Particle Filter Tracking Architecture
for use Onboard Unmanned Aerial Vehicles

A Thesis Proposal
Presented to
The Academic Faculty

by

Ben T. Ludington

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Electrical and Computer Engineering
Georgia Institute of Technology
November 2005
ABSTRACT

Because of their ability to reach unique vantage points, rotary-winged unmanned aerial vehicles are well suited for visual target tracking missions. However, these missions are very burdensome for the vehicle operator since he must interpret the incoming video frames, update the orientation of the camera, and update the position of the vehicle to ensure the target is successfully tracked. This research seeks to provide an automated system architecture, which will perform these three tasks with minimal operator input.

The majority of the research is focused on the video interpretation task. A particle filter is chosen to estimate the position of the target within each video frame. The particle filter is a recursive, sample-based state estimation tool that is capable of approximating non-Gaussian distributions governed by nonlinear models. Much of the recent particle filter research can be classified into three categories, data fusion, efficiency improvement, and color model adaptation. The data fusion works describe methods to blend information from various sensors in a structured manner. Most of the research relies upon static systems, where the fusion and other measurement parameters remain constant. However, the few exceptions show the promise of adaptive fusion systems, where the parameters change based on the tracking conditions. The efficiency improvement works describe methods to handle the particle filter’s inherently large computational load to achieve real-time performance. It is typically handled by changing the number of particles based upon properties of the particle distribution. The color model adaptation works allow the reference color model to change as the tracking conditions change. This allows more accurate color measurements to be made.

The proposed system introduces a real-time, adaptive particle filter, which fuses information from multiple sensors. The parameters of the particle filter, including the number of particles, are all controlled using properties of the particle distribution. The output of the
particle filter is used along with a conventional, covariance-method, linear predictor to update the camera orientation. The position of the vehicle is updated by blending information from the vehicle’s dynamics as well as the terrain to determine an acceptable observation point. This point is sent to the vehicle as the next waypoint.

Initial testing demonstrates the effectiveness of the real-time, adaptive particle filter. However, further work is required to further automate the adaptation scheme and to better determine the confidence in the estimate. Simulations of a simplified camera orientation controller and initial flight tests show the potential of the overall system when used with an effective waypoint generator. The proposed system is expected to contribute:

- A unified control structure that adapts both the particle filter parameters as well as the number of particles.
- An automatic initialization process for the particle filter.
- A camera orientation command generator for tracking applications.
- A model and terrain-based waypoint generator for tracking applications.
- A target tracking architecture for use onboard a rotary-wing unmanned aerial vehicle.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vii</td>
</tr>
<tr>
<td>DISCLAIMER STATEMENT</td>
<td>viii</td>
</tr>
<tr>
<td>NOMENCLATURE</td>
<td>ix</td>
</tr>
<tr>
<td><strong>I PROBLEM DESCRIPTION AND BACKGROUND</strong></td>
<td></td>
</tr>
<tr>
<td>1.1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1 Research Motivation and Proposed System Overview</td>
<td>1</td>
</tr>
<tr>
<td>1.1.2 Problem Statement</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Origin and History of the Problem</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1 State Estimation</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1.1 Kalman Filtering and Its Relatives</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1.2 Grid-based Approximation Methods</td>
<td>6</td>
</tr>
<tr>
<td>1.2.2 Particle Filtering</td>
<td>7</td>
</tr>
<tr>
<td>1.2.2.1 Data Fusion</td>
<td>8</td>
</tr>
<tr>
<td>1.2.2.2 Efficiency Improvements</td>
<td>9</td>
</tr>
<tr>
<td>1.2.2.3 Color Model Adaptation</td>
<td>11</td>
</tr>
<tr>
<td>1.2.3 Linear Prediction</td>
<td>12</td>
</tr>
<tr>
<td>1.2.4 Preliminary Research</td>
<td>13</td>
</tr>
<tr>
<td>1.2.5 Preliminary Research Results</td>
<td>18</td>
</tr>
<tr>
<td>1.2.5.1 Particle Filtering</td>
<td>18</td>
</tr>
<tr>
<td>1.2.5.2 Camera Controller</td>
<td>24</td>
</tr>
<tr>
<td>1.2.5.3 Flight Test Results</td>
<td>27</td>
</tr>
<tr>
<td><strong>II PROPOSED RESEARCH</strong></td>
<td></td>
</tr>
<tr>
<td>2.1 Particle Filter Improvements</td>
<td>29</td>
</tr>
<tr>
<td>2.2 Camera Orientation Controller</td>
<td>31</td>
</tr>
<tr>
<td>2.3 Camera Position Controller</td>
<td>31</td>
</tr>
<tr>
<td>2.4 Expected Contributions</td>
<td>32</td>
</tr>
</tbody>
</table>
LIST OF TABLES

1 Soldier Tracking Parameters. ................................................. 20
2 Truck Tracking Parameters. .................................................. 21
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Proposed system schematic.</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Automatic initialization routine.</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>Soldier video automatic initialization.</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>Truck video automatic initialization.</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>Frames 1, 19, 37, 55, 73, and 91 of the soldier output video.</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>Soldier tracking error.</td>
<td>22</td>
</tr>
<tr>
<td>7</td>
<td>Soldier tracking parameters.</td>
<td>22</td>
</tr>
<tr>
<td>8</td>
<td>Frames 1, 75, 150, 225, 300, and 375 of the truck output video.</td>
<td>23</td>
</tr>
<tr>
<td>9</td>
<td>Truck tracking error.</td>
<td>23</td>
</tr>
<tr>
<td>10</td>
<td>Geometry of the target’s ground location estimate.</td>
<td>24</td>
</tr>
<tr>
<td>11</td>
<td>Target path and FOV center ground path.</td>
<td>26</td>
</tr>
<tr>
<td>12</td>
<td>Error between the target position in the frame and the center of the frame.</td>
<td>26</td>
</tr>
<tr>
<td>13</td>
<td>Waypoint generation.</td>
<td>32</td>
</tr>
<tr>
<td>14</td>
<td>Waypoint generation with obstructions.</td>
<td>33</td>
</tr>
</tbody>
</table>
DISCLAIMER STATEMENT

Section I (background/literature survey) of this proposal was prepared without input from my research advisor or any other person. While technical writing guides may have been referred to, I did not solicit or receive assistance from any other person while preparing this portion of this document.

Signed:_____________________________ Ben T. Ludington

Approved:_________________________ Dr. George J. Vachtsevanos
NOMENCLATURE

Mathematical Symbols

\(a_p\) predictor coefficients
\(A^i_k\) area of particle \(i\) at time \(k\) in pixels\(^2\)
\(e_h\) signed difference between target’s horizontal frame position and frame center
\(e_v\) signed difference between target’s vertical frame position and frame center
\(F_k\) discrete system model matrix at time \(k\)
\(FOV_h\) horizontal angular field of view width
\(FOV_v\) vertical angular field of view width
\(H_f\) frame height in pixels
\(H_k\) discrete measurement model matrix at time \(k\)
\(h^i_k\) \(\{h_{k,j}^i, j = 1, \ldots, N_B\}\), histogram of particle \(i\) at time \(k\)
\(h_{\text{ref}}\) \(\{h_{\text{ref},j}, j = 1, \ldots, N_B\}\), reference histogram
\(I\) set of all particle indices
\(J\) set of updated particle indices
\(M^i_k\) sum of the difference pixels within particle \(i\) at time \(k\)
\(N(x, \sigma^2)\) Gaussian distribution with mean \(x\) and variance \(\sigma^2\)
\(N_S\) number of particles (samples)
\(p\) predictor order
\(Q_k\) system noise covariance matrix at time \(k\)
\(R_k\) measurement noise covariance matrix at time \(k\)
\(v_k\) measurement noise random variable at time \(k\)
\(W_f\) frame width in pixels
\(w^i_k\) weight of particle \(i\) at time \(k\)
\(w^i_{C,k}\) color weight of particle \(i\) at time \(k\)
\(w^i_{M,k}\) motion weight of particle \(i\) at time \(k\)
\(w^i_{S,k}\) smoothness weight of particle \(i\) at time \(k\)
\(w_k\) system noise random variable at time \(k\)
x  target ground position relative to the camera in the x direction
x_k  state at time k
x_k^i  particle i at time k
x_{k,1}^i  horizontal position of particle i at time k
x_{k,2}^i  vertical position of particle i at time k
x_t  target position coordinate
x_v  camera position coordinate
\hat{x}_t  predicted target ground position relative to the camera in the x direction
y_t  target position coordinate
y_v  camera position coordinate
y_k  measurement at time k
y_{1:k}  \{y_i, i = 1, \ldots, k\}, set of measurements from time 1 to time k
y_k^C  color measurement at time k
y_k^M  motion measurement at time k
y_k^S  smoothness measurement at time k
z_k^i  randomly selected high motion pixel
z_v  camera altitude
\beta_{RW}  adaptable weighting term between 0 and 1
\psi  camera pan
\psi_e  pan error
\sigma_C  adaptable term for the color likelihood model
\sigma_j  adaptable term for the jump portion of the system update model
\sigma_M  adaptable term for the motion likelihood model
\sigma_{RW}  adaptable term of the random walk portion of the system update model
\sigma_S  adaptable term for the smoothness likelihood model
\theta  camera tilt
\theta_e  tilt error

