + \rho_8)$. This noise distribution is overly simplistic and may ignore some factors, but it is sufficient for the experiments reported in this paper. In total, there are nine free parameters $(\eta, \rho_1, \rho_2, \cdots, \rho_8)$ that must be determined through hand tuning or calibration. In our simulation, we find optimal free parameters free using gradient descent.

2.3.5 Update Algorithm

Given all the preliminaries specified in the previous sections, the update algorithm is just the standard extended Kalman filter, with modification to handle cases where the color region tracker fails due to occlusions or the target leaving the field of view. When no sensor measurement $\mathbf{z}_t$ is available, we simply predict the system state and allow diffusion of the state covariance without sensor measurement correction. When we do have a sensor measurement but the estimated state is far from the predicted state, we reset the filter, using the existing robot position and orientation but fixing the relative target state to that predicted by $\mathbf{h}^{inv}(\mathbf{z}_t)$ and fixing the elements of $\mathtt{P}_t$ by propagating the sensor measurement error for $\mathbf{z}_t$ through $\mathbf{z}^{inv}(\mathbf{z}_t)$ as previously explained in Section 2.3.4. Here is a summary of the algorithm:

```
1. Input $\mathbf{z}_0$.
2. Calculate $\hat{\mathbf{x}}_0$ and $\mathtt{P}_0$.
3. For $t = 1, \ldots, T$, do
   a predict $\hat{\mathbf{x}}_t^- = \mathbf{f}(\hat{\mathbf{x}}_{t-1}, \mathbf{u_{t-1}})$,
   b calculate $\mathtt{J}_{f_t}$ and $\mathtt{Q}_t$,
   c predict $\mathtt{P}_t^- = \mathtt{J}_{f_t} \mathtt{P}_{t-1} \mathtt{J}_{f_t}^T + \mathtt{Q}_t$.
   d If $\mathbf{z_t}$ is unavailable,
      i let $\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^-$,
      ii let $\mathtt{P}_t = \mathtt{P}_t^-$.
   e Otherwise
      i calculate $\mathtt{J}_{h_t}$, $\mathtt{S}_t$, and Kalman gain
        $\mathtt{K}_t = \mathtt{P}_t^- \mathtt{J}_{h_t}^T (\mathtt{J}_{h_t} \mathtt{P}_t^- \mathtt{J}_{h_t}^T + \mathtt{S}_t)^{-1}$,
      ii estimate $\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^- + \mathtt{K}_t(\mathbf{z_t} - \mathbf{h_t}(\hat{\mathbf{x}}_t^-))$,
```

ii update the error estimate $\mathtt{P}_t = (\mathtt{I} - \mathtt{K}_t \mathtt{J}_{h_t})\mathtt{P}_t^-$.

f If $\|\hat{\mathbf{x}}_t - \hat{\mathbf{x}}_t^-\| > \sigma$, reset the filter.

3 Experimental Setup

We tested the performance and efficacy of the proposed method by carrying out experiments, not only on synthetic data, but also in real-world scenarios. Before deploying to the real AR.Drone, we performed a simulation to test model correctness and effectiveness. We then carried out real-world experiments in both indoor and outdoor environments. We compare the proposed joint estimation model with two baselines: 1) relative localization without any filtering, and 2) relative localization with Kalman filter-based sensor correction but no joint estimation of purser and target state. We analyze the data quantitatively for indoor and qualitatively for outdoor environments.

For the simulation experiment, we generated pose data without any noise both for the AR.Drone and target object. The data consist of the target's simulated position with appropriate velocity and simulated AR.Drone's odometry (orientation and linear velocities). We used these as ideal ground truth trajectories. We then produced noisy simulated odometry readings. In order to generate corresponding noisy trajectories, we added noise to the AR.Drone odometry data then accumulated the data to obtain a trajectory. We also added noise to the 2D target bounding box. Direct calculation of pursuer and target positions from the noisy odometry and noisy target bounding boxes gives us baseline method 1, i.e., sensor-only relative localization without filtering. We simulated a camera generating 640×480 images at 20 fps with a focal length of 550 pixels (horizontal field of view 60°).

For the real-world experiments, we performed quantitative evaluation in indoor environments, where we acquired ground truth data for the target and purser using circular markers pasted on the two objects. The circular markers were of a known diameter. We use a fixed, calibrated camera to track the markers in order to estimate, as accurately as possible, real-world ground truth for both the target and the AR.Drone using Krajník and Nitsche's method [23]. See Fig. 6 for a photo of experiment setting. We use these ground truth data for computing root mean square error (RMSE). We also tested the proposed visual tracking with the AR.Drone's front facing monocular camera in this indoor environment.

