2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

3-D Path Planning and Target Trajectory Prediction for the Oxford Aerial Tracking System

Heiko Helble and Stephen Cameron

***Abstract*— For the Oxford Aerial Tracking System (OATS) we are developing a robot helicopter that can track moving ground objects. Here we describe algorithms for the device to perform path planning and trajectory prediction. The path planner uses superquadratic potential fields and incorporates a height change mechanism that is triggered where necessary and in order to avoid local minima traps. The goal of the trajectory prediction system is to reliably predict the target trajectory during occlusion caused by ground obstacles. For this we use two artificial neural networks in parallel whose retraining is automatically triggered if major changes in the target behaviour pattern are detected. Simulation results for both systems are presented.**

I. INTRODUCTION

There has recently been a resurgence of interest in small autonomous helicopter systems, as the increase in their perceived utility has coincided with the availability of better hardware. In particular, it is now possible to fit enough computing power on-board such small devices to achieve useful mission objectives, whilst the considerable problems of ensuring that the vehicle can fly in a stable fashion can be largely relegated to small and lightweight commercial control boxes called automatic flight control systems (AFCSs). The Oxford Aerial Tracking System (OATS) is designed to explore the design of such a vehicle, whose high-level goal is that of visual tracking. The hardware components of OATS consist of a miniature helicopter, a commercial AFCS, a small on-board computer, radio links, and a camera mounted on a two-axis gimbal. This hardware is designed to be capable of autonomous visual tracking of ground targets, the latter being selected in flight from a ground station. The hardware components and basic tracking algorithms employed in OATS are described in more detail in [1], [2], and at the time of writing this paper the status of this hardware is that the tracking algorithms are being commissioned on the helicopter.

In parallel to the main hardware development, an important part of OATS is the development of high-level mission software in simulation. Our main simulator software is called *HeliSim*, and the framework of *HeliSim* and its use in the development of the visual tracking algorithms was given in [1]. The work presented here introduces two additions to OATS's current capabilities, enhancing the overall autonomy of the system. Experiments with *HeliSim* have shown that automatic path planning is an essential ability for a small UAV operating in known environments. In contrast to fixed-wing drones flying at high altitudes, our UAV helicopter is designed for manoeuvring at low altitudes. Hence, the first addition to OATS discussed in this paper concerns collision avoidance between the robotic helicopter and obstacles found close to the ground level (e.g. trees, buildings, power lines). The second additional feature for OATS addresses the problem of target occlusion during tracking.

In [2] we introduced a re-acquisition strategy suitable for stationary (occluded) targets; here we extend this idea to targets exhibiting complex movement patterns (e.g., aversion). Since such movement patterns created by different types of ground targets exhibit non-repetitive dynamic properties that can not be modelled with a single dynamic model, observation and subsequent *learning* is required for successful trajectory prediction. We therefore allow for training of artificial neural networks, thus confronting this problem in a novel way.

The rest of this paper is organised as follows. Related work is reviewed in §II, and §III explains our approach for path planning, specifically focussing on the capabilities of our robotic platform of performing changes in flight altitude. §IV considers target trajectory prediction, and §V concludes and looks forward to the planned field trials of the OATS system.

II. BACKGROUND

Potential field approaches [3]–[5] are a popular choice for path planners that have to work quickly, and have been adapted for uncertain environments [6], [7], and in particular our work is based on the use of superquadric potential functions [8], [9]. Several papers have explicitly considered the problem of path planning for a UAV. [10], [11] considers the application of evolution-based path planning to the motion of a UAV through a field of obstacles at uncertain locations, and another evolution based approach [12] calculates a curved path with desired characteristics in a 3-D terrain using B-Spline curves. [13] incorporates probabilities into mission planning for UAVs flying through an area with multiple sources of threat, and [14] integrates a probabilistic roadmap planner into the run-time system of their UAV.

[15] deals with a strategy of path planning for a UAV to follow a ground vehicle. The proposed strategy acts on the assumption that the ground vehicle may change its heading and speed while the UAV maintains a fixed air speed. [16] presents a planner for UAVs using a *Voronoi Graph*, and [17], [18] incorporates the kinematic and dynamic properties

Authors' address: Oxford University Computing Laboratory, Parks Road, Oxford OX1 3QD, UK. `heiko.helble` and `stephen.cameron@comlab.ox.ac.uk`

of a small UAV helicopter so that a non-linear control law for manoeuvre execution can be provided. [19] use octrees to effectively extend the search for a UAV path into three dimensions.

[20] presents an approach for tracking vehicles in road traffic scenes using an explicit occlusion reasoning step. A contour based tracker is employed which uses intensity and motion boundaries; the tracking is based on an affine motion model; and the estimation process is decomposed into two linear Kalman filters (one for estimating the motion parameters and one for estimating the vehicle shapes and contours). [21] describes the use of a Kalman filter for visually tracking objects with the possibility of small occlusions, and also an extended Kalman filter (EKF) to model more complex movements.

