Predicting the movements of robot teams using generative models

Simon Butler and Yiannis Demiris

Abstract When a robot plans its actions within an environment containing multiple robots, it is often necessary to take into account the actions and movements of the other robots to either avoid, counter, or cooperate with them, depending on the scenario. Our predictive system is based on the biologically-inspired, simulation-theoretic approach that uses internal generative models in single-robot applications. Here, we move beyond the single-robot case to illustrate how these generative models can predict the movements of the opponent's robots, when applied to an adversarial scenario involving two robot teams. The system is able to recognise whether the robots are attacking or defending, and the formation they are moving in. It can then predict their future movements based on the recognised model. The results confirm that the speed of recognition and the accuracy of prediction depend on how well the models match the robots' observed behaviour.

1 Introduction

There are many situations where it is beneficial to model and predict the behaviour of other robots in the environment. For example: if each robot is operating independently then it is important to avoid collisions by predicting each robot's trajectory; in an adversarial setting, for example RoboCup soccer (Kitano et al, 1998), it is crucial to be able to neutralise and counter the opponents' actions; and in environments where the robots have a common goal, co-operation is usually necessary to help solve the task, or to avoid duplicating work.

Simon Butler
EEE Dept., Imperial College London, UK, e-mail: simon.butler02@imperial.ac.uk

Yiannis Demiris
e-mail: y.demiris@imperial.ac.uk

The obvious way to obtain the objectives of the other robots is to simply communicate with them and ask what they intend to do. However in many situations this is infeasible, because an opponent will be unwilling to hand over its plans and tactics, or the robots may not share a common language so they will not be able to understand the request, or even when communication fails it is important to have a fallback mechanism to continue the task. Therefore, in these cases, the intentions of the robots must be predicted based purely on observation.

In this work, an observation-based predictive system will be applied to an adversarial, real-time engagement between two teams. This domain is time-sensitive—in crisis situations the decision making time is crucial; there are numerous autonomous robots (which could also represent humans, vehicles or aircraft); it has a rich structure that allows intention recognition at different levels of a command hierarchy; the environment, including team objective and allocation, is dynamic and constantly under review.

2 Background

It can be seen that recognising and predicting the goals and intentions of teams of robots is essentially a problem of model matching. Each robot collects information of the state of the environment and this data can be acted on by taking either a *descriptive* or *generative* approach.

The descriptive approach uses the extraction of low-level features to match against pre-existing representations. For example, in the multi-agent domain, Devaney and Ram (1998) analyse spatio-temporal traces for coordinated motion, whether moving apart or together, however this may not scale easily to more complex plans. Using a similar technique Sukthankar and Sycara (2006) create spatio-temporal models to encode group behaviour, then match traces to these models. Although, again, more complex behaviours may not be able to be described using spatio-temporal models alone. In the domain of a soccer match Beetz and Kirchlechner (2005) try to extract explicit rules from training data. However, their approach could be prone to over-fitting if only a limited amount of training data is available, hence making it difficult to apply the results to classify new situations.

Within the generative approach a set of latent (hidden) variables are introduced that encode the causes that *can produce* the observed data. Using these variables for a recognition and prediction task involves modifying the parameters of the generating process until the generated data can be favourably compared against the observed data. This approach has been used in the single-agent domain by using graphical models, such as Hidden Markov Models (HMMs) (Bui et al, 2002), or by using internal models to perform prediction through simulation (Demiris and Khadhouri, 2006; Demiris, 2007).