The outdoor environment is a more desirable environment for testing the proposed methods for joint localization and visual twith the AR.Drone. However, we are deprived of ground truth data because we have tracked the target for a longer period of time and the fixed camera was unable to keep the markers in its field of view for the entire experiment duration. Therefore, we carried out a qualitative outdoor evaluation in which we tested the stability and smoothness of the proposed method's estimates.

In both indoor and outdoor environments, the target object was a person wearing distinctive clothing

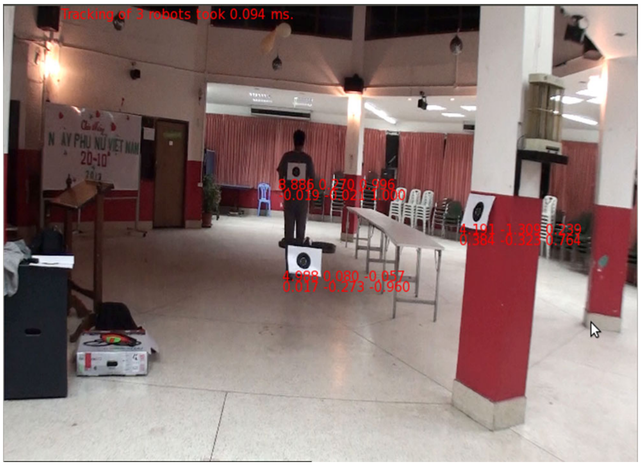


Fig. 6 Markers on the target and AR.Drone providing world-coordinate positions with reference to an additional calibrated camera

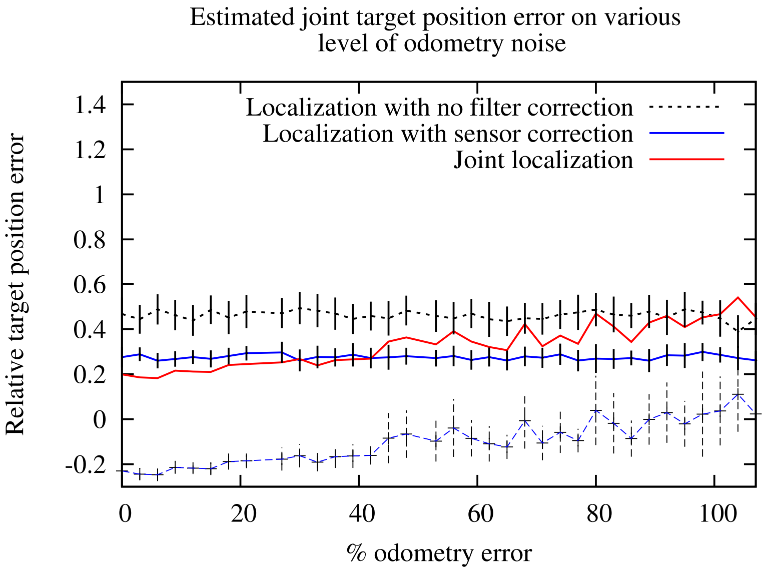


Fig. 7 Experiment I: We simulated the pursuit robot with different levels of odometry noise and measured the average relative target position error for the three estimation methods. Error bars denote 95 % confidence intervals

and a background containing trees, grass, chairs, shadows, and diffuse light. The target moved with varying velocities and directions and was followed by the AR.Drone under teleoperation. The operator adjusted the drone's position to follow target as smoothly as possible. Since the goal of this paper is to asses localization only, we did not use an autonomous position controller for the pursuer; this module will be assessed in future work.

Concerning the visual tracker, we have already reported speed and accuracy tests for the algorithm on different platforms with different hand-held cameras [3]. In this paper we test the proposed visual tracker specifically on the front-facing monocular camera of the AR.Drone, both in indoor and outdoor environments. Additionally we test our method with the online pursuit video sequence published by Klein et al. [22]. The results show how the proposed method recovers from occlusion.

4 Results

In this section, we provide detailed results and analysis for each experiment outlined in the previous section.

We define two baseline methods as points of comparison with the proposed joint state estimation method. The first baseline method does not use any filter while tracking the target [1, 6]; the sensor-based relative target position estimate is simply accepted. Hence it is named "Localization with no filtering." The second baseline uses sensor measurement correction based on an ordinary extended Kalman filter [9] to smooth the target's estimated trajectory in the pursuer robot's coordinate system. This baseline is named "Localization with sensor correction." We do not explicitly incorporate the pursuer's odometry measurements either jointly or separately in the baseline methods. The proposed method, however, jointly corrects the pursuer's odometry with the sensor measurement to produce better performance. The proposed joint localization method integrates the target and robot pose in a joint state space model, as explained in Section 2.3.

