Model Driven State Estimation for Target Pursuit

Abdul Basit*,†, Matthew N. Dailey*, and Pudit Laksanacharoen‡

*Computer Science and Information Management, Asian Institute of Technology
Pathumthani, Thailand. Email: Abdul.Basit@ait.asia, mdailey@ait.asia
†University of Balochistan, Address, City
‡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.

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.

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] and cam-shift [6]. 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 odometry, 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 [7], [8] and target dynamics with a model of the color region tracking sensor via an extended Kalman filter [9]. 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 dynamics in the world coordinate frame. We define the system state at time \( t \) to be

\[
x_t = [x_t, y_t, z_t, \dot{x}_t, \dot{y}_t, \dot{z}_t, x'_t, y'_t, z'_t, \gamma'_t, \beta'_t, \alpha'_t]^T,
\]

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, \((x'_t, y'_t, z'_t)\) is the pursuit robot’s position, and \((\gamma'_t, \beta'_t, \alpha'_t)\) 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., \((x'_0, y'_0, z'_0) = (0, 0, 0)\) 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

\[
u_t = [d^L_t, d^R_t]^T,
\]

where \(d^L_t\) is the distance traveled by the left wheel and \(d^R_t\) 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

\[
x_{t+1} = f(x_t, u_t) + \nu_t,
\]

where \(\nu_t \sim \mathcal{N}(0, Q_t)\). \(f(x_t, u_t)\) 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.

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

\[
d_s = \frac{d^L_t + d^R_t}{2}.
\]

The change in yaw in the robot coordinate system is

\[
d_\alpha = \frac{d^L_t - d^R_t}{l},
\]

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

\[
\begin{bmatrix}
  d_x \\
  d_y
\end{bmatrix} = R \begin{bmatrix} 1 - \cos(d_\alpha) \\
  \sin(d_\alpha) \end{bmatrix}.
\]

Note that in addition to arc motions, this also covers the special case of rotation in place, where \(d^L_t = -d^R_t\) and \(R = 0\). However, for the special case of straight motion, when \(d^L_t = d^R_t\) 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 represented by \(R_t\) at time \(t\):

\[
\begin{bmatrix}
  x'_{t+1} \\
  y'_{t+1} \\
  z'_{t+1}
\end{bmatrix} = \begin{bmatrix} x_t^r \\
  y_t^r \\
  z_t^r
\end{bmatrix} + R_t \begin{bmatrix} d_x \\
  d_y \end{bmatrix},
\]

in detail, expanding \(R_t\), we get

\[
\begin{bmatrix}
  x'_{t+1} \\
  y'_{t+1} \\
  z'_{t+1}
\end{bmatrix} = \begin{bmatrix} x_t^r + \frac{dx_s}{ds}c_\alpha s_\beta + \frac{dy_s}{ds} (c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma) \\
  \frac{dz_s}{ds}c_\beta + \frac{dy_s}{ds} (s_\alpha c_\beta s_\gamma + c_\alpha c_\gamma) \\
  \frac{dx_s}{ds} s_\beta + \frac{dy_s}{ds} c_\alpha c_\beta s_\gamma
\end{bmatrix},
\]

where \(c_\cdot\) and \(s_\cdot\) are shorthand for the cosine and sine functions.

To determine \(\alpha_{t+1}, \beta_{t+1}, \) and \(\gamma_{t+1}\), we let \(R\) be the rotation matrix corresponding to a rotation of \(d_\alpha\) around the \(z\) axis in the robot ground plane. Then the new orientation of the vehicle, expressed as a rotation matrix, is

\[
R_{t+1} = R_t R_t^	op.
\]
To extract Euler rotations from $R_{t+1}$, we use
\[ \gamma_{t+1} = \text{atan2}(r_{32}, r_{33}) \]
\[ \beta_{t+1} = \text{atan2} \left( -r_{31}, \sqrt{r_{32}^2 + r_{33}^2} \right) \]
\[ \alpha_{t+1} = \text{atan2}(r_{21}, r_{11}) , \]
where $r_{ij}$ represents the $(i, j)$-th element of $R_{t+1}$.

