Robotics and Autonomous Systems 74 (2015) 1–14

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Joint localization and target tracking with a monocular camera

Abdul Basit [a,b,*], Matthew N. Dailey [a], Jednipat Moonrinta [a], Pudit Laksanacharoen [c]

[a] *Asian Institute of Technology, Klong Luang Pathumthani (12121), Thailand*
[b] *University of Balochistan, Quetta (87300), Pakistan*
[c] *King Mongkut's University of Technology North Bangkok, Bangsue (10800), Bangkok, Thailand*

HIGHLIGHTS

- Joint localization fuses target dynamics and pursuit robot kinematics to improve trajectories estimation.
- An adaptive histogram similarity threshold correctly suspend tracking and localization when target is occluded.
- A fast target redetection method avoids false detections and improves accuracy.
- Redetection successfully reinitializes visual tracking and state estimation correction.

ARTICLE INFO

Article history:
Received 3 March 2014
Received in revised form
28 April 2015
Accepted 27 May 2015
Available online 10 June 2015

Keywords:
Joint localization
Pursuit robot
Unmanned ground vehicles
Differential drive kinematics
Target dynamics
Tracking suspension
Histogram backprojection
Target redetection
Tracking reinitialization

ABSTRACT

Localization capabilities are necessary for autonomous robots that need to keep track of their position with respect to a surrounding environment. A *pursuit robot* is an autonomous robot that tracks and pursues a moving target, requiring accurate localization relative to the target's position and obstacles in the local environment. Small unmanned ground vehicles (SUGVs) equipped with a monocular camera and wheel encoders could act as effective pursuit robots, but the noisy 2D target position and size estimates from the monocular camera will in turn lead to overly noisy 3D target pose estimates. One possible approach to relative localization for pursuit robots is, rather than simply tracking and estimating a relative robot–target position in each frame, *joint localization*, in which the purser and target are both localized with respect to a common reference frame. In this paper, we propose a novel method for joint localization of a pursuit robot and arbitrary target. The proposed method fuses the pursuit robot'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 improved results compared to standard filters for estimating relative target trajectories in a 3D coordinate system. For visual tracking, we also introduce an adaptive histogram matching threshold for suspending tracking when the target is lost in a cluttered environment. When tracking is suspended, rather than traversing the entire image to search for a reappearance of the target, we only search the part of the image segmented by histogram backprojection and correctly reinitialize the tracker. The experimental results show that the joint localization method outperforms standard localization methods and that the visual tracker for pursuit robot can deal effectively with target occlusions.

1. Introduction

Small unmanned ground vehicles (SUGVs) such as the PackBot, SARGE and Gladiator [1] are useful for gathering information about environments where access by human beings is either impossible or dangerous. They are portable, lightweight and inexpensive. Such robots are becoming increasingly common in military applications and disaster areas such as the damaged Fukushima nuclear plant.

One intriguing application of small autonomous vehicle is *pursuit*. Robot pursuit applications include human security, e.g., following and monitoring important people or pursuing suspicious people in security or military contexts. Although pursuit can be accomplished through teleoperation, [2,3], in situations where fast decision making is crucial, pursuit robots should be capable of making decisions autonomously.

In our work, we investigate the feasibility of autonomous pursuit of targets using SUGVs. We use a custom-made all-terrain

* Corresponding author at: Asian Institute of Technology, Klong Luang Pathumthani (12121), Thailand.
E-mail addresses: abasit@uob.edut.pk, abdulbasitkhan@gmail.com (A. Basit), mdailey@ait.asia (M.N. Dailey), jednipat@ait.ac.th (J. Moonrinta), stl@kmutnb.ac.th (P. Laksanacharoen).

http://dx.doi.org/10.1016/j.robot.2015.05.012
0921-8890/© 2015 Published by Elsevier B.V.

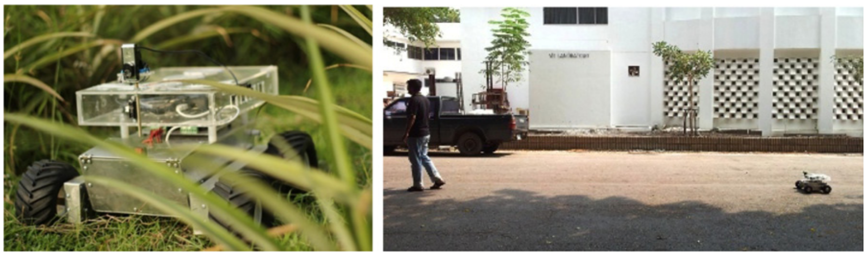

Fig. 1. All-terrain robot for tracking and pursuit of arbitrary target objects using a monocular camera.

surveillance robot similar to the iRobot PackBot that is equipped with teleoperation capabilities and mounted with a monocular camera, as shown in Fig. 1. We focus on the special case of monocular vision-based localization and tracking of the target's position relative to the purser in a 3D coordinate system, without the use of any 3D sensor.

Although the use of a monocular camera as the main sensor simplifies the design and lowers the cost of the robot, it presents additional challenges. First, since depth estimates based on monocular cues will necessarily be extremely noisy, to obtain usable target position estimates from the moving camera, we need a smooth and stable sensor modeling and state filtering technique. Second, tracking an object during target pursuit requires a tracker that is both sufficiently accurate and sufficiently fast to keep track of the target in real time.

In the following subsections, we introduce related work in target pursuit, localization, and 2D target tracking. Afterwards, we outline the specific contributions of our work.

1.1. Target pursuit

The problem of autonomous pursuit and person following has been addressed in the robotics literature. A great deal of the work has benefitted from the use of laser range finders. Kobilarov et al. [4] propose a method for a mobile robot equipped with a laser range finder and an omnidirectional camera to track and follow a person in outdoor environments. Each laser scan is processed and converted into connected components (blobs) that may be obtained from humans present in the laser field of view. Additional measurements are obtained from the regions in the image corresponding to each of these objects. Awai et al. [5] use a camera and laser range finder with a mobile robot for person following capable of autonomous return. Person tracking is performed using HOG features, color information, and the shape of range data. The mobile robot operates in 2D in a static, flat environment. Ohshima et al. [6] achieve person following using a laser range finder, the output of which is combined with odometry data to localize the specific target and other objects and to estimate the target's motion. The target moves slowly, less than 1 m/s. The pursuer stays close to the target to avoid interference by other objects. Sung et al. [7] propose a method to identify legs in laser scans, group the legs into people, and track a target person robustly over a time period using defined attributes.

Other related work has used stereo imaging. Yoshimi et al. [8] developed a robot that can follow a person using vision-based target detection and can also avoid obstacles with the help of ultrasonic sensors. The authors use stereo vision for tracking people. They describe an algorithm for extracting information on the distance to each feature point, the speed of the target, and the color and texture of the target's clothing, obtaining stable tracking and robust operation in dynamic environments. Chen et al. [9] present a vision-based algorithm that uses Lucas–Kanade feature detection and matching to find the location of the target in the image and control the robot. Matching is performed between a pair of stereo images including matches between successive video frames. The authors use a RANSAC algorithm to segment the sparse disparity map and estimate a motion model for the target and background.

In more recent work, Choi et al. [10] propose a method for multi-target detection and tracking of people with occlusions in indoor environments. Results from various detection algorithms such as face, skin color, upper body, depth based shape, and motion detectors are fused using a specialized sampling and particle filter procedure.

All of the aforementioned methods use laser ranging, stereo, or depth cameras to obtain 3D information for tracking the target, and most have only been tested in indoor environments. Detectors such as those proposed by Choi et al. would complement and possibly improve the accuracy of the methods reported upon in this paper, at the cost of additional processing time. However, a single, monocular camera would be much simpler to integrate and more cost effective in a commercial product than these ranging sensors, so our focus is on obtaining high accuracy tracking of a single target with the low cost, light weight, and fast processing speed of algorithms incorporating only a monocular camera and color-based appearance models.

On the other hand, there are numerous challenges to the use of such a simple sensor. Monocular cameras do not directly provide any 3D information. It may also be easier for a target to blend into the background in a monocular video stream, especially outdoors. Any solution must thus effectively integrate partial, noisy information from multiple observations and must handle occlusions and target loss robustly. Our method addresses these challenges.

1.2. Localization

Localization refers to sequential state estimation techniques used by robots to keep track of their position with respect to the surroundings. Localization is used extensively in SLAM [11,12], visual tracking [13], and 3D reconstruction. Some of the widely used methods include the extended Kalman filter, particle filters [14], and grid-based methods. The EKF is the simplest, fastest, and most widely used technique [15–17] for mobile robot localization.

Most methods for mobile robot localization combine odometry data with measurements of bearing and/or distance to static landmarks. In pursuit applications, however, the pursuer must localize itself not only with respect to the static environment, but also with respect to the target. For this reason, our joint localization method treats the target itself as a mobile landmark for localization, integrating pursuer kinematics and target dynamics to obtain smooth, accurate localization estimates for both the pursuer and the target.

1.3. 2D object tracking

We categorize the common 2D image-based tracking algorithms as to whether they utilize *feature matching*, *optical flow*, or *feature histograms*. Feature matching algorithms such as SIFT [18], SURF [19], and shape matching algorithms such as contour matching [20] are potentially useful for tracking, but they are slow. Calonder et al. [21] introduce BRIEF (Binary Robust Independent Elementary Features) to improve the speed performance of the SIFT descriptor, but it is very sensitive to in-plane rotations. Rublee et al. [22] introduce ORB (Oriented FAST and Rotated BRIEF) to overcome the limitations of BRIEF. Dalal et al. [23] introduce the histogram of oriented gradients (HOG) descriptor for pedestrian detection in static images, and Munaro at el. [24] use HOG descriptors in a classification phase to track people with a mobile robot and RGB-D sensor. All of these methods are potentially effective, but they are too computationally expensive to be considered for real-time tracking by a moving robot with modest computational resources, and they would require additional support [25,26,24] to model target appearance for purposes of target redetection after it leaves and reenters the field of view tracking by a moving robot with modest computation resources.

Optical flow methods [27] may be within reach in terms of speed, but they do not maintain an appearance model. This means they are unable (by themselves) to recover from occlusions and the target leaving the field of view. Zajdel et al. [28] extend an optical flow technique to enable a mobile robot to track and redetect the target when the target is lost and reappears in camera's field of view, but only indoor environments were considered, and the optical flow algorithm used for tracking is unable to track the target's shape and size.