Acronyms and Abbreviations

DARPA  Defense Advanced Research Projects Agency
FOV  Field Of View
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>fps</td>
<td>frames per second</td>
</tr>
<tr>
<td>GUST</td>
<td>Georgia Tech UAV Simulation Tool</td>
</tr>
<tr>
<td>HSV</td>
<td>Hue Saturation Value</td>
</tr>
<tr>
<td>KLD</td>
<td>Kullback-Leibler Distance</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi-layer Perceptron</td>
</tr>
<tr>
<td>pdf</td>
<td>Probability Density Function</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
</tbody>
</table>
CHAPTER I

PROBLEM DESCRIPTION AND BACKGROUND

1.1 Introduction

1.1.1 Research Motivation and Proposed System Overview

Unmanned aerial vehicles (UAVs) are well suited for surveillance missions such as visual tracking since, “one unmanned sentinel could survey the same area as ten (or more) human sentries” [29]. Rotary-winged UAVs are particularly adapt at visual tracking due to their ability to rapidly change directions and hover. Also, as pointed out by the Department of Defense, the use of UAVs for tracking results in the “greater probability that the mission will be successful” [30]. For the mission to be successful, the current technology level requires the operator’s undivided attention to (1) continuously interpret the incoming image frames, (2) update the orientation of the camera, and (3) update the position of the vehicle. Not only is this burdensome for the operator, it is also unsafe since the operator may be too distracted to notice anomalies. The purpose of the proposed research is to decrease the load on the operator by implementing an architecture that will allow the three target tracking tasks to be completed autonomously.

A particle filter is proposed to complete the first task, image interpretation. In this task, the position, or state, of the target is estimated within each incoming video frame. A particle filter is chosen because it is able to estimate non-Gaussian distributions that evolve according to nonlinear models. These distributions are common in visual tracking tasks because of the background clutter. The increased performance of the particle filter comes with an inherently large computational burden. However, techniques for handling this burden are proposed. An automatic initialization scheme, adaptation scheme, and neural network-based confidence indicator are also proposed. The particle filter improvements allow it to successfully track targets in images in real-time.

Once the position of the target is estimated within each frame, it’s three dimensional
coordinate can be estimated using a flat terrain model. Then, the future target locations can be predicted. A conventional, covariance method linear predictor is proposed for this task. This choice is made because the covariance method is “typically more accurate than the autocorrelation method” [13]. While the method for finding the predictor coefficients is more complex than the autocorrelation method, it can still be done efficiently [26].

The proposed system then sends the predicted target location directly to the camera controller. It is assumed that vehicle is equipped with a controller that is capable of pointing the camera at a given ground position. The GTMax unmanned research helicopter is equipped with such a controller [15]. Pointing the camera at the future target location ensures the target will remain within the camera’s field of view (FOV). The predicted target location is also used to determine future waypoints for the vehicle. Three factors are considered for the proposed waypoint generation. First, the distance from the target is considered to ensure the target is kept close enough to the camera for the tracking task will be successful. Second, linear obstructions, such as a line of trees or row of buildings, are considered to ensure the target is not occluded for an extended time. Finally, the vehicles dynamics are considered to ensure there are no rapid attitude changes that cannot be compensated by the camera controller.

The proposed system is shown schematically in Figure 1. Frames from the camera are processed by the particle filter to estimate the position of the target. The output of the particle filter is then used to predict future target locations. The camera orientation controller and the waypoint generator close the loop by sending commands to the camera and to the vehicle.

1.1.2 Problem Statement

An automatic target tracking system is required to increase the autonomy of a UAV and decrease the operator’s workload. The system should consistently track targets using data collected from video. While tracking the target, the orientation of the camera should update to keep the target near the center of the FOV. The position of the vehicle should change to keep the vehicle close enough to the target that the tracking will be successful while
considering the location of linear obstructions as well as any required attitude changes to the vehicle. The target will be operating in a cluttered environment making the image interpretation task the most challenging component of the system.
1.2 Origin and History of the Problem

The subsystems of the proposed target tracking architecture each have been thoroughly studied in the past. Modern state estimation techniques will be discussed along with aspects of linear prediction.

1.2.1 State Estimation

When tracking targets in video, the goal is to estimate the position, or state, of the target within each video frame using information from all the previous measurements. This is given by the probability density function (pdf) over the state space,

\[ p(x_k|y_{1:k}) , \]

where \( x_k \) is the state at time \( k \), and \( y_{1:k} = \{ y_i, i = 1, \ldots, k \} \) is the set of measurements up to time \( k \). The optimal, Bayesian solution is given in two steps, prediction and update. In the prediction step, an estimate of the system dynamics is used to estimate the prior pdf. For example, if the dynamics are assumed to be governed by a first order Markov process, the prior pdf is given by the Chapman-Kolmogorov equation

\[ p(x_k|y_{1:k-1}) = \int p(x_k|x_{k-1}) p(x_{k-1}|y_{1:k-1}) \, dx_{k-1} , \]

where \( p(x_k|x_{k-1}) \) is given by the system model. In the update step, the posterior pdf is found after the measurement is taken at time \( k \). The posterior is given by Bayes’ rule

\[ p(x_k|y_{1:k}) = \frac{p(y_k|x_k) p(x_k|y_{1:k})}{p(y_k|y_{1:k-1})} , \]

where \( p(y_k|x_k) \) is given by the measurement model.

Equations 2 and 3 form the optimal, Bayesian tracking solution [3]. Without imposing significant restrictions on the system, these equations cannot be solved in a closed form. However, there are a number of techniques to estimate the solution.

1.2.1.1 Kalman Filtering and Its Relatives

When the distribution of Equation 1 is Gaussian and the system model is linear, the well-known Kalman filter forms the optimal solution. Since the Kalman filter requires Gaussian
distributions, the solutions are completely constrained once the mean and the covariance are found. In other words, when the system model can be given as

\[ x_k = F_k x_{k-1} + w_{k-1} \]  
\[ y_k = H_k x_k + v_k, \]

where \( w_{k-1} \) and \( v_k \) are zero mean, independent, white-noise, Gaussian, random variables with covariance matrices \( Q_{k-1} \) and \( R_k \), respectively, then the estimation solution is given by

\[ p(x_k|x_{k-1}) = N \left( m_{k-1|k-1}, P_{k-1|k-1} \right) \]  
\[ p(x_k|y_{1:k-1}) = N \left( m_{k|k-1}, P_{k|k-1} \right) \]  
\[ p(x_k|y_k) = N \left( m_{k|k}, P_{k|k} \right), \]

where the mean and covariance at each time is given using the predict and update process

\[ m_{k|k-1} = F_k m_{k-1|k-1} \]  
\[ P_{k|k-1} = Q_{k-1} + F_k P_{k-1|k-1} F_k^T \]  
\[ m_{k|k} = m_{k|k-1} + K_k \left( y_k - H_k m_{k|k-1} \right) \]  
\[ P_{k|k} = P_{k|k-1} - K_k H_k P_{k|k-1}, \]

where \( N \left( m, P \right) \) is a Gaussian distribution with mean \( m \) and covariance \( P \) and

\[ S_k = H_k P_{k|k-1} H_k^T + R_k \]  
\[ K_k = P_{k|k-1} H_k^{T} S_k^{-1}. \]

The derivation of these equations is given by Kalman in [18] and is summarized in [13]. Equations 9 – 14 are easily implemented in real-time, and are commonly used in many control applications. However, the linear, Gaussian assumptions are overly restrictive for use in many circumstances. Two methods have become popular for cases when the system model is nonlinear.

