Abstract

Recent military and civil actions worldwide have highlighted the potential utility for unmanned aerial vehicles (UAVs). Both fixed wind and rotary aircraft have contributed significantly to the success of several military and surveillance operations. Future combat operations will continue to place unmanned aircraft in challenging conditions such as the urban warfare environment, where surveillance is particularly challenging. These challenges as well as the reduced autonomy, and operator workload requirements of current unmanned vehicles present a roadblock to their success. It is anticipated that future operations will require multiple UAVs performing in a cooperative mode, sharing resources and complementing other air or ground assets to accurately calculate a target’s position. This paper reviews the current status of UAV target tracking technologies with emphasis on recent developments aimed at UAV improved autonomy and discusses future directions and technological challenges that must be addressed in the immediate future.

1. INTRODUCTION

Because of their ability to reach unique vantage points without endangering a human operator, camera equipped unmanned aerial vehicles (UAVs) are effective platforms for military and civilian surveillance missions. Small UAVs are particularly useful in urban military operations when it is necessary to see around the corner. However, urban surveillance is challenging because of the inherent clutter and occlusions in the environment. To effectively deal with this challenge, we have broken the problem down into three steps.

In the first step, the incoming video is processed using a particle filter to determine the location of the target. The particle filter is a sample-based approach to Bayesian state estimation that is able to approximate non-Gaussian distribution that evolve according to non-linear dynamics. The following section describes the particle filter framework and discusses methods of dealing with the algorithms large computational load.

After the particle filter estimates the position of the target, a reasoning layer provides situational awareness to deal with occlusions or other target tracking shortcomings. This layer uses a truth maintenance system to determine a set of scenarios that may have resulted in a degradation of the target tracking algorithm. The system will then use these scenarios to develop an effective strategy to remedy the situation. The truth maintenance system is introduced following the particle filter tracking algorithm.

Finally, an adversarial reasoning module is placed in the highest layer to produce strategies that deal with evading targets using a swarm of UAVs. By using a differential game framework, which is discussed after the truth maintenance system, a team of air vehicles is able to contain a target that is attempting to escape. The framework decomposes a complete game into a set of two player games, which are more easily solved. The adversarial reasoning module allows for group coordination as depicted on the Department of Defense’s UAV autonomy capability trend [1] in figure 1.
2. PARTICLE FILTERS FOR VISUAL TRACKING

The particle filter is a state estimation tool that is able to approximate a non-Gaussian distribution that evolves according to nonlinear dynamics using a randomly-selected set of weighted samples [2]. It is an effective tool for visual tracking because non-Gaussian distributions are prevalent due clutter in each image frame. As with the Kalman filter or other Bayesian techniques, the estimated distribution is recursively computed using the two step process of prediction and update. In the prediction step, the particle set is propagated forward one step in time using the system update model. In the update step, measurements are used to update the particle weights. The particles are resampled at each step by their weights. This results in the lower weighted particles being replaced by higher weighted particles. As the number of particles increases, the sampled-based distribution converges to the true distribution [3]. When using a particle filter to track targets in images, the state space is typically four dimensional and describes a rectangle. Therefore, the higher weighted particles correspond to the peak of the distribution and are rectangles that are near the target, while the lower weighted particles correspond to the tails of the distribution and are rectangles that are further away from the target.

A particle filter has been implemented that uses a system update model that is similar to [4], where the model is made up of two components. The first component assumes the target moves smoothly from one frame to the next. Therefore, the particles are moved according to a Gaussian random walk model. The second component accounts for the times when the smoothness assumption is violated, which can occur if the target becomes occluded or briefly moves outside of the FOV. This portion of the model is another Gaussian distribution that is centered around a randomly selected pixel where motion has been detected. Motion is detected in a pre-processing step. The two components of the model are combined using a convex combination, and the size of each contribution is changed throughout the tracking process depending on the performance of the system.

After the particles are moved, measurements are taken. Here, color and motion measurements are used. However, other cues can also be used depending on the sensor capabilities. Color measurements are taken by comparing the histogram of each particle to a reference histogram as in [5]. The reference histogram is either part of the a priori knowledge or is generated by manually selecting the target in the first frame. The particles that have histograms that are close to the reference histogram are assigned higher weights. Motion measurements are taken by subtracting the value (in the hue-saturation-value sense) of each pixel within the particle from the corresponding pixel in the previous frame and taking the absolute value. The differences are summed over the particle and normalized by the particle area. The particles that have the higher motion measurements are assigned higher weights. The total weight is generated by assuming the measurements are independent and taking the product of the weights generated by each measurement. The contribution of each measurement source is changed throughout the tracking process depending on the performance of the system.