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. In this paper, for the visual tracker, we thus consider the problem of *redetecting* the target object once the target is lost in occlusion or from image field of view. We assume that the goal is to search for the target object in every frame without any bias as to where the object might appear.

To execute this target search in the entire image, the common approach in computer vision would be the sliding window over the image at multiple scales and compare with a target model. To improve the naive sliding window search, Porikli [29] propose an "integral histogram" method using integral images. Perreault and Hebert [30] 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. [31] take a similar approach to obtain histograms over sliding windows by efficiently updating the histogram using previously calculated histograms for overlapping windows.

However, although it is possible to compute sliding window histograms in constant time per window location, the computation may still not be fast enough if multiple window sizes and aspect ratios must be considered, and furthermore, finding a single best rectangular window still does not give a precise object shape and orientation. Chen et al. [32,33] address the speed issue by scattering randomly generated elliptical regions over the image in a first rough detection phase and address the precision issue by performing fine searches from the more likely candidate regions.

CAMSHIFT (Continuously Adaptive Mean Shift) [34,35] is a fast and robust feature histogram tracking algorithm potentially useful for mobile robots in outdoor environments. The method begins with manual initialization from a target image patch. It then tracks that region using a combination of color histograms, the basic mean-shift algorithm [36,37], and an adaptive region-sizing step. It is scale and orientation invariant. To the best of our knowledge, no other method is comparable in terms of processing speed. The main drawback of CAMSHIFT is that if the target leaves the field of view or is occluded, the algorithm either reports an error or starts tracking a completely different object. In the work reported in this paper, we eliminate that drawback.

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.

1.4. Contributions

The challenges in pursuit target tracking with a monocular camera are that (1) the 2D tracker must be sufficiently fast and accurate, (2) it must be possible to estimate a 3D position based on the 2D tracker's output, and (3) there must be a way to overcome the extreme noise inherent in estimating 3D positions from monocular cues.

In this paper, we introduce a method that addresses these challenges. To reduce target position estimation error caused by noisy monocular depth cues, we propose a joint localization method. The model maintains an estimate of the state of the target, assuming a linear dynamical model, as well as an estimate of the pursuit robot's state, assuming differential drive robot kinematics [38,39]. We thus fuse information from 2D visual tracking and the SUGV's wheel encoders with knowledge of the robot's kinematics in a joint localization filter to obtain superior state estimation.

To handle cases where the target is occluded or leaves the scene, we additionally propose a fast method for suspending visual tracking. The decision to suspend tracking is based on an adaptive threshold applied to the dissimilarity between the CAMSHIFT region's color histogram and the stored appearance model, as well as heuristic limitations on changes in the tracking window's size.

To ensure that the target is re-acquired when a lost target reappears, we also propose a real-time target object redetection method. The redetection method is based on backprojection of the appearance model, thresholding of the per-pixel likelihood, and connected components analysis, resulting in a collection of candidate regions, the best of which is selected if it is sufficiently likely according to the appearance model.

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

This paper includes revised and extended materials from two conference papers on state estimation [40] and the visual tracker [41]. The state estimation method was previously only tested in simulation, and the visual tracker was only tested in isolation from the pursuit robot. The proposed method in this paper is also proposed and tested with the Unmanned Aerial Vehicle (UAV) [42]. In this paper, we provide complete descriptions of each algorithm, and we evaluate the method using a complete, real-world implementation of the joint localization method, incorporating the 2D visual tracker with the pursuit robot. We provide additional detailed experimental results in real-world scenarios both indoors and outdoors. We have also carried out additional evaluations of the tracker with both online data and the pursuit robot's camera and with multiple objects in the scene.

2. Algorithm design

In this section, we briefly describe the basic algorithm flow for the entire system, then in later sections, we separately describe the proposed joint localization method and the 2D visual tracker for SUGVs in detail.

The proposed algorithm for state estimation by pursuit robots combines a visual object tracking algorithm using monocular camera and a filter incorporating both target dynamics and

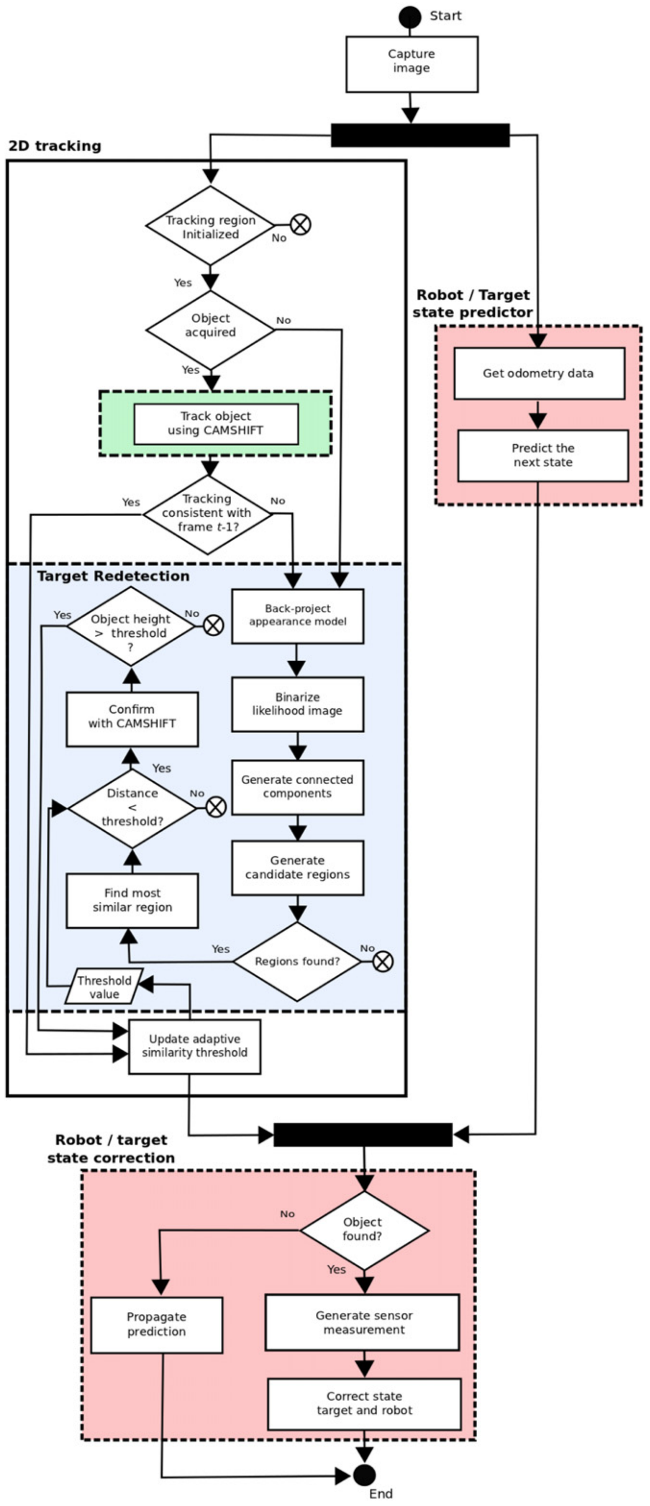
Start
Capture image
2D tracking
Tracking region Initialized
No
Yes
Object acquired
No
Yes
Track object using CAMSHIFT
Tracking consistent with frame t-1?
Yes
No
Robot / Target state predictor
Get odometry data
Predict the next state
Target Redetection
Object height > threshold ?
Yes
No
Back-project appearance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

Fig. 2. Algorithm flow diagram. The green box contains the CAMSHIFT 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 the EKF motion model for fusing and filtering the odometry and target tracking data. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

differential drive kinematics to produce stable and accurate target and robot state estimates simultaneously.

We refer the reader to Fig. 2. So long as the targetis in the scene, on each input image, we apply the standard CAMSHIFT algorithm. On each frame, CAMSHIFT returns an estimated target position and size in the 2D image. To transform the 2D information to 3D, we take the ray from the camera's center through the image plane at the center of the 2D bounding box and calculate the depth of the target along that ray using the (assumed given) height of the target in an absolute frame. Meanwhile, we acquire odometry data in the form $(dx, dy, d\theta)$, where (dx, dy) describes the linear displacement of the robot and $d\theta$ describes the angular displacement of the robot. The target tracking result and odometry are fed to the EKF, which predicts the new state based on the previous estimate then corrects the state estimate based on the new measurements. If the target is occluded or leaves the scene, we stop 2D tracking and state estimate correction and run the redetection algorithm until the target reappears, at which time we restart the normal flow of the algorithm. In the following sections, we describe the joint localization and visual tracking steps in detail.

3. Joint estimation of robot and target state

In this section, we introduce the state estimation model. We model the robot's state, the target's state, and the color region tracking sensor just described in the extended Kalman filter framework.

3.1. System state

The system state expresses the pursuit robot's position and the target's position and dynamics 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 pursuit robot's position, and $(\gamma_t^r, \beta_t^r, \alpha_t^r)$ is the pursuit robot's 3D orientation (roll, pitch and yaw). The positions and orientations are expressed in the world coordinate frame.

3.2. State transition

We assume that initially, the specific position and orientation of robot coordinate with respect to world coordinate frame is either given to us or the **robot position is considered** (0, 0, 0) and its orientation is assumed aligned to world coordinate orientation (0, 0, 0). We further assume that the vehicle has differential drive kinematics and is equipped with two encoders, one for each drive wheel. The odometry control vector is

$$\mathbf{u}_t = \left[d_t^L \quad d_t^R\right]^T, \tag{2}$$

where d_t^L is the distance traveled by the left wheel and d_t^R is the distance traveled by the right wheel. The distances are calculated from the number of ticks per revolution, the wheel base, and the diameter of the wheels. The motion model is

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\nu}_t, \tag{3}$$

where $\boldsymbol{\nu}_t \sim \mathcal{N}(\mathbf{0}, \mathbb{Q}_t)$. $\mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$ has two components. The first component models the kinematics of a differential drive robot, and the second component is a first order linear dynamical system for the **target's motion**. See Fig. 3 for a schematic. We describe each component in turn.

