A State Estimation Approach for Live Aircraft Engagement in a C2 Simulation Environment

Arno Duvenhage
Willem H. le Roux
Council for Scientific and Industrial Research
Meiring Naude Rd
Pretoria
South Africa
aduvenhage@csir.co.za, whleroux@csir.co.za

Keywords:
Alpha-Beta, Pure Pursuit, C2, Dead Reckoning, Tracking, State Estimation, Guidance

ABSTRACT: *The increased use of simulations during live air defense exercises requires interoperability between different Command and Control (C2) systems and simulators. By accepting air picture or sensor tracks from each other, C2 systems and simulators can react to or engage targets between live and simulated worlds in various ways. State estimation of live aircraft can be a challenge when data from dedicated aircraft transponders is not available. In such cases air picture data have to be used. Often the data is then available only through C2-oriented communications protocols that may introduce temporal and spatial errors. This paper presents an algorithm for modeling aircraft within a real-time simulation environment using only track position data from external C2 systems and simulators. The predicted flight path is smoothed and realistic aircraft orientation, such as banking in a turn, is also introduced.*

1 Introduction

Command and Control (C2) systems are often interconnected during live exercises using the relevant C2 communications links. These C2 links are often low to medium bandwidth links with data packets subject to jitter and transmission delays. This can cause unpredictable delays and unpredictable or low update rates which can make using the data for modeling problematic. The quality of the plot and track data is subject to noise present in the relevant air picture or sensor tracks. The nature of the C2 protocol and the link can also degrade data.

C2 equipment and systems may be replaced with real-time simulators. This is done to reduce the cost of live exercises or to experiment with systems that don't exist or have not been commissioned yet. Real-time engagement of live aircraft for example in a simulated environment can only be accomplished if the actual aircraft are recreated or injected into the simulation environment in some way. This involves modeling the aircraft with an algorithm that can, in real time, recreate a realistic flight path from plot and track data available through the C2 links.

This paper will start with an overview of the aircraft injection concept and the algorithm used to inject aircraft into a simulation environment. Thereafter a detailed discussion on a specific example implementation of the algorithm will be given.

1.1 Aircraft Injection

The modeled aircraft flight path should be complete and smooth. In a simulation environment any amount of jerk (abrupt changes in acceleration) in target motion might cause undesirable behavior within the simulated systems; some models may contain state estimation techniques for prediction or threat evaluation that are sensitive to jerk. The aircraft's position, velocity and orientation have to be calculated at each simulation time step, while keeping the aircraft behavior consistent with the flight dynamics of the relevant aircraft type.

A trade-off between spatial and temporal accuracy have to be considered as well. There should be a minimal lag between the modeled aircraft state and its counterpart in the relevant external system or simulator – Engaging targets across live-simulator or simulator-simulator boundaries require a high degree of temporal accuracy to keep the C2 timelines as correct as possible.

It is important to note that the algorithm presented in this paper does not perform the same function as dead reckoning (DR). Dead reckoning is a technique used by simulation technologies like the Distributed Interactive Simulation (DIS) to reduce the frequency and consequently the bandwidth requirements of distributing state updates and deals with network issues [1]. This is done by predicting a model's state on all hosts until the model's controlling host deems a state update necessary. The model's controlling host will send out a state update if the prediction error becomes too large. These error correcting updates will correct the model state on all hosts and will cause the model to *jump* on all hosts viewing the model. Larger errors are tolerated if bandwidth requirements are more strict [2].

The algorithm presented in this paper operates with whatever data it gets from the relevant communications links. The algorithm, though not similar to DR, is however similar to the approach presented by Aggarwal et al [3] in the sense that it will correct for jitter and delays present in computer networks. The algorithm models aircraft from external plot and track position updates that may arrive at low and unpredictable rates. The algorithm can however be used to smooth out the jumps caused by dead reckoning state updates (error correcting updates) on hosts looking at for example an aircraft model – in this case the standard DR prediction algorithms will be replaced by the algorithm presented in this paper.