The first method is the extended Kalman filter, which was introduced in [1]. In this method, the system model is approximated by its first order Taylor series expansion. While
this technique can be effective when the nonlinearities are relatively small, large nonlinearities or large model errors can result in prohibitively large estimation errors or even divergence of the filter. The other method is the unscented Kalman filter, which was introduced in [16] and [17]. This technique uses a set of *sigma points* to approximate the mean and covariance of the distribution in Equation 1. The nonlinear system model is then applied to the points at each time step, and measurements are used as in the traditional Kalman filter to update the distribution. The resulting points form an estimate of the new posterior distribution. While the unscented Kalman filter typically outperforms the extended Kalman filter, it is still only able to approximate the mean and the covariance of Gaussian distributions. Therefore, this method will have large tracking errors when used for visual tracking since the distributions are typically non-Gaussian.

In cases when the distribution of Equation 1 is non-Gaussian, the Gaussian sum filter can be used [1]. In this approach, the distribution is decomposed into a sum of Gaussian distributions. Then, each Gaussian distribution is passed through a Kalman filter (or extended or unscented Kalman filter). This results in a bank of Kalman filters. The output of each filter is then weighted and summed. While this filter has been successful in cases when the noise is relatively small, it tends to require numerous reinitializations when the noise increases. These repeated reinitializations may result in filter divergence.

### 1.2.1.2 Grid-based Approximation Methods

As implied by the name, grid-based approximations decompose the state space into a finite number of discrete states. Then, the probability within each state is evaluated following the two-step process of predict and update using the equations in [3]. These methods are typically used when the system is governed by a hidden Markov model as described in [34]. However, as the number of discrete states increases, the required computational burden becomes overwhelming. The burden can be reduced by truncating the state space, but this results in a higher estimation error.
### 1.2.2 Particle Filtering

The particle filter uses a set of samples to approximate the distribution of Equation (1):

$$p(x_k|y_{1:k}) \approx \sum_{i=1}^{N_s} w^i_k \delta(x_k - x^i_k),$$

where $x^i_k$ is a particle, or point in the state space, and $w^i_k$ is its corresponding weight. The weights are scaled so their sum is unity. As in the other techniques, the particles are updated in two steps, prediction and update. In the prediction step, the particles are moved in the state space using the system model and sampling from $p(x^i_k|x^i_{k-1})$. Then, after measurements are taken, the weights are updated using the measurement likelihood model

$$w^i_k \propto p(y_k|x^i_k).$$

The particles are resampled at each time step according to their weights. Therefore, the previous weights, $w^i_{k-1}$, may be neglected. As the number of particles increases, the sample-based approximation converges to the true pdf [8, 9]. By using a set of non-deterministic points, the particle filter can accurately approximate non-Gaussian distributions that evolve according to nonlinear models, and the particle filter typically yields less tracking error than Kalman filter based techniques [25].

The particle filter was first used as the bootstrap filter by Gordon, Salmond, and Smith, where it was compared to the extended Kalman filter in a four-dimensional bearings only tracking problem [12]. The particle filter successfully estimated the state of the non-linear system, while the extended Kalman filter diverged. The particle filter became more popular after Isard and Blake used it to track curves in highly cluttered images in what they called the CONDENSATION algorithm [14]. Objects with distinct edges were consistently tracked in images. For example, a particle filter was used to track the edge of a single leaf in a video taken of a bush blowing in the wind. This proved the particle filter’s unique ability to process multimodal (non-Gaussian) distributions. Recently, a particle filter was used to track objects in video using only color cues [31]. The particle set was a group of rectangles that were placed within the image. A histogram was populated for each particle, and these histograms were compared to a reference histogram to generate measurements. While the
particle filter performed well, the filter was initialized manually and the filter was unable to process the frames in real-time with modest computational resources.

Much of the other target tracking particle filter work has been in three areas, data fusion, efficiency improvements, and color model adaptation. The data fusion works allow the particle filter to process information from various sensors in an efficient and effective manner. The efficiency improvement works allow the particle filter to perform in real time even with the inherently large computational burden. The color model adaptation works allow the measurements to adapt to the changing tracking conditions.

1.2.2.1 Data Fusion

In many state estimation problems, information is available from a variety of sensors. For example, in visual tracking shape, color, size, and motion data are typically available. However, some cues may be more useful than others. The data fusion works enable the particle filter to focus on the useful cues while ignoring the other cues.

In the simplest approach, Perez, Vermaak, and Blake use independent cues to track objects in video [32]. They show the benefit of using motion cues in addition to color cues. When the filter is attempting to acquire (or reacquire) a target, the motion measurements are much more useful than when the filter is locked on a target. When the filter is locked on, the tracking can be done using only color measurements. They also use audio cues to further aid the tracking of a speaker in a room. Since all of the cues are independent, the combined weight of each particle is obtained by simply multiplying the weights generated from each of the individual cues. While the authors point out that the utility of each cue changes as the tracking conditions, they do not attempt to change their measurement model.

Loy, et al. use multiple cues to track faces in images [24]. In their approach, a cue processor generates separate distributions for each cue as well as the distribution from all the cues. They assume that the distribution created from all the cues is the true distribution and the distributions from the most useful cues will be the closest to the true distribution. Therefore, more computational resources are assigned to the cues that are more useful
so that these cues are updated at the frame rate. The other cues are processed in the background and may not be updated at the frame rate. The same approach is used by Apostoloff and Zelinsky to visually track lane markers on roadways [2]. Triesh and von der Malsburg use a similar approach to visually track faces [35]. However, instead of using the utility of each cue to manage the computation resources, they use the utility to weight the contribution of each cue to the total measurement.

Collins and Liu use a discriminative technique to measure the utility of their cues when tracking vehicles in images [6]. The technique relies upon the simple idea that when using a useful cue, the measurements near the target will be much different than the measurements in the surrounding area. Therefore, they take measurements around each particle in addition to the measurements within each particle. The cues that result in the largest difference between the two are deemed to be the most useful. While this technique automatically selects the most useful cues, it requires additional computations that might be too burdensome to allow real-time tracking in video.

Instead of fusing the cues only in the measurement step, Gatica-Perez et al. rely on the particle filter structure to fuse the information [11]. They use audio and edge-based measurements to track speakers in a meeting room. The audio cues are used to update the position of the particles (in the prediction stage) and edge measurements are used to update the weights. This process allows more particles to be placed in regions where the target is more likely to be.

1.2.2.2 Efficiency Improvements

As the number of particles increases, the particle filter error decreases. However, increasing the number of particles increases the computational burden. If the burden becomes too large, the filter will not achieve real-time performance. When tracking objects in images, the computational burden is typically large since histograms are required for each particle to generate color data. Therefore, managing this burden is especially important. The most obvious way to manage the burden is by adjusting the number of particles based on the filter’s performance.
Changing the number of particles is first suggested in [4]. The error between the true and particle-based distributions is shown to be dependent on the number of particles. Boers argues that it is not necessary to keep the number of particles constant. This work set the stage allowing the number of particles to change during filtering.

In the approach developed by Bolic, Hong, and Djuric, the number of particles is controlled by a heuristic rule set [5]. The size of the particle set is increased if the number of particles with weights below a predefined threshold is too large or if the difference between the predicted and actual observations is too large. The rules for decreasing the number of particles is analogous. These simple rules allow the adapting particle filter to track a target just as well as a fixed particle filter but with a lower computational burden.

In a different approach, Karlsson and Gustafsson develop a control structure for the number of particles [19]. The control structure consists of two particle filters with a different number of particles. If the statistical properties of the output of the two filters are similar, the number of particles is decreased. Otherwise, the number of particles is increased. This approach was successfully used in a data association problem.

Kullback-Leibler Distance (KLD) techniques are developed by Fox to bound the error, or KLD, between the true posterior and the particle-based approximation [10]. The bound on this error is shown to be dependent on the number of particles. Since the true posterior is not available, the predicted distribution is compared to the particle-based posterior approximation. The number of particles is increased when the KLD between the prior and the posterior is large, and decreased when the two distribution are similar. While this technique is more structured than the two previous techniques, it requires a substantial overhead. The author reports that the approach may not improve real-time performance.