Trajectory prediction in robot soccer is considered in [22] using linear models and neural networks. Time series prediction is a popular application for neural networks in areas such as traffic forecasting in ATM networks [23] and financial forecasting [24]; [25] includes a discussion of choosing an appropriately sized input window which is particularly relevant to our work.

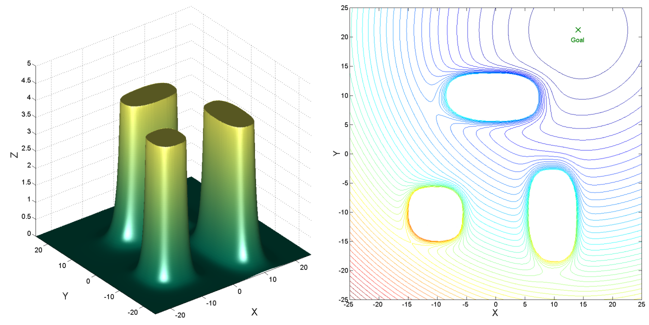
III. PATH PLANNING FOR OATS

A. Prerequisites

We are interested in solutions to the problem of finding a collision free and safe route for a small unmanned aerial vehicle from a given (GPS) coordinate to a second one. In this scenario, we assume the availability of a detailed elevation map of the flight area that contains all major fixed obstacles (e.g. buildings, mountains, power lines). Our goal is to command the UAV to fly to new coordinates by sending it a high level set of instructions via the wireless link. Once a new command is received by the UAV, it should automatically calculate a possible path between its current configuration and the goal configuration. The path consists of a series of roughly equidistant coordinates, which can successively be fed into our existing Automatic Flight Control System (AFCS). The AFCS then compiles these inputs into a set of low-level motion primitives and executes them accordingly. The sequence is complete when the UAV approaches the given goal configuration.

Ideally the UAV should stay at a roughly constant height above the ground in order for its camera to obtain a good view of the object being tracked. However, in certain situations altitude changes are unavoidable: obstacles such as power lines (which are at the same height or higher than the current UAV flight height) can not be simply bypassed by going around or under them. Here, the automatic path planner needs to find a path that takes the UAV *above* the obstacle. Consequently, the UAV first needs to ascend to a certain height above the obstacle in question and then descend to the previous height above ground once the obstacle has been passed.

Finally, as the achievable UAV positioning accuracy during flight relies on GPS (which has limited accuracy) and depends on external disturbances (wind), a well dimensioned safety margin between the UAV and the obstacles is crucial in order to avoid collisions.



(a) 3-D view repulsive potential field (b) Contour total potential field

Fig. 1. Superquadric potential field composition

B. Potential Fields Based on Superquadric Isopotential Contours

Our path planner uses a potential field method which offers the following desirable properties: guaranteed obstacle avoidance while keeping a safety distance; generation of a smooth trajectory; low computational load; and expandable for inclusion of dynamic obstacles. While many different attractive potential functions have been proposed in the literature, the most commonly used attractive potential field takes the form of a quadratic well function:

$$U_{att}(\mathbf{q}) = \frac{1}{2}\,\xi\,\|\mathbf{q}_{Tar} - \mathbf{q}_{Rob}\|^2 \qquad (1)$$

where ξ is a positive scaling factor, $\mathbf{q}_{Tar}$ and $\mathbf{q}_{Rob}$ are the target and robot positions, and $\|\cdot\|$ is the Euclidean distance function. This simple function has the advantages of providing a linear control law with constant gain, and that all potentials are approximately quadratic for small displacements ([8]).

For a repulsive field we use a superquadric [9], which allows us to model all sorts of different shapes. The general superquadratic isopotential contour is:

$$\left[\left(\frac{x}{f_1(x,y,z)}\right)^{2n} + \left(\frac{y}{f_2(x,y,z)}\right)^{2n}\right]^{\frac{2m}{2n}} + \left(\frac{z}{f_3(x,y,z)}\right)^{2m} = 1 \qquad (2)$$

where f_1, f_2 and f_3 are scaling functions, and m and n are exponential parameters. For the polyhedral geometric obstacles used here we follow the refinement of (2) given in [8] to obtain constant functions for f_1, f_2 and f_3, and (also following [8]) we choose m and n to try to reduce the likelihood of unwanted local minima. Since we will use the vertical profile formed by an obstacle's repulsive potential field in §III-D, we set the z value for each obstacle equal to the height of the corresponding obstacle as defined in the elevation map of the flight area.

The overall potential U_{Total} is a linear combination of the attractive and repulsive potentials, with weights chosen to set the relative importance of achieving obstacle avoidance and reaching the goal. Fig. 1 shows how the potential field based on superquadric isopotential contours is composed: Fig. 1(a) shows a 3-D view of the field due to three rectangular obstacles, and Fig. 1(b) shows a contour plot of U_{total} when the centre of the attractive well is located in the top right hand corner of the plot.