1.2 Algorithm Overview

The algorithm includes both state estimation and guidance (block diagram shown in Figure 1).

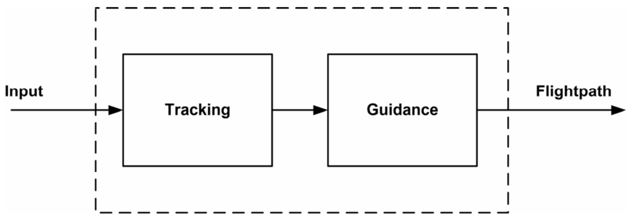

Figure 1: Algorithm Overview

The basic principle is that the guidance component guides the simulated aircraft from its current position on an *ad hoc* basis (in real-time) to position predictions calculated by the tracking component. The position predictions are made for the next expected input sample time each time a new input sample is received.

The tracking component is responsible for:

- Estimating the position of the aircraft from the incoming data that may contain errors,
- estimating the time of the next input sample,
- compensating for temporal variance and lag in incoming position data, and
- mitigating possible lag caused by the guidance algorithm by being able to predict guidance input into the future.

The tracking component can be a kinematic tracking filter like an α-β or α-β-γ type tracking filter [4]: These are 2nd and 3rd order kinematic filters that are easy to implement compared to the more complex Kalman, particle and Interacting Multiple Model (IMM) type tracking filters that may also be used. Gray and Foster [5] discuss a coefficient selection method and also show how the properties of a steady state Kalman filter compare to the properties of α-β and α-β-γ filters. α-β and α-β-γ can deliver similar results to a steady state Kalman filter, but is less complex to implement [5].

Noh et al [6] and Berg [7] present alternative approaches to adjusting the Kalman gain for tracking maneuvering targets. Berg's approach is proven in anti-aircraft gun fire control field tests and is computationally less complex, compared to extended Kalman filters (EKFs). Noh's approach provides similar results to IMM tracking filters by using a fuzzy gain calculation system.

Morelande and Challa [8] discuss particle filters and the interacting multiple model, probabilistic data association, extended Kalman filter (IMM-PDA-EKF) algorithm. They state that the Kalman filter performs worse when used for tracking than a particle filter in large clutter densities, but performs better when the measurement are precise. Bugallo et al [9] also compare the performance of particle and Kalman filters for tracking maneuvering targets. They discuss a cost-reference particle filtering (CRPF) algorithm which has a superior performance to both standard particle filters (SPF) and extended Kalman filters (EKF).

The guidance component is responsible for:

- Guiding the aircraft to the predicted aircraft positions on an *ad hoc* basis (in real-time as new predictions are made),
- calculating the aircraft orientation at the desired fidelity (discussed in Section 3),
- limiting aircraft movement to stay within the dynamics of the relevant aircraft type, and
- ensuring that the aircraft motion contains no jerk.

Guidance laws that can be used for the guidance component are pure pursuit and proportional navigation. Belkhouche et al [10] and Naeem et al [11] present different applications of the pure pursuit algorithm and discuss the similarity between pure pursuit and proportional navigation. Pure pursuit guidance can also be called a tail chase algorithm since the pursuer always tries for a trajectory directly toward the target. Proportional navigation on the other hand over steers in an attempt to keep the line of sight angle between the pursuer and target constant. The line of sight angle can be defined as the angle between the line of sight from the pursuer to the target and the pursuer trajectory. Menon and Ohlmeyer [12] give an alternative but equivalent definition of pure pursuit and proportional navigation. They compare the two within a missile simulation.

Belkhouche et al [13] discuss alternative tracking of moving targets for application in robotics and present an algorithm that includes obstacle avoidance as well. Advanced high fidelity navigation with a proper airframe model and an autopilot is also possible. Using a six degree of freedom airframe model would result in a realistic flight path with realistic aircraft orientation and velocity. Model Predictive Control (MPC) uses a model of a process to predict and control future process outputs based on a specific control sequence and can be applied to guidance. The process being controlled can be linear or non-linear and a Genetic Algorithm (GA) can be used to optimize the control sequence [11].