The simulation-based approach is linked to the "simulation theory" perspective on human cognition that states that an observer's cognitive and motor structures have a duel role: both acting overtly, and simulating and imagining actions and their consequences (Hesslow, 2002). A number of architectures have been inspired by this. One such approach (dubbed HAMMER) uses a hierarchical system of inverse (plan) and forward (predictor) models, with each model pair (collectively known as internal models) being configured to a particular action (Demiris and Khadhouri, 2006). The system finds the matching action by using the model that has the lowest error between the predicted state from each forward model and the observed state. This architecture has the advantage that it works well in on-line situations because it will always produce the most closely matching model at any point, rather than other approaches which make a selection then backtrack if it fails. A review by Demiris (2007) shows the applicability of the use of these internal models to multi-agent systems. Related work by Takács et al (2007) on segmenting spatio-temporal traces of multiple agents paves the way for applying inverse models to distinct groups of agents.

3 Objectives

The goal of this work is to investigate the simulationist generative approach in multi-agent systems: whether an opponent's intentions can be inferred by creating plausible plans (or hypotheses) that, as time progresses, are simulated to find the closest match with the observed behaviour, i.e., prediction through the generation of multiple competing plans. This is based on the assumption that the objective that was used to create the best-matching plan correlates with the intent of the opponent.

In the chosen scenario for this work, two opposing teams of heterogeneous robots are fully observable within a simulated, large-scale, outdoor environment, and each opponent may have many possible objectives at different levels of abstraction. Such objectives vary from "eliminate the opponents' bases" to "defend areas of strategic importance" at the high level, and at the low level the objectives are of the form "attack target with a wedge formation" or "surround target".

It is assumed that models exist for generating plans for the commander's own team, and therefore hypotheses for the opponent's team are created by applying these models to the opponent's robots, in other words, taking a "what would I do in that situation" approach (Demiris, 2007; Demiris and Khadhouri, 2006). Additionally, the use of generative models has the advantage that they can be used not only to recognise and predict, but also to produce movement.

4 System Architecture

Starting at the lowest level, taking a generative approach to recognition and prediction of plans requires a method to generate and evaluate certain primitive actions that a team of robots can perform. The biologically-inspired HAMMER architecture provides a starting point for the system (see figure 1). It is comprised of three main

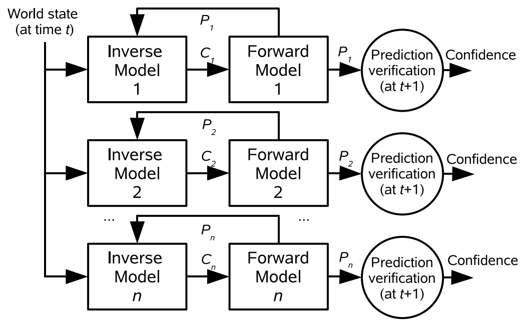

Fig. 1 The HAMMER architecture. Multiple inverse models receive the world state and suggest possible commands (C_1-C_n), which are formed into predictions of the next world state by the corresponding forward model (P_1-P_n). These predictions are verified on the next time step, resulting in a set of confidence values.

components: the **inverse models** (plan generators), the **forward models** (predictors) and the **evaluator** (Demiris and Khadhouri, 2006; Demiris, 2007).

An inverse model takes the current world state and an assigned goal, and outputs the required waypoints and robot parameters that, under some specified constraints, it believes are necessary for each robot to achieve the goal. Each parallel instance of a plan is paired with an instance of the forward model that provides an estimate of the events that will occur if the generated plan is followed. At each time step this estimate is returned to the inverse model to tune any parameters of the actions to achieve the desired goal.

To determine which of these inverse/forward-model pairs most accurately describes the events that are occurring, periodically the output of each forward model is compared with the actual world state. These comparisons result in confidence values that behave as an indicator of how closely the observed events match each particular prediction, and they are subsequently accumulated over time until such a point that one model pair achieves a clear separation from the others. This model can then be simulated further into the future to provide a prediction of upcoming events.

5 Implementation

The implementation of the system is divided into three main sections (as shown in figure 2): the simulators that host the human-controlled teams of robots; the simulators running in a slave mode, executing the internal models; and the evaluator that bridges the two, sending the state of the human-controlled world to the slave simulators and subsequently gathering and analysing their predictions.