4.1 Experiment I: Joint Localization Simulation

First we demonstrate the efficacy of fusing target dynamics and AR.Drone kinematics in one state space model in a simulation experiment. We simulated scenarios containing different levels of AR.Drone odometry noise, while keeping the sensor noise constant. The noise ranged from 0 % to 110 %. The proposed method shows significant improvement over the baseline methods so long as noise level is below 30 %. Performance deteriorates above the 40 % noise level. The performance of the other two methods remains constant over odometry noise levels because they do not use odometry data.

We also evaluated the joint localization model with synthetic data in which a moving UAV chased a moving target. At each iteration of localization, the RMSE is computed between the estimated and ground truth target positions in the robot coordinate frame, using each of the three methods. We also suspend joint localization when visual tracker suspends target tracking and wait for the detector to redetect the target and reinitialize the visual tracker, also reinitialize the joint localization. See the first graph of Fig. 8c. The proposed method provides more stable and smooth estimates than the baseline methods.

We performed statistical tests on the RMSE of each method compared to ground truth for odometry noise of 15 %. The result is shown in the bottom of Fig. 8c. The proposed method outperforms the existing methods by 65 % and 27 %.

4.2 Experiment II: Indoor Joint Localization

In Experiment II, we carried out indoor experiments with quantitative analysis. We used ground truth data for comparison between the proposed and existing methods.

At every iteration of each algorithm, at time t, we transformed the target position (x_t, y_t, z_t) to the AR.Drone coordinate reference frame. We calculated root mean squared error between each method's estimate and ground truth. See Fig. 8a and b top row. We collapse the results of multiple indoor runs into two cases, a large open room and a corridor.

The proposed method clearly obtains smoother and more stable estimates compared to the baseline methods. Although relative localization with sensor correction produces better results when compared