Both the tracking and guidance components can introduce lag (output lagging behind input maneuvers), but this can be mitigated by the tracker: The tracker can predict/estimate the aircraft state into the future enough to compensate for or mitigate lag in the algorithm. Finally it is also important to stress that the tracking component effectively runs on the input timestamps (not synchronized to simulation time) and outputs predictions in simulation time. The guidance component on the other hand is synchronized to simulation time. Alternatively the algorithm can be seen as transforming data from an arbitrary input time and update rate to aircraft states synchronized to the simulation time and update rate.

2 Background

Motivation for the development of the aircraft injection algorithm is provided in this section, together with some cases where it has been applied successfully.

2.1 Traditional Tracking and Jerk

Prediction and state estimation techniques are concerned with tracking a maneuvering target, using a combination of state prediction and state update operations. The state update operations correct for incorrect state predictions of maneuvering targets. These corrections normally introduce some form of discontinuity in the output causing an abrupt change in state relating to a perceived target jerk. Jerk in target motion might cause undesirable behavior within the simulated systems.

Traditional filtering techniques used for state estimation (like α-β, α-β-γ, Kalman and particle tracking filters) all suffer from state updates that cause jumps (error correcting updates) in the output if the tracker can't estimate the target maneuver accurately enough. Figure 2 shows the output of an α-β tracker and the jumps in the output of the tracker where state updates occur can be seen.

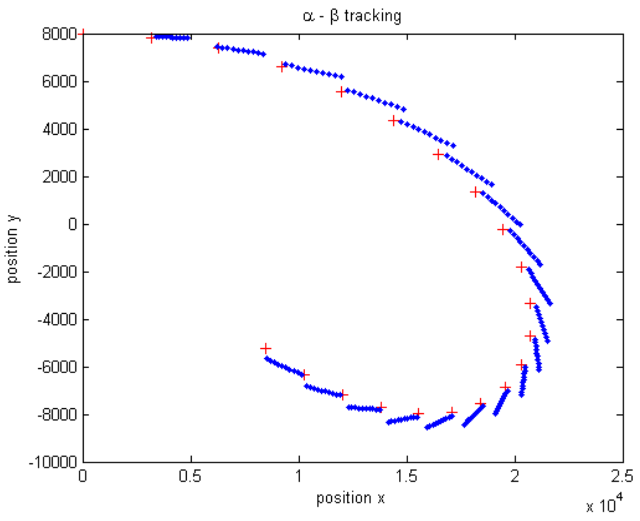

Figure 2: Target position observations (crosses) being tracked by an α-β filter (dotted lines).

2.2 Successful Application

The algorithm has successfully been used to model live aircraft from track position data from real designation, as well as tracking radars connected via a C2 protocol gateway to an air defense simulation. This allows a simulated air defense battery to engage live aircraft in a virtual environment. Operator in the loop fire control terminal mock-ups also allows air-defense doctrine to be evaluated without the need to physically deploy weapon systems.

Ships have also been modeled from Automatic Identification System (AIS) data received from sensors located on a military corvette. The AIS is a maritime identification system that allocates a unique identifier to each vessel.

The algorithm has also successfully been used to share air picture data between two completely different simulations running at different update rates and different simulation times. The sharing of air picture data enables collaboration during exercises among different C2 simulators interconnected with the relevant C2 links [14].

The simulated aircraft motion (including aircraft orientation) contributed to the perceived realism of the simulation environment. Recording the simulation results allowed pilots and air defense personnel to review and discuss the live exercises.

3 Aircraft Behavior and Dynamics

The flight path of a fixed wing aircraft will look different to the flight path of a rotary wing aircraft. The behavior of military aircraft is also much different to the behavior of civilian aircraft. The algorithm parameters can be varied to be more applicable to different types of targets if the simulation has access to aircraft identification and classification information (The modeled aircraft should not accelerate, turn or move unrealistically).