3.2.1. Pursuit robot motion

For the robot's motion, we first introduce some intermediary variables for convenience. The linear distance traveled by the robot over the interval in the robot coordinate is

$$d_s = \frac{d_t^L + d_t^R}{2}. \tag{4}$$

The change in yaw in the robot coordinate system is

$$d_\alpha = \frac{d_t^L - d_t^R}{l}, \tag{5}$$

where l is the wheel base (the distance between the two wheels). If $d_\alpha \neq 0$, we can write the turning radius

$$R = \frac{d_s}{d_\alpha}. \tag{6}$$

With finite R, the robot's displacement in the robot coordinate system is defined by

$$\begin{bmatrix} d_x \\ d_y \end{bmatrix} = R \begin{bmatrix} \sin(d_\alpha) \\ 1 - \cos(d_\alpha) \end{bmatrix}. \tag{7}$$

Note that in addition to arc motions, this also covers the special case of rotation in place, where $d_t^L = -d_t^R$ and $R = 0$. However, for the special case of straight motion, when $d_t^L = d_t^R$ and $d_\alpha = 0$, we take the limit of Eq. (7) as $R \to \infty$ to obtain

$$\begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} d_s \\ 0 \end{bmatrix}. \tag{8}$$

To convert the robot's relative motion in the robot's ground plane into the world coordinate frame, we must rotate by the orientation of the robot's ground plane represented by R_t at time t:

$$\begin{bmatrix} x_{t+1}^r \\ y_{t+1}^r \\ z_{t+1}^r \end{bmatrix} = \begin{bmatrix} x_t^r \\ y_t^r \\ z_t^r \end{bmatrix} + \mathrm{R}_t \begin{bmatrix} d_x \\ d_y \\ 0 \end{bmatrix}, \tag{9}$$

expanding R_t in detail, we get

$$\begin{bmatrix} x_{t+1}^r \\ y_{t+1}^r \\ z_{t+1}^r \end{bmatrix} = \begin{bmatrix} x_t^r \\ y_t^r \\ z_t^r \end{bmatrix} + \begin{bmatrix} d_x \mathrm{c}_{\alpha_t} \mathrm{c}_{\beta_t} + d_y(\mathrm{c}_{\alpha_t} \mathrm{s}_{\beta_t} \mathrm{s}_{\gamma_t} - \mathrm{s}_{\alpha_t} \mathrm{c}_{\gamma_t}) \\ d_x \mathrm{s}_{\alpha_t} \mathrm{c}_{\beta_t} + d_y(\mathrm{s}_{\alpha_t} \mathrm{s}_{\beta_t} \mathrm{s}_{\gamma_t} + \mathrm{c}_{\alpha_t} \mathrm{c}_{\gamma_t}) \\ -d_x \mathrm{s}_{\beta_t} + d_y \mathrm{c}_{\beta_t} \mathrm{s}_{\gamma_t} \end{bmatrix}, \tag{10}$$

where $\mathrm{c}_\cdot$ and $\mathrm{s}_\cdot$ are abbreviations for the cosine and sine functions.

To determine α_{t+1}, β_{t+1}, and γ_{t+1}, we let $\mathbb{R}$ be the rotation matrix corresponding to a rotation of d_α around the z axis in the robot's ground plane. Then the new orientation of the vehicle, expressed as a rotation matrix, is

$$\mathrm{R}_{t+1} = \mathrm{R}_t \mathbb{R}. \tag{11}$$

We can easily extract Euler rotations from the above equation.

3.2.2. Target motion

We assume the simple linear dynamics

$$\begin{aligned} x_{t+1} &= x_t + \Delta_t \dot{x}_t \\ y_{t+1} &= y_t + \Delta_t \dot{y}_t \\ z_{t+1} &= z_t + \Delta_t \dot{z}_t \\ \dot{x}_{t+1} &= \dot{x}_t \\ \dot{y}_{t+1} &= \dot{y}_t \\ \dot{z}_{t+1} &= \dot{z}_t \end{aligned} \tag{12}$$

for the target object's state.

3.2.3. Linearization

Since $\mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$ is nonlinear and we must approximate the system described in Eq. (3) by linearizing around an arbitrary point $\hat{\mathbf{x}}_t$. We write

$$\mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) \approx \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t) + \mathrm{J}_{f_t}(\mathbf{x}_t - \hat{\mathbf{x}}_t), \tag{13}$$

where J_{f_t} is the Jacobian

$$\mathrm{J}_{f_t} = \left[\frac{\partial \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)}{\partial \mathbf{x}_t} \right] \tag{14}$$

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

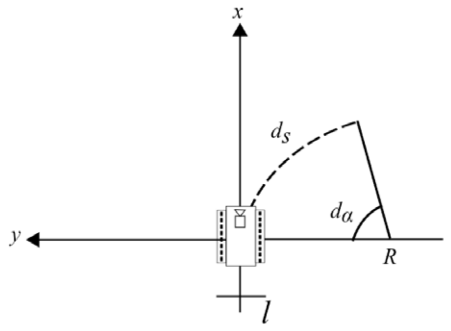

Fig. 3. Schematic of robot motion model. l is the wheel base, d_s is the distance traveled (arc length), R is the turning radius, and d_α is the angle turned.

3.3. Sensor model

The target 2D position, size and orientation are received from the tracking and redetection functions. See Section 4.

The measurement from such an algorithm is simply a bounding box at time t:

$$\mathbf{z}_t = \left[u_t, v_t, w_t^{img}, h_t^{img} \right]^T. \tag{15}$$

We model the sensor with a function $\mathbf{h}(\cdot)$ mapping the system state $\mathbf{x}_t$ to the corresponding sensor measurement

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \boldsymbol{\zeta}_t, \tag{16}$$

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

For a pinhole camera with focal length f and principal point (c_x, c_y), ignoring any in-plane rotation of the cylindrical object, we can write

$$\begin{aligned} u_t &= (f x_t^{cam} + c_x)/z_t^{cam} \\ v_t &= (f y_t^{cam} + c_y)/z_t^{cam} \\ w_t^{img} &= f w_0 / z_t^{cam} \\ h_t^{img} &= f h_0 / z_t^{cam}, \end{aligned} \tag{17}$$

where (h_0, w_0) is the assumed target's height and width.

Here $\mathbf{x}_t^{cam} = [x_t^{cam}, y_t^{cam}, z_t^{cam}, 1]^T$ is the homogeneous representation of the rigid transformation of the target's center into the camera coordinate system:

$$\mathbf{x}_t^{cam} = \mathrm{T}_t^{W/C} \begin{bmatrix} x_t \\ y_t \\ z_t \\ 1 \end{bmatrix}, \tag{18}$$

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

$$\mathrm{T}_t^{W/C} = \mathrm{T}^{R/C} \mathrm{T}_t^{W/R}. \tag{19}$$

$\mathrm{T}_t^{W/R}$ is the rigid transformation from the world coordinate system to the robot coordinate at time t, and $\mathrm{T}^{R/C}$ is the (fixed) transformation from the robot coordinate system to the camera coordinate system. We can write

$$\mathrm{T}_t^{W/R} = \begin{bmatrix} \mathrm{R}_t^T & -\mathrm{R}_t^T \mathbf{x}_t^r \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{20}$$

where $\mathbf{x}_t^r = (x_t^r, y_t^r, z_t^r)$ and R_t^T represents the orientation of the robot in the world coordinate frame.

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

$$\mathrm{J}_{h_t} = \left[\frac{\partial \mathbf{h}(\mathbf{x}_t)}{\partial \mathbf{x}_t} \right] \tag{21}$$

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

3.4. Initialization

To initialize the system, we need an a-priori state vector $\mathbf{x}_0$. As previously explained, we assume that the robot is at the origin of the world coordinate system or that an alternative initial position 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 write

$$\hat{\mathbf{x}}_0 = [x_0, y_0, z_0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T = \mathbf{h}^{inv}(\mathbf{z}_0). \tag{22}$$

We only need to estimate the initial world-coordinate position (x_0, y_0, z_0) of the target from $\mathbf{z}_0$. We first obtain an initial position $[x_0^{cam}, y_0^{cam}, z_0^{cam}, 1]^T$ in the camera coordinate frame then, noting that the robot frame at time 0 is also the world frame, we can map to the world coordinate frame by

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} = \left(\mathtt{T}^{R/C}\right)^{-1} \begin{bmatrix} x_0^{cam} \\ y_0^{cam} \\ z_0^{cam} \\ 1 \end{bmatrix}. \tag{23}$$

Inspecting the system in Eq. (17), 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 get $z_0^{cam} = fh_0/h_0^{img}$. We use h_0^{img} instead of w_0^{img} on assumption that the user-specified bounding box is more accurate vertically than horizontally.

3.5. 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:

$$\mathtt{S}_t = \lambda^2 \begin{bmatrix} (w_t^{img})^2 & 0 & 0 & 0 \\ 0 & (h_t^{img})^2 & 0 & 0 \\ 0 & 0 & (w_t^{img})^2 & 0 \\ 0 & 0 & 0 & (h_t^{img})^2 \end{bmatrix}. \tag{24}$$

We use $\lambda = 0.15$ in our simulations. 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:

$$\mathtt{P}_0 = \mathtt{J}_{h^{inv}}\mathtt{S}_0\mathtt{J}_{h^{inv}} + \mathrm{diag}(0, 0, 0, \eta, \eta, \eta, 0, 0, 0, 0, 0, 0). \tag{25}$$

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

We assume for simplicity that the state transition noise covariance $\mathtt{Q}_t$ is diagonal. We let

$$v_t = \sqrt{\dot{x}_t^2 + \dot{y}_t^2 + \dot{z}_t^2}, \tag{26}$$

and then 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 robot's position be $\Delta_t^2(\rho_5 d_s + \rho_6)$, and we let the entries of $\mathtt{Q}_t$ corresponding to the robot's orientation be $\Delta_t^2(\rho_7 d_s + \rho_8)$. This noise distribution is overly simplistic and ignores many factors, but it is sufficient for the experiments reported upon in this paper. In total, there are nine free parameters $(\eta, \rho_1, \rho_2, \dots, \rho_8)$ that must be determined through hand tuning or calibration. In our simulation we find the optimal parameters using gradient decent.

