Model Driven State Estimation for Target Pursuit

Abdul Basit
Computer Science and
Information Management
Asian Institute of Technology
Pathumthani, Thailand
Email: st107840@ait.ac.th

Matthew N. Dailey
Computer Science and
Information Management
Asian Institute of Technology
Pathumthani, Thailand
Email: mdailey@cs.ait.ac.th

Pudit Laksanacharoen
Mechanical Engineering
King Mongkut's University of
Technology (North Bangkok)
Bangsue, Bangkok, Thailand
Email: stl@kmutnb.ac.th

***Abstract*—Autonomous target pursuit is an extremely useful technology for surveillance applications. In this paper, we derive and evaluate, in a realistic simulation, a novel tracking algorithm for vision-based pursuit. We assume a simple ground-based surveillance robot equipped with a single monocular camera. For the sensor, we propose the use of a color histogram based region tracker. We integrate models of the robot's kinematics and the target's dynamics with a model of the color region tracking sensor via an extended Kalman filter. Detailed simulation results demonstrate that the tracking algorithm substantially reduces the relative position estimation error introduced by noisy color region tracking. The algorithm thus enables target pursuit based on an extremely noisy but simple and low cost sensor.**

I. Introduction

Video surveillance using an array of fixed camera sensors is difficult due to the limited resolution and field of view of each camera. The problem can be partly addressed through the use of remotely operated pan-tilt-zoom cameras, but it is still impossible to achieve complete coverage at high resolution throughout a given security zone. Surveillance cameras on mobile platforms such as ground or airborne robots can complement a stationary surveillance camera system, adding increased range and precision of surveillance.

However, it is difficult for an operator to manually teleoperate robots, particularly when he or she would like to track and follow a particular target of interest moving at a natural or even evasive speed. As one solution to this problem, we are exploring technology allowing a security operator to identify a suspicious target in the video feed from a mobile robot then command the robot to attempt to keep the target in view as it moves through the environment, behind obstacles, and so on.

We call this problem *autonomous target pursuit*. In autonomous target pursuit, besides obstacle modeling and navigation, one of the central problems is to use the camera to keep track, over time, of the target relative to the pursuit robot's position as both are moving. Our interest is to use robots such as the iRobot PackBot for autonomous pursuit. Here we focus on systems like our own simple all-terrain surveillance robot (see Fig. 1), which is equipped with teleoperation capabilities and a single monocular camera.

Autonomous pursuit with a monocular camera sensor requires us to track arbitrary objects. Researchers have proposed several methods for arbitrary target object tracking. Objects with well-defined edges can be tracked using *contour based* methods [1]. *Feature based* tracking methods extract reliable features such as SURF [2] and SIFT [3] from the object of interest and track them over time. These methods are very robust but computationally intensive, so they require approximation and/or hardware acceleration to achieve real time performance.

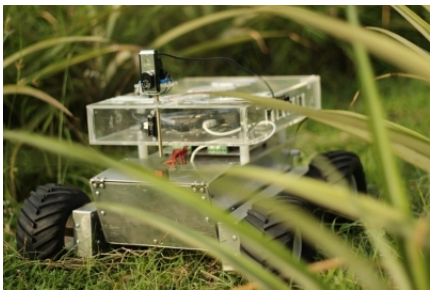

Fig. 1. All-terrain robot for tracking and pursuit of arbitrary objects using a monocular camera.

In terms of speed and simplicity, the most important category of object tracking methods is based on *histogram matching* [4]. The most popular color histogram based methods are probably mean shift [5], [6] and cam-shift [7]. Due to these methods' speed and simplicity, we propose their use for tracking arbitrary objects during autonomous pursuit with a single monocular camera. The main limitation of these methods, however, is that they are all extremely noisy, especially in cluttered outdoor environments. While both robot egomotion estimation and target tracking from fixed sensors are well-understood problems, joint estimation of pursuer and target trajectories in real time, using a monocular camera and color histogram tracking as the sensor, is a challenging unsolved problem.