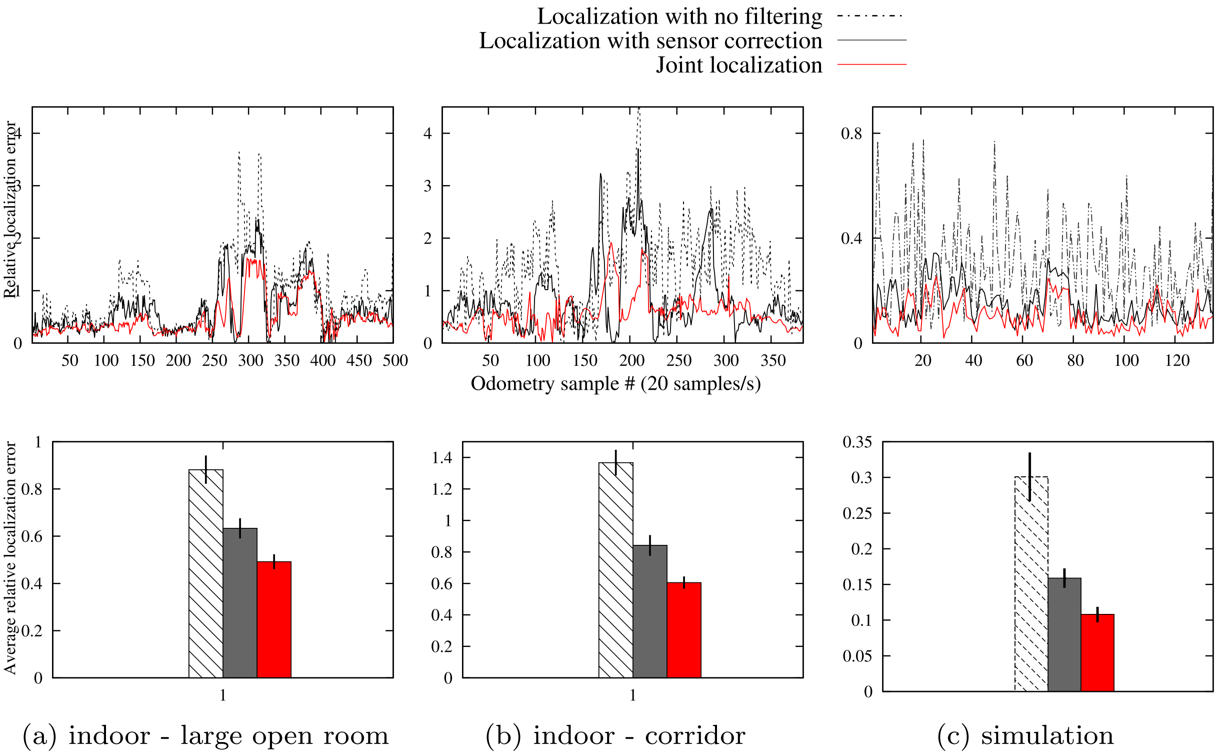


Fig. 8 Experiment I and II: Quantitative evaluation with real world (indoor) and synthetic data. First row shows relative target position root mean squared error (RMSE) of the target in the robot coordinate frame. Second row reflects accumulated average error with 95 % confidence interval. **a** Real world indoor experiment in closed hall. The proposed method provides smooth and stable estimates, 44 % and 22 % better than the baseline methods. **b** Indoor experiment in a corridor. RMSE is 56 % and 28 % lower than with the baseline methods (**c**) Simulation with synthetic data. The proposed method outperforms tracking method without model-based correction by 65 % and outperforms the sensor-based correction method by 27 %

with localization with no filter, our joint localization method, fusing sensor measurements and odometry data, produces additional improvement.

As with the simulation data, we computed the average root mean square error over all runs. The results are shown in the bottom row of Fig. 8a and b. The results show that the filtering methods help smooth the raw sensor measurements, but the proposed joint localization method outperforms not only the method with no filter but also relative localization with sensor correction. Joint localization improves relative target position estimation over both baseline methods.

4.3 Experiment III: Outdoor Joint Localization

As mentioned earlier, though quantitative analysis is not possible outdoors, we did perform an outdoor experiment to ensure the practicality of the proposed approach. We evaluated our method at several stages of the visual tracking and localization process. First we show results when the target is standing still and the UAV is simply hovering. See Fig. 9a. Next we show results when the target is moving with variable velocity and the UAV is accelerating to catch up with the target. See Fig. 9b. Finally, we show results when sensor noise increases due to changes in background and lighting. See Fig. 9c.

Refer to Fig. 9a. The estimated X position varies between 5.5 and 7.5 meters, a range of 2 meters. Recall from Fig. 1 that the target's position X is the distance of the target from the AR.Drone in the forward direction. The proposed method's estimated trajectory, shown in red, is smoother and has lower variance than that of the other methods. The spikes in the trajectory reflect noisy variation in the estimated 2D bounding box's size. The joint localization method infers smooth transitions between neighboring intervals. The estimated Y position varies between 0 and