C. Plan Generation and Selection

In a traditional potential field method we search for the goal by following the negative gradient of the field; however, local minima in the field can cause this method to fail. Instead, we *sample* the potential in a directed search. Specifically, we compute U_{Total} at N_s equidistant values on a circle of radius r around the current robot position (x, y). For values $r = 1$ and $N_s = 32$ with (x_i, y_i), $i = 1, 2, ..., N_s$ this generates 32 one-step plans where we "predict" one step ahead. The set of plans can be viewed as "the robot is at (x, y), move it to (x_i, y_i)". The plan to be executed is then found by choosing i^* such that:

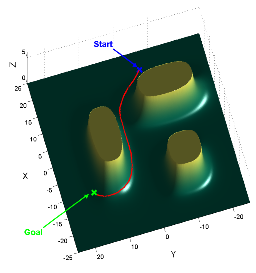
$$U_{Total}(x_{i^*}, y_{i^*}) \leq U_{Total}(x_i, y_i), \quad i = 1, 2, ..., N_s. \tag{3}$$

Thus, we determine the direction $\theta(k)$ which results in minimising U_{Total} defined in (3). In the last step, the robot is commanded to take a step of length λ in the direction $\theta(k)$. This approach does not need analytical gradient information since we do not calculate the gradient of the U_{Total} function explicitly. Note that higher values of N_s are computationally more expensive, but also provide more precise directional commands. (In the context of OATS the cost of this planning is not significant.)

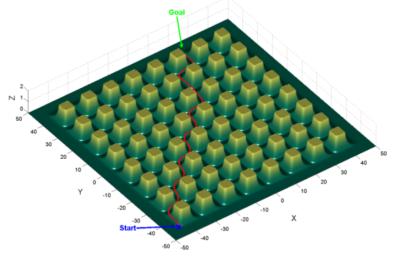
D. Avoiding Local Minima by using the Third Dimension

One limitation of our path planning approach is the existence of local minima for certain obstacle configurations. Examples are "U" shaped convex obstacles or very long obstacles, causing the robot to become trapped on its way towards the goal position. These local minima are the result of the unpredictable shape of the total potential field after the superposition of the attractive and repulsive potential fields. Furthermore, there might not exist a possible path in the plane from the start configuration to the goal configuration, because it is blocked by a number of obstacles.

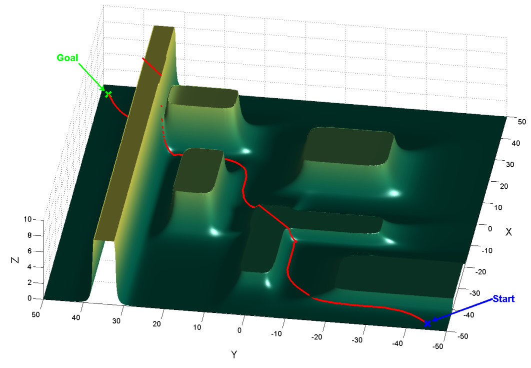
We can overcome these limitations by controlling the altitude of the helicopter. We monitor the progress of the robot during path planning in order to detect situations in which it has become trapped in a local minimum or has been stopped by an obstacle configuration that can not be bypassed because no 2-D path exists. Progress P is measured as the Euclidean distance the robot has travelled during the last m time steps based on the reference position R_{t-m} and the current robot position R_t. If $P < \frac{1}{2}m\lambda$ (where λ is the step size taken at each time step during path planning and m is a natural number) we switch from path planning behaviour (PPB) to obstacle overfly behaviour (OOB). Once OOB is active, we determine which of the obstacles surrounding the current robot position R_t is nearest to it based on the Euclidean metric; we call this obstacle O_n. The height of O_n is then determined from the elevation map and labelled H_o, and the maximum overfly altitude A_{max} for the robot is given by $A_{max} = \mu H_o$, with $\mu > 1$. Typically, a value for



(a) Path between 3 obstacles



(b) Path in obstacle grid



(c) 3-D path planning

Fig. 2. Path planning results for different obstacle configurations

μ is chosen between 1.2 and 2, yielding an obstacle overfly altitude between 120% and 200% relating to H_o.

In the following example of OOB, the robot is commanded to fly towards the goal configuration as the crow flies, following the contour of the repulsive potential field of O_n. Thus, the robot vertically ascends (or descends, respectively) in steps of λ, always keeping the vertical distance of μ times the height given by the repulsive potential field of O_n until A_{max} is reached. The contour following crow line flight towards the goal configuration continues until O_n is overflown and the original flight altitude is restored. At this point, we switch back to PPB until the goal configuration is reached. In case of further local minima or blocking obstacles in the path of the robot, the "PPB $\rightarrow$ OOB $\rightarrow$ PPB" cycle is repeated until the goal has been reached.

E. Results

Path planning results are shown in Fig. 2; the path generated by our planner is represented by a red line connecting the start and goal positions. (The red line consists of individual equidistant data points with $\lambda = 0.1$.) Fig. 2(a) shows an example path that runs between the obstacle configuration from Fig. 1. In Fig. 2(b), the robot has to find its way through a grid of 64 obstacles located in close vicinity to one another. Starting in the bottom corner of the field, it successfully sidles towards the goal position, which is located at the far end of the field. An example of the 3-D path planning capabilities of our implementation is given in Fig. 2(c). The robot starts in the bottom right corner but is stopped by a "V" shaped obstacle, forming a classic local minimum in its path towards the goal. Our path planning algorithm deals with the situation by commanding the robot to ascend to an altitude greater than the blocking obstacle, as described in §III-D.