Our approach is to perform rough global localization of the pursuit robot and target in an obstacle map. For optimal localization of the robot and target, one would use encoder-based odemtry, visual odometry, landmark observations, all the priori knowledge of the target and a model of the target's trajectory, but it would be extremely difficult to incorporate all of this knowledge in a real time tracking algorithm.

In this paper we therefore derive and evaluate, in a realistic simulation, a novel tracking algorithm for vision based pursuit. We integrate models of differential drive robot kinematics [8],

[9] and target dynamics with a model of the color region tracking sensor via an extended Kalman filter [10]. Detailed simulation results demonstrate that the tracking algorithm substantially reduces the relative position estimation error introduced by noisy color region tracking.

The algorithm thus enables target pursuit based on an extremely noisy but simple and low cost sensor, a monocular camera with color region tracking.

II. Joint Estimation of Robot and Target State

In this section, we describe the tracking algorithm in detail. We model the robot's state, the target's state, and the color region tracking sensor in the extended Kalman filter framework.

A. 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

$$\mathbf{x}_t = [x_t, y_t, z_t, w_0, h_0, \dot{x}_t, \dot{y}_t, \dot{z}_t, x_t^r, y_t^r, z_t^r, \gamma_t^r, \beta_t^r, \alpha_t^r]^T,$$

where (x_t, y_t, z_t) is the target's position, (w_0, h_0) is the target's size (the object is assumed to be cylindrical), $(\dot{x}_t, \dot{y}_t, \dot{z}_t)$ 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). All positions and orientations are expressed in the world coordinate frame.

B. State Transition

We assume that initially, the world coordinate frame is aligned with the robot coordinate frame, i.e., $(x_0^r, y_0^r, z_0^r) = (0, 0, 0)$ and $(\gamma_0, \beta_0, \alpha_0) = (0, 0, 0)$. (Alternatively, a specific initial position and orientation could be given.) 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

$$\mathbf{u}_t = \begin{bmatrix} d_t^L & d_t^R \end{bmatrix}^T,$$

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 received from each encoder, the number of ticks per revolution, and the diameter of the wheel. The motion model is

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

where $\nu_t \sim \mathcal{N}(\mathbf{0}, \mathtt{Q}_t)$. $\mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$ has two components. The first component models the kinematics of a differential drive robot with constant linear and angular velocity over short time periods (acceleration is modeled as noise). See Fig. 2 for a schematic. The second component is a first order linear dynamical system for the target's motion. We describe each component in turn.

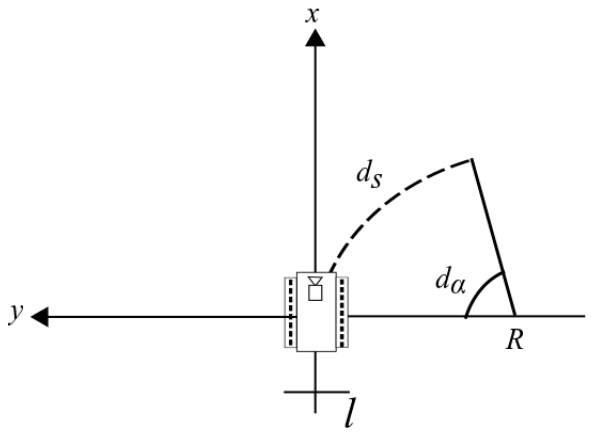

Fig. 2. Schematic of robot motion model. l is the wheel base, d_s is the distance traveled (arc length), R is the turning radius, and d_α is the angle turned.