3.6. 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 leaves 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 $\mathtt{P}$ by propagating the sensor measurement error for $\mathbf{z}_t$ through $\mathbf{h}^{inv}(\mathbf{z}_t)$ as previously explained in Section 3.5. Here is the modified EKF algorithm:

```
1. Input $\mathbf{z}_0$.
2. Calculate $\hat{\mathbf{x}}_0$ and $\mathtt{P}_0$.
3. For $t = 1, \dots, T$, do
   (a) Predict $\hat{\mathbf{x}}_t^- = \mathbf{f}(\hat{\mathbf{x}}_{t-1}, \mathbf{u}_{t-1})$.
   (b) Calculate $\mathtt{J}_{ft}$ and $\mathtt{Q}_t$.
   (c) Predict $\mathtt{P}_t^- = \mathtt{J}_{ft}\mathtt{P}_{t-1}\mathtt{J}_{ft}^T + \mathtt{Q}_t$.
   (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 $\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^- + \mathtt{K}_t(\mathbf{z}_t - \mathbf{h}_t(\hat{\mathbf{x}}_t^-))$.
       iii. 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.
```

4. Visual tracking and redetection

In this section, we briefly outline the visual tracking and detection algorithms before providing details. Although any tracker that returns a 2D bounding box for the target object could be used, in the experiments reported in this paper, we use CAMSHIFT, which is based on the principle of color histogram backprojection. The idea is to build a color histogram from an initial image I_0, then, on each successive frame, to calculate, for each pixel in the image, the probability of that pixel's color according to the histogram. Obtaining these probabilities is called backprojection. Backprojection here should not be confused with the notion from projective geometry of backprojecting a 2D image point to obtain 3D ray.

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 redetection when candidate target most similar to the appearance model is similar enough and large enough.
6. Run CAMSHIFT to confirm redetected region. If confirmed, reinitialize; otherwise, return to redetection.

4.1. Initialization and CAMSHIFT tracking

On the first frame $\mathtt{I}_0$, we expect the user to provide the target object's position and size in the form of a bounding box (x_0^c, y_0^c, w, h). In our implementation, in operation, the user selects the initial bounding box with a mouse.

Although our methods are compatible with any fast feature histogram-based image region tracker, in our experiments, we use CAMSHIFT [34]. CAMSHIFT combines traditional mean shift with an adaptive region sizing step. Given an initial position of the detection window in frame $t-1$, for frame t, it computes a backprojection of the appearance model onto the image and then estimates the center of mass of the backprojection among the pixels in a region of interest slightly expanded from the detection window from time $t-1$. The method then shifts the detection window to the calculated center of mass and repeats the estimation and shifting process until convergence. Finally, the moments $M_{..}$ of the backprojection in the region of interest are calculated and used to set the size of the detection window for frame I_t as follows [43].

$$x_c = \frac{M_{10}}{M_{00}}, \qquad y_c = \frac{M_{01}}{M_{00}}, \qquad r = \frac{M_{20}/x_c^2}{M_{02}/y_c^2},$$
$$w = \sqrt{2M_{00}} \cdot r, \qquad h = \sqrt{2M_{00}}/r.$$

On each frame, if CAMSHIFT returns a reasonable bounding box, the parameters of the adaptive suspension threshold are updated. The bounding box check and threshold calculation are described in the next section.

4.2. Suspending tracking

CAMSHIFT works extremely well so long as the target appearance remains consistent and distinctive with respect to the background. However, when the target object leaves the scene, is occluded, or impinges a background region with a similar color distribution, the tracking region tends to change rapidly in size, growing into background regions or moving to a different location completely. When this happens during a target pursuit application, lest the pursuit motion planner becomes confused, it is important to suspend tracking and attempt to redetect the target object.

To achieve this, as a first measure, we impose simple constraints on the target detection window's location and size. If the target object's estimated size or location changes by an amount inconsistent with robot and target dynamics, clearly, the tracker is lost and needs to be reinitialized.

However, such simple location and size constraints are not sufficient. We find that in cluttered scenes, when the target is partially or wholly occluded or leaves the scene, CAMSHIFT tends to get distracted by background regions, oftentimes without a sufficiently large change in position or size to flag suspension.

We therefore, before committing to CAMSHIFT's estimate of the target at time t, verify the quality of the candidate detection region using an adaptive threshold applied to the dissimilarity between the candidate region's color histogram $\mathbf{h}_t^r$ and the appearance model $\mathbf{h}^m$. We use the default OpenCV histogram comparison function [44], which returns a distance based on the Bhattacharyya coefficient

$$d_t \equiv d(\mathbf{h}_t^r, \mathbf{h}^m) = \sqrt{1 - \sum_i \sqrt{\mathbf{h}_t^r(i) \cdot \mathbf{h}^m(i)}}. \tag{27}$$

(The implementation also normalizes the histograms to sum to 1.) The resulting distance varies between 0, for identical histograms, to 1, for non-overlapping histograms.

The histogram comparison threshold that we apply to d_t is computed adaptively. We keep running estimates of the distance measure's mean and standard deviation

$$\mu_n = \mu_{n-1} + \frac{d_t - \mu_{n-1}}{n},$$
$$\sigma_n = \sqrt{\frac{(n-2)\sigma_{n-1}^2 + (d_t - \mu_{n-1})(d_t - \mu_n)}{t-1}}, \tag{28}$$

and then, for $t > 1$, we suspend tracking when we obtain a new distance that deviates too far from the running mean, i.e., when

$$d_t > \mu_n + \theta\sigma_n.$$

Algorithm 1 TARGET TRACKING: track target object, suspend tracking, and call redetection algorithm when target is not clear.

```
Input: $(\mathbf{b}_0 \leftarrow (x_0^c, y_0^c, h_0^{img}, w_0^{img}), \mathtt{I}_0)$
 1: $\mathtt{H}^m \leftarrow$ CALCHIST$(\mathtt{I}_0, \mathbf{b}_0)$
 2: $t \leftarrow 1$
 3: $n \leftarrow 0$
 4: $TargetAcquired \leftarrow true$
 5: $\beta_t \leftarrow 0$
 6: while ($\mathtt{I}_t$ = read frame from IO) do
 7:     if $TargetAcquired$ then
 8:         $[\mathbf{b}_t, d_t] \leftarrow$ TRACK$(\mathtt{I}_t, \mathtt{H}^m, \mathbf{b}_{t-1})$
 9:     else
10:         $[\mathbf{b}_t, d_t] \leftarrow$ REDETECT$(\mathtt{I}_t, \mathtt{H}^m, \beta_{t-1}, \mathbf{b}_{t-1})$
11:     end if
12:     if $(n = 0)$ then
13:         $n \leftarrow 1$
14:         $\mu_n \leftarrow d_t$
15:         $\sigma_n \leftarrow 0$
16:         $\beta_t \leftarrow \mu_n$
17:     else if $(n = 1)$ then
18:         $n \leftarrow 2$
19:         update $\mu_n$ and $\sigma_n$
20:         $\beta_t \leftarrow \mu_n + \theta\sigma_n$
21:     else
22:         $\beta_t \leftarrow \mu_n + \theta\sigma_n$
23:         if $(d_t \le \beta_t)$ And $(h_t^{img} > h_{t-1}^{img} \times 0.5)$ then
24:             $n \leftarrow n + 1$
25:             update $\mu_n$ and $\sigma_n$
26:             $TargetAcquired \leftarrow true$
27:         else
28:             $\mathbf{b}_t \leftarrow \mathbf{b}_{t-1}$
29:             $TargetAcquired \leftarrow false$
30:         end if
31:     end if
32:     $t \leftarrow t + 1$
33: end while
```

Algorithm 2 TRACK: target tracking function.

```
1: function TRACK$(\mathtt{I}, \mathtt{H}^m, \mathbf{b})$
2:     $\mathtt{P} \leftarrow$ BACKPROJECT$(\mathtt{I}, \mathtt{H}^m)$
3:     $\mathbf{b} \leftarrow$ CAMSHIFT$(\mathtt{P}, \mathbf{b})$
4:     $\mathtt{H}^r \leftarrow$ CALCHIST$(\mathtt{I}, \mathbf{b})$
5:     $d \leftarrow$ COMPAREHIST$(\mathtt{H}^r, \mathtt{H}^m)$
6:     return $\mathbf{b}, d$
7: end function
```

θ is a threshold on the z-score of the newly measured distance. We use $\theta = 3$ in our experiments. Algorithm 1 provides a formal definition of the algorithm.

4.3. Target redetection

While tracking is suspended, on every frame, we need to execute the target redetection phase of the algorithm. The flow is shown in the light blue box in Fig. 2, and an example of the result of each step is shown in Fig. 4. We detail each step here.

4.3.1. Backproject the appearance model

The color histogram $\mathtt{H}^m$ gives us the probability $P(\mathtt{I}_t(x,y) \mid \text{target})$ of observing a pixel with color $\mathtt{I}_t(x,y)$ given that the pixel is actually in the target region. Backprojection of the appearance model $\mathtt{H}^m$ simply means that we generate a new image $\mathtt{P}$ such that $\mathtt{P}_t(x,y) = P(\mathtt{I}_t(x,y) \mid \text{target})$ according to $\mathtt{H}^m$. $\mathtt{P}$ will have in general several clusters of pixels with high values, indicating some

Algorithm 3 Target Redetection: redetect candidate region and reinitialize the target tracker.

```
1: function ReDetect($\mathtt{I}, \mathtt{H}^m, \beta, \mathbf{b}$)
2:     $\mathtt{P} \leftarrow$ BackProject($\mathtt{I}, \mathtt{H}^m$)
3:     $\mathtt{M} \leftarrow$ Binarize($\mathtt{P}$)
4:     $\mathtt{R} \leftarrow$ Open($\mathtt{M}$)
5:     $C \leftarrow$ GenerateComponents($\mathtt{R}$)
   ▷ Filter out the smaller region 30% below the target size at $t$ and compare the similarity with the target object.
6:     $\mathbf{r} \leftarrow$ ExtractSimilarRegion($C, \beta, \mathtt{H}^m, \mathbf{b}$)
7:     if $\mathbf{r} \neq$ Null then
8:         $\mathbf{r} \leftarrow$ CAMShift($\mathtt{P}, \mathbf{r}$)
9:         $\mathtt{H}^r \leftarrow$ CalcHist($\mathtt{I}, \mathbf{r}$)
10:        $d \leftarrow$ CompareHist($\mathtt{H}^r, \mathtt{H}^m$)
11:    else
12:        $\mathbf{r} \leftarrow \mathbf{b}$
13:        $d \leftarrow 1$
14:    end if
15:    return $\mathbf{r}, d$
16: end function
```

degree of consistency of the region with $\mathtt{H}^m$. An example is shown in Fig. 4(b).