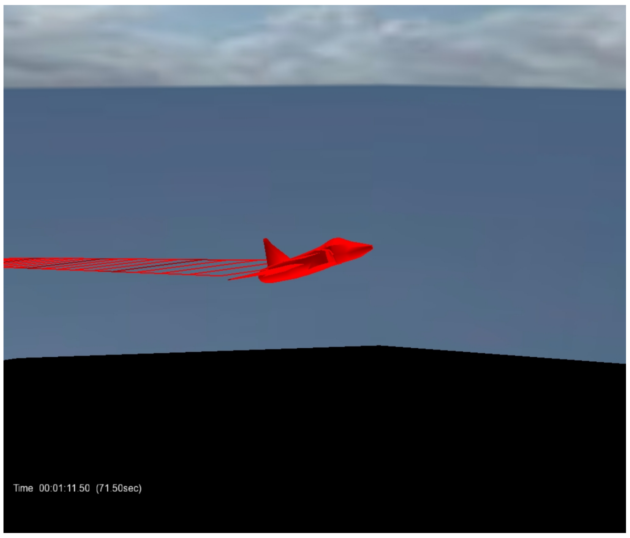

Figure 3: Slow flying aircraft model with a large angle of attack.

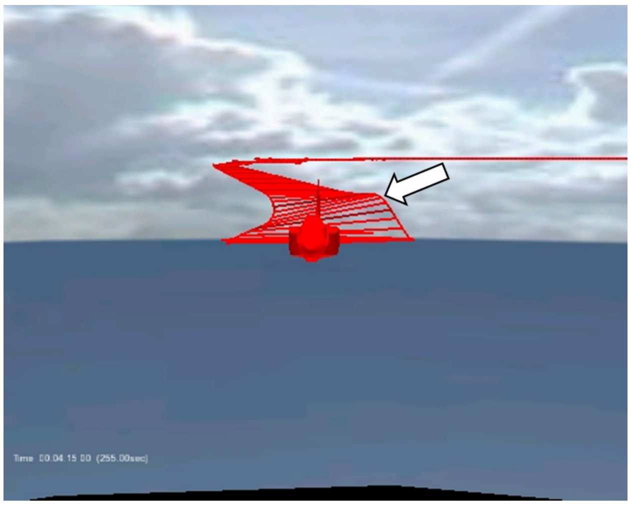

Figure 4: Aircraft model banking in a turn.

Rotary wing aircraft for example lean forward when accelerating and lean back when stopping. Fixed wing aircraft on the other hand have an angle of attack that is approximately inversely proportional to its airspeed (as shown in Figure 3). Figure 4 shows a modeled aircraft banking or rolling with an angle roughly proportional to the angular acceleration in a turn when maneuvering. The angle of attack also leads the pitch angle (i.e. angle of attack leads change in velocity) for most maneuvers (shown in Figure 5).

The correct orientation of the aircraft is required when calculating Radar Cross Section (RCS) values required by radar models for example. It may also be of value (enhance sense of realism) when the models are visualized in 3D and even 2D viewers, terminals or scenario planners. The images presented in this section show an aircraft modeled with a specific implementation of the algorithm discussed in the next section (Figure 3, Figure 4 and Figure 5). The fidelity of the modeled aircraft's orientation can range from superficial (only for visualization purposes) to very accurate, depending on the requirements and the guidance algorithm used.

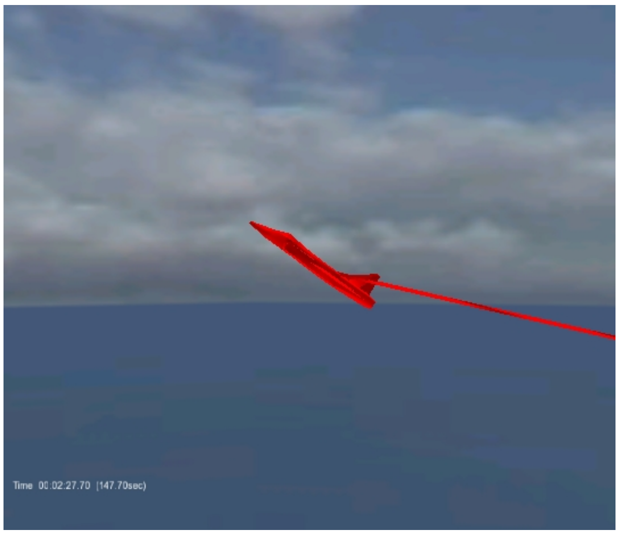

Figure 5: Aircraft model angle of attack leading pitch up maneuver.