After this first local minimum is negotiated successfully and the original flight altitude is restored, the robot continues its path. However it is confronted by a second blocking obstacle (the long structure on the left hand side of the figure), which represents a power line in our simulation, and the robot is forced to perform an altitude change once again. The goal position, located on the far left corner, is finally reached successfully and the path planning for this task is complete. (Of course, if the desire for the helicopter to travel at a particular altitude were relaxed then it would be possible to filter out unnecessary height changes in order to save fuel.)

IV. TRAJECTORY PREDICTION FOR OATS

A. Prerequisites

OATS has the aim to autonomously track arbitrary ground objects from within a UAV helicopter. In this section of the paper we describe a novel technique we have developed that allows predicting of the future trajectory of the tracked ground object. The motivation for this research is twofold. Firstly, the tracked target can become temporarily occluded due to obstacles that are in the line of sight between robot and target. Secondly, the tracked target might try to hide from the observer; that is, purposely manoeuvre behind obstacles like buildings and trees so that the observer loses its track on the target. The use of an Extended Kalman Filter was rejected due to the lack of a dynamic model for the task at hand. We rather have to act on the assumption that the target we are trying to track possesses highly non-linear dynamics that do not follow any fixed rules: this stems from the fact that we simply do not know beforehand whether we are going to track a car, pedestrian, ship or tank (say). In addition, we have to assume that we might be dealing with a target which is aware of the airborne observer and is therefore trying to hide from it.

Hence, our solution to this problem is to observe the behaviour pattern of the target and *learn* its dynamics in the process. Artificial neural networks (ANNs) have the ability to be used as an arbitrary function approximation mechanism which learns from observed data [26]; they have been used for tasks such as regression analysis, time series prediction, classification, filtering and clustering.

B. Neural Networks for Target Trajectory Prediction

We are trying to predict the trajectory of a target which is moving in a two dimensional plane, and so we use two independent feed-forward ANNs for the x and y coordinates. (We use these coordinates rather than (r, θ) (say) to avoid the looping that occurs with angular coordinates.) Each of these two identical ANNs are composed of a multi-layer feed-forward back-propagation network which employs 6 neurons in the input layer, 10 neurons in the hidden layer, and 3 neurons in the output layer, with a nonlinear tan-sigmoid transfer function being applied to the units in the hidden and the output layer. This structure was selected based on empirical observations from experiments with several different ANN configurations.

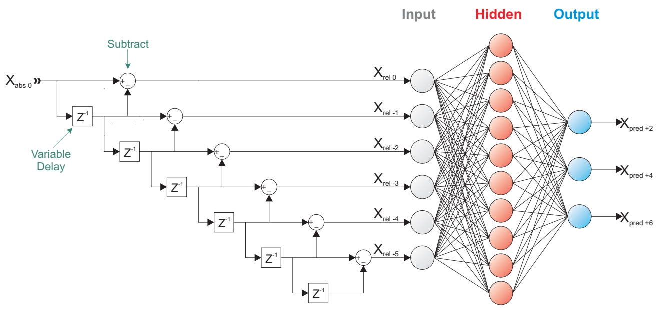


Fig. 3. Artificial neural network for target trajectory prediction

Fig. 3 shows the structure of ANN$_x$ including the symbolic input data processing blocks. On the left, the absolute x coordinate of the tracked target at present time t_0 is fed into the network (denoted by $X_{abs(0)}$ in the figure). In order to gain a set of data points that represent a history of the past six x coordinates, the signal is passed through delay blocks that are aligned in a linear sequence. After each delay, the signals $X_{abs(m)}$, $m = 0, -1, ..., -6$ are tapped and converted into relative signals:

$$X_{rel(n)} = X_{abs(n)} - X_{abs(n-1)}, \qquad n = 0, -1, ..., -5. \quad (4)$$

The advantage of using coordinates which are relative compared to the target position at time t_0 comes from the fact that the network generalises in this way and does not consider absolute values. Hence, the network is able to make a correct prediction for target behaviour patterns it is presented with no matter which absolute coordinates they occurred at. In the following, the six relative signals $X_{rel(n)}$ are connected to the six neurons of the ANN input layer. At each time step the ANN predicts three target coordinates $X_{pred(k)}$, $k = +2, +4, +6$ which are available at the three neurons of the ANN output layer.

In order to predict further into the future than the six time steps given by $X_{pred(+6)}$, an additional time span of past values has to be considered. We achieve this by using variable delay blocks. This has the advantage that the actual ANN structure does not have to be changed and the number of neurons in each layer can remain constant. Consequently, for an integer delay of d we get:

$$\begin{array}{ll} X_{abs(m)}, & m = (0, -1, \ldots, -6)\, d \\ X_{rel(n)} = X_{abs(n)} - X_{abs(n-d)}, & n = (0, -1, \ldots, -5)\, d \\ X_{pred(k)}, & k = (+2, +4, +6)\, d \end{array} \quad (5)$$

Even though the input data the ANN sees for values of $d > 1$ exhibits “gaps”, experimental results suggest that this does not affect its prediction accuracy excessively.