4.3.2. Binarize likelihood image

In this step, to eliminate weak matches between $\mathtt{I}_t$ and the appearance model, we threshold $\mathtt{P}$ using the standard Otsu method to obtain a binary image $\mathtt{M}$ indicating candidate target pixels.

4.3.3. Generate connected components

In this step, we apply morphological erosion and dilation to $\mathtt{M}$ to eliminate noise and fill gaps, then extract the connected components. In our experiments, we use a square structuring element 3 pixels wide. An example is shown in Fig. 4(c).

4.3.4. Generate candidate regions

In this step, we eliminate any connected components with an area less than 30% of the target object's size in the last frame before tracking was suspended, then we find the rectangular bounding box of each surviving connected component. If no candidate regions remain, we continue to the next frame. An example of surviving bounding boxes is shown in Fig. 4(d) and overlaid on the original image in Fig. 4(e).

4.3.5. Find most similar region

In this step, we obtain the color histogram of each region surviving the previous step and compare with the appearance model $\mathtt{H}^m$ using Eq. (27). If the smallest distance is below the adaptive threshold calculated in the tracking phase, we reinitialize CAMSHIFT using the corresponding best region. An example of successful reinitialization is shown in Fig. 4(f). Pseudocode for the visual tracking algorithm is shown in Algorithm 3.

5. Experimental setup

The main objective of the proposed method is to use the pursuit robot's sensors (monocular camera and wheel encoders) to estimate a smooth, accurate trajectory for the target relative to the pursuit robot. Our evaluation thus proceeds in two parts. First, we evaluate the joint localization model, which combines the visual tracker's sensor measurements with wheel encoder data and the kinematic model to generate a temporally smoothed trajectory taking the pursuit robot's kinematics into account. Then,

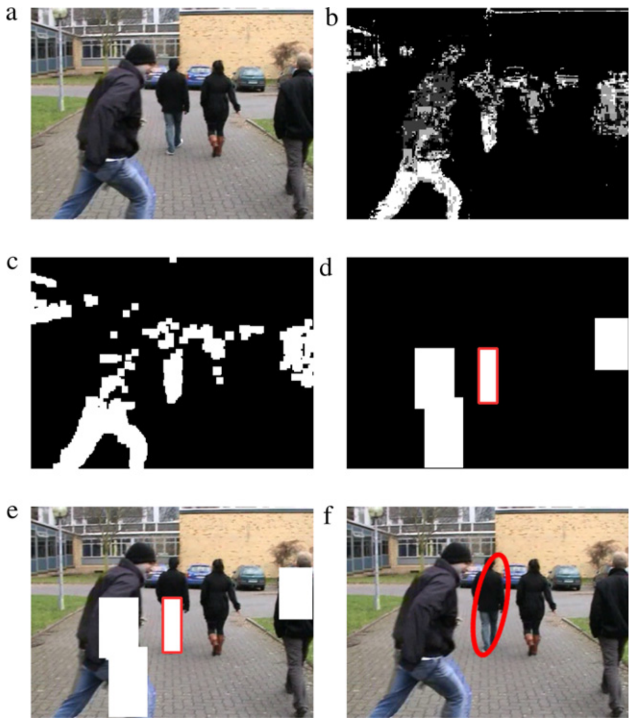

Fig. 4. Example target redetection steps. (a) Example image I_t containing the target. (b) Backprojection $\mathtt{P}_t$ of appearance model. (c) Binarized and filtered versions of $\mathtt{P}_t$. (d) Bounding boxes of candidate regions after filtering out small connected components. (e) Candidate regions from (d) overlaid on $\mathtt{I}_t$. (f) Reinitialized CAMSHIFT result. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

we evaluate the visual tracker, which provides an instantaneous estimate of the target's position relative to the pursuit robot.

For the joint localization algorithm, we performed experiments under three conditions: (1) simulation, which allows detailed analysis of the effects of different noise levels, (2) real world indoor environments, which allow ground truth trajectories to be carefully measured, and (3) real world outdoor environments, which allow realistic, albeit qualitative, evaluation of the robot's and target's estimated trajectories.

Experiment I briefly evaluates the method in simulation. The pursuit robot moved at a constant speed of 1 m/s along a curved trajectory, and the simulated target moved with a constant speed of 1 m/s along a piecewise linear path of three segments. The simulation also included a period of time in which the robot and target are traveling parallel to each other, with the target outside the pursuer's camera's field of view. During these times, no sensor measurements are observed (see Fig. 5(a)). We hypothesized that the proposed method would perform better than a simplistic smoothing method when the pursuit robot's odometry measurements are accurate and worse than the simple smoothing method when the pursuit robot's odometry measurements are extremely noisy. To test this hypothesis, we varied the simulated pursuit robot's odometry noise from 0 (perfect odometry) to 108% of the distance traveled and observed our algorithm's resulting estimation error.

To test the performance of the proposed method in the real world, in Experiment II, the pursuit robot moved to follow a human target performing arbitrary simple and complex (curved) movements. The tests were in an indoor environment to obtain detailed quantitative performance measurements under realistic visual tracking noise. We pre-calibrated the camera's intrinsic parameters, measured and marked the floor in order to accurately measure ground truth trajectories of the robot and target, then recorded video while the target and pursuer were standing still then moving simultaneously. The target's motion followed a curve

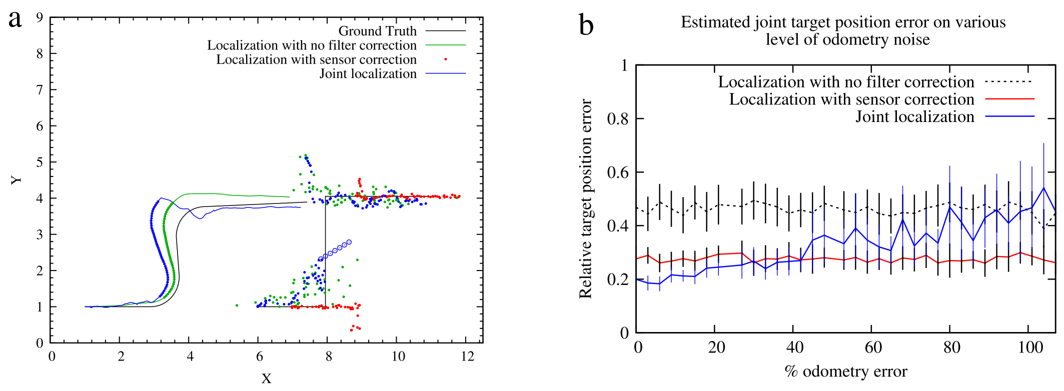

Fig. 5. Experiment I results. (a) Absolute paths from one sample run. Pursuer (left) and target (right) moved along the black curves. Odometry noise and sensor noise were fixed throughout the experiment. Blue path and blue points show robot path and target locations estimated from raw odometry and sensor data without filtering. Green path and green points show robot path and target locations estimated from raw odometry and sensor correction. Red path and red points show robot path and target locations estimated with model-based filtering algorithm of Section 3. Thick lines delimited by filled squares denote the period of time in which the target was outside the pursuer's camera's field of view. Red open squares show target positions estimated from raw sensor measurements without any filtering. The filter is able to smooth the noisy target positions, but absolute position estimates are biased due to accumulation of odometry error. (b) We repeated the scenario shown in part (a) 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

with left and right motion, whereas the pursuer veered off to the left because of wheel slippage but kept the target in view. The sensor measurements always came from the actual visual tracking algorithm tested in Experiments IV and V. As in Experiment I, we compare the error in the robot-relative positions of the target over each run with either raw sensor measurements, simple smoothing, or the proposed method. For further comparison, we also examine the trajectories of the pursuer and target according to the ground truth and estimates.

In Experiment III, we performed a more realistic, albeit qualitative, evaluation in which we teleoperated the pursuit robot to follow a human target moving along arbitrary curved trajectories in an outdoor environment. Due to the difficulty in obtaining ground truth data in the outdoor case, we simply visually compare the stability and smoothness of the relative trajectories of the proposed method in comparison to the two baseline methods.

Finally, for the visual tracker, we collected visual data in both simple and complex outdoor environments using video from handheld cameras. To extensively test the method's ability to recover from occlusion of the target, we included a video sequence by Klein et al. [45]. See Fig. 9(b). The target was always a human who in every video either moved out of the camera's field of view or was occluded one or more times. In some cases the camera rotated to reacquire the target, and in others, the target moved back into the field of view. All four scenes were scenes in which the visual tracker is mostly successful at tracking so long as the target is clearly visible in the scene. All four videos were acquired at 30 fps. Two videos, "b" and "c", were acquired at 320×240, whereas videos "a" and "d" were acquired at 640×480. Experiment IV tests the visual tracker's accuracy over these video sequences, and Experiment V tests its runtime performance. The low resolution videos were left out of the runtime performance evaluation in Experiment V.

6. Results and discussion

In this section, we first provide a detailed description of the baseline methods for comparison, and then we provide detailed results and analysis for each of the five experiments 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 [34,35]; the sensor-based relative target position estimate is simply accepted. Hence it is named "Localization with no filter correction". The second baseline uses sensor measurement correction based on an ordinary extended Kalman filter [46] 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 3.

6.1. *Experiment I: Simulation*

We tested the joint localization algorithm in simulation as previously described. Example estimated absolute trajectories for the robot and target with sensor noise 20% and odometry noise 9% under the three models are shown in Fig. 5(a). The estimated absolute paths of the robot and target contain error accumulated over the entire run, but in target pursuit, the target's position *relative to the pursuer* is much more important than the absolute position, so in the remaining results, we analyze, over time, the error in the target's estimated position relative to the robot.

To prove the efficacy of joint localization coupling target dynamics and pursuit robot kinematics in improving localization, we simulated the robot's odometry measurements with varying noise levels. For each noise level, we repeated the experiment 10 times, with the same setup, the same robot and target trajectories, and the same motion model parameters, but we obtained different samples from the sensor and odometry noise distributions. For each noise level, we first generated one synthetic simulation data sequence, optimized the motion model, then ran the 10 separate simulations using the same set of motion model parameters. Fig. 5(b) shows the average relative target position error as a function of odometry error level. As expected, sensor-only estimation and the reduced model show no sensitivity to odometry error level, but our method is indeed sensitive to the odometry error level. However, for the reasonable range of odometry error (30% or less), our method still substantially outperforms sensor-only estimation and modestly outperforms the reduced model.