5.1 Simulator

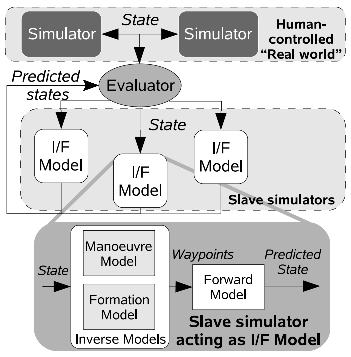

Fig. 2 System implementation. This shows the human-controlled simulator instances at the top of the diagram, feeding the system state into the evaluator, which sends it to each inverse/forward-model pair and receives the corresponding predicted state. These states are then compared to get a confidence for each inverse model.

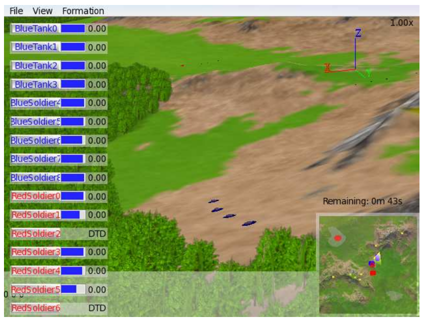

Fig. 3 Simulator screenshot. This shows the interface for the human-controlled teams. The status of each robot is shown down the left, and there is a minimap showing the locations of the robots in the bottom right.

When robots interact with each other, they can affect the state of the environment and of themselves, for example a tackle in RoboCup or the destruction of a vehicle in a military scenario. Hence, for the implementation of our architecture, we obtain the "real-world" state from humans commanding teams of robots within a realistic, distributed, computer-simulated environment. The engine of the simulator is based on Delta3D (Darken et al, 2005), which is an open-source project to integrate various software libraries, such as Open Scene Graph (OSG), Open Dynamics Engine (ODE), Character Animation Library 3D (Cal3D), Game Networking Engine (GNE), etc., into a coherent platform for simulation and games.

The 3D engine (OSG) was used to model a large outdoor terrain (see the screenshot in figure 3), with the height and other features (texture, trees, buildings, etc) of the terrain being displayed based on 2D feature maps. The physics engine (ODE) was used to accurately model the movement of the various agents and vehicles, with additional control of their aiming and firing mechanisms. The noise introduced by the physics engine provides a stochastic element to the outcome of scenarios.

When using the simulator, the human commanders are responsible for choosing goal positions for their robots, so they are free to perform manoeuvres and formations as they see fit.

5.2 Distributed Computing Infrastructure

A peer-to-peer approach was taken to allow each instance to simulate the local commander's robots (as can be seen at the top of figure 2). Positional events, and other events such as projectile detonations, are sent on each frame of the simulation over the network to the opponent, and representations of those events are shown on the opponent's instance. For example, a 3D mesh of an opponent is moved to the position that was computed on the opponent's instance.

Additionally, a distributed, network-based, approach was used to implement the predictive component of the system, which alleviates some of the computational burden of the execution of many models in parallel. This means that each instance of the simulator can be run on a different PC, or many simulators run on a multicored PC.

A separate program on the network acts as an *evaluator*, sitting between the "real world" simulation and the slave client simulators that are acting as inverse/forward-model pairs. Detailed state information is obtained from the world simulation periodically. This state is used to initialise the slave clients and commands are sent to activate the desired inverse model. The simulation is run at a faster than real-time and the predictions fed back to the evaluator. Due to the faster simulation, the result of many different commands can be predicted before needing to be compared with the actual state.