In a related approach, Kwok, Fox, and Meila use KLD-sampling along with a windowing technique [20, 21]. The windowing approach distributes the particles in time. For instance, if the computational resources allow 15 particles to be updated when each measurement is taken, and 60 particles are required for the error to be reduced to an acceptable level then the window will be four measurements wide. However, the entire set of 60 particles is used to form the distribution approximations. The windowing technique allows the KLD-based
particle filter to function in real-time when used in a robot localization problem. Liu, Liu, and Lu used a similar windowing technique to track faces in images [23]. However, in their case, the number of particles was kept constant and the window width was always two frames wide. The particles were assigned to the individual frames based on weights. The particles with high weights were placed in one frame, while the particles with low weights were sent to the other frame. This resulted in real-time performance when used with a simplified color model.

1.2.2.3 Color Model Adaptation

Recently, color has emerged as a powerful tracking tool when used within a particle filter [22, 23, 27, 28, 31, 32]. Color measurements are typically taken by comparing a particle’s histogram to a reference histogram. The reference histogram is either part of the a priori knowledge or it is generated manually in the first frame. Since lighting and background conditions can change in addition to the orientation of the target, the reference histogram should change as well.

Vermaak et al. develop a scheme to update the reference histogram using only the particles that are near the target [36]. The particles are checked to be sure the probability of a target being present is above a threshold and that this target is in motion. This check avoids over eager adaptation, which could occur when particles are measuring clutter. Nummiaro, Koller-Meier, and Van Gool opt for a simpler technique to change the reference histogram [27, 28]. When a particle’s weight rises above a predetermined threshold, a new reference histogram is constructed using a convex combination of the previous reference histogram and the particle’s histogram. The size of the contribution of the particle’s histogram is kept constant throughout the tracking process. Li and Zheng extend this work to allow the size of the contribution to change [22]. In this approach, the state space is extended to include a dimension for the size of the contribution. This allows different contribution sizes to be evaluated by the particle filter. The contribution size that results in the best tracking is used at each iteration.
1.2.3 Linear Prediction

To account for inherent processing and communication delays, the camera will be com-
mmanded to point at future target positions once the target position is estimated using a
particle filter. Since the UAVs in the scope of this work operate at low altitudes (< 500 ft.)
and have relatively wide FOVs (> 40 degrees horizontal) the speed of the prediction be-
comes more important than the accuracy. Therefore, a linear predictor is a good candidate
for this task.

The goal of linear prediction is to estimate future values of a signal using a linear
combination of past signal values. The problem is to find the predictor coefficients, \( a_k \) so
that the estimate

\[
\hat{x}_t = - \sum_{k=1}^{p} a_k x_{t-k}
\]

is as close as possible, in a least squares sense to the future value, \( x_t \). The solution to the
equation is given by the normal equations \[13\]

\[
\sum_{k=1}^{p} a_k \sum_t x_{t-k} x_{t-i} = - \sum_t x_t x_{t-i}, \quad 1 \leq i \leq p.
\]

This equation can be written in matrix form as

\[
\begin{bmatrix}
  r_0 & r_1 & r_2 & \cdots & r_{p-1} \\
  r_1 & r_0 & r_1 & \cdots & r_{p-2} \\
  r_2 & r_1 & r_0 & \cdots & r_{p-3} \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  r_{p-1} & r_{p-2} & r_{p-3} & \cdots & r_0
\end{bmatrix} \begin{bmatrix}
  a_1 \\
  a_2 \\
  a_3 \\
  \vdots \\
  a_p
\end{bmatrix} = \begin{bmatrix}
  r_1 \\
  r_2 \\
  r_3 \\
  \vdots \\
  r_p
\end{bmatrix},
\]

where

\[
r_i = \sum_t x_t x_{t-i}
\]

The range of summation in Equations \[18\] and \[20\] depends on the modeling method used \[26\].
In the correlation method, the range of summation is \(-\infty \leq t \leq \infty\). However, since the
value of \( x \) is not known for all time, a rectangular window that begins at time \( S \) and ends at
time \( T \) is typically used and unknown values are set equal to zero. This causes the array of
equation \[19\] to be Toeplitz. Therefore, the normal equations can be solved efficiently using
the Levinson recursion \[13\].
However, setting the unknown values to zero increases the prediction error. In the covariance method, only known values are used for the prediction. The range of summation in Equation 18 is $S + p \leq t \leq T$, and Equations 19 and 20 become

$$
\begin{bmatrix}
  r_{1,1} & r_{1,2} & r_{1,3} & \cdots & r_{1,p} \\
  r_{2,1} & r_{2,2} & r_{2,3} & \cdots & r_{2,p} \\
  r_{3,1} & r_{3,2} & r_{3,3} & \cdots & r_{3,p} \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  r_{p,1} & r_{p,2} & r_{p,3} & \cdots & r_{p,p}
\end{bmatrix}
\begin{bmatrix}
  a_1 \\
  a_2 \\
  a_3 \\
  \vdots \\
  a_p
\end{bmatrix}
= -
\begin{bmatrix}
  r_{1,0} \\
  r_{2,0} \\
  r_{3,0} \\
  \vdots \\
  r_{p,0}
\end{bmatrix},
$$

(21)

The matrix of Equation 21 is no longer Toeplitz. However, it is near Toeplitz, and it can be solved efficiently using the method described in [26].

1.2.4 Preliminary Research

A particle filter was used to track two types of targets in video using measurements from multiple sources. The particle (state) space is four dimensional and described a rectangle placed in the image. Therefore, each particle is completely described by the coordinates of the top left corner of the rectangle as well as the width and height. The output of the particle filter is the expected value of the sample-based approximation. This is found by taking a weighted average of the particle set

$$
E(x_k|y_{1:k}) \approx \frac{N_x}{\sum_{i=1}^{N_x} w_i^k x_i^k}
$$

(23)

The particle filter implements several novel features. First, it initializes automatically. Then the system model and measurement model are updated online using a heuristic rule set based on the estimated performance of the filter. The number of particles is also changed online using the same rule set, which allows the filter to operate in real-time.

In the previous work described above, the filters are initialized in one of three ways. In the most prevalent technique, the particles are placed manually in the first frame. When this is not acceptable, the particles are either placed using color data or uniformly.
cases where color is used, regions are identified in the first frame that are near the target’s color. However, the initialization only relies upon one color. For example, every pixel is measured to see how red it is. Then particles are assigned to regions that are red. The other case, placing the particles uniformly, is not efficient and is not an acceptable solution when the images are highly cluttered.

In the preliminary work, an automatic initialization routine is used. The routine relies upon color, motion, and size data to place the particles in the initial frame. The routine is shown schematically in Figure 2. First, two time adjacent images are sent through brickwall chromatic filters to form a binary image. The passbands of the filters are determined using a priori data. After this step, the only pixels that remain are similar in color to the target. The filtered images are then subtracted pixel-wise and their absolute value is taken. This yields motion data. If the target is expected to be large (greater than 10% of the image size) a coarse image is made. The value of each pixel in the coarse image is the sum of the corresponding 10 pixel by 10 pixel area in the difference image. The pixels are then sent to a comparitor to form a final binary image. The regions with HIGH pixels are grouped to form candidate regions, and particles are assigned to the candidate regions based on size.

![Figure 2. Automatic initialization routine.](image)

After the particles are initially placed in the state space, the recursive portion of the particle filter begins. At each step, the particles are first resampled according to their weights. Resampling at each step avoids the problem of sample degeneracy as described in [3] and [9], and has been successful in other visual trackers [31, 32]. Once the new set of particles is selected, the particles are propagated one time step forward using the system
As in [32], the system model assumes that the target moves smoothly from one frame to the next. Therefore, the model is dominated by a Gaussian, random walk distribution. However, as demonstrated in [11], the use of measurements in the prediction stage can be advantageous. Therefore, another Gaussian distribution is included in the model. This distribution is centered on a randomly selected pixel where motion has been detected, denoted as \( z^i_k \). The motion detection is done as in the automatic initialization routine. Therefore, the complete system model is given as

\[
p(x^i_k | x^i_{k-1}) = \beta_{RW} N(x^i_{k-1}, \sigma^2_{RW}) + (1 - \beta_{RW}) N(z^i_k, \sigma^2_j),
\]

where \( N(x, \sigma^2) \) is a Gaussian distribution with mean \( x \) and variance \( \sigma^2 \), \( \beta_{RW} \) is a scaling term that controls how large the contribution of each portion of the model is, and \( \sigma_{RW} \) and \( \sigma_j \) control how spread each Gaussian distribution is. The values of \( \beta_{RW}, \sigma_{RW}, \) and \( \sigma_j \) are discussed later when the adaptation scheme is described.