4 Aircraft Injection Algorithm

This section will detail a specific example of the algorithm implemented as part of a threat injector service that was added to a ground based air defense simulation. This was done to enable the engagement of live aircraft during exercises. The algorithm can be called an *alpha-beta-pure-pursuit* algorithm since it uses an α-β tracker and a pure pursuit guidance algorithm.

4.1 Aircraft Injection Requirements

The primary goal of the threat injector service is to model live aircraft in a ground based air defense simulation for live aircraft engagement. The threat injector service operates on plot and track data from real and simulated systems. The data is expected to contain temporal and spatial errors caused by noise and unpredictable network delays. The relevant protocol's quantization of values, like position and timestamps, can also introduce spatial and temporal errors.

Simulated threat evaluation and weapon systems may use a modeled aircraft's velocity vector to calculate intercept points for example. These calculations are often very sensitive to irregularities in target motion, especially in target velocity, and require target state updates at high and reliable rates. The state of a modeled aircraft should be available in real-time at the simulation update rate. The aircraft motion should contain no jerk and be consistent with the flight dynamics of the relevant live aircraft type (even if the input data contains errors).

The orientation of the injected aircraft also has to be modeled, but for visualization purposes only. This means that the orientation does not have to be correct, but only provide appropriate visual queues to add realism to the aircraft behavior when visualized.

4.2 Variable Observation Rate α-β Tracker

The α-β filter is a 2nd order *kinematic* filter that assumes the target moves at a constant velocity between measurement updates. This is not the case for maneuvering targets, although the assumption may hold for sections of a flight profile. This specific implementation allows for variable update rates and delayed observations.

Blackman describes the α-β filter in [15]. Equation 1 and Equation 2 are used to update the filter state (output position, x_s, and output velocity, v_s) using the difference between the input observation, x_o, and the predicted state, x_p. This is done each time a new target observation is available. T is the time since the last state update.

Equation 1: State update

$$x_s(k) = x_p(k) + \alpha\left[x_o(k) - x_p(k)\right]$$

Equation 2: State update

$$v_s(k) = v_s(k-1) + \frac{\beta}{T}\left[x_o(k) - x_p(k)\right]$$

Equation 3 is used for state prediction. State prediction is done between observations and to correct for delayed observations and variable update rates. x_p is recalculated with the new observation timestamp, before performing the state update in order to correct for delayed observations or variable update rates. The gains (α and β) control how much noise in the input affect the system, but changing the gains also affect the tracking performance. Larger values for α and β will allow the filter to converge to a steady state faster, but make it more sensitive to input noise; Smaller values for α and β will make it less sensitive to input noise, but the filter will converge slower.

Equation 3: State prediction

$$x_p(k+1) = x_s(k) + T.v_s(k)$$

The Benedict Border relationship (Equation 4) given in [4] was used to calculate β from α.

Equation 4: Benedict Border relation

$$\beta = \frac{\alpha^2}{2-\alpha}$$

4.3 Pure Pursuit Guidance

Pure pursuit guidance moves the pursuer (or aircraft in this case) directly toward its target i.e. tries to keep the line of sight angle zero [10]. Pure pursuit guidance will cause the pursuer to trail the target and intercept it from behind. This behavior is desirable since the *alpha-beta-pure-pursuit* algorithm tries to follow and predict a target's movements instead of intercepting the target. The aircraft velocity and acceleration is limited to the aircraft dynamics, since excessive delays and errors can cause the algorithm to become unstable (runaway behavior).