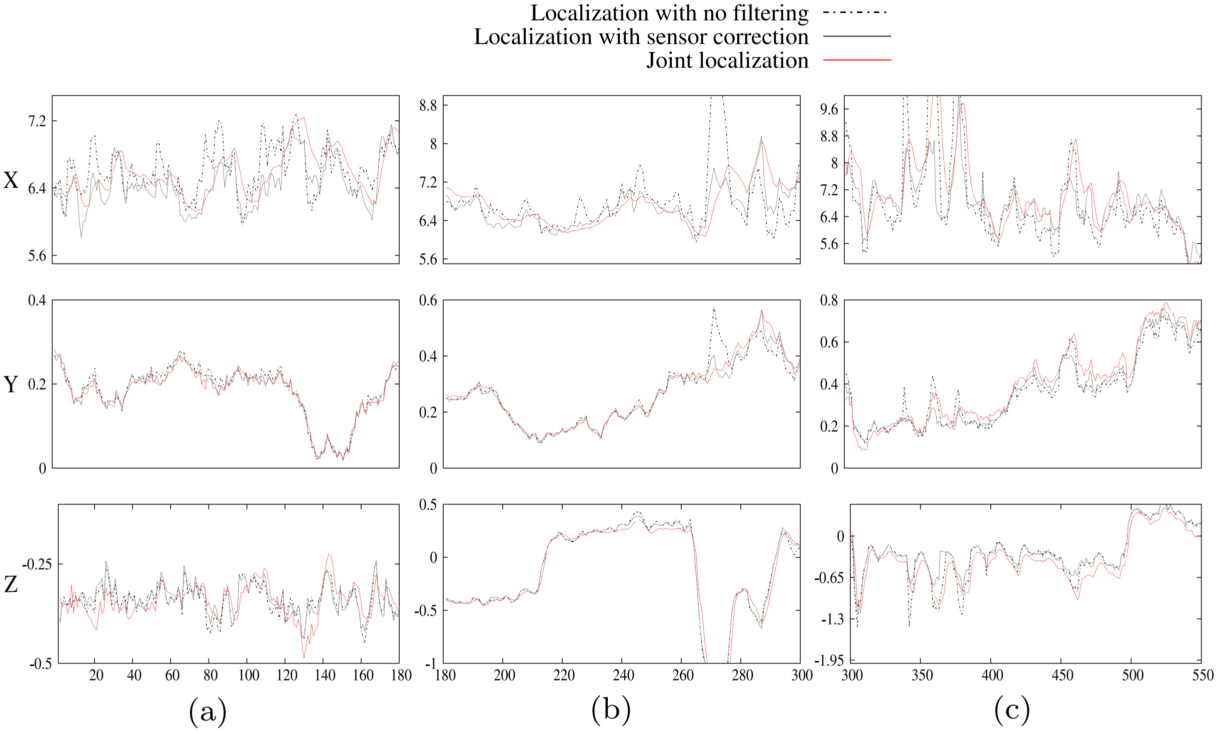


Fig. 9 Experiment III: outdoor qualitative evaluation. Stability and smoothness of estimated target position (X, Y, Z) in the robot coordinate frame, where X is forward, Y is left, and Z is upward. Each subplot's ordinate shows the target's position on one axis in meters, while the abcissa shows time varying between 1 and 550 over all three subfigures. **a** Target is stationary and UAV is simply hovering. The peaks at $t = 130$ in Y and Z are generated when UAV moves to adjust its position relative to the target. **b** Target is moving and UAV is accelerating at $t = 210$ and decelerating at $t = 260$ to catch up with the target. **c** Visual sensor noise increases because of lighting and clutter in the background. Higher noise on X axis shows higher variance due to 2D bounding box error

0.4 meters. The variation is lower on this axis, reflecting lower noise in the estimated left-right position of the 2D bounding box. The proposed method again performs better than the baseline methods. Finally, the estimated Z position varies between -0.5 and -1.2 meters. We observe more spikes or jerking movements on this axis, generally caused by the UAV's pitch control. The Z position estimates are again smoother than for other baseline methods. Overall, the joint localization method clearly performs better than the baseline methods.

Next, refer to Fig. 9b. During this period, the target is moving away from AR.Drone, which is attempting to catch up. The AR.Drone begins accelerating at $t = 200$ and decelerating at $t = 265$. The estimated X position varies between 5.5 and 8.0 meters. The large black-dotted spike indicates extreme sensor noise when the AR.Drone started decelerating. $\dot{x}_t^r$ is minimized when the pitch $\beta_t^r = 0$. The same spike can be observed in the Y target position in the robot coordinate frame. Here the noise varies between 0 and 0.6 meters. Clearly the proposed method is smoother and more stable along both the X and Y axis, even with the velocity variation of the AR.Drone. Finally, the estimated Z position varies between -1 and 0.5 meters. The change in Z at $t = 213$ is caused by the UAV's forward pitch, causing it to move. When the AR.Drone decelerates, we obtain noise in sensor and odometry measurements, so we observe more variability between $t = 270$ and $t = 295$. The overall performance of joint localization is clearly better than the other methods.