The performance of the filter is estimated by examining properties of the particle distribution. A neural network is used to map the properties of the particle distribution, such as maximum weight, maximum color weight, and spatial spread of the particles to an estimate of the shape of the distribution. When a distinct peak is detected in the distribution, the particle filter is assumed to be performing well, and the parameters can be adjusted accordingly. The system update and measurement models can both be updated to reflect the filter’s performance. Also, the number of particles can be decreased when the filter is performing well. By reducing the number of particles, the inherently large computational burden of the particle filter can be decreased.

Figure 2 shows a set of output frames as the particle filter tracked a truck moving across a field. The video was taken from the GTMax research unmanned helicopter [6] and the frames were processed offline after converting the video to a series of frames at 10 frames
per second. In this case, the helicopter was hovering and the orientation of the camera was manually controlled. The output of the particle filter is shown as the red and blue rectangles. The ten highest weighted particles are represented by blue rectangles, while the remained of the particle set is represented by red rectangles. The weighted average of the particles is shown as the white crosshairs. As can be seen, the particle filter correctly estimated the position of the target. The tracking error is shown in Figure fig:terror

![Figure 2. Using a particle filter to track a truck from a UAV.](image)

![Figure 3. Particle filter tracking error.](image)

**3. TRUTH MAINTENANCE SYSTEMS FOR SITUATIONAL AWARENESS**

Tracking the target in real-time is a challenging task. It forms the core of any architecture and provides information on which related reasoning and inferencing mechanisms work. Tracking algorithms are limited in their reasoning abilities. There is usually no persistence associated with them. Also we do not get situation awareness out of the algorithms themselves. In this section we present a framework for automated reasoning that works on top of the tracking and image analysis algorithms to generate an overall picture for the reasoner and can provide insight into the state of the tracking to a human observer. We use a popular problem-solving class of systems called Truth-Maintenance Systems (TMS). The TMS work with inference engines and maintain the truth in the system by revising sets of beliefs as new information becomes available that may contradict existing information. TMS can solve problems where algorithmic solutions do not exist and are very suitable for large-scale spaces [7]. Like other knowledge-based problem solvers, the TMS work on domain knowledge. There are several shortcomings in conventional problem solvers that are addressed by a TMS. For example, a TMS by design provides an explanation for its reasoning process. It also recovers from inconsistencies such as incorrect inputs. This input can be an erroneous sensor or it can make some incorrect assumptions. When the input makes the situation clear, the assumption may prove to be incorrect. At this point the bad assumption and its related inferencing are retracted. A TMS also maintain its previous inferences and therefore avoids performing the same processing again and again [8].

Tracking problems require maintaining an internal awareness of the situation. This awareness becomes especially useful in situations where the tracking algorithms begin to lose confidence in their tracking abilities. Such situations may arise due to clutter in the images, partial or full occlusions, change in the image quality due to light or weather conditions, etc. A typical tracker can potentially lose track of its target(s) in these situations and improving the tracking algorithm alone may not be enough to address this problem. As shown in Figure 4, we add a reasoning layer that maintains data structures across video frames. This layer receives tracking information about a target from the tracking layer that is also responsible for image analysis. The reasoning layer maintains a database of current situation. The data is maintained in terms of the coordinates of the target, its features being tracked, and other relevant information such as presence of buildings, trees
As new situations arise, such as occlusion due to the presence of a building, the reasoning layer (guided by tracking database) generates scenarios that may be relevant here. These scenarios are treated as assumptions by the inferencing mechanism. Some assumptions may be related to expected time of occlusion etc. As time passes by if the target does not emerge from occlusion, new assumptions will be added, possibly taking into account other possibilities such as target is aware of tracking, target’s final destination, dismounted targets. The tracker is made aware of the situation and its control as a result may decide to hover around during this time.

**Figure 4. Truth maintenance system for target tracking**

### 4. DIFFERENTIAL GAMES FOR ADVERSARIAL REASONING

While a UAV is tracking a target, the target may become aware of the UAV and attempt to avoid the aircraft. Due to limited maneuverability of the UAV and possible obstacles such as buildings and trees, the target may be able to leave the field of view of the UAV. However, when dealing with swarms of UAVs, it is possible to overcome the limitations of the individual UAV by allowing the vehicles to work in concert, thereby guaranteeing that the target will remain within the field of view of at least one UAV. Hence, when a target attempts to avoid the UAV, which is currently maintaining a visual lock on the target, the other UAVs must be able to predict the future actions of the target and positioning themselves such that the target can be handed off effectively. Naturally, to determine how the vehicles should be deployed, it is necessary for the swarm to determine the target’s escape strategies by considering the constraints imposed by the environment, the targets maneuvering capabilities, the current location of all the UAVs in the swarm and the maneuvering capabilities of the UAVs.