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 is

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

The change in yaw in the robot coordinate system is

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

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

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

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

$$\begin{bmatrix} d_x \\ d_y \end{bmatrix} = R \begin{bmatrix} 1 - \cos(d_\alpha) \\ \sin(d_\alpha) \end{bmatrix}. \tag{2}$$

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 Equation 2 as $R \to \infty$ to obtain

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

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

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

in detail, expanding $\mathtt{R}_t$, we get

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

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

To determine α_{t+1}, β_{t+1}, and γ_{t+1}, we let $\mathtt{R}$ be the rotation matrix corresponding to a rotation of d_α around the z axis in the robot ground plane. Then the new orientation of the vehicle, expressed as a rotation matrix, is

$$\mathtt{R}_{t+1} = \mathtt{R}_t\mathtt{R},$$

To extract Euler rotations from $\mathtt{R}_{t+1}$, we use

$$\begin{aligned}\gamma_{t+1} &= \operatorname{atan2}(r_{32}, r_{33})\\ \beta_{t+1} &= \operatorname{atan2}\left(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}\right)\\ \alpha_{t+1} &= \operatorname{atan2}(r_{21}, r_{11}),\end{aligned}$$

where r_{ij} represents the (i,j)-th element of $\mathtt{R}_{t+1}$.

2) Target motion: We assume the simple linear dynamics

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

for the target object's state. In this paper we assume the object's size is fixed and known, so in fact $w_t = w_0$ and $h_t = h_0$ for all t.

3) Linearization: Since $\mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$ is nonlinear and we will be using a Kalman filter, we must approximate the system described in Eq. 1 by linearizing around an arbitrary point $\hat{\mathbf{x}}_t$. We write

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

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

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

evaluated at $\hat{\mathbf{x}}_t$.

C. Sensor model

We assume the robot's target tracking camera is mounted in a fixed, nearly vertical position with roll (rotation around the principle axis) close to 0. We further assume that the system incorporates a 2D tracking algorithm capable of producing, at time t, an estimate of the 2D bounding box of the object's projection into the camera plane. In our application, the operator initially selects the bounding box of the target to be pursued in the first video frame, then we use the standard CAMSHIFT algorithm from OpenCV to track the object from frame to frame.

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

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

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 $\mathbf{h}(\cdot)$ mapping the system state $\mathbf{x}_t$ to the corresponding sensor measurement

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \zeta_t,$$

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

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

$$\begin{aligned}u_t &= (f x_t^{cam} + c_x)/z_t^{cam}\\ v_t &= (f y_t^{cam} + c_y)/z_t^{cam}\\ w_t^{img} &= f w_0/z_t^{cam}\\ h_t^{img} &= f h_0/z_t^{cam}.\end{aligned} \tag{3}$$

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:

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

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

$$\mathtt{T}_t^{W/C} = \mathtt{T}^{R/C}\mathtt{T}_t^{W/R}.$$

$\mathtt{T}_t^{W/R}$ 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. In detail, if from the robot's orientation at time t, we obtain the rotation matrix

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

we can write

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

where $\mathbf{x}_t^r = (x_t^r, y_t^r, z_t^r)$.

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

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

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

D. Initialization

To initialize the system, we need an a-priori state vector $\mathbf{x}_0$. As previously explained, we assume 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 measurment $\mathbf{z}_0$ and write

$$\hat{\mathbf{x}}_0 = [x_0, y_0, z_0, w_0, h_0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T = \mathbf{h}^{inv}(\mathbf{z}_0).$$

Since w_0 and h_0 are assumed known, 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 $[x_0^{cam}, y_0^{cam}, z_0^{cam}, 1]$ 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

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

Inspecting the system in Eq. 3, 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 $z_0^{img} = fh_0/h_0^{img}$ on the assumption that the user-specified bounding box is more accurate vertically than horizontally.

E. 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:

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

We use $\lambda = 0.1$ in our simulations. For the initial state error, 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:

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

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

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

$$v_t = \sqrt{\dot{x}_t^2 + \dot{y}_t^2 + \dot{z}_t^2}.$$

We let the entries of $\mathtt{Q}_t$ corresponding to the target postion be $\Delta_t^2(\rho_1 v_t^2 + \rho_2)$, and we let the entries of $\mathtt{Q}_t$ correspoding 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.