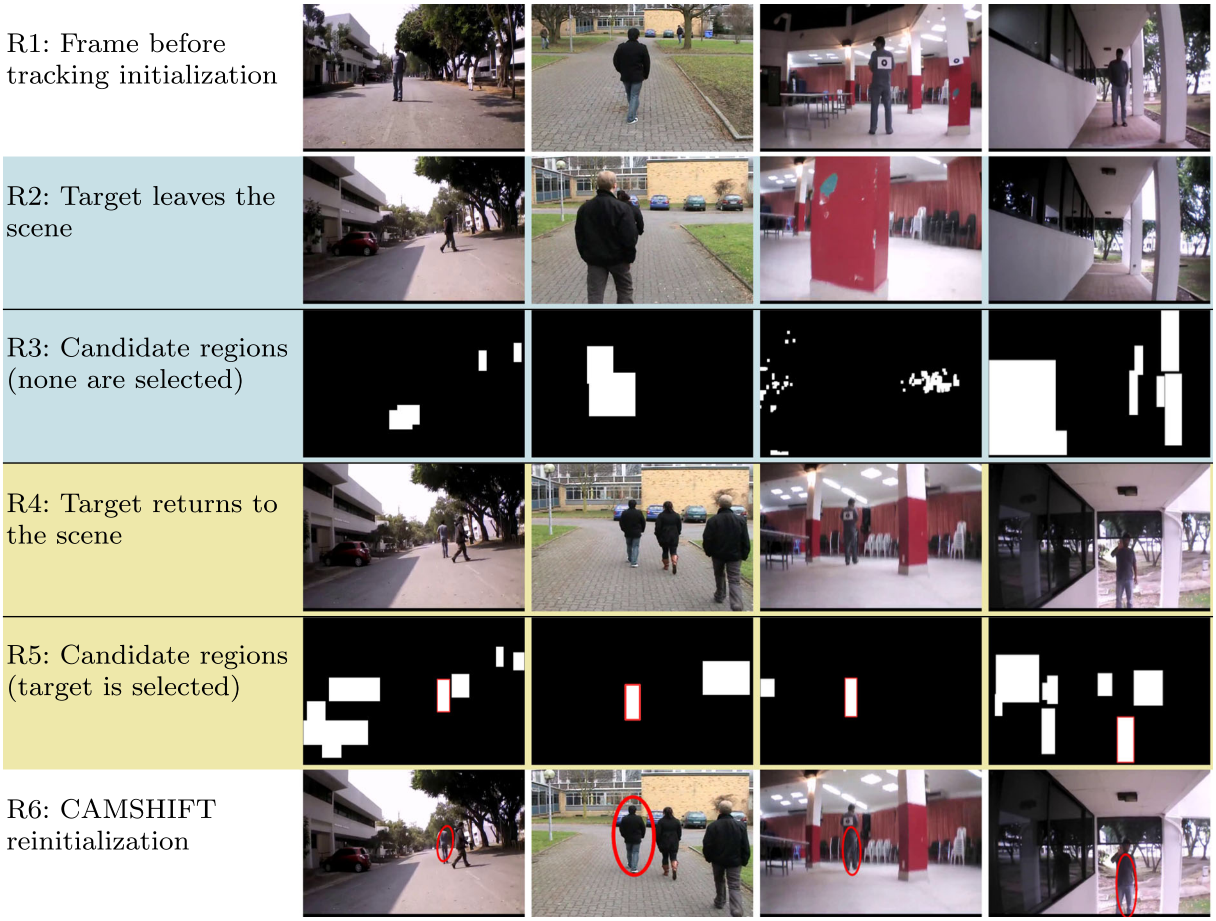

R1: Frame before tracking initialization
R2: Target leaves the scene
R3: Candidate regions (none are selected)
R4: Target returns to the scene
R5: Candidate regions (target is selected)
R6: CAMSHIFT reinitialization

Fig. 10 Experiment IV: Visual tracker evaluation. The proposed method was tested in different environments with different background and target objects. Each column shows images from a different video. Rows show results of each step of processing. Blue colored rows show processing when the target is not in the scene. Yellow colored rows show the same processing steps when the target returns to the scene

Now we turn to Fig. 9c. We observe an increased number of spikes in the trajectories caused by error in the 2D bounding box estimates. This noise results in unstable prediction, challenging the proposed joint localization method. The estimated position of the target varies between 5.4 and 10 meters, the estimated Y position varies between 0 and 0.8 meters, and the estimated Z position varies between -1.5 and 0.6 meters. The proposed method performs better than the other two methods in terms of stability and smoothness.

4.4 Experiment IV: Visual Tracking

We tested the visual tracker in both indoor and outdoor environments. In Fig. 10, the first column shows outdoor results with our AR.Drone. The second column shows results of processing an online pursuit video sequence by Klein et al. [22], and the last two columns show indoor results with our AR.Drone in a large open room and corridor.

Based on the first frame of each video (R1 in Fig. 10), we initialized CAMSHIFT tracking by manually providing a bounding box for the human target in the scene. We then ran the proposed tracking, suspension, and redetection method to the end of each video.

During tracking, our method incrementally updated the mean μ_t and standard deviation σ_t of the distance d_t between the appearance model H^m and the tracked target's color histogram H_t^r. In almost all cases, when the target left the scene, the distance d_t exceeded the adaptive threshold, except for a few cases in which the redetection algorithm found a sufficiently similar object in the background.

Table 1 Experiment IV: tracking accuracy. For each video, we report the number of frames the target is visible and occluded or outside the FOV, the existence of multiple objects in the image scene, the percentage of frames containing the target in which target was correctly tracked, and the percentage of frames not containing the target in which the target was falsely detected. Misses occur when tracking is incorrectly suspended

Video	Frames w. target		Multiple	Detection ratio	
	present	absent	objects	true	false
a	389	50	Yes	96.8%	2.0%
b	997	20	Yes	98.9%	0.0%
c	541	150	No	88.8%	5.3%
d	426	174	No	97.3%	3.4%

Table 2 Experiment IV: Runtime performance of visual tracking and redetection algorithms on two different processors

Redetection Phase		Tracking Phase	
Core i3	Atom N280	Core i3	Atom N280
41.455ms	91.232ms	16.340ms	49.234ms

Rows R2 and R3 in Fig. 10 show example images acquired when the target was absent from the camera field of view. At this point in each video, CAMSHIFT tracking is suspended and the redetection algorithm is running, correctly reporting the absence of the target from the field of view.

Rows R4 and R5 in Fig. 10 show example images acquired after the target has returned to the field of view. The proposed method eventually successfully identifies the candidate region among the possible candidates. In the figure, the selected region is surrounded by a red rectangle.