5.3 Evaluation Process

To analyse each inverse/forward-model pair, the evaluator calculates the normalised vectors of the movement from the robot's previous position to both the predicted the actual positions. The dot product of these vectors is then taken and scaled by the shortest vector, as a proportion of the longest vector to give the confidence of that prediction. This metric has the desired characteristics that if the robot moves towards or away from the predicted position then the confidence approaches 1 or -1 respectively, or if it moves perpendicular to the predicted position then the confidence is zero. An additional condition was added so that if the magnitudes of both vectors are less than one metre then the confidence is 1, regardless of heading. This is to reduce errors from the case where the robot drifts or slides slightly and therefore requires a dead-zone with a radius of 1m, within which the robot is counted as stationary. Therefore the confidence, c, can be expressed as

$$c = \begin{cases} 1 & \text{if } |\mathbf{a}| < 1 \text{ and } |\mathbf{p}| < 1, \\ \hat{\mathbf{a}} \cdot \hat{\mathbf{p}} \frac{\min(|\mathbf{a}|,|\mathbf{p}|)}{\max(|\mathbf{a}|,|\mathbf{p}|)} & \text{otherwise.} \end{cases}$$

where $\mathbf{a}$ is the vector from the start position to the *actual* position and $\mathbf{p}$ is the vector from the start position to the *predicted* position, and $\hat{\mathbf{a}}$ and $\hat{\mathbf{p}}$ are the normalised vectors.

6 Experiments & Results

The experimental setup consists of two human commanders each running an instance of the simulator and controlling their (red or blue) team. The state of each team is sent over the network to each instance, and also to the *evaluator*. In this case the predictive system was applied against the blue team's robots (the *opponent*).

To test the feasibility of the generative approach, several inverse models were created. Each robot can have both a *manoeuvre* model and a *formation* model applied to it. The *manoeuvre* inverse models that are available for execution are either *attack* or *defend*, these are defined as follows:

Attack The robot finds the nearest enemy robot and moves towards it until it is in "weapons" range and has clear line-of-sight. At this point it stops and "fires at the enemy".

Defend The robot finds the nearest enemy robot, and if it has clear line-of-sight then the robot moves in the opposite direction until hidden from its sight.

The *formation* inverse models are either *wedge*, *circle*, *line* or *column* and are shown in figure 4. Each formation is described by a weighted graph (as defined by Ji and Egerstedt (2007)), so each robot must keep a distance (scaled by the weight) from their neighbours and this distance is assumed to be a constant 50m for this experiment. Therefore their target position is a linear combination of the offsets to each of the positions required to maintain the distance to each of the connected neighbours. Each formation requires a robot to act as the leader and this was implemented by using a heuristic measure to select the robot that is nearest to the target position. This then allows the other positions to be assigned by nearest neighbour, starting with the leader.

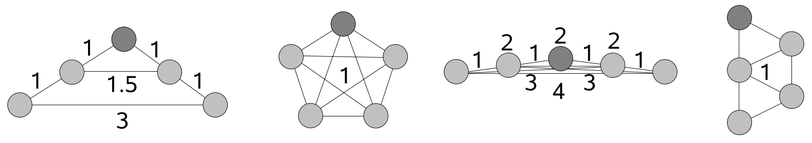

Fig. 4 Weighted graphs describing each formation. From left to right: *wedge*, *circle*, *line* and *column*. The numbers indicate the weight of the connection, higher numbers mean increasing repulsive forces. Those with just one weight specified indicates that all the weights take this value. The darker grey circle indicates the robot that acts as the leader of the formation.

The evaluator launches two instances of the simulator running in a slave mode, and sets a queue of models to simulate against the opponent's blue robots. Due to our approach, the manoeuvre and formation models cannot be treated independently, therefore all combinations must be evaluated. This is because the specific movements of the robots in formation depend on the the route determined by the manoeuvre model. For example, features of the terrain affect the speed of each robot, and any local object avoidance will cause the formation to be disrupted. However, these disruptions can be accounted for by simulating the formation along the route determined by the manoeuvre model. Therefore on one instance the *attack* model with each of the formations is simulated and on the other the *defend* model is used, again with each of the formations.