During the simulations, on each iteration of localization, we computed the RMSE between the estimated and ground truth target positions in the robot coordinate frame, using each of the three methods. We also suspended relative target localization when the visual tracker suspended target tracking. In these cases, we wait for the detector to redetect the target and reinitialize the visual

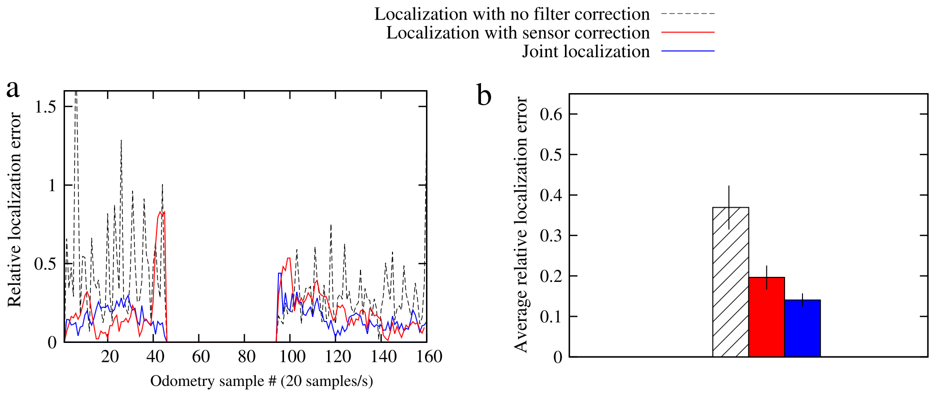

Fig. 6. Experiment I results, continued. The proposed method is compared to the two baseline methods in simulation with 9% odometry and 15% sensor error. (a) Relative position estimation error over time for one representative run. (b) Average relative error over the full run shown in (a). Error bars denote 95% confidence intervals. Joint localization is 58.8% and 26.3% better than the baseline methods.

tracker, then we reinitialize localization. See Fig. 6(a) for results of a sample run with 9% odometry error and 15% visual tracker sensor error. We performed statistical tests on the RMSE of each method over this run. See Fig. 6(b). The proposed method significantly outperformed both baseline methods by 58.8% and 26.3%.

We conclude that the proposed method provides more stable and smooth estimates of the target's relative trajectory than do the baseline methods.

6.2. Experiment II: Real world, indoors

We recorded ground truth measurements, odometry data, and camera images while teleoperating the pursuit robot to follow a human target indoors. Since our goal was to evaluate the extent to which target tracking improves relative position error, we selected an arbitrary run in which the target remained in the field of view. Fig. 7(a) and (b) show the robot and target trajectories in the x–y plane. Fig. 7(c) shows the relative target position error at each point in the run for the same three algorithms tested in Experiment III, and Fig. 7(d) shows the average (RMSE) relative position error of the three state estimation methods over the entire run. Fig. 7(e) shows sample results of visual tracking.

The pursuer and target initially remained still. We recorded the initial position of the pursuer as $(0, 0, 0)$ in the world coordinate system and measured the target's initial true position as $(470, 0, 80)$ cm. Then the target moved away from the pursuer, veering left and right, and we teleoperated the pursuer to follow. Due to wheel slippage, the pursuer veered diagonally in the $+y$ direction but kept the target in view. Resulting estimated trajectories are compared to the ground truth in Fig. 7(a) and (b). As expected, thetarget was easily tracked by all of the filtering methods while both target and pursuer were still. During target and pursuer motion, however it is clear from the data that filtering is much better than the very noisy raw data, and joint localization provides smoother and more stable estimates than the baseline methods.

The results show that both joint localization and localization with sensor correction consistently outperform localization with no filter correction. On average, the proposed joint localization estimates are 40.5% better than localization with no filter and 27.6% better than localization with sensor correction. To test whether the difference between joint localization and localization with sensor correction is significant, we performed a t-test on difference in the sample means and found a significant difference ($t = -5.16$, $p < 0.001$).

In Fig. 7(c), at $t = 25$ there is a spike in the relative distance error. This is due to noise in the visual tracker, which reported a change in the height of the target in the image, causing a change in the estimated distance to the target from the robot. Near $t = 150$ there is a point at which the pursuer takes an arc motion to the left while the target moves on a complex trajectory. Here we observe another increase in the noise in the sensor measurements. The raw sensor measurement based estimate is extremely noisy at this point, whereas the proposed localization model shows a more smooth and less noisy trajectory compared to the other two methods.

6.3. Experiment III: Real world, outdoors

As previously explained, due to the difficulty in obtaining metric ground truth data in an outdoor environment, we only perform a qualitative evaluation of the method's ability in the outdoor environment. In Fig. 8, we show the resulting estimated position of the target in the world coordinate frame over an arbitrary run in which the target was always in the field of view. The target moved away from the robot, describing a complex trajectory. When needed, we transform the estimated pursuer-relative position of the target into the world coordinate frame using the assumption that the pursuit robot began at position $(0, 0, z_r, 0, 0, 0)$ in the world coordinate frame, where z_r is the vertical offset of the robot's center from the ground, and the target began at its first raw sensor-based estimated position relative to the robot.

The results show clearly that the proposed method provides less noisy, more smooth, and more stable estimates of the target's trajectory than do the other two methods.

Spikes and abrupt changes in the estimated trajectories can be observed after $t = 140$, especially evident after $t = 150$. At this time, the robot was performing a somewhat complex motion, a sudden arcing motion in a counter clockwise direction. At this point the bounding box received from the visual tracker becomes very noisy, and combined with the jerky motion of the pursuit robot, we obtain noisy target position estimates from the raw sensor measurements. However, the proposed method estimates a smoother trajectory compared to the baseline methods. The results could be further improved if we used a method to remove extremely noisy sensor measurements; we could then in principle obtain smooth trajectories like those before $t = 140$ in Fig. 8.

6.4. Experiment IV: Visual tracking accuracy

In the first frame of each video (R1 in Fig. 9), we initialized CAMSHIFT tracking by selecting the human target in the scene. We then ran the proposed tracking, suspension, and redetection method to the end of each video.

During tracking, we 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^r_t. In almost all cases, when the target left the scene, the distance d_t exceeded the

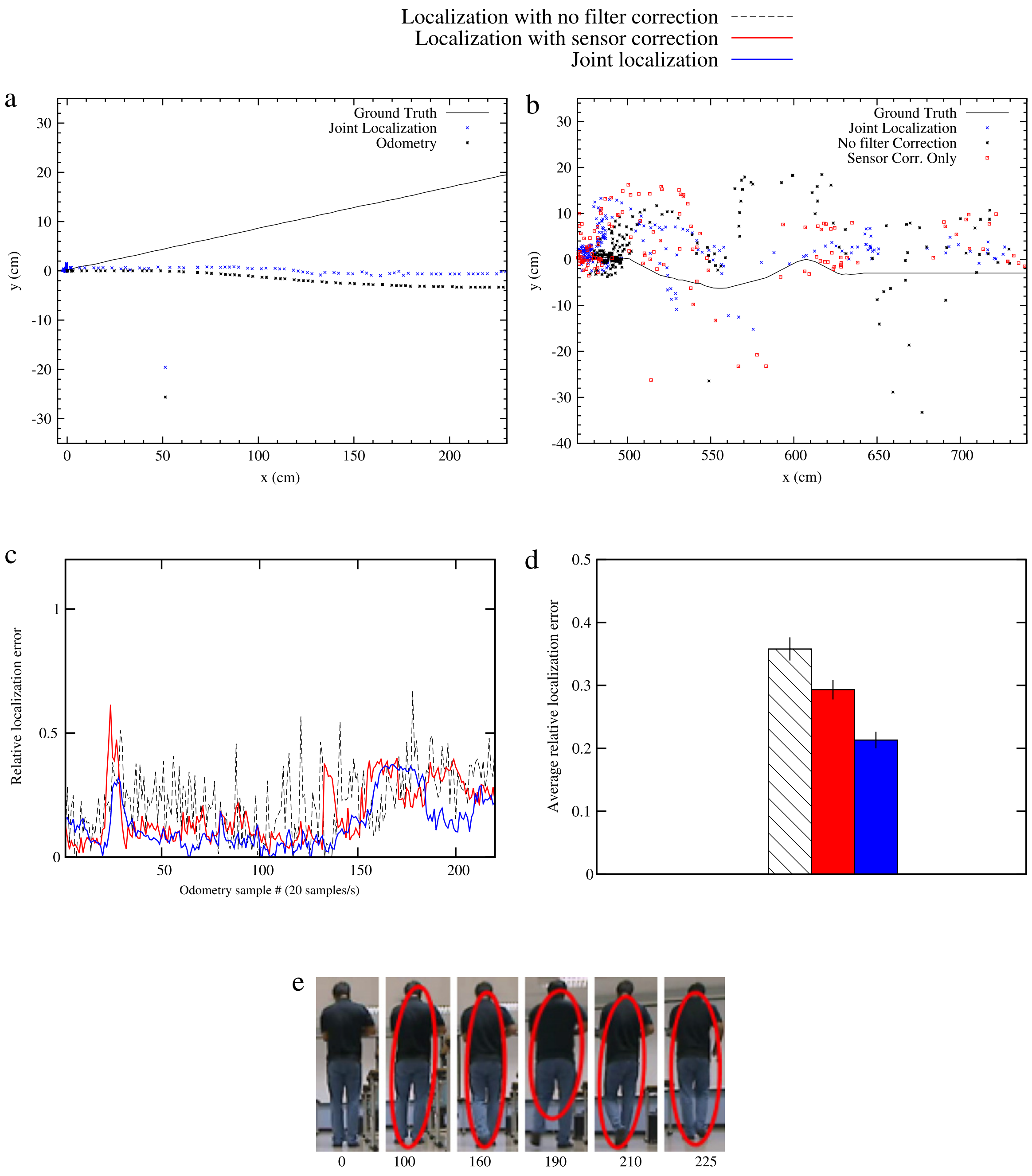

Fig. 7. Experiment II results. Real world data, indoor environment, with ground truth. (a) Top view of pursuer's ground truth and estimated trajectory during the experiment. Initial position was assumed to be (0, 0, 0) in the world frame. (b) Top view of target's ground truth and estimated trajectory. Initial position was measured to be (470, 0, 80) cm in the world frame. (c) Relative position estimation error over time. (d) Average relative error over the full run shown in (c). Error bars denote 95% confidence intervals. Joint localization is 40.5% better than localization with no filter and 27.6% better than localization with sensor correction. (e) Sample visual target tracking results.

adaptive threshold, except for a few cases in which the redetection algorithm found a sufficiently similar object in the background.