F. 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 $\mathbf{h}^{inv}(\mathbf{z}_t)$ as explained in Section II-E. 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 $\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^- + \mathtt{K}_t(\mathbf{z}_t - \mathbf{h}_t(\hat{\mathbf{x}}_t^-))$
      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

III. Experiments and Results

We performed two simulation experiments to validate the efficacy of the proposed method for correcting the trajectory of the target relative to the robot. For both experiments, we generated a synthetic trajectory for the robot and target and added random odometry noise and sensor error. The simulated robot moved at a constant speed of 1 m/s along the S-shaped curve shown in green on the left of Fig. 3, and the simulated target moved with a constant speed of 1 m/s along the three straight paths shown in green on the right of Fig. 3. Included in the simulation is a period of time where the robot and target are travelling parallel to each other with the target outside the robot's camera's field of view. During this time, no sensor measurements are observed. In Experiment I, we fixed the odometery noise to a typical level, Guassian with standard deviation equivalent to 9% of the distance traveled. In Experiment II, we varied odometry noise from 0 to 80% of the distance traveled and observed our algorithm's resulting estimation error. In both experiments, to model the sensor measurements, we first project the actual synthetic target into the image, find the bounding box, then add synthetic noise to the bounding box parameters. The noise was Gaussian with standard deviation for u_t and w_t^{img} equal to 20% of w_t^{img} and standard deviation for v_t and h_t^{img} equal to 20% of h_t^{img}. Both experiments used the same simulated camera generating 640×480 images at 20 fps with a focal length of 550 pixels (horizontal field of view 60°.

A. Experiment I (fixed odometery noise)

The results for Experiment I, in which we fixed the odometry noise to 9% and the sensor noise to 20%, are shown in Fig. 3. We observe that the odometry-only estimates of the

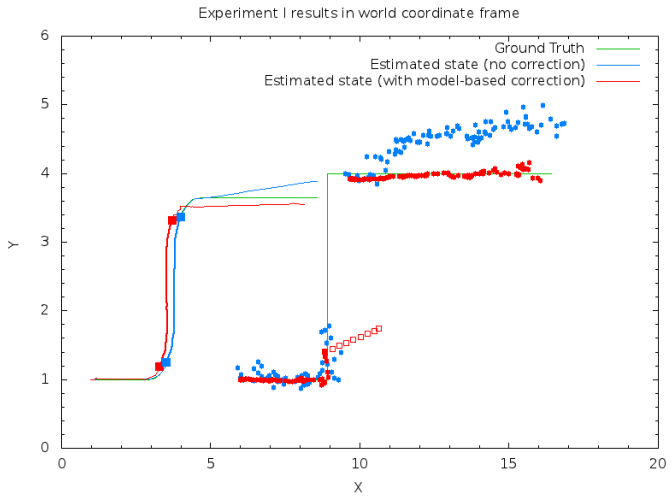

Fig. 3. Experiment I results. Robot (left) and target (right) moved along the green lines. Odometry noise and sensor noise were fixed throughout the experiment. Blue path and blue points show estimated robot path and sensor measurements estimated directly from odometry and the sensor measurements without filtering. Red path and red points show estimated robot path and sensor measurements with model-based filtering algorithm of Section II. Thick lines delimeted by filled squares denote the period of time in which the target was outside the camera view. Red open squares show the propagation of the estimated target position without any measurements. The filter is able to smooth the noisy target positions but absolute position measurements are biased due to robot positioning error.

target's positions are extremely noisy, due to combined effect of accumulated odometry error and the error in the sensor measurements themselves. The corrected target trajectory is much more smooth than the sensor-only path and is much closer to the true trajectory initially, when the estimated robot state is very close to the true state. When the target abruptly changes direction, the estimated target path is less accurate, showing hysteresis due to the use of the estimated target velocity. When the target leaves the field of view, we see that the the predictions quickly deviate from the true target path, but the estimate recovers when the target reappears and the filter is reinitialized.

Both the sensor-only estimated target trajectory and the corrected target trajectory are relatively far from the true target path towards the end of the simulation, but this is clearly due to the accumulated odometry error. In target pursuit, the target's position *relative to the pursuer* is much more important than the absolute position, so we next analyze, over time, the error in the target's estimated position relative to the robot. The data are shown in Fig. 4.