Fig. 4 illustrates for two cases ($d = 1$ and $d = 2$) how the variable delay affects the creation of a set of history coordinates to be fed into the ANN. In both cases, the absolute target coordinate at present time t_0 is represented by a grey shaded field labelled ’Abs. 0’, located in the centre of the timeline. Fields to its left stand for coordinates recorded prior and carry negative indices. Likewise, fields to its right are posterior coordinates carrying positive indices. For $d = 1$, the six relative coordinates are calculated from ’Abs. 0’ to

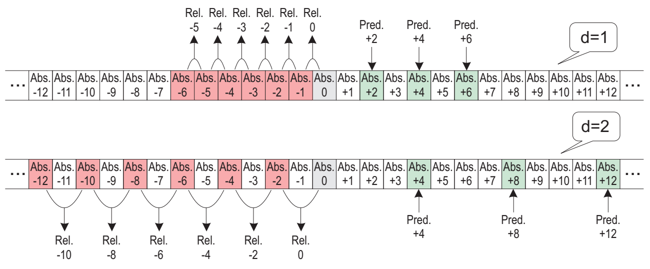


Fig. 4. Timeline of coordinates fed into the ANN for trajectory prediction. Red fields denote prior, grey present, and green posterior coordinates.

'Abs. -6' (fields shaded in red) and the ANN predicts the relative coordinates 'Pred. +2', 'Pred. +4', and 'Pred. +6'. The corresponding absolute prediction coordinates (green shaded fields) are calculated by adding the relative prediction coordinates to the 'Abs. 0' coordinate. The same procedure applies for $d = 2$, but interleaved prior coordinates are used (odd negative indices).

C. Learning Target Behaviour Patterns

In general, there are two different styles of training an ANN; *incremental training* where the weights and biases of the network are updated each time an input is presented to the network, and *batch training* where the weights and biases are only updated after all the inputs are presented (offline). At first glance, the possibility of learning target behaviour patterns incrementally seems more desirable since inputs and targets can be presented as sequences to the ANN online during target tracking. However, this approach has some profound disadvantages compared to the batch learning method:

- It is not possible to compute the objective function for any fixed set of weights. Consequently, it is hard to determine if progress is being made during training;
- Neither can the objective function be computed on a training set nor the error function on a validation set, since these data sets are not stored;
- Cross-validation and bootstrapping can not be used to estimate generalisation errors;
- The objective function can not be computed to any desired precision.

Hence, incremental learning is generally more difficult and unreliable than batch learning. Our experiments with trajectory prediction showed furthermore that progress is very slow when incremental learning is employed compared to using the batch learning method.

The initial weights for the ANN described in §IV-B are determined by training the network on a set of prerecorded target tracking data, which is split into training, test, and validation subsets for this purpose. We then use the Levenberg-Marquardt (LM) algorithm for training until a satisfactory network performance is achieved (MSE $< 10^{-5}$). This algorithm represents the current state-of-the-art learning method for training a moderate-sized feed-forward ANN due to its robustness and its capability to find a solution in many cases, even if it starts off very far from the final minima.

We constantly monitor the trajectory prediction performance during target tracking by computing a running average of the prediction errors. If it falls below a certain level (e.g. because the target changed its behaviour pattern), an ANN retraining sequence is initiated automatically. For this purpose, a variable sized target coordinate buffer is kept up-to-date at all times. When retraining the ANNs becomes necessary, the data in the buffer is split into training, test, and validation subsets. Subsequently the LM algorithm is applied to these subsets and the new ANN weights are then used for prediction. For common buffer sizes, this procedure is executed on the target platform in near real-time ($t < 500$ms).

A noteworthy detail in this context is the correlation between target coordinate buffer size and the maximal number of time steps we are trying to predict into the future (which depends on the value chosen for d). For the time t_{fill} it takes to fill the coordinate buffer we get:

$$t_{fill} = t_{step}\, N_{buff}\, d \tag{6}$$

where t_{step} is the duration of each time step in milliseconds, N_{buff} is the size of the coordinate buffer, and d is the delay in time steps used in the variable delay blocks of the ANN coordinate pre-processing stage. The buffer size is furthermore the factor which limits the duration of observed behaviour patterns t_{patt} to which the ANN is capable of adapting itself to. Thus, it is only possible to exactly learn patterns whose repetition frequency is the same or less than the size of the used coordinate buffer. Consequently, the following condition has to be met: $t_{patt} \leq t_{fill}$. If this is not the case, the ANN prediction performance decreases rapidly.