2) Target motion: We assume the simple linear dynamics
\[ x_{t+1} = x_t + \Delta_t \hat{x}_t \]
\[ y_{t+1} = y_t + \Delta_t \hat{y}_t \]
\[ z_{t+1} = z_t + \Delta_t \hat{z}_t \]
\[ \hat{u}_{t+1} = u_t \]
\[ \hat{h}_{t+1} = h_t \]
\[ \hat{x}_{t+1} = \hat{x}_t \]
\[ \hat{y}_{t+1} = \hat{y}_t \]
\[ \hat{z}_{t+1} = \hat{z}_t \]
for the target object’s state. In this paper we assume the object’s size is fixed and known, so in fact $\hat{u}_t = u_0$ and $\hat{h}_t = h_0$ for all $t$.

3) Linearization: Since $f(x_t, u_t)$ 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 $\hat{x}_t$. We write
\[ f(x_t, u_t) \approx f(\hat{x}_t, u_t) + J_{f_t}(x_t - \hat{x}_t) , \]
where $J_{f_t}$ is the Jacobian
\[ J_{f_t} = \left[ \frac{\partial f(x_t, u_t)}{\partial x_t} \right] \]
evaluated at $\hat{x}_t$. We omit the delimited Jacobian calculations.

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:
\[ z_t = \left[ u_t, v_t, w_{t \text{img}}^\text{cam}, h_{t \text{img}}^\text{cam} \right]^T , \]
where $(u_t, v_t)$ is the center and $w_{t \text{img}}^\text{cam}$ and $h_{t \text{img}}^\text{cam}$ are the width and height of the bounding box in the image. We model the sensor with a function $h(\cdot)$ mapping the system state $x_t$ to the corresponding sensor measurement
\[ z_t = h(x_t) + \zeta_t , \]
with $\zeta \sim \mathcal{N}(0, \Sigma_t)$.

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
\[ u_t = (f x_{t \text{cam}}^\text{cam} + c_x) / z_t \]
\[ v_t = (f y_{t \text{cam}}^\text{cam} + c_y) / z_t \]
\[ w_{t \text{img}}^\text{cam} = f w_0 / z_t \]
\[ h_{t \text{img}}^\text{cam} = f h_0 / z_t . \]
Here $x_{t \text{cam}}^\text{cam} = [x_{t \text{cam}}^\text{cam}, y_{t \text{cam}}^\text{cam}, 1]^T$ is the homogeneous representation of the rigid transformation of the target’s center into the camera coordinate system:
\[ x_t^\text{cam} = T_{t \text{W/C}}^W [ x_t \]
\[ y_t \]
\[ z_t \]
\[ 1 ] , \]
where the transformation $T_{t \text{W/C}}^W$ is defined as
\[ T_{t \text{W/C}}^W = T_{R/C \text{W/R}}^R . \]
$T_{t \text{W/R}}$ is the rigid transformation from the world coordinate system to the robot coordinate at time $t$, and $T_{R/C \text{W/R}}$ 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
\[ R_t = \begin{bmatrix}
    c_{\alpha_t} c_{\beta_t} & c_{\alpha_t} s_{\beta_t} & -s_{\alpha_t} \\
    s_{\alpha_t} c_{\beta_t} & s_{\alpha_t} s_{\beta_t} & c_{\alpha_t} \\
    s_{\beta_t} & -c_{\beta_t} & 0
\end{bmatrix} , \]
we can write
\[ T_{t \text{W/R}} = \begin{bmatrix} R_t^T & -R_t^T x_t^r \\ 0^T & 1 \end{bmatrix} , \]
where $x_t^r = (x_t^r, y_t^r, z_t^r).
As with the transition model, to linearize $h(x_t)$ around an arbitrary point $\hat{x}_t$, we require the Jacobian
\[ J_{h_t} = \left[ \frac{\partial h(x_t)}{\partial x_t} \right] \]
evaluated at an arbitrary point $\hat{x}_t$.

D. Initialization
To initialize the system, we need an a-priori state vector $x_0$. 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 measurement $z_0$ and write
\[ x_0 = [x_0, y_0, z_0, u_0, h_0, 0, 0, 0, 0, 0, 0, 0, 0] \]
\[ = h^\text{img} (z_0) . \]
Since $u_0$ 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 $z_0$. We first obtain an initial position
\[ [x_{0 \text{cam}}^\text{cam}, y_{0 \text{cam}}^\text{cam}, z_{0 \text{cam}}^\text{cam}]^T \]
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

\[
\begin{bmatrix}
x_0 \\
y_0 \\
z_0 \\
1
\end{bmatrix} = \left( \begin{bmatrix} R & C \end{bmatrix} \right)^{-1} \begin{bmatrix}
x_{0}^{\text{cam}} \\
y_{0}^{\text{cam}} \\
z_{0}^{\text{cam}} \\
1
\end{bmatrix}.
\]

