`UT^2`: Human-like Behavior via Neuroevolution of Combat Behavior and Replay of Human Traces

Jacob Schrum, Igor V. Karpov and Risto Miikkulainen
{`schrum2,ikarpov,risto`}`@cs.utexas.edu`

1 Architecture

The University of Texas at Austin's entry in the CEC 2011 Human-like Bots Competition is `UT^2`, which stand for **U**niversity of **T**exas in **U**nreal **T**ournament.

The `UT^2` bot uses a behavior-based architecture in which a list of behavior modules is cycled through in priority order on every logic cycle. Each behavior module has a trigger, and if a module's trigger fires on a given cycle, then that module defines the behavior of the agent for the given logic cycle. The full architecture is shown in Fig. 1.

In terms of Computational Intelligence, the two most interesting features of `UT^2` are that its combat behavior is defined via an evolved neural network (in the `Battle Controller`), and both its navigation and its routine for getting unstuck make use of a database of traces of human behavior in UT2004 (via the `Human Retrace Controller`).

This architecture is based on `UT^2-2010`: a previous version of `UT^2` that came 2nd in BotPrize 2010 [1]. Full details are available in two chapters for the upcoming book Believable Bots. Therefore, this abstract focuses on how `UT^2` has been improved since BotPrize 2010.

2 Evolved Battle Controller

`UT^2-2010`'s `Battle Controller` was learned using multiobjective constructive neuroevolution [2]. A new controller was evolved against native UT2004 bots. Evolution occurred in the relatively small map `DM-1on1-Albatross` to assure that as much of the bot's time was spent in combat as possible. The set of objectives used was reduced to a simple set of three: damage dealt (maximize), damage received (minimize), and number of collision events with level geometry (minimize).

Several input sensors were added to the bot's neural network, most interesting of which are the mimicry sensors. These are sensors that heuristically determine whether an opponent is executing