D. Results

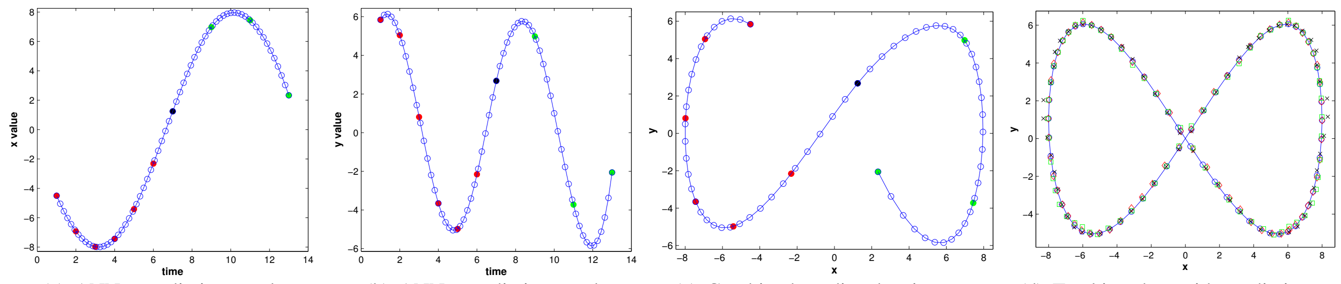
The trajectory prediction algorithm introduced in this section has been implemented in Matlab/Simulink and tested within the UAV helicopter simulation framework *HeliSim* we presented in [1].

All results shown in Fig. 5 were obtained from real-time experiments with our ANN implementation using a delay value of $d = 5$ time steps. For this value of d, (5) gives us the following three arithmetic series:

$$\begin{aligned} m &= 0, -5, \ldots, -30 \\ n &= 0, -5, \ldots, -25 \\ k &= +10, +20, +30 \end{aligned} \tag{7}$$

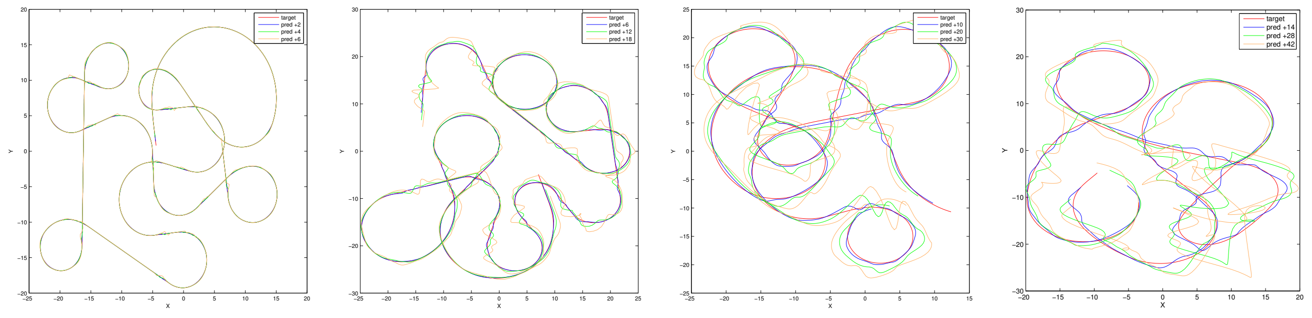
where k denotes that we are predicting three target coordinates that lie in the future, the most distant one being 30 time steps ahead of the current target position. To do so, we use six relative coordinates that lie in the past (denoted by n), which are gained from seven absolute coordinates (denoted by m) reaching back a maximum of 30 time steps. In Fig. 5(a) to 5(c) red dots indicate prior, black dots present, and green dots posterior coordinates.

Fig. 5(a) and Fig. 5(b) are plots generated from the outputs of ANN$_x$ and ANN$_y$, respectively. Even though HeliSim constantly predicts three posterior coordinates during runtime, the calculation of intermediate coordinates between



(a) ANN$_x$ prediction results (b) ANN$_y$ prediction results (c) Combined predicted trajectory (d) Tracking data with prediction

Fig. 5. Target trajectory prediction results for $d = 5$. Blue circles (○) connected by a solid blue line mark observed target coordinates; and red diamonds (♢), green squares (□) and black crosses (×) denote the first, second and third prediction control points respectively.



(a) Time delay $d = 1$. (b) Time delay $d = 3$. (c) Time delay $d = 5$. (d) Time delay $d = 7$.

Fig. 6. Target trajectory prediction results for complex random trajectories as produced by a manoeuvring car. The figures show an overlay of the actual (observed) trajectory and the three predicted trajectories $pred(k), k = (+2, +4, +6)d$.

these three control coordinates is only triggered when the target becomes occluded. For filling out the gaps between the control coordinates, interpolating cubic splines are employed [27]. By choosing the spline sample density in such an extent that the number of interpolated values equals the number of missing data points between each pair of control coordinates, a seamless trajectory is obtained that possesses a coordinate for each time step. The cubic spline is shown as a blue line in Fig. 5(a) and Fig. 5(b); the interpolated data points are denoted by blue circles.