Once the particles are moved, measurements are taken. First, color measurements are taken. A histogram is created for each particle. A HSV color model is used, and the histogram is 10 hue bins by 10 saturation bins wide with an additional 10 bins for value. The value bins are scaled down by a factor of 10 so hue and saturation dominate the histogram. The HSV model was used successfully in [31] and is more robust to changes in lighting that other color models. Each measurement is compared to a reference histogram using a distance measure related to the Bhattacharyya similarity coefficient. As pointed out in [7], this distance is a metric. The distance measure is

\[
D_C(h^i_k, h_{ref}) = \left(1 - \frac{1}{\sqrt{\sum_{j=1}^{N_B} h^i_{k,j} h_{ref,j}}} \right)^{\frac{1}{2}},
\]

where \( h^i_k = \{h^i_{k,j}, j = 1, \ldots, N_B\} \) is the histogram for particle \( x^i_k \) and \( h_{ref} = \{h_{ref,j}, j = 1, \ldots, N_B\} \) is the reference histogram. For the preliminary work, the reference histogram is generated by manually selecting regions in 10 randomly selected frames. The histograms for these ten regions are averaged and used as the reference. The color likelihood is found from this
distance,
\[ p(y^C_k|x^i_k) \propto \exp \left( -\frac{D_C(h^i_k, h_{ref})}{\sigma_C^2} \right) , \tag{26} \]
where \( \sigma_C^2 \) controls the slope of the exponential curve. An exponential model is used because it has been successful in previous work [11, 22, 27, 28, 31, 32], and because the slope can be easily changed by adapting \( \sigma_C^2 \).

Motion measurements are also collected for each particle. The value (in the HSV sense) of each pixel within the particle is subtracted from the corresponding pixel value in the previous image, and the absolute value is taken. The resulting differences are summed over the particle to yield a motion measurement. This measurement is then turned into a distance,
\[ D_M(x^i_k) = \left( 1 - \frac{M^i_k}{A^i_k} \right)^\frac{1}{2} , \tag{27} \]
where \( M^i_k \) is the sum of the difference pixels in particle \( x^i_k \) and \( A^i_k \) is the area of particle \( x^i_k \).

This distance is also a metric as shown in Appendix A. As with the color measurements, the distance is used to form the motion likelihood,
\[ p(y^M_k|x^i_k) \propto \exp \left( -\frac{D_M(x^i_k)}{\sigma_M^2} \right) , \tag{28} \]
where \( \sigma_M^2 \) is analogous to \( \sigma_C^2 \).

As pointed out in [3], when the particles are resampled at each step, the filter can become too sensitive to outliers. To combat this and to smooth the output, another term is included in the measurement model. This term rewards the particles that remain close to the previous filter output. As with the color and motion measurements, first a distance measure is created,
\[ D_S(x^i_k) = \left[ (x^i_{k,1} - x_{k-1,1})^2 + (x^i_{k,2} - x_{k-1,2})^2 \right]^\frac{1}{2} , \tag{29} \]
where \( x^i_{k,1} \) and \( x^i_{k,2} \) are the horizontal and vertical coordinates of particle \( x^i_k \), respectively and \( x_{k-1,1} \) and \( x_{k-1,2} \) are the horizontal and vertical coordinates of the filter output at time \( k-1 \), respectively. This a Euclidean distance measure, and is clearly a metric. The distance is used to form the smoothing likelihood,
\[ p(y^S_k|x^i_k) \propto \exp \left( -\frac{D_S(x^i_k)}{\sigma_S^2} \right) \]

where $\sigma^2_S$ is analogous to $\sigma^2_C$ and $\sigma^2_M$.

As in [32], all of the measurements are assumed to be independent. Therefore, the total measurement likelihood is the product

$$p \left( y_k^i | x_k^i \right) = p \left( y^C_k | x_k^i \right) p \left( y^M_k | x_k^i \right) p \left( y^S_k | x_k^i \right). \tag{31}$$

The particle weights are proportional to each of measurement likelihoods.

$$w^{i,C}_k \propto p \left( y^C_k | x_k^i \right), \quad w^{i,M}_k \propto p \left( y^M_k | x_k^i \right), \quad w^{i,S}_k \propto p \left( y^S_k | x_k^i \right), \tag{32}$$

and the total weight is

$$w^i_k \propto w^{i,C}_k w^{i,M}_k w^{i,S}_k \propto p \left( y_k | x_k^i \right). \tag{33}$$

As discussed earlier, the performance of the particle filter with respect to tracking error and computational efficiency can be improved by adapting the filter’s parameters to the changing tracking conditions. The adaptation approach is similar to [5], where a heuristic rule set is formulated. The first step in the adaptation is to roughly measure the performance of the filter. This is done by examining properties of the particle distribution. The maximum weight generated by the color measurements is examined. If it is above a predetermined threshold, the filter is assumed to be tracking the target well. The spread of the particles is also examined. If the ten highest weighted particles are all within a square that is smaller than a predetermined threshold, the filter is assumed to be tracking the target well.

When the filter is performing well, changes are made to the filter that allow it to function more efficiently. When the filter is not performing well, efficiency is traded off to reduce the tracking error. The changes are made to the system model, the measurement model, and the number of particles.

When the filter is tracking well, the system model is changed to reduce the particle movement from one step to the next. This is done by increasing $\beta_{RW}$ while decreasing $\sigma_{RW}$ and $\sigma_j$. Also, the particles that remain close to the previous filter output are rewarded more by decreasing $\sigma_S$.

As [32] demonstrated, motion cues are much more useful when the filter is acquiring or re-acquiring the target. However, once the filter has locked onto the target, the tracking
can be done almost entirely based on color. Therefore, when the filter is tracking well, $\sigma_C$ is decreased while $\sigma_M$ is increased. This results in the filter responding more to smaller color changes.

To improve the efficiency, the number of particles is decreased when the filter is tracking well. While this results in a higher bound on the error, it typically does not increase the actual error when implemented experimentally. The number of particles is gradually decreased through the use of a convex combination of the current number of particles and a predetermined number of particles that is based on the performance level. Also, when the filter is tracking well, it becomes unnecessary to update the entire particle set at each time step. As argued in [20, 21, 23] the non-updated particles are still useful when formulating the sample based approximation. However, they are penalized

$$w_k^n \propto w_{k-1}^n, \; n \notin J \subset \{1, 2, \ldots, N_S\},$$

where $J$ is the set of updated particles, and $\ell_k^n = 0.4$. The size of $J$, like the other filter parameters, is determined by the heuristic rule set. A sample rule would take the following form:

If the maximum color weight is above threshold $A$ and the spread of the top ten particles is less than $B$ then set $\beta_{RW}$, $\sigma_{RW}$, $\sigma_j$, $\sigma_S$, $\sigma_C$, and $\sigma_M$ to certain predetermined values. Also, change the number of particles, and change the size of $J$.

Once the particle filter determines the position of the target in the frame, trigonometric relations can be used to estimate the position of the target in three-dimensional space if the terrain is assumed to be flat, or a unit vector can be projected onto the ground. The coordinates of the target are sent to the linear predictor, which generates the commands for the camera orientation and provides input to the waypoint generator.

### 1.2.5 Preliminary Research Results

#### 1.2.5.1 Particle Filtering

A particle filter was written in C++ to track a soldier as he maneuvered in an urban environment and to track a truck from a UAV. The video of the soldier was collected using
a hand held camera by an operator located atop a nearby building. The video of the truck was collected onboard the GTMax unmanned helicopter. In both cases, the videos were converted into a series of frames at 10 frames per second (fps).

When displaying the results, the particles are shown in frames as red and blue rectangles. The ten highest weighted particles are shown in blue, the remaining in red. The brightness of the red corresponds to the weight, with the higher weighed particles shown with brighter red. The output, or weighted average, of the particle filter is shown as white cross hairs.

The output of the automatic initialization routine for the soldier video is shown in Figure 3. The chromatic filter passband is centered at \((H, S, V) = (0.475, 0.150, 0.450)\), where the HSV values range from zero to one. The passband is 0.25 wide in hue, 0.1 wide in saturation and 0.3 in value. The passband was determined experimentally. Sixty particles are placed in the first frame, and 53 of them are placed near the soldier as he leaned against the car. The remaining seven particles are placed near a cluttered region around the mailbox in the left side of the image. These particles will either move toward the target or be replaced by other, higher weighted particles in the subsequent frames.