In each of the four cases shown, CAMSHIFT is correctly reinitialized, as shown in row R6 of Fig. 10.

Over the four videos, the target was successfully tracked in 95.6% of the frames in which the target was in the scene, with false positives occurring in only 4.4% of the frames in which the target was not in the scene. The accuracy results on a per-video basis are summarized in Table 1.

We tested the runtime performance of the system on two different hardware configurations, a 2.26 GHz Intel Core i3 laptop running 32-bit Ubuntu Linux 11.10 and a 1.6 GHz Intel Atom N280 single core netbook running 32-bit Lubuntu 11.10. The results are summarized in Table 2. Both redetection and tracking run at high frame rates, with the worst case of just over 10 fps for redetection on the Atom processor. The method is clearly feasible for onboard execution by a mobile robot with modest computational resources.

5 Conclusion

In this paper, we propose a joint localization method that fuses target dynamics and UAV kinematics into one state space model, and we demonstrate how to integrate sensor measurements from a moving monocular camera into the resulting filter. We show that joint localization produces more stable, smooth, and noise-resistant trajectories than those produced by standard filters. The joint localization filter performs well even when there are sudden changes in the sensor measurement indicating an erroneous detection or a rapid change of the target object's position.

We have also proposed a computationally efficient and accurate visual tracker that can deal with situations where the object disappears from the field of view. The target's 3D position is inferred by the tracker from the bounding box of its projection into the 2D image. The result is an estimated trajectory sufficiently smooth to use in a pursuit robot. In a series of experiments, we show that the proposed filtering technique outperforms traditional filtering methods.

In future work, we plan to further investigate adaptive and discriminative modeling of the target and the background, to address situations in which the target's appearance varies as it moves through areas with different lighting conditions and backgrounds.

Acknowledgments AB was supported by graduate fellowships from the University of Balochistan, Quetta, the Higher Education Commission of Pakistan, and the Asian Institute of Technology, Thailand. WSQ was supported by fellowships from the National University of Science and Technology (NUST), Pakistan, and the Asian Institute of Technology, Thailand. TK was supported by EU-ICT project 600623 'STRANDS' and CZ-MSMT project LH11053. We thank Niraj Shakya for help with experimental data collection.

References

1. Allen, J.G., Xu, R.Y.D., Jin, J.S.: Object Tracking using CamShift Algorithm and Multiple Quantized Feature Spaces. In: Pan-Sydney Area Workshop on Visual Information Processing, vol. 36, pp. 3–7 (2004)
2. Barrientos, A., Colorado, J., del Cerro, J., Martinez, A., Rossi, C., Sanz, D., Valente, J.: Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots. J. Field Robotics **28**(5), 667–689 (2011)
3. Basit, A., Dailey, M.N., Lakanacharoen, P., Moonrinta, J.: Fast target redetection for CAMSHIFT using back-projection and histogram matching. In: International Conference on Computer Vision Theory and Applications (VISAPP) (2014)
4. Basit, A., Dailey, M.N., Laksanacharoen, P.: Model driven state estimation for target pursuit. In: IEEE International Conference on Control, Automation, Robotics & Vision, pp. 1077–1082 (2012)
5. Bi, Y., Duan, H.: Implementation of autonomous visual tracking and landing for a low-cost quadrotor. Opt. Inter. J. Light Elec. Opt. **124**(18), 3296–3300 (2013)
6. Bradski, G.: Real Time Face and Object Tracking as a Component of a Perceptual User Interface. In: IEEE Workshop on Applications of Computer Vision, pp. 214–219 (1998)
7. Bristeau, P.J., Callou, F., Vissière, D., Petit, N., et al.: The Navigation and Control technology inside the AR.Drone micro UAV. In: 18th IFAC World Congress, pp. 1477–1484 (2011)
8. Chen, H., Houkes, Z.: Model-based recognition and classification for surface texture of vegetation from an aerial sequence of images. In: Proceedings of SPIE, vol. 3222, pp. 236–245 (1997)
9. Comaniciu, D., Ramesh, V.: Mean shift and optimal prediction for efficient object tracking. In: Proceedings of the International Conference on Image Processing, 2000, vol. 3, pp. 70–73 (2000)
10. Comaniciu, D., Ramesh, V., Meer, P.: Real-Time Tracking of Non-Rigid Objects using Mean Shift. In: IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 142–149 (2000)
11. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. IEEE Trans. Pattern Anal. Mach. Intell. **25**(5), 564–577 (2003)
12. Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.: MonoSLAM: Real-time single camera SLAM. IEEE Trans. Pattern Anal. Mach. Intell. **29**(6), 1052–1067 (2007)
13. Denman, S., Chandran, V., Sridharan, S.: An adaptive optical flow technique for person tracking systems. Pattern Recogn. Lett. **28**(10), 1232–1239 (2007)
14. Funk, N.: A study of the Kalman filter applied to visual tracking. Tech. Rep., University of Alberta, CMPUT (2003)
15. Gktoan, A., Sukkarieh, S., Bryson, M., Randle, J., Lupton, T., Hung, C.: A rotary-wing unmanned air vehicle for aquatic weed surveillance and management. J. Intell. Robot. Syst. **57**(1-4), 467–484 (2010)
16. Gupte, S., Mohandas, P., Conrad, J.: A survey of quadrotor unmanned aerial vehicles. Southeastcon. In: IEEE Southeastcon 2012, pp. 1–6 (2012)
17. Herwitz, S., Johnson, L., Dunagan, S., Higgins, R., Sullivan, D., Zheng, J., Lobitz, B., Leung, J., Gallmeyer, B., Aoyagi, M., et al.: Imaging from an unmanned aerial vehicle: Agricultural surveillance and decision support. Comput. Electron. Agric. **44**(1), 49–61 (2004)
18. Higuchi, K., Shimada, T., Rekimoto, J.: Flying sports assistant: External visual imagery representation for sports training. In: 2nd Augmented Human International Conference (AH), 2011, pp. 7:1–7:4 (2011)
19. Jiménez-Berni, J.A., Zarco-Tejada, P.J., Suarez, L., Fereres, E.: Thermal and narrowband multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle. IEEE Trans. Geosci. Remote Sens. **47**(3), 722–738 (2009)
20. Karavasilis, V., Nikou, C., Likas, A.: Visual tracking by adaptive Kalman filtering and mean shift. In: Artificial Intelligence: Theories, Models and Applications, pp. 153–162. Springer (2010)
21. Kim, J., Shim, D.: A vision-based target tracking control system of a quadrotor by using a tablet computer. In: 2013