The results show clearly that model-based correction of the noisy color region tracking sensor measurements consistently outperforms sensor-only estimation. Fig. 5 shows an overall comparison between the relative error of sensor-only estimation and model-based correction. On average, the corrected target position estimates are more than 50% better than the sensor-based measurements.

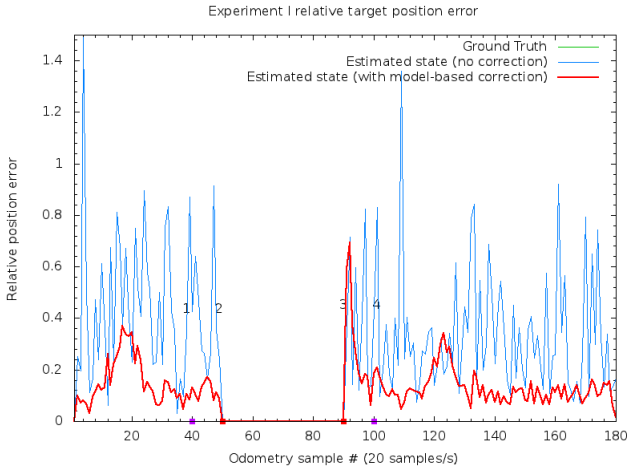

Fig. 4. Experiment I results (relative position error). The relative position error of the sensor-only estimates are shown in blue, and the relative position error after model-based correction are shown in red. Sample 40, labeled "1," is the point at which the target makes a 90° left turn (refer to Fig. 3). Sample 50, labeled "2," is the point at which the target leaves the camera field of view. Sample 90, labeled "3," is the point at which the target reappears in the camera field of view. Sample 100, labeled "4," is the point at which the target makes a 90° right turn. The filter reduces the relative position error and also reduces the variance of the relative position error.

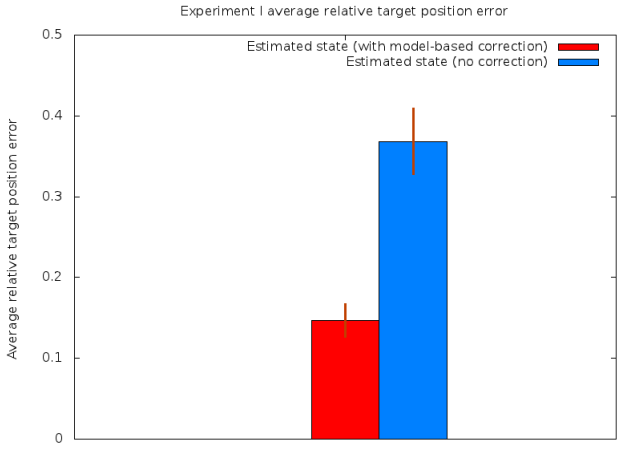

Fig. 5. Experiment I results (average relative position error). Error bars denote 95% confidence intervals. The filter reduces the relative position error by more than 50%.

B. *Experiment II*

Our method attempts to optimally combine odometry information and sensor measurements to improve upon the relative target position estimation error of sensor-only estimation. This means that our method will perform better with more accurate odometry and conversely worse with less accurate odometry. This might limit the ability of our method when the pursuit robot is moving over rough terrain, for example. To test this expected dependency of model-based correction on accurate odometry, in Experiment II, we compared the performance of sensor-only estimation and model-based correction under increasing odometry error. We began with odometry error of

0 (the robot always has exact knowledge of its position) and gradually increased the standard deviation of the noise added to the odometry measurements up to 80% of the distance traveled. For each noise level, we repeated the experiment 10 times, with the same setup and robot and target trajectories as Experiment I. Fig. 6 shows the average relative target position error as a function of odometry error level. As expected, sensor-only estimation shows no sensitivity to odometry error level, but our method is indeed sensitive to the odometry error level. However, for the reasonable range (20% or less), our method still substantially outperforms sensor-only estimation.