4.4 MATLAB Implementation

The alpha-beta pure pursuit algorithm was implemented in MATLAB to verify the algorithm and provide some further insight on improving it.

The α-β tracker estimates the state of the aircraft from the input observations and predicts the position of the next observation. The prediction is made with the assumption that the input update rate is constant, but the α-β tracker also corrects for delays and variable update rates. Pure pursuit guidance is then applied to intercept the predictions made by the tracker. This is done by moving in a straight line from the current aircraft position towards the prediction with a speed calculated to reach the prediction at the expected time of the next input observation.

Figure 6 shows the output of the alpha-beta pure pursuit algorithm. The difference (or maneuvering lag) between the input and output of the algorithm is caused by the tracking filter's inability to estimate the target maneuver, but this behavior can be expected from most tracking filters.

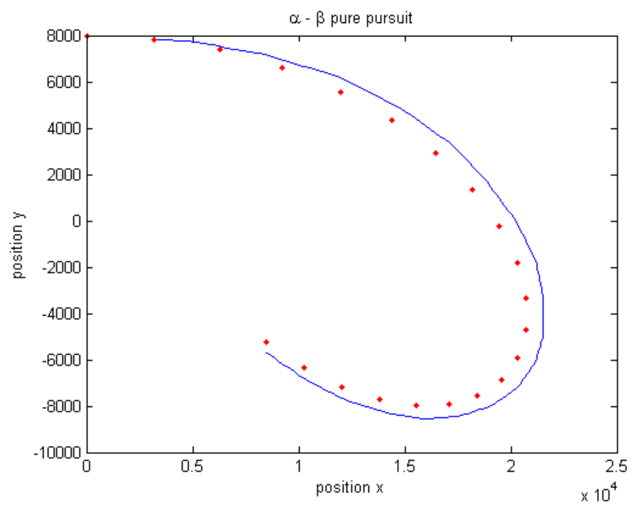

Figure 6: Target (dotted line) and output of alpha-beta pure pursuit algorithm (solid line).

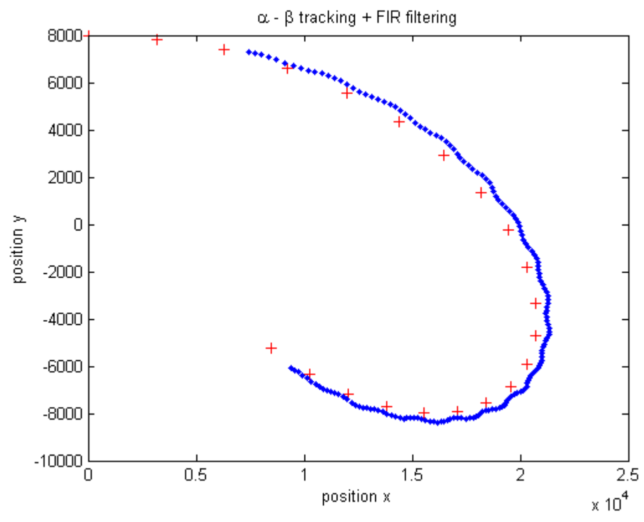

Figure 7: Output of the algorithm (dotted line) if no guidance is done.

4.5 FIR filtering

The guidance algorithm output is lastly sent though a Finite Impulse Response (FIR) filter to simulate aircraft dynamics (smooth out the flight path and remove any jerk). The aircraft's smoothed velocity is then calculated using the filtered positions, while the aircraft's smoothed acceleration is calculated using the smoothed velocities. A higher order FIR filter (more coefficients) will smooth the output more than a lower order FIR filter, but a higher FIR filter has a larger phase lag. Phase lag causes the algorithm output to lag behind aircraft maneuvers which is undesirable.

Figure 7 shows what the output of the algorithm would look like if no guidance is done i.e. a FIR filter is applied directly to the α-β tracker output shown in Figure 2. Comparing this to the smooth output shown in Figure 6 clearly shows the added benefit of the guidance algorithm: The main disadvantage of using only a FIR is the phase lag; with pure pursuit guidance the flight path doesn't suffer from as much phase lag.