International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1165–1172 (2013)

22. Klein, D.A., Schulz, D., Frintrop, S., Cremers, A.B.: Adaptive Real-Time Video-Tracking for Arbitrary Objects. In: IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 772–777 (2010)
23. Krajník, T., Nitsche, M., et al.: External Localization System for Mobile Robotics. In: International Conference on Advanced Robotics. IEEE, Montevideo (2014)
24. Lan, Y., Thomson, S.J., Huang, Y., Hoffmann, W.C., Zhang, H.: Review: Current status and future directions of precision aerial application for site-specific crop management in the USA. Comput. Electr. Agri. **74**(1), 34–38 (2010)
25. Lou, J., Yang, H., Hu, W.M., Tan, T.: Visual vehicle tracking using an improved EKF. In: Asian Conference of Computer Vision (ACCV), pp. 296–301 (2002)
26. Murphy, D.W., Cycon, J.: Applications for mini VTOL UAV for law enforcement. In: Enabling Technologies for Law Enforcement and Security, pp. 35–43. International Society for Optics and Photonics (1999)
27. Perreault, S., Hebert, P.: Median filtering in constant time. Image Processing. IEEE Trans. Image Process. **16**(9), 2389–2394 (2007)
28. Porikli, F.: Integral histogram: a fast way to extract histograms in cartesian spaces. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005, vol. 1, pp. 829–836 (2005)
29. Pupilli, M., Calway, A.: Real-time camera tracking using a particle filter. In: British Machine Vision Conference (2005)
30. Qureshi, W.S., Alvi, A.B.N.: Object tracking using mach filter and optical flow in cluttered scenes and variable lighting conditions. World Acad. Sci. Eng. Technol. **3**(12), 642–645 (2009)
31. Shimin, F., Qing, G., Sheng, X., Fang, T.: Human tracking based on mean shift and Kalman filter. In: Artificial Intelligence and Computational Intelligence, vol. 3, pp. 518–522 (2009)
32. Sizintsev, M., Derpanis, K., Hogue, A.: Histogram-based search: A comparative study. In: IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008, pp. 1–8 (2008)
33. Ta, D.N., Chen, W.C., Gelfand, N., Pulli, K.: Surftrac: Efficient tracking and continuous object recognition using local feature descriptors. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2937–2944 (2009)
34. Welch, G., Bishop, G.: An introduction to the kalman filter. Tech. rep., Chapel Hill (1995)
35. Yokoyama, M., Poggio, T.: A contour-based moving object detection and tracking. In: Visual Surveillance and Performance Evaluation of Tracking and Surveillance, pp. 271–276 (2005)
36. Zhou, H., Yuan, Y., Shi, C.: Object tracking using SIFT features and mean shift. Comp. Vision Image Underst. **113**(3), 345–352 (2009)