![Figure 3. Soldier video automatic initialization.](image)

The output of the automatic initialization routine for the truck video is shown in Figure 4. The chromatic filter passband is centered at \((H, S, V) = (0.160, 0.125, 0.260)\), and is 0.28 wide in hue, 0.25 wide in saturation, and 0.02 wide in value. Here, 75 particles are placed in the first frame. The increase in the number of particles is due to the smaller target size and increased clutter. In this case, all 75 particles are placed near the target.

The range of parameter values for the soldier tracking video is shown in Table 1 and
the range of parameter values for the truck tracking video is shown in Table 2. These values are determined experimentally. The differences between the two are due to the truck appearing much smaller in the video and the increase in clutter in the truck video. Also, the camera was subjected to more motion in the truck case since the camera was mounted on a helicopter. Therefore, the contribution of the motion measurements is decreased. (In both cases, the four dimensional state is \([x, y, width, height]^T\), where \(x\) and \(y\) are the coordinates of the top left corner.)

```
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\beta_{RW})</td>
<td>0.5 – 0.85</td>
</tr>
<tr>
<td>(\sigma_{RW})</td>
<td>([4, 4, 1.5, 1.5]^T - [8, 8, 1.5, 1.5]^T)</td>
</tr>
<tr>
<td>(\sigma_j^2)</td>
<td>([4, 4, 1.5, 1.5]^T - [8, 8, 1.5, 1.5]^T)</td>
</tr>
<tr>
<td>(\sigma_C^2)</td>
<td>0.005 – 0.01</td>
</tr>
<tr>
<td>(\sigma_M^2)</td>
<td>4</td>
</tr>
<tr>
<td>(\sigma_S^2)</td>
<td>10 – 60</td>
</tr>
<tr>
<td>(N_S)</td>
<td>30 – 80</td>
</tr>
<tr>
<td>Size of (J)</td>
<td>80% – 95% of (N_S)</td>
</tr>
</tbody>
</table>
```

Six output frames from the soldier video are shown in Figure 5. As can be seen in this sample of frames, both the particles and the particle filter output track the soldier as he maneuvers in the urban environment. The only time the particle filter looses the target is when he is outside of the camera’s FOV, which occurs near frame 45. When he reemerges, the particles quickly move toward him near frame 60. The plot of the tracking error is shown in Figure 6. The tracking error is the difference between a set of manually...
Table 2. Truck Tracking Parameters.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta_{RW}$</td>
<td>0.7 - 0.85</td>
</tr>
<tr>
<td>$\sigma_{RW}$</td>
<td>$[8, 8, 0.1, 0.1]^T$</td>
</tr>
<tr>
<td>$\sigma_j$</td>
<td>$[8, 8, 0.1, 0.1]^T$</td>
</tr>
<tr>
<td>$\sigma_C$</td>
<td>0.002 - 0.01</td>
</tr>
<tr>
<td>$\sigma_M^2$</td>
<td>8</td>
</tr>
<tr>
<td>$\sigma_S^2$</td>
<td>4 - 60</td>
</tr>
<tr>
<td>$N_S$</td>
<td>50 - 95</td>
</tr>
<tr>
<td>Size of $J$</td>
<td>97% of $N_S$</td>
</tr>
</tbody>
</table>

determined target positions and the filter output.

Figure 5. Frames 1, 19, 37, 55, 73, and 91 of the soldier output video.

For the soldier tracking, the average processing time per frame is 60 milliseconds and the longest frame took 78 milliseconds, which is well under the 100 milliseconds required for real-time operation. The processing time was measured on a computer with an Intel Pentium M 730 processor, 1 GB of RAM, and a ATI Mobility Radeon X300 128 MB video card. The real-time performance was possible because of the filter’s ability to adapt to changes in tracking conditions.

The set of filter parameters is shown in Figure 7. When the filter is tracking the soldier well, the number of particles is decreased, the particle movement is decreased, and the importance of color measurements is increased. However, when the target begins to leave the FOV near frame 40, the particle filter determines that the tracking conditions have
changed. Therefore, the number of particles is increased, the particle movement is increased, and the importance of motion measurements is increased. Once the filter reacquires the target near frame 60, the filter transitions back to a more efficient state.

![Figure 6. Soldier tracking error.](image)

Six output frames for the truck video are shown in Figure 8. Again, the particle filter successfully tracked the target. In this case the tracking was successful even with a high level of clutter. The tracking error is shown in Figure 9. For the truck tracking, the average processing time per frame was 50 milliseconds and the longest frame took 94 milliseconds. Again, the filter performed in real-time.

![Figure 7. Soldier tracking parameters.](image)
Figure 8. Frames 1, 75, 150, 225, 300, and 375 of the truck output video.

Figure 9. Truck tracking error.
1.2.5.2 Camera Controller

A camera controller simulation was written in Matlab to test the linear prediction-based control architecture. The simulation consists of four parts. The first part converts the simulated target ground position to a position in a video frame. The second part estimates the target’s ground position using the video frame position. The third part predicts future target locations. The fourth part models the camera motion as a double integrator in both pan and tilt.

The target position in the video frame is determined by measuring the distance the target is from the center of the FOV in both horizontal and vertical directions. The horizontal and vertical directions are oriented with the FOV as it is projected on the ground. These distances are then converted to angles using trigonometric relations, and the target is displayed in the video frame using these angles.

In the second portion of the simulation, the above process is reversed. The ground position of the target is estimated using the frame position of the target. The geometry of the problem is shown in Figure 10. The camera is located at \((x_v, y_v, z_v)\) and goal is to find the target’s coordinates, \((x_t, y_t, 0)\).

![Figure 10. Geometry of the target’s ground location estimate. The rectangle in the upper right shows the corresponding video frame.](image)

The camera’s orientation is described by its pan and tilt, \(\psi\) and \(\theta\), respectively. The roll
is only used to compensate for the vehicle’s roll so the top of the frame is always close to parallel with the horizon. The error in pan and tilt are approximated using the horizontal and vertical distances between the target’s frame position and the center of the frame

$$
\psi_e \approx e_h \cdot \frac{FOV_h}{W_F}, \quad \theta_e \approx e_v \cdot \frac{FOV_v}{H_F},
$$

where $FOV_h$ and $FOV_v$ are the horizontal and vertical dimensions of the FOV (in radians), respectively, and $W_F$ and $H_F$ are the width and height of the video frame (in pixels), respectively ($e_h$ and $e_v$ are also measured in pixels). Then, using a flat terrain model, the target’s ground position is given by

$$
x_t = x_v + z_v \cdot \left[ \cos(\psi) \tan \left( \frac{\pi}{2} - \theta - \theta_e \right) + \sin(\psi) \frac{\tan(\psi_e)}{\sin(\theta + \theta_e)} \right],
$$

$$
y_t = y_v + z_v \cdot \left[ \sin(\psi) \tan \left( \frac{\pi}{2} - \theta - \theta_e \right) - \cos(\psi) \frac{\tan(\psi_e)}{\sin(\theta + \theta_e)} \right].
$$

Trigonometry can be replaced with linear algebra. First, a vector is found in the camera axis that is pointed at the target. This vector is rotated into the body axis of the helicopter through the use of a direction cosine matrix. Then, the target’s ground location is given as the intersection of the rotated vector with the ground plane. This technique allows for roll to also be included when calculating the target’s location. For the preliminary work, the trigonometric equations are used.

Once the current target location is known, it is sent to the linear predictor. For the preliminary work, a third order predictor is used. The linear predictor then sends the future target locations to the camera controller. A proportional, derivative controller is used to model the camera controller. To model the error in the particle filter output, zero-mean, Gaussian, white noise with variance of four pixels squared is added to the frame position of the target. The results are shown in Figures 11 and 12 when the camera is 150 ft. above the ground.

The FOV moved with the target. As can be seen in Figure 12 after the target moved into the FOV, the error was less than 30 pixels. The image size is 180 by 120 pixels. Therefore, the target stayed well within the FOV. The use of prediction reduced the error...
Figure 11. Target path and FOV center ground path.

Figure 12. Error between the target position in the frame and the center of the frame.
by 47%.

While this preliminary work shows the efficacy of the camera control approach, it also shows the need for the position of the vehicle to change as the target moves. When the target moves further from the camera, the tracking becomes more challenging due to the reduced target image quality. This results in a larger tracking error (in pixels), which can cause even larger differences between the target’s location and the commanded position of the FOV center. Therefore, a portion of the proposed work is expected to add another layer of autonomy to the system that will allow the camera position to change based on the target’s location.