Rows R2 and R3 in Fig. 9 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 the target in the field of view.

Rows R4 and R5 in Fig. 9 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 case, CAMSHIFT is correctly reinitialized, as shown in row R6 of Fig. 9.

Over the four videos, the target was successfully tracked in 95.12% of the total frames, the target was missed in 1.99% of the total frames, and a false target was detected and tracked in 4.67% of the total frames. The accuracy results on a per-video basis are summarized in Table 1. The video data sequences are available at the first author's website [47].

6.5. *Experiment V: Visual tracking run time performance*

We tested the runtime performance of the visual tracker 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. We measured the average time required for tracking (the standard

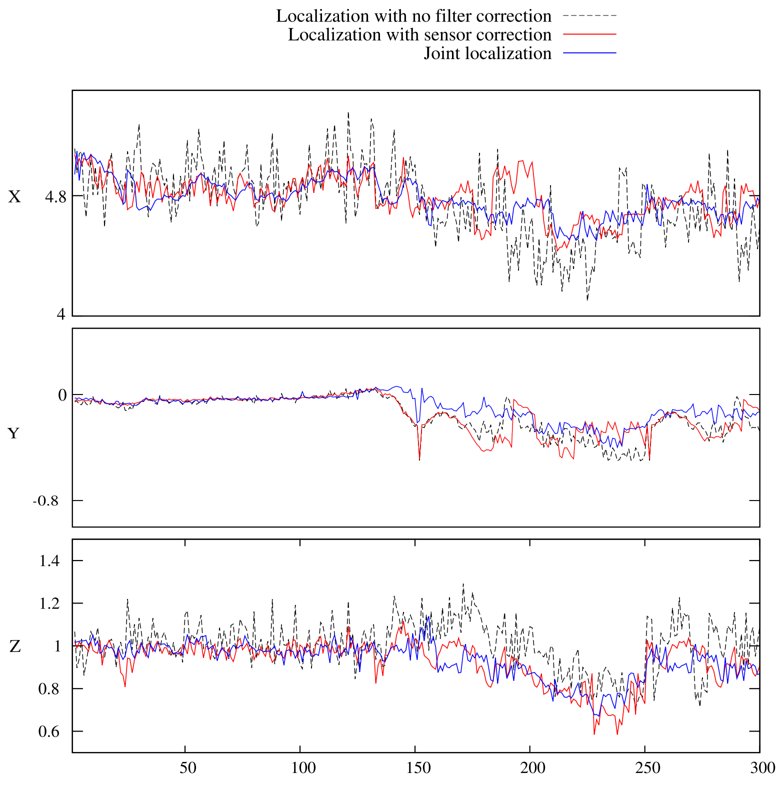

Fig. 8. Experiment III results. Estimated positions (x_t, y_t, z_t) of the target in the pursuer coordinate system over time in one run of the outdoor environment. In our coordinate system, x is forward, y is left–right, and z is up–down.

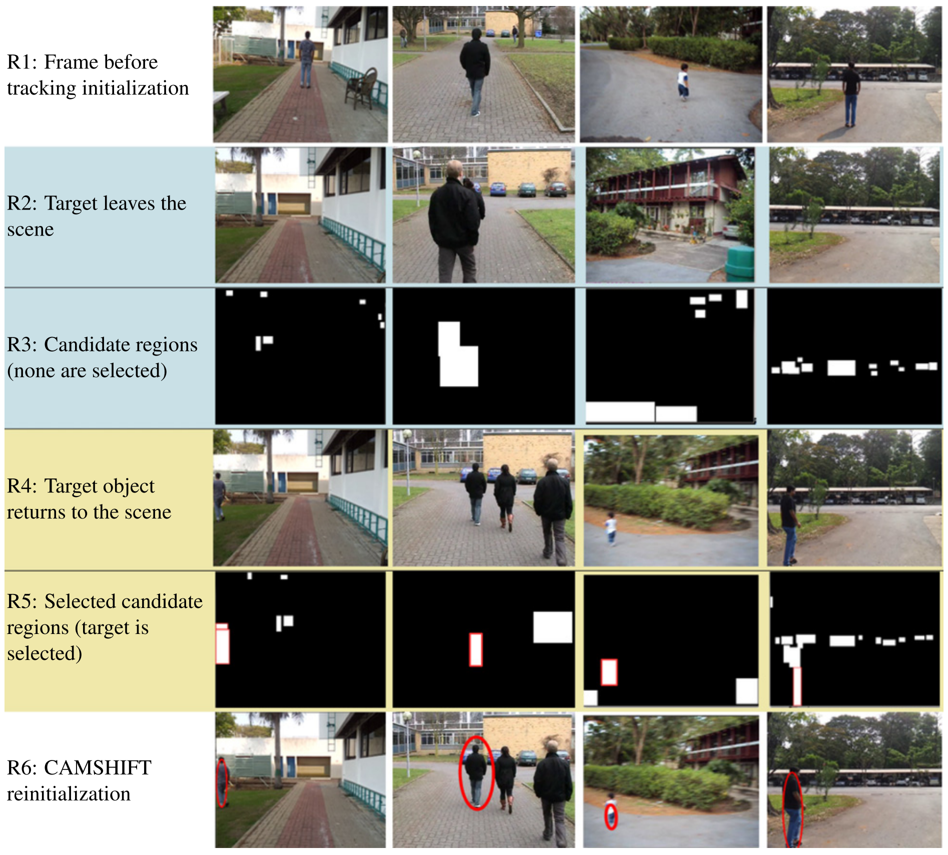
R1: Frame before tracking initialization
R2: Target leaves the scene
R3: Candidate regions (none are selected)
R4: Target object returns to the scene
R5: Selected candidate regions (target is selected)
R6: CAMSHIFT reinitialization

Fig. 9. Experiment IV results. The proposed method was tested in different outdoor 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. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Experiment IV results. For each video, we report the number of frames in which the target is visible and occluded or outside the field of view, whether multiple targets are visible, as well as the number of hits, misses, and false detections for objects other than the actual target.

Video	Frames w. target		Multiple objects	Detection ratio		
	Present	Occluded/Absent		Hit	Miss	False
a	637	35	No	597 (88.84%)	3 (0.45%)	78 (11.61%)
b	997	20	Yes	1005 (98.82%)	0 (0.0%)	12 (1.18%)
c	389	50	No	425 (96.81%)	7 (1.59%)	21 (4.78%)
d	426	174	No	584 (97.33%)	6 (1%)	22 (3.67%)
Online dataset						
e	947	0	No	909 (95.9%)	18 (1.9%)	20 (2.12%)
f	305	0	No	278 (91.15%)	0 (0%)	27 (8.85%)
g	453	45	Yes	439 (96.91%)	9 (1.99%)	5 (1.1%)

Table 2
Experiment V results. Runtime performance of tracking and redetection algorithms on two different processors.

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

CAMSHIFT routine plus histogram distance measurement and adaptive threshold update) and target redetection over all relevant frames in the three high resolution test videos.

The results are summarized in Table 2. Both algorithms 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 on-board execution by a mobile robot with modest computational resources.

As discussed in Section 1.3, there are many alternatives for tracking besides CAMSHIFT. Many of them would potentially provide better person detection accuracy and tracking performance while the target is in the field of view. Indeed, these methods would be complementary to our approach—as already explained, any method that provides a noisy 2D bounding box for the target could be used with our localization method without any other changes. Although to the best of our knowledge, none of the state of the art detection and tracking methods are sufficiently fast to run in real time on commodity embedded systems hardware, this will certainly change in the future, leading to the idea for future work of combining person detection and tracking using HOG or other state-of-the-art methods with color-based or texture-based appearance modeling.

7. Conclusion

In this paper, we propose a joint localization method for pursuit SUGVs that integrates target dynamics and pursuit robot kinematics 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 camera-centered or robot-centered coordinate system. We show that joint localization produces more stable, smooth, and noise-resistant trajectories than those produced by standard filters or a system without any filter. 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 also propose a computationally fast visual tracker for pursuit robots. We propose an adaptive histogram similarity threshold used to suspend the visual tracker when the target is occluded or leaves the field of view. We find that visual trackers need proper reinitialization when the target reappears in the camera's field of view. To accomplish this, we introduce a fast redetection algorithm based on histogram backprojection that searches for the target over the entire image in real time. Experiments IV and V demonstrate that the visual tracker is robust and sufficiently fast to run on embedded systems with modest resources typical of mobile robots.

We believe that up till now, research on target pursuit applications has been far from the point of commercial exploitation, due in large part to the high cost and bulk of the 3D sensors necessary. The developments reported upon in this paper dramatically reduce the sensory apparatus necessary for efficient and effective target pursuit, replacing the complex sensors with a single monocular camera, thus bringing the concept closer to the point of practical implementation in commercial products.

Although we have demonstrated that the proposed method is appropriate for joint localization of a moving pursuer and target, it would be insufficient for target tracking scenarios in which accurate global localization of the pursuer is required. Global localization is the province of SLAM, in which we maintain a joint posterior over stationary landmark states and the moving robot. However, the two approaches are complementary—moving target landmarks could easily be integrated with most SLAM methods (especially those based on EKFs). The proposed sensor and system state evolution models could be used directly in nearly any probabilistic SLAM algorithm. The target observation would provide additional constraints on the pursuer's state estimate, possibly improving its global localization accuracy, and the static landmark observations, by further constraining the pursuer's state estimate, would in turn improve the target's global localization accuracy.

In future work, first, we plan to remove the limitation of a fixed real-world target height, by automatically estimating the height of the target object by tracking it over time. Second, we will integrate the method with a complete processing stream including path planning and obstacle avoidance.

Acknowledgments

This research was supported by a Royal Thai Government research grant to MND and PL. AB was supported by graduate fellowships from the University of Balochistan, the Higher Education Commission of Pakistan, and the Asian Institute of Technology. We are grateful for the support of Sergio Goncalves and Phongsathorn Eakamongul, who assisted with robot hardware and control software.