The required 2-D trajectory can be obtained by combining the results gained by ANN$_x$ and ANN$_y$ (Fig. 5(c)); note that the plots shown in Fig. 5(a) and Fig. 5(b) utilise *time* on the x-axis and *position* on the y-axis, whereas the 2-D plot shown in 5(c) displays the target's x position on the x-axis and its y position on the y-axis. Fig. 5(d) shows a plot of the original tracking data together with the predicted coordinates. The target describes a "figure-of-eight" in the plane. To make it harder for the ANNs to learn and subsequently predict this trajectory, changes in velocity were added to the target motion. This is identifiable from the varying distances between the dots in the plot representing the target's position after each time step. The shape of the figure-of-eight performed by the simulated target in the experiment was furthermore altered by compressing it in the x direction.

The result plots shown in Fig. 5 demonstrate how well our prediction algorithm performs in this task. The achieved error values of the calculated running averages for the three prediction control points were:

$$E_{2d} = 0.02, \quad E_{4d} = 0.05, \quad E_{6d} = 0.10. \qquad (8)$$

The size of the windows used for calculating the running averages in (8) were thereby set equal to the coordinate buffer size defined by N_{buff}.

The second trajectory prediction experiment (Fig. 6) included in this paper describes the simulation of a manoeuvring car (controlled by a joystick input device in real-time) as the tracking target. Apart from frequent and random changes in direction, this also involves variations in velocity and acceleration, making for a very complex overall movement pattern. The results shown in Fig. 6(a)–6(d) use increasing delay values ($d = 1, 3, 5, 7$), which yield increasing posterior prediction steps. Table I gives an overview on how the choice of d affects the average errors obtained from the experiments shown in Fig. 6. Naturally, this gives rise to the fact that the further we are trying to predict into the future, the more the prediction accuracy deteriorates. However, we can confirm that even a delay value of $d = 7$, as used in the experiment shown in Fig. 6(d), still produces valuable trajectory prediction information. This is largely due to the fact that the OATS helicopter only requires a rough estimate of the target coordinates in order to reacquire the target (a target search strategy is triggered during prolonged times of occlusion, see [2]).

We learnt from our experiments that the size of the coordinate buffer used for recording target behaviour patterns plays a crucial role for the obtainable trajectory prediction accuracy. As a rule of thumb, the buffer should at least accommodate the number of coordinates necessary to describe one cycle of the observed target pattern to be learnt during batch training. For more complex patterns that do not show any repetitive behaviour, it is found that an increased buffer

TABLE I
AVERAGE TRACKING PREDICTION ERROR VALUES FOR THE EXPERIMENTS SHOWN IN FIG. 6

Delay	Post. Prediction	E_{2d}	E_{4d}	E_{6d}
d=1	k=+2,+4,+6	0.062	0.125	0.187
d=3	k=+6,+12,+18	1.129	2.231	3.269
d=5	k=+10,+20,+30	1.905	3.592	4.861
d=7	k=+14,+28,+42	2.624	4.811	6.062

size usually yields more accurate prediction results.

V. CONCLUSIONS AND FUTURE WORKS

We have presented two novel techniques suitable for small UAVs with visual target tracking capabilities. The results we obtained from experiments with the *HeliSim* framework suggest potential enhancements to OATS in its current configuration. By combining conventional path planning techniques based on potential fields with our robot's ability to perform altitude changes, we successfully showed how local minima and blocking obstacles (that can not be circumvented in the plane) can be bypassed without the risk of collisions occurring in the process. The trajectory prediction method demonstrates how artificial neural networks can be employed for learning target movement patterns that stem from visual object tracking. The results suggest that two feed-forward ANNs run in parallel are capable of adapting themselves (via batch learning) well enough so that complex movement patterns can be predicted with high accuracy.

Future work on OATS will be concerned with implementing the techniques brought forward in this paper into the UAV system software. We furthermore plan on carrying out field experiments with our robotic helicopter. These experiments will involve several different moving ground targets for tracking. In addition, OATS will be deployed in cluttered environments (as found in urban regions), thus heavy use of the automatic path planner will be inevitable.

REFERENCES

[1] H. Helble and S. Cameron, "OATS: Oxford aerial tracking system," in *Towards Autonomous Robotic Systems*, Surrey University, Guildford, UK, September 2006, pp. 72–77. [Online]. Available: http://tinyurl.com/2jevsz

[2] H. Helble and S. Cameron, "OATS: The Oxford aerial tracking system," *Rob. and Aut. Systems*, 2007, accepted.

[3] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[4] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion*. MIT Press, 2005.

[5] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2002, pp. 2383–2388. [Online]. Available: http://tinyurl.com/frv4x

[6] T. Hague and S. Cameron, "Motion planning for non-holonomic industrial robot vehicles," in *Proc. IROS-91*. Osaka: IEEE, Nov. 1991, pp. 1275–1280.

[7] J. Barraquand and J. Latombe, "Robot motion planning: A distributed representation approach," *Int. J. Rob. Research*, vol. 10, no. 6, pp. 628–649, 1991.