Inspecting the system in Eq. 3, we can find \( x_{t}^{\text{cam}} \) and \( y_{t}^{\text{cam}} \) given \( u_t \) and \( v_t \) if \( z_{t}^{\text{cam}} \) is known. We can obtain \( z_{t}^{\text{cam}} \) from \( w_{t}^{\text{img}} \) or \( h_{t}^{\text{img}} \). We use \( z_{t}^{\text{img}} = h_{t}^{\text{img}} / z_{t}^{\text{img}} \) 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 \( S_t \). 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:

\[
S_t = \lambda^2 \begin{bmatrix}
(u_t^{\text{img}})^2 & 0 & 0 & 0 \\
0 & (h_t^{\text{img}})^2 & 0 & 0 \\
0 & 0 & (u_t^{\text{img}})^2 & 0 \\
0 & 0 & 0 & (h_t^{\text{img}})^2
\end{bmatrix}.
\]

We use \( \lambda = 0.2 \) in our simulations. For the initial state error denoted by \( P_0 \), we propagate the measurement error for \( z_0 \) through \( H^{\text{inv}}(z_0) \) and take into account the initial uncertainty about the target’s velocity:

\[
P_0 = J_{h^{\text{inv}}} S_t J_{h^{\text{inv}}} + \text{diag}(0, 0, 0, 0, 0, \eta, \eta, 0, 0, 0, 0, 0, 0).
\]

\( \eta \) is a constant and \( J_{h^{\text{inv}}} \) is the Jacobian of \( H^{\text{inv}}(\cdot) \) evaluated at \( z_0 \).

We assume for simplicity that the state transition noise covariance \( Q_t \) is diagonal. We let 

\[
v_t = \sqrt{\dot{x}_t^2 + \dot{y}_t^2 + \dot{z}_t^2},
\]

and then we let the entries of \( Q_t \) corresponding to the target position be \( \Delta^2 (\rho_1 \dot{v}_t^2 + \rho_2) \), and the entries of \( Q_t \) corresponding to the target velocity be \( \Delta^2 (\rho_3 \dot{v}_t^2 + \rho_4) \). We let the entries of \( Q_t \) corresponding to the robot’s position be \( \Delta^2 (\rho_5 \dot{v}_t^2 + \rho_6) \), and we let the entries of \( Q_t \) corresponding to the robot’s orientation be \( \Delta^2 (\rho_7 \dot{v}_t^2 + \rho_8) \). 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 \((\eta, \rho_1, \rho_2, \cdots, \rho_8)\) that must be determined through hand tuning or calibration. In our simulation we find the optimal parameters using gradient descent.

### 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 \( z_t \) 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 \( H^{\text{inv}}(z_t) \) and fixing the elements of \( P_t \) by propagating the sensor measurement error for \( z_t \) through \( H^{\text{inv}}(z_t) \) as previously explained in Section II-E. Here is a summary of the algorithm:

1. **Input** \( z_0 \).
2. **Calculate** \( \hat{x}_0 \) and \( P_0 \).
3. **For** \( t = 1, \ldots, T \), **do**
   a. **Predict** \( \hat{x}_t = f(\hat{x}_{t-1}, u_{t-1}) \)
   b. **Calculate** \( J_{f_t} \) and \( Q_t \)
   c. **Predict** \( P_t^{-1} = J_{f_t} P_{t-1} J_{f_t}^T + Q_t \)
   d. **If** \( z_t \) is unavailable
      i. **Let** \( \hat{x}_t = \hat{x}_t^- \)
      ii. **Let** \( P_t = P_t^- \)
   e. **otherwise**
      i. **Calculate** \( J_{h_t}, S_t, \) and Kalman gain
         \[ K_t = P_t^{-1} J_{h_t}^T (J_{h_t} P_t J_{h_t}^T + S_t)^{-1} \]
      ii. **Estimate** \( \hat{x}_t = \hat{x}_t^- + K_t (z_t - h_t(\hat{x}_t^-)) \)
      iii. **Update the error estimate** \( P_t = (I - K_t J_{h_t}) P_t^- \)
   f. **If** \( \| \hat{x}_t - \hat{x}_t^- \| > \sigma \), **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 synthetic trajectories 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 curved trajectory shown in black 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 black on the right of Fig. 3. During most of the simulation, the target remains inside the robot’s camera’s field of view. But the simulation also includes a period of time in which the robot and target are travelling parallel to each other with the target outside the robot’s camera’s field of view. During these times, no sensor measurements are observed. In Experiment I, we fixed the odometry noise to a typical level, Gaussian with standard deviation equal to 9% of the distance traveled. In Experiment II, we varied the odometry noise from 0 (perfect odometry) to 108% of the distance traveled and observed our algorithm’s resulting estimation error. We compare the estimation error under our model with the estimation error obtained by assuming no sensor noise (“no correction” in the results figures) and a simplified version of model-based estimation in which the relative position of the target with respect to the robot is smoothed using a Kalman filter with first-order linear dynamics. Both experiments used the same simulated camera generating 640 \times 480 images at 20 fps with a focal length of 550 pixels (horizontal field of view 60°.) To model the sensor measurements, we first project the actual synthetic target into the image, find the bounding box and, then add synthetic noise to the bounding box parameters. The noise was Gaussian with a standard deviation for \( u_t \) and \( v_t^{\text{img}} \) equal to 20% of \( u_t^{\text{img}} \) and a standard deviation for \( v_t \).
were \( \rho_1, \cdots, \rho_8, \eta \) controlling the assumed noise level in the motion model under each experimental condition, we perform gradient decent on the mean squared error between the predicted and ground truth robot-relative target positions over the intervals in the simulation where the target is visible. This means that when we are comparing model-based estimation to sensor-based estimation or the reduced model based procedure, we are reporting the best possible results for each model.

A. Experiment I (fixed odometry noise)

In this experiment, we fixed the odometry noise to 9% and the sensor noise to 20%. The optimal parameters from the gradient decent for the full model-based correction method were \( \rho_1=0.0000, \rho_2=0.373397, \rho_3=0.0000, \rho_4=0.0000, \rho_5=0.268323, \rho_6=0.0000, \rho_7=127.752168, \rho_8=0.327067, \eta=1.498932 \), and the optimal values for the reduced model were \( \rho_1=0.912168, \rho_2=0.0000, \rho_3=0.000052, \rho_4=1.505585, \rho_5=1.505585 \). The estimated robot and target trajectories over the simulation in each condition are shown in Fig. 3. We observe that the corrected robot trajectory is less smooth than those from the methods that do not correct the robot’s path using sensor measurements. Furthermore, all of the estimated target trajectories are relatively far away from the ground truth, because of accumulated odometry error. However, the filtered target trajectories are much smoother than the raw sensor-based estimates. Furthermore, the target path filtered with the full model appears to be somewhat smoother than that of the reduced model. In any case, in target pursuit, and \( h_{t, \text{img}} \) equal to 20% of \( h_{t, \text{img}} \). To determine the nine free parameters \( \{\rho_1, \cdots, \rho_8, \eta\} \) controlling the assumed noise level 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, with the full model or the reduced model, consistently outperforms sensor-only estimation. Fig. 5 shows an overall comparison between the relative error of sensor-only estimation and the two model-based correction methods. On average, the corrected target position estimates are 64.58% better than the raw sensor-based estimates without correction, and they are 19.04% better than the estimates based on the reduced model. To test whether the difference between the full and reduced model are significant, we performed a t-test for difference in the sample means and a significant difference (\( t=-3.11, p=0.001976 \)).

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. 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 108% 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. For each noise level, we first generated synthetic simulation data, optimized the motion model, then ran 10 simulations. Fig. 6 shows the average relative target position error as a function of odometry error level.
As expected, sensor-only estimation and the reduce 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.

We note that the improvement in accuracy of the full model relative to the reduced model is modest and only significant at low odometry error rates. This is because the relative position of the target to the robot is quite stable in the relative simple scenario. We have performed additional experiments with more challenging robot and target trajectories and velocity profiles and we find that the disparity between the two methods increases as the difficulty of the scenario increases. Due to length limitations, we do not report these additional experiments here.

IV. CONCLUSION

Target pursuit is an extremely useful technology for surveillance applications. However, to our knowledge there is no existing ground-based system able to track and pursue arbitrary targets autonomously in complex outdoor scenarios. 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. The absolute target position estimates are not very accurate.

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.

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 geometry 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