References

[1] G.R. Gerhart, C.M. Shoemaker, D.W. Gage, PackBot: a versatile platform for military robotics, in: Unmanned Ground Vehicle Technology VI, 2004.
[2] N. Irimie, A. Zorila, A. Nan, P. Schiopu, Remote control of a small unmanned ground vehicle (SUGV), in: Proc. SPIE 7821, Advanced Topics in Optoelectronics, Microelectronics, and Nanotechnologies, 2010.
[3] J.J. Carafano, A. Gudgel, The pentagon's robots: Arming the future, Heritage Found. Backgr. 2096 (2007) 1–6.
[4] M. Kobilarov, G. Sukhatme, J. Hyams, P. Batavia, People tracking and following with mobile robot using an omnidirectional camera and a laser, in: IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2006, pp. 557–562.

[5] M. Awai, T. Shimizu, T. Kaneko, A. Yamashita, H. Asama, HOG-based person following and autonomous returning using generated map by mobile robot equipped with camera and laser range finder, in: Intelligent Autonomous Systems 12. Vol. 194, Springer, Berlin, Heidelberg, 2013, pp. 51–60.
[6] A. Ohshima, S. Yuta, Tracking and following people and robots in crowded environment by a mobile robot with SOKUIKI sensor, in: Distributed Autonomous Robotic Systems 8, Springer, 2009, pp. 575–584.
[7] Y. Sung, W. Chung, Human tracking of a mobile robot with an onboard LRF (Laser Range Finder) using human walking motion analysis, in: International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), IEEE, 2011, pp. 366–370.
[8] T. Yoshimi, M. Nishiyama, T. Sonoura, H. Nakamoto, S. Tokura, H. Sato, F. Ozaki, N. Matsuhira, H. Mizoguchi, Development of a person following robot with vision based target detection, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2006, pp. 5286–5291.
[9] Z. Chen, S.T. Birchfield, Person following with a mobile robot using binocular feature-based tracking, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2007, pp. 815–820.
[10] W. Choi, C. Pantofaru, S. Savarese, Detecting and tracking people using an RGB-D camera via multiple detector fusion, in: IEEE International Conference on Computer Vision Workshops (ICCV), IEEE, 2011, pp. 1076–1083.
[11] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges, in: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI, IJCAI, Acapulco, Mexico, 2003.
[12] A.J. Davison, I.D. Reid, N.D. Molton, O. Stasse, Monoslam: Real-time single camera SLAM, IEEE Trans. Pattern Anal. Mach. Intell. 29 (6) (2007) 1052–1067.
[13] J. Lou, H. Yang, W.M. Hu, T. Tan, Visual vehicle tracking using an improved EKF, in: Asian Conference of Computer Vision, ACCV, 2002, pp. 296–301.
[14] M. Pupilli, A. Calway, Real-time camera tracking using a particle filter, in: British Machine Vision Conference, 2005.
[15] F. Shimin, G. Qing, X. Sheng, T. Fang, Human tracking based on mean shift and Kalman filter, in: International Conference on Artificial Intelligence and Computational Intelligence, AICI, Vol. 3, 2009, pp. 518–522.
[16] N. Funk, A study of the kalman filter applied to visual tracking, University of Alberta, Project for CMPUT 652.
[17] V. Karavasilis, C. Nikou, A. Likas, Visual tracking by adaptive kalman filtering and mean shift, in: Artificial Intelligence: Theories, Models and Applications, Springer, 2010, pp. 153–162.
[18] D. Lowe, Object recognition from local scale-invariant features, in: The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999. Vol. 2, 1999, pp. 1150–1157.
[19] H. Bay, T. Tuytelaars, L. Gool, Surf: Speeded up robust features, in: Computer Vision—ECCV 2006, Vol. 3951, Springer, Berlin, Heidelberg, 2006, pp. 404–417.
[20] M. Yokoyama, T. Poggio, A contour-based moving object detection and tracking, in: Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005, pp. 271–276.
[21] M. Calonder, V. Lepetit, C. Strecha, P. Fua, Brief: Binary robust independent elementary features, in: Computer Vision—ECCV 2010, Springer, 2010, pp. 778–792.
[22] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, Orb: an efficient alternative to sift or surf, in: IEEE International Conference on Computer Vision (ICCV), IEEE, 2011, pp. 2564–2571.
[23] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). Vol. 1, IEEE, 2005, pp. 886–893.
[24] M. Munaro, F. Basso, E. Menegatti, Tracking people within groups with RGB-D data, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2012, pp. 2101–2107.
[25] H. Zhou, Y. Yuan, C. Shi, Object tracking using sift features and mean shift, Comput. Vis. Image Underst. 113 (3) (2009) 345–352.
[26] D.-N. Ta, W.-C. Chen, N. Gelfand, K. Pulli, SURFTrac: Efficient tracking and continuous object recognition using local feature descriptors, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2009, pp. 2937–2944.
[27] S. Denman, V. Chandran, S. Sridharan, An adaptive optical flow technique for person tracking systems, Pattern Recognit. Lett. 28 (10) (2007) 1232–1239.
[28] W. Zajdel, Z. Zivkovic, B. Krose, Keeping track of humans: Have i seen this person before?, in: IEEE International Conference on Robotics and Automation, ICRA, 2005, pp. 2081–2086.
[29] F. Porikli, Integral histogram: a fast way to extract histograms in Cartesian spaces, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR, Vol. 1, 2005, pp. 829–836.
[30] S. Perreault, P. Hebert, Median filtering in constant time, IEEE Trans. Image Process. 16 (9) (2007) 2389–2394.
[31] M. Sizintsev, K. Derpanis, A. Hogue, Histogram-based search: A comparative study, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2008, pp. 1–8.
[32] X. Chen, H. Huang, H. Zheng, C. Li, Adaptive bandwidth mean shift object detection, in: IEEE Conference on Robotics, Automation and Mechatronics, 2008, pp. 210–215.
[33] X. Chen, Q. Huang, P. Hu, M. Li, Y. Tian, C. Li, Rapid and precise object detection based on color histograms and adaptive bandwidth mean shift, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2009, pp. 4281–4286.
[34] G. Bradski, Real time face and object tracking as a component of a perceptual user interface, in: IEEE Worksop on Applications of Computer Vision, WACV, 1998, pp. 214–219.
[35] J.G. Allen, R.Y.D. Xu, J.S. Jin, Object tracking using camshift algorithm and multiple quantized feature spaces, in: Pan-Sydney Area Workshop on Visual Information Processing, Vol. 36, 2004, pp. 3–7.
[36] D. Comaniciu, V. Ramesh, P. Meer, Real-time tracking of non-rigid objects using mean shift, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Vol. 2, 2000, pp. 142–149.
[37] D. Comaniciu, V. Ramesh, P. Meer, Kernel-based object tracking, IEEE Trans. Pattern Anal. Mach. Intell. 25 (5) (2003) 564–577.
[38] S.M. LaValle, Planning Algorithms, Cambridge University Press, Cambridge, UK, 2006, Available at http://planning.cs.uiuc.edu/.
[39] M. Nitulescu, Theoretical aspects in wheeled mobile robot control, in: Automation, Quality and Testing Robotics, Vol. 2, 2008, pp. 331–336.
[40] A. Basit, M.N. Dailey, P. Laksanacharoen, Model driven state estimation for target pursuit, in: IEEE International Conference on Control, Automation, Robotics & Vision, ICARCV, 2012, pp. 1077–1082.
[41] A. Basit, M.N. Dailey, P. Lakanacharoen, J. Moonrinta, Fast target redetection for CAMSHIFT using back-projection and histogram matching, in: International Conference on Computer Vision Theory and Applications, VISAPP, 2014.
[42] A. Basit, W.S. Qureshi, M.N. Dailey, K. Tomáš, Joint localization of pursuit quadcopters and target using monocular cues, J. Intell. Robot. Syst. 78 (3–4) (2015) 613–630.
[43] Exner, Bruns, Kurz, Grundhöfer, Bimber, Fast and robust CAMShift tracking, in: IEEE International Workshop on Computer Vision for Computer Games, CVCG, 2010. URL: http://www.jku.at/cg/content/e48345/.
[44] G. Bradski, A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, OŔeilly Media, Inc., 2008.
[45] D.A. Klein, D. Schulz, S. Frintrop, A.B. Cremers, Adaptive real-time video-tracking for arbitrary objects, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2010, pp. 772–777.
[46] D. Comaniciu, V. Ramesh, Mean shift and optimal prediction for efficient object tracking, in: International Conference on Image Processing, ICIP, Vol. 3, 2000, pp. 70–73.
[47] A. Basit, Test video sequences, 2014. URL: http://www.cs.ait.ac.th/abasit/research_data.html.

Abdul Basit received the B.S. in Computer Science from Shaheed Zulfikar Ali Bhutto Institute of Science and Technology (SZABIST), Karachi, Pakistan, in 2005. He received the M.Sc. degree in Computer Science from the Asian Institute of Technology (AIT), Thailand, in 2010. He is currently a Ph.D. candidate in Computer Science at AIT and a Lecturer at the University of Balochistan (UoB). His research interests lie in machine vision for control, automation, and robotics.

Matthew N. Dailey received the B.S. and M.S. in Computer Science from North Carolina State University and the Ph.D. in Computer Science and Cognitive Science from the University of California, San Diego. He spent two years as a Research Scientist with Vision Robotics Corporation of San Diego, CA USA and two years as a Lecturer in the Computer Science and Information Technology programs at Sirindhorn International Institute of Technology, Thammasat University, Thailand. In 2006, he joined the Computer Science and Information Management Department at the Asian Institute of Technology, Thailand, where he is now an Associate Professor. His research interests lie in machine learning, machine vision, robotics, systems security, and high performance computing.

Jednipat Moonrinta received the B.Eng. in Computer Engineering from Chiang Mai University, Thailand, in 2008 and the M.Eng. in Computer Science from the Asian Institute of Technology, Thailand, in 2010. In 2010, he joined the Computer Science and Information Management research center at the Asian Institute of Technology, Thailand, where he is a Research Associate. His research interests are in machine vision, robotics, image processing, and machine learning.

Pudit Laksanacharoen received the B.Eng. in Mechanical Engineering from King Mongkut's University of Technology Thonburi, Thailand and the M.S. and Ph.D. in Mechanical Engineering from Case Western Reserve University in Cleveland OH, USA in 2001. He is now an Associate Professor in Mechanical and Aerospace Engineering at King Mongkut's University of Technology North Bangkok, Thailand. In 2004, he was a Visiting Researcher at the Hirose-Fukushima Robotics Lab at the Tokyo Institute of Technology, Japan. His research interests are in biologically inspired robots, creative mechanical design, and rehabitation robots.