4.6 Aircraft Orientation

The aircraft orientation generated by this algorithm is largely superficial (an approximation for visualization purposes only) and calculated based on the aircraft's smoothed (output of FIR filtering process) velocity and acceleration. The difference between the aircraft north-east-down (NED) axis and the aircraft body axis is shown in Figure 8.

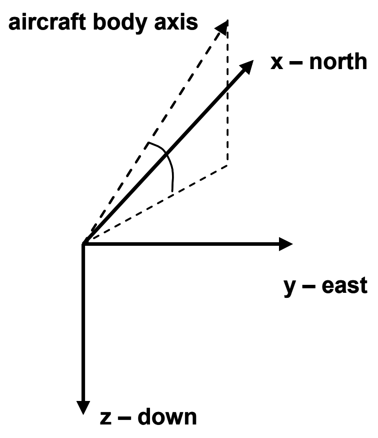

Figure 8: Aircraft axis

The aircraft heading and pitch is calculated based on the aircraft's velocity. The pitch angle is then scaled to make it look as if the pitch angle leads the change in velocity during pitch up or pitch down maneuvers (Figure 5). The aircraft's pitch is augmented with an angle of attack that is inversely proportional to the square of the aircraft's speed (Figure 3).

The current normalized velocity (from smoothed velocity) and previous normalized velocity vectors are used to calculate the roll angle. The cross product of the two vectors results in the axis around which the aircraft is rotating during a maneuver. The dot product of that axis and a vector pointing up (negative NED z-axis) then results in an indication of the aircraft's yawing (turning around NED z-axis). This is then used as the roll indicator since an aircraft usually banks when turning. Figure 4 shows the aircraft rolling while making a turn. Subtle coupling of the aircraft heading with the roll indicator can also add to the perceived realism by causing a slight sideslip during a roll maneuver.

Aircraft maneuvers usually lag behind the relevant instigating change in orientation i.e. an aircraft would for example start rolling before being able to perform a banking maneuver. The modeled aircraft orientation, as illustrated in the *alpha-beta-pure-pursuit example*, is however calculated from the aircraft maneuvers. This will have the effect of the modeled aircraft for example not banking before a turn, but banking during the turn [16].

5 Conclusion

This paper presented an algorithm for modeling aircraft within a real-time simulation environment using only track position data from external C2 systems and simulators. The test implementation of the algorithm was based on a variable update rate α-β predictor and pure pursuit guidance.

The algorithm successfully modeled aircraft from real and simulated data:

- Aircraft modeled from live plot and track data were successfully engaged in a simulation environment,
- aircraft modeled from shared air picture tracks allowed two C2 simulators to collaborate during exercises, and
- modeled aircraft were visualized during exercises and demos and the orientation of the modeled aircraft added to the perceived realism of the simulation environment.

6 Future Work

One drawback of the current test implementation of the algorithm is the lag caused by the FIR filter and to a certain extent the α-β filter as well. It would be worth while to investigate phase correction or alternative filtering and tracking techniques that specifically minimize phase lag.

Future work on the guidance component could include:

- Using heuristics to check for valid aircraft behavior (e.g.. using a flight envelope ellipsoid to limit aircraft movement).

- Using a high fidelity (three or six degree of freedom) airframe model coupled with navigation for the guidance component.

- Using a fixed or variable thrust guided missile model as the guidance component.

Research on the added benefit of more advanced prediction and guidance techniques would indicate whether to focus on improved prediction or improved guidance algorithms in future implementations.

7 References

[1] Samir Torki, Patrice Torguet and Cedric Sanza: "Optimizing Dead Reckoning with Classifier Systems" Euro SIW, June 2007.

[2] Wentong Cai, Francis B.S. Lee and L. Chen: "An Auto-adaptive Dead Reckoning Algorithm for Distributed Interactive Simulation" Workshop on Parallel and Distributed Simulation, pp. 82-89, 1999.