[8] R. Volpe and P. Khosla, "Manipulator control with superquadric artificial potential functions," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, no. 6, pp. 1423–1436, Nov/Dec 1990. [Online]. Available: http://tinyurl.com/jabv6

[9] H. Löffelmann and E. Gröller, "Parameterizing superquadrics," in *Third International Conference in Central Europe in Computer Graphics and Visualisation*, vol. 3, Plzen, Czech Republic, February 1995, pp. 162–172. [Online]. Available: http://tinyurl.com/jrf4c

[10] D. Rathbun, S. Kragelund, A. Pongpunwattana, and B. Capozzi, "An evolution based path planning algorithm for autonomous motion of a UAV through uncertain environments," in *Digital Avionics Systems Conference*, vol. 2, 2002, pp. 8D2:1–12. [Online]. Available: http://tinyurl.com/kkek3

[11] D. Rathbun and B. Capozzi, "Evolutionary approaches to path planning through uncertain environments," in *1st Unmanned Aerospace Vehicles, Systems, Technologies, and Operations Conference and Workshop*. Portsmouth, VA, USA: American Institute of Aeronautics and Astronautics, May 2002, pp. 20–23. [Online]. Available: http://tinyurl.com/f7dhp

[12] I. K. Nikolos, K. P. Valavanis, N. C. Tsourveloudis, and A. N. Kostaras, "Evolutionary algorithm based offline-online path planner for UAV navigation," *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, vol. 33, pp. 898–912, December 2003. [Online]. Available: http://tinyurl.com/jmm2c

[13] A. Dogan, "Probabilistic path planning for UAVs," in *IEEE International Symposium on Intelligent Control*, 2003, pp. 608–613. [Online]. Available: http://tinyurl.com/fupmh

[14] P. O. Pettersson and P. Doherty, "Probabilistic roadmap based path planning for an autonomous unmanned aerial vehicle," in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2004, appeared in: Workshop on Connecting Planning and Theory with Practice. [Online]. Available: http://tinyurl.com/f4fay

[15] J. Lee, R. Huang, A. Vaughn, X. Xiao, J. K. Hedrick, M. Zennaro, and R. Sengupta, "Strategies of path-planning for a UAV to track a ground vehicle," in *Autonomous Intelligent Networks and Systems (AINS)*, 2003. [Online]. Available: http://tinyurl.com/zofyj

[16] D. B. Kingston, "Implementation issues of real-time trajectory generation on small UAVs," Ph.D. dissertation, Brigham Young University, April 2004. [Online]. Available: http://tinyurl.com/l7926

[17] E. Frazzoli, M. A. Dahleh, and E. Feron, "Robust hybrid control for autonomous vehicle motion planning," in *39th IEEE Conference on Decision and Control*, vol. 1, Sydney, NSW, Australia, December 2000, pp. 821–826. [Online]. Available: http://tinyurl.com/krnx7

[18] E. Frazzoli, M. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," in *American Control Conference*, vol. 1, Arlington, VA, USA, June 2001, pp. 43–49. [Online]. Available: http://tinyurl.com/eeehz

[19] A. Dattasharma and S. S. Keerthi, "An augmented Voronoi roadmap for 3D translational motion planning for a convex polyhedron moving amidst convex polyhedral obstacles," *Theoretical Computer Science*, vol. 140, no. 2, pp. 205–230, April 1995.

[20] D. Koller, J. Weber, and J. Malik, "Robust multiple car tracking with occlusion reasoning," University of California at Berkeley, Tech. Rep., January 1994. [Online]. Available: http://tinyurl.com/o5h4r

[21] E. Cuevas, D. Zaldivar, and R. Rojas, "Kalman filter for vision tracking," Freie Universität Berlin, Tech. Rep., August 2005, Technical Report B 05-12. [Online]. Available: http://tinyurl.com/oawml

[22] S. Behnke, A. Egorova, A. Gloye, R. Rojas, and M. Simon, *Predicting Away Robot Control Latency*. Padua, Italy: Springer, June 2003, vol. 3020, pp. 712–719, robot world cup soccer and rescue competitions and conferences No7. [Online]. Available: http://tinyurl.com/z7lgf

[23] T. Edwards, D. Tansley, N. Davey, and R. Frank, "Traffic trends analysis using neural networks," in *International Workshop on Applications of Neural Networks to Telecommuncations*, vol. 3, 1997, pp. 157–164. [Online]. Available: http://tinyurl.com/g54wk

[24] C. L. Giles, S. Lawrence, and A. C. Tsoi, "Noisy time series prediction using a recurrent neural network and grammatical inference," *Machine Learning*, vol. 44, no. 1-2, pp. 161–183, July 2001. [Online]. Available: http://tinyurl.com/jseqm

[25] R. Frank, N. Davey, and S. Hunt, "Time series prediction and neural networks," *Journal of Intelligent and Robotic Systems*, vol. 31, no. 1-3, pp. 91–103, May 2001. [Online]. Available: http://tinyurl.com/genuq

[26] J. Hertz, R. Palmer, and A. Krogh, *Introduction to the Theory of Neural Computation*. Perseus Books, 1990, ISBN 0-201-51560-1.

[27] C. D. Boor, *A Practical Guide to Splines*, ser. Applied Mathematical Sciences. New York: Springer, 1978, vol. 27.