1.2.5.3 Flight Test Results

A similar particle filter was used to track a target during a flight test of the GTMax 2 rotorcraft UAV. The GTMax 2 is equipped with a commercially available web-camera that can move in both pan and tilt (Axis 213). The particle filter functioned on a laptop within the ground station. Due to extraordinary schedule constraints, the filter was not initialized automatically as described previously. Instead, the ground operator would click on the target to be tracked in an image. A histogram was taken in the manually selected region and used as the reference histogram. The initial particle set was also placed around the manually selected region.

The flight-test results were similar to those from above. However, since the images were sent over a 802.11(b) network, the delay between frames was 0.5 seconds in the best of cases. This required the target to move very slowly (5 mph) for the tracking to be successful. If the particle filter was placed onboard the helicopter, these delays would be dramatically reduced allowing the target to move more quickly without losing the track.

The camera controller relied upon existing software to convert the two-dimensional target position in the frame to a three-dimensional target location on the ground and kept the camera pointing at the target location. To smooth the movement of the camera, the target location was filtered. Therefore, even though the target was kept in the FOV, it was not necessarily in the center of the FOV. The proposed prediction-based controller is expected
to perform better. The waypoint generation also relied upon existing software, however it
did not have any knowledge of obstacles. Therefore, it simply moved the helicopter with
very low acceleration limits to keep the helicopter in the vicinity of the target.

This preliminary research shows the particle filter’s ability to track targets as they move
in cluttered video. Through the use of adaptation, the particles are placed more effectively
and the measurements become more useful. This allows the size of the particle set to be
reduced, which results in increased efficiency without degrading the filter’s performance.
However, as will be discussed in the next chapter, determining the correct particle filter
parameters is still a time intensive process that is done in an admittedly *ad hoc* manner.
Much of the proposed particle filter work centers around a change to the algorithm that
will simplify the parameter selection process while choosing better parameters.
CHAPTER II

PROPOSED RESEARCH

The goal of the proposed target tracking architecture is to, (1) enable the particle filter to robustly track targets as they move in cluttered images in real-time, (2) update the orientation of the camera to keep the target within the camera’s FOV, and (3) update the position of the camera (and vehicle) to ensure the tracking is successful.

2.1 Particle Filter Improvements

As seen in the preliminary work, changing the particle filter parameters within the system and measurement models improves the performance of the filter and allows the number of particles to be decreased in cases when the filter is performing well. However, the process of determining the values of the parameters within the rule set is little more than guess and check. In addition to being time consuming, this process must be redone for each tracking video.

To overcome this problem, the ability of the particle filter to test multiple hypotheses will be used. The performance estimate will determine if the values of the various parameters should increase or decrease. Then, a subset of the particles will be assigned values for the parameters according to a Gaussian distribution. The mean of the Gaussian distribution will be controlled by the performance estimate. Once the measurements are taken, the parameters will be evaluated. The system model parameters, $\beta_{RW}$, $\sigma_{RW}$, and $\sigma_j$, can be evaluated by examining the individual weights as in [22]. However, the measurement model parameters, $\sigma_C$, $\sigma_M$, and $\sigma_S$, will need to be evaluated differently since they each directly effect the weights. The performance of the particle filter will serve as a measure of how well the measurement model performed. If the filter performs better, then the parameters will be updated to the new values. Otherwise, they will be left unchanged. The number of particles, and the size of the updated portion, $J$, will still be governed directly by the
While the addition of these auxiliary variables, $\beta_{RW}$, $\sigma_{RW}$, and $\sigma_j$, to the state space might require additional particles, the increase is expected to be rather small. The mean of the three parameters will be controlled using the output of the performance estimate as in the preliminary research. Therefore, the parameters will not need to be as diverse to find the correct value for the parameters. This will result in only a small increase in the size of the particle set.

The reference color histogram will also be allowed to change using the method described in [27] and [28], where histograms of particles with high weights are blended with the old reference histogram to form a new reference histogram. While the method used in [22] resulted in better performance, it might require too many additional particles since the state space was augmented to allow the particles to determine the size of the contribution. Thus, real-time performance may become too challenging. However, the method will be tested to determine if a balance can be found between performance and processing time.

Since the output of the particle filter is used to make decisions at higher levels of the architecture, a confidence measurement should be available to aid in this decision making. While the value of the maximum color weight and the spread of the particles were used to estimate the performance of the filter, more information could be included in the performance estimate, resulting in a better estimate. A neural network is proposed to blend the following values from the particle filter: maximum color weight, standard deviation of the color weights, maximum motion weight, standard deviation of the motion weights, and spread of the ten highest weighted particles. The network should output one of four levels to give a heuristic measure of the filter’s performance. A multi-layer perceptron (MLP) is expected to give an acceptable classification of the performance when used with a back-propagation training algorithm [33]. The training data will be generated manually, off-line from previous filter output. A three layer MLP was used to estimate the filter performance in the preliminary work. Using a set of 100 training frames, the MLP correctly classified 85% of the frames. The accuracy should increase as more training data is generated. The output of the neural network will replace the simple rules that were used in the preliminary
research.

2.2 Camera Orientation Controller

After the particle filter estimates the position of the camera in the frame, the three-dimensional target location will be approximated using the method described in the preliminary research. If the neural network confidence indicator determines the filter is tracking the target well, the estimated target location will be used within the linear predictor to estimate the location of the target one time step into the future. The future target location will then be sent to the camera controller. If the confidence in the output is low, then the current target position will be used to update the camera orientation.

2.3 Camera Position Controller

The position of the camera and vehicle is updated by determining a set of waypoints for the vehicle to travel to. The choice of waypoints in this case should be based on three factors, (1) the target location, (2) required vehicle attitude change, and (3) linear ground obstacles such as a line of trees or a building wall.

The waypoint generation should keep the vehicle close enough to the target that the particle filter will be able to successfully track the target in the incoming frames. However, forcing the vehicle to follow every move of the target will result in large attitude changes. This could cause the orientation of the camera to change too rapidly for the controller to keep up with, which is clearly undesirable. Therefore, the waypoint is chosen to severely limit the acceleration of the vehicle. The actual limit will be determined experimentally. The two cases for waypoint selection are shown in Figure 13. In both figures the gray circle represents the set of points the helicopter can reach in a fixed time with a fixed acceleration (it is a circle since the vehicle is assumed to be at zero velocity; if the vehicle is moving it would be a trumpet shape). In (a), the target is predicted to be outside of the circle. Thus, the closest the helicopter can get to the target without breaking the acceleration limit is the point where the circle intersects the line connecting the target and the helicopter. In (b), the target is within the acceleration limit circle. Therefore, the next waypoint is offset
from the predicted target location to ensure the camera will not be pointing directly down.

![Waypoint generation](image)

**Figure 13.** Waypoint generation.

The shaded circle represents the area the vehicle can reach in a fixed time with a fixed acceleration limit. In (a) the target is outside of the circle; therefore, the next waypoint is the point on the circle closest to the target. In (b) the target is inside the circle; therefore, the next waypoint is the target’s location with a small offset to ensure the camera is not directly over the target. The offset is represented by the dashed line.

Finally, the waypoints should also keep known linear obstructions from occluding the view of the target. This case is shown in Figure 14. Here a line of trees is between the target and the vehicle, and the target is close to the obstruction. If the waypoint is assigned as in Figure 13 (a), the trees might block the view of the target. Therefore, the acceleration limit circle is modified to reflect only the points where the target is visible, and the point closest to the target is chosen as the next waypoint. When determining the possible waypoints (the shaded regions in Figures 13 and 14), only simple, kinematic equations will be used.

### 2.4 Expected Contributions

The proposed particle filter-based target tracking architecture is expected to contribute:

- A unified control structure that adapts both the particle filter parameters as well as the number of particles.
Figure 14. Waypoint generation with obstructions.
The circle represents the area the vehicle can reach in a fixed time with a fixed acceleration limit. However, only the shaded regions are allowed for waypoint generation since the other points will not result in a clear view of the target. The next waypoint is given as the point in the shaded region closest to the target.

- An automatic initialization process for the particle filter.
- A camera orientation command generator for tracking applications.
- A model and terrain-based waypoint generator for tracking applications.
- A target tracking architecture for use onboard a rotary-wing unmanned aerial vehicle.

This will result in the increased autonomy of a UAV.

2.5 Remaining Work

- Implement the neural network within the particle filter. The code for the neural network will be developed in Matlab and then moved to C++. The training data will be manually labeled using the previous filter output.