For ease of analysis, there are three robots on the blue team and one robot on the red team. Each slave simulator is initialised with the world state then run for a pre-specified duration (five seconds) at an increased time scale (4x) and the positions of the robots are returned to the controller within two seconds. The instance is then reset and the process repeated with the next model in the queue, upon the receipt of the next world state. Therefore a result for each model is obtained every 8 seconds.

As can be seen from the trace of each robot's positions for this experiment (shown in figure 5), the red team's objective is to at first to stay hidden behind a ridge, then once the opponent's robots are close enough, move over the ridge and attack. The blue team's robots move together in an attacking wedge formation, towards the hidden red robot. When they move closer to the red robot, they move into a line formation (at around timestep 60). Then when the red robot comes into range they stop and fire at it, until it is destroyed.

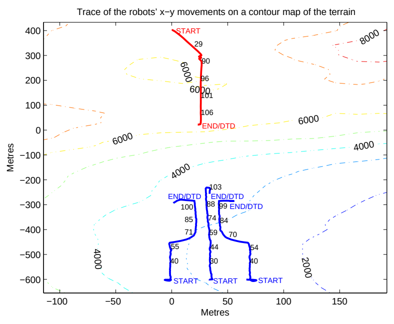

Fig. 5 Unit movements. A contour map (dashed lines), and a trace (thick lines) of the movements of blue robots (starting at the bottom, in a valley), and the red robot (starting at the top, on a ridge). The numbers on the traces represents the time step at that point, and the numbers on the contours represent the height of the terrain.

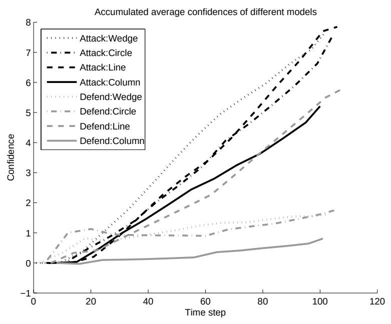

Fig. 6 Accumulated confidences of each model Showing the *attack* and *defend* inverse models, and the formations *wedge*, *circle*, *line* and *column* applied to each. The change in formation at timestep 60 is not clear.

The overall confidence of each combination of inverse models is calculated by averaging the confidence of each individual robot within the team, repeated for each simulator instance. The accumulated confidence values are shown in figure 6. From this we can see the separation between the attacking or defending models, but the change in formation that occurs at around timestep 60 is not clear. This is because the averaging of the confidence takes place over all the robots in the formation, however, the leader of the formation will always have the same confidence regardless of the formation being performed. Therefore we can separate the inverse models by

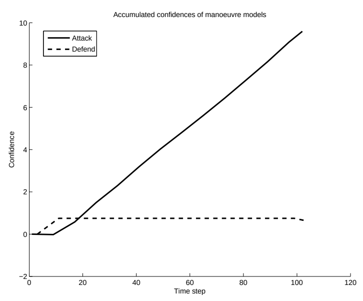

Fig. 7 Accumulated confidences for attack or defend models. Using the average confidence of the leader, a clear separation can be seen between attack and defend.

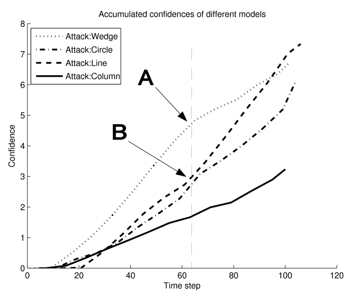

Fig. 8 Accumulated confidences for formation models. Using the average confidence of the formation excluding the leader, the change from wedge to line formation can be seen at points A and B respectively.

averaging the confidence values of the leader in all the formations for each *manoeuvre* model (shown in figure 7) and by averaging the confidences of the remaining robots each of the formations (i.e., excluding the leader) that apply to the winning *manoeuvre* model (shown in figure 8). Now the change in formation that occurs around timestep 60 can clearly be seen by the slowing of the confidence of the wedge formation, and the gaining in confidence of the line formation.