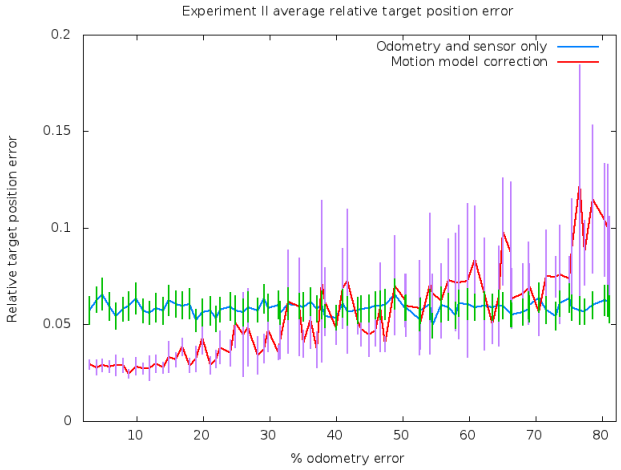

Fig. 6. Experiment II results. We repeated Experiment I with different levels of odometry noise and measured the average relative target position error for sensor-only estimation and model-based correction. Error bars denote 95% confidence intervals.

IV. Conclusion

Target pursuit is an extremely useful technology for surveillance applications. However, there is no existing system able to track and pursue arbitrary targets autonomously. In this paper, we take first steps towards the goal of enabling a relatively simple ground-based robot with a monocular camera to track an arbitrary target during pursuit of that target. For the sensor, we propose the use of a color histogram based region tracker. The only manual intervention needed is that the operator must describe the initial bounding box of the target in the video feed, and the only prior knowledge needed is the height of the target in the real world.

Although 2D color region tracking has obvious benefits — such trackers are extremely fast and rarely lose their target — they are quite noisy. In a first experiment, we show that our filter, which incorporates models of both the color region tracking sensor and the sensor's (robot's) motion, substantially reduces the relative position estimation error incurred by the noisy sensor. In a second experiment, we show that the filter is quite robust to reasonable levels of odometry error.

There are some limitations to this preliminary work. The method has not yet been tested in the real world. Color region tracking alone may not be suitable for all environments and all targets. We currently require knowledge of the target's height to achieve accurate tracking results. The absolute target position estimates are not very accurate.

In future work, we plan to address all of these issues, by testing the algorithm on our ROS-based pursuit robot testbed, experimenting with more sophisticated trackers, treating unknown target geomtry as a hidden estimation problem, and using real time visual odometry to further improve upon odometry error during pursuit.

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.

References

[1] M. Yokoyama and T. Poggio, "A contour-based moving object detection and tracking," in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, oct. 2005, pp. 271 – 276.

[2] D.-N. Ta, W.-C. Chen, N. Gelfand, and K. Pulli, "Surftrac: Efficient tracking and continuous object recognition using local feature descriptors," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, june 2009, pp. 2937 –2944.

[3] H. Zhou, Y. Yuan, and C. Shi, "Object tracking using sift features and mean shift," *Computer Vision and Image Understanding*, vol. 113, no. 3, pp. 345 – 352, 2009.

[4] Y. Wei and L. Tao, "Efficient histogram-based sliding window," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, june 2010, pp. 3003 –3010.

[5] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2, 2000, pp. 142–149.

[6] ——, "Kernel-based object tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 5, pp. 564–577, May 2003.

[7] J. G. Allen, R. Y. D. Xu, and J. S. Jin, "Object tracking using camshift algorithm and multiple quantized feature spaces," in *2003 Pan-Sydney Area Workshop on Visual Information Processing (VIP2003)*, ser. CRPIT, M. Piccardi, T. Hintz, S. He, M. L. Huang, and D. D. Feng, Eds., vol. 36. Sydney, Australia: ACS, 2004, pp. 3–7. [Online]. Available: http://crpit.com/confpapers/CRPITV36Allen.pdf

[8] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[9] M. Niulescu, "Theoretical Aspects in Wheeled Mobile Robot Control," in *Automation, Quality and Testing Robotics*, vol. 2, 2008, pp. 331–336.

[10] G. Welch and G. Bishop, "An introduction to the kalman filter," *In Practice*, vol. 7, no. 1, pp. 1–16, 2006.