[3] Sudhir Aggarwal , Hemant Banavar and Amit Khandelwal: "Accuracy in Dead-Reckoning Based Distributed Multiplayer Games" ACM SIGCOMM workshop on Network and System Support for Games, pp. 161-165, 2004.

[4] Dirk Tenne and Tarunraj Singh: "Optimal Design of α-β-(γ) Filters" American Control Conference, June 2000.

[5] John E. Gray and George J. Foster: "Filter Coefficient Selection Using Design Criteria" Systems Theory, Vol. 28, pp. 275-279, April 1996.

[6] S.Y. Noh, J.B. Park and Y.H. Joo: "Intelligent Tracking Algorithm for Manoeuvring Targets using Kalman Filter with fuzzy gain" IET Radar Sonar Navig., Vol. 1, No. 3, pp. 241-247, 2007.

[7] Russel F. Berg: "Estimation and Prediction for Manoeuvring Target Trajectories" IEEE Transactions on Automatic Control, Vol. 28, No. 3, pp. 294-304, March 1983.

[8] Mark R. Morelande and Subhash Challa: "Manoeuvring Target Tracking in Clutter using Particle Filters" IEEE Transactions on Aerospace and Electronic Systems, Vol. 41, No. 1, pp. 252-270, January 2005.

[9] M.F. Bugallo, S. Xu and P.M. Djuric: "Performance Comparison of EKF and Particle

Filtering methods for Manoeuvring Targets" Stony Brook University, Stony Brook, October 2006.

[10] Fethi Belkhouche, Boumediene Belkhouce and Parviz Rastgoufard: "A linearized model for the *kinematic* equations of the pure pursuit guidance law" AIAA Guidance, Navigation and Control Conference and Exhibit, August 2004.

[11] W. Naeem, R. Sutton and S.M. Ahmad: "Pure Pursuit Guidance and Model Predictive Control of an Autonomous Underwater Vehicle for Cable/Pipeline Tracking" World Maritime Technology Conference, October 2003.

[12] P.K. Menon and Ernest J. Ohlmeyer: "Integrated Guidance-Control Systems for Fixed-Aim Warhead Missiles" AIAA Missile Sciences Conference, November 2000.

[13] Fethi Belkhouche, Boumediene Belkhouce and Parviz Rastgoufard: "Line of Sight Robot Navigation Toward a Moving Goal" IEEE Transactions on Systems, Man, and Cybernetics, Vol. 32, No. 2, pp. 255-267, April 2006.

[14] J.J. Nel, W.H. le Roux, O. van der Schyf and Lt Col M. Mostert: "Modelling Joint Air Defence Doctrinal Issues with a Link-ZA based Integration of two C2 Simulators – A Case Study" Military Information and Communications Symposium of South Africa (MICSSA), Vol. 3, July 2007.

[15] Samuel S. Blackman, Multiple-Target Tracking with Radar Applications, Artech House, 1986.

[16] J.M. Rolfe and K.J. Staples, Flight Simulation, Chapter 3, Cambridge, 1997.

ARNO DUVENHAGE is a Researcher for the Council for Scientific and Industrial Research (CSIR), South Africa. He joined the CSIR's Mathematical and Computational Modeling Research Group in January 2005 as a Software Engineer. Arno's current work involves modeling and simulation for acquisition decision support, focusing on air defense, specializing in distributed and networked systems. Arno has a BEng Degree in Computer Engineering from the University of Pretoria, South Africa, and is currently specializing in software engineering.

HERMAN LE ROUX has been with the South African Council for Scientific and Industrial Research since April 1998 and is at present a Principal Engineer in the Mathematical and Computational Modeling Research Group. He is involved in Modeling and Simulation-based Decision Support, specifically for the South African National Defense Force. Interests include information fusion, biometrics, artificial intelligence and software engineering. Le Roux completed a Masters Degree in Computer Engineering at the University of Pretoria in 1999 and is currently pursuing a PhD in Information Fusion.