7 Conclusions & Future Work

The work reported here constitutes a first study into the "simulation theory" approach to intention recognition when applied to the multirobot domain. Our results indicate that such approach holds good promise as it is able to recognise and predict whether robots are attacking or hiding from each other, and the formation that they are following in a simulated adversarial scenario. The system is also able to predict the future positions of the robots, assuming the same models hold.

It is still an open problem of how to best combine the confidences of each individual robot into an overall confidence for the inverse model. Immediate extensions include the parameterisation of the inverse models so that if, for example, a robot is moving slowly due to damage, the speed of the robot set by the inverse model can be tuned to match. This tuning can be based on the feedback of the confidence of the previous prediction to the inverse model.

A drawback of this approach is the high computational cost required to evaluate the inverse/forward-model pairs for each possible hypothesis. There are several solutions to this, such as, assigning preconditions to each of the models so they are only executed under certain circumstances, or to briefly run each model and to rank them based on their confidence to find the most promising ones.

Longer-term predictions could be produced by inverse models that do not depend directly on the world state, but combine the confidence of the low-level inverse models (e.g. formations or manoeuvres) with higher-level statistics (e.g. robot losses or speed of movement), to generate predictions that have a lower spatial and temporal resolution. For example, a high-level inverse model such as “defend base” would predict that the movement of robots is low, and that they are highly concentrated, with the centre of mass being in the base, and would be reinforced if the *defend* inverse model has a high confidence. This can lead to a system that is useful when using the simulator to alert the commander when an attack is imminent, or to infer the high-level intentions of the opponent, so the commander can better form an effective strategy.

We are currently persuing an extension to this system that assumes that the environment is only partially observable, for example through the implementation of the architecture on our team of outdoor P3-ATs operating as a multi-robot system. This makes the deployment of sensors to detect the position of the opponent an important part of the strategy. This can be done by making assumptions on the locations of the opponent and running the inverse/forward models, then ranking the results by the threat posed by each prediction, then deploying the sensors to verify the assumptions, covering the most risky models.

Acknowledgements We would like to thank Bálint Takács, Tom Carlson, Murilo Fernandes Martins and Paschalis Veskos for their comments.

References

Beetz M, Kirchlechner B (2005) Computerized real-time analysis of football games. IEEE pervasive computing 4(3)

Bui H, Venkatesh S, West G (2002) Policy recognition in the abstract hidden markov models. Journal of Artificial Intelligence Research 17:451–499

Darken R, Mcdowell P, Johnson E (2005) The Delta3D open source game engine. IEEE computer graphics and applications 25(3)

Demiris Y (2007) Prediction of intent in robotics and multi-agent systems. Cognitive Processing 8(3):151–158

Demiris Y, Khadhouri B (2006) Hierarchical attentive multiple models for execution and recognition of actions. Robotics and autonomous systems 54(5)

Devaney M, Ram A (1998) Needles in a haystack: Plan recognition in large spatial domains involving multiple agents. In: National Conference on Artificial Intelligence

Hesslow G (2002) Conscious thought as simulation of behaviour and perception. Trends in Cognitive Sciences 6(6):242–247

Ji, Egerstedt M (2007) Distributed coordination control of multiagent systems while preserving connectedness. IEEE Transactions on Robotics 23(4):693–703

Kitano H, Asada M, Kuniyoshi Y, Noda I, Osawai E, Matsubara H (1998) Robocup: A challenge problem for AI and robotics. RoboCup-97: Robot Soccer World Cup I pp 1–19

Sukthankar G, Sycara K (2006) Robust recognition of physical team behaviors using spatio-temporal models. In: AAMAS ’06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, ACM, New York, NY, USA, pp 638–645

Takács B, Butler S, Demiris Y (2007) Multi-agent behaviour segmentation via spectral clustering. In: Proceedings of the AAAI-2007, PAIR Workshop, AAAI Press, pp 74–81