To accomplish this, a differential pursuit-evasion game framework is used to determine the escape strategies of the evading target and the actions needed to be taken by the swarm to ensure that the target does not escape.

The main result from the work done on differential games is the so-called Hamilton-Jacobi-Bellman-Issacs (HJBI) equation given by

\[ \frac{\partial V}{\partial t} + H(x,t,DV) = 0, \]  

where the Hamiltonian \( H \) is

\[ H(x,t,DV) = \min_{u_p} \max_{u_e} \left[ f_1(x,t,u_p,u_e) \cdot \frac{\partial V}{\partial x_1} + \cdots + f_N(x,t,u_p,u_e) \cdot \frac{\partial V}{\partial x_N} + L(x,t,u_p,u_e) \right] \]

given the cost functional

\[ J = \phi(x(t_f)) + \int_0^{t_f} L(x,t,u_p,u_e) \, dt \]

The control inputs for the UAVs and the evading target are \( u_p \) and \( u_e \), respectively. \( \phi() \) is the termination cost and \( L() \) is the integrated cost functional. \( V \) is the value of the game, that is, given a particular state of the game, \( V \) describes the cost of intercepting the evading targets. The system is governed by the following system dynamics:

\[ \dot{x}_1 = f_1(x,t,u_p,u_e) \]
\[ \vdots \]
\[ \dot{x}_N = f_N(x,t,u_p,u_e), \]

where \( x \in \mathbb{R}^N \).

Since initial conditions are expressed over the entire state space, the interception cost will have to be determined for all the states. Hence, since the size of the state space is \( N = m \cdot n \), where \( m \) is the number of players and \( n \) is the size of the individual players’ state space, the computational complexity of the problem increases exponentially with an exponent of \( m \cdot n \). The computational complexity suggests that means must be sought that will reduce the computational burden and make it feasible for real-time implementation of the scheme.
A hierarchical decomposition technique is used to reduce the computational burden of the multiplayer stochastic game problem. The decomposition is done in three stages. In the lowest level, the interception strategy for each of the pursuers is determined by solving several two-player stochastic differential pursuit-evasion games. In the middle level, one or more pursuers are assigned to each evading target based either on a target importance measure or on the time to intercept. In the highest level, a dynamic performance-based region of responsibility (DPRR) is computed for each of the pursuers assigned to a particular target.

The decomposition approach essentially reduces the multiplayer differential game problem into several much simpler two-player problems as shown in Figure 5. By performing the decomposition, the cooperation between the pursuing players is not considered. Hence, the mid- and high-level steps in the decomposition approach are designed to reintroduce cooperation in an intelligent fashion. Based on the estimated interception time derived in the low-level decomposition stage, each of the pursuers is assigned a DPRR. If an evader is in a pursuer’s DPRR, it is that pursuer’s responsibility to intercept the evading target. However, if there are no evaders in a particular pursuer’s DPRR, the pursuer will move toward a virtual target. The virtual target is the point on the boundary of the pursuer’s DPRR with the largest difference between the estimated time to capture and the time it takes the evader to reach the point. This point is where it is most likely for the evader to cross into the pursuer’s DPRR. Figure 6 illustrates the target assignment.

The evading target in the center of the figure is attempting to escape the three pursuers. Only the pursuer at the top of the picture is attempting to intercept the evading target directly. The other two pursuers are heading toward the virtual targets in an attempt to block the evader’s possible escape routes, that is, they are essentially performing containment maneuvers. It should be noted that the DPRR for each of the pursuers is updated regularly; consequently, the tasks assigned to the pursuers also changes frequently depending on the actions taken by the evading target.

The decomposition approach is advantageous, since it is computationally much simpler to solve the two-player games than general multiplayer games. The approach is able to adapt rapidly to changes in the scenario, that is, if new pop-up targets are encountered the roles of each of the pursuers is reassigned to effectively handle the unexpected change to the scenario. Additionally, the algorithm can easily be implemented in a distributed fashion, that is, the computational resources onboard all of the UAVs can be utilized effectively.

5. CONCLUSIONS

As the autonomy and reliability of UAVs improve, UAVs will find themselves being relied upon in more complex situations. This paper presented a target tracking system that can be used in the complex situation of tracking targets in urban warfare environments. The hierarchical system was presented from the bottom up. It includes a particle filter visual tracker, a truth maintenance system for situational awareness, and an adversarial reasoner to allow effective UAV teaming. Such a system will provide robust and reliable information while keeping a human operator out of harms way.
6. ACKNOWLEDGEMENTS

This work was partially supported by a US Army SBIR Phase II Project under contract W911NF-06-C-0018 via Impact Technologies LLC and by the DARPA/IXO HURT program and the Air Force Research Laboratory under contract FA865004-C-7142 via Northrop Grumman. The authors thank these organizations for their support.

References