- Update the parameter adaptation scheme as described above. The system model parameters will be chosen within the structure of the particle filter, and measurement model parameters will be chosen using both the structure of the particle filter and the output of the confidence indicator.
• Allow the reference color histogram to change. The method in [22] will be studied to see if the size of the contribution of the new histogram can be found through an auxiliary variable. If this results in too much of a degradation in processing time, the size of the contribution will be controlled using the output of the confidence indicator.

• Continue to test the particle filter using data collected from flight testing. The GTMax and its twin, the GTMax 2, are expected to be used extensively in the near future as part of the Heterogeneous, Urban, Reconnaissance, Search, and Target Acquisition Team (HURT) project, which is sponsored by the Defense Advanced Research Projects Agency (DARPA). The video from this project should provide ample testing data.

• Implement the camera orientation controller in C++ and interface it with the Georgia Tech UAV Simulation Tool (GUST). This will allow real camera and vehicle models to be used.

• Implement the waypoint generator. The system described above will first be implemented in Matlab. Upon successful testing in Matlab, the system will be transitioned into C++ and interfaced with the GUST.

• Implement the entire system within the GUST, and simulate ground targets. The GUST is currently capable of simulating camera views and the resulting images. The camera orientation controller and waypoint generator will run within the GUST, while the particle filter will be added to it. The ground target’s motion will be governed by a unicycle model, and its positions will be displayed in the simulator. The system is expected to automatically track the targets in simulation.

• Move the particle filter, camera orientation controller, and waypoint generator to the onboard computers of the GTMax and interface the software with the available frame grabber. This will allow the closed-loop tracking during flight.

• Measure the performance of the system both in simulation and in flight. The following metrics will be used to measure performance:
– Percent of lost track. A track is defined to be lost when no particles are in the vicinity of the target for 10 seconds.

– Tracking error. This is the distance between the target’s position and the ground position of the center of the FOV.

– Processing time. This is the average and worst case processing time per frame for the filter.

2.6 Required Facilities

The proposed research will be performed in software. The requirements for this work are two computers with Microsoft Visual C++ 6.0. The first must be capable of running the GUST and have TV out capabilities. The GUST is available from the UAV Laboratory in the Daniel Guggenheim Aerospace Engineering School and has been used extensively for a variety of other projects. The second must be able to run the particle filter from the preliminary research in real-time.
APPENDIX A

PROOF OF MOTION METRIC

The distance generated by the motion measurements,

\[ D_M \left( x_i^k \right) = \left( 1 - \frac{M_i^k}{A_i^k} \right)^{\frac{1}{2}}, \quad (39) \]

is a metric. The distance is a measure of how far the particle is from full motion. Full motion occurs when \( \frac{M_i^k}{A_i^k} = 1 \). The distance between the motion of two particles can be written as

\[ D_M \left( x_i^k, x_j^k \right) = \left| \frac{M_i^k}{A_i^k} - \frac{M_j^k}{A_j^k} \right|^{\frac{1}{2}}. \quad (40) \]

This is a metric.

**Proof.** For the distance to be a metric, the following three items must be valid.

1. \( D_M \left( x_i^k, x_j^k \right) = D_M \left( x_j^k, x_i^k \right) \)

   This condition is trivially satisfied since \( \left| \frac{M_i^k}{A_i^k} - \frac{M_j^k}{A_j^k} \right|^{\frac{1}{2}} = \left| \frac{M_j^k}{A_j^k} - \frac{M_i^k}{A_i^k} \right|^{\frac{1}{2}} \).

2. \( D_M \left( x_i^k, x_j^k \right) \geq 0 \)

   Trivially, \( \left| \frac{M_i^k}{A_i^k} - \frac{M_j^k}{A_j^k} \right|^{\frac{1}{2}} > 0 \) unless \( \frac{M_i^k}{A_i^k} = \frac{M_j^k}{A_j^k} \), then \( D_M \left( x_i^k, x_j^k \right) = 0 \).

3. \( D_M \left( x_i^k, x_j^k \right) + D_M \left( x_j^k, x_l^k \right) \geq D_M \left( x_i^k, x_l^k \right) \)

   There are six cases to study.

Case 1. \( \frac{M_i^k}{A_i^k} \leq \frac{M_j^k}{A_j^k} \leq \frac{M_l^k}{A_l^k} \)

\[
\left[ D_M \left( x_i^k, x_j^k \right) + D_M \left( x_j^k, x_l^k \right) \right]^2 = \left( \frac{M_i^k}{A_i^k} - \frac{M_j^k}{A_j^k} \right) + 2D_M \left( x_i^k, x_j^k \right)D_M \left( x_j^k, x_l^k \right)
\Rightarrow D_M \left( x_i^k, x_j^k \right) + D_M \left( x_j^k, x_l^k \right) = \left[ D_M \left( x_i^k, x_j^k \right)^2 + 2D_M \left( x_i^k, x_j^k \right)D_M \left( x_j^k, x_l^k \right) \right]^{\frac{1}{2}} \geq
\]

\[ D_M \left( x_i^k, x_l^k \right) \]
Case 2. \( \frac{M^l}{A^l} \leq \frac{M^u}{A^u} \leq \frac{M^l}{A^l} \)

\[ D_M \left( x^l_k, x^j_k \right) \geq D_M \left( x^l_k, x^j_k \right) \]

\[ \Rightarrow \ D_M \left( x^l_k, x^j_k \right) + D_M \left( x^l_k, x^j_k \right) \geq D_M \left( x^l_k, x^j_k \right) + D_M \left( x^l_k, x^j_k \right) \geq D_M \left( x^l_k, x^j_k \right) \]

Case 3. \( \frac{M^l}{A^l} \leq \frac{M^u}{A^u} \leq \frac{M^l}{A^l} \)

\[ D_M \left( x^l_k, x^j_k \right) \geq D_M \left( x^l_k, x^j_k \right) \]

\[ \Rightarrow \ D_M \left( x^l_k, x^j_k \right) + D_M \left( x^l_k, x^j_k \right) \geq D_M \left( x^l_k, x^j_k \right) + D_M \left( x^l_k, x^j_k \right) \geq D_M \left( x^l_k, x^j_k \right) \]

Case 4. \( \frac{M^l}{A^l} \leq \frac{M^u}{A^u} \leq \frac{M^l}{A^l} \)

\[ D_M \left( x^l_k, x^j_k \right) \geq D_M \left( x^l_k, x^j_k \right) \]

\[ \Rightarrow \ D_M \left( x^l_k, x^j_k \right) + D_M \left( x^l_k, x^j_k \right) \geq D_M \left( x^l_k, x^j_k \right) + D_M \left( x^l_k, x^j_k \right) \geq D_M \left( x^l_k, x^j_k \right) \]

Case 5. \( \frac{M^l}{A^l} \leq \frac{M^u}{A^u} \leq \frac{M^l}{A^l} \)

\[ D_M \left( x^l_k, x^j_k \right) \geq D_M \left( x^l_k, x^j_k \right) \]

\[ \Rightarrow \ D_M \left( x^l_k, x^j_k \right) + D_M \left( x^l_k, x^j_k \right) \geq D_M \left( x^l_k, x^j_k \right) + D_M \left( x^l_k, x^j_k \right) \geq D_M \left( x^l_k, x^j_k \right) \]

Case 6. \( \frac{M^l}{A^l} \leq \frac{M^u}{A^u} \leq \frac{M^l}{A^l} \)

\[ \left[ D_M \left( x^l_k, x^j_k \right) + D_M \left( x^l_k, x^j_k \right) \right]^2 = \left( \frac{M^l}{A^u} - \frac{M^l}{A^l} \right) + 2D_M \left( x^l_k, x^j_k \right) \]

\[ \Rightarrow \ D_M \left( x^l_k, x^j_k \right) + D_M \left( x^l_k, x^j_k \right) = \left[ D_M \left( x^l_k, x^j_k \right)^2 + 2D_M \left( x^l_k, x^j_k \right) D_M \left( x^l_k, x^j_k \right) \right]^{1/2} \geq \]

\[ D_M \left( x^l_k, x^j_k \right) \]
Since all three items are true, the distance is a metric.
REFERENCES


HEGAZY, T., LUDINGTON, B., and VACHTSEVANOS, G., “Reconnaissance and Sureveillance in Urban Terrain with Unmanned Aerial Vehicles” in Proceedings of 16th IFAC World Congress, (Prague, Czech Republic), July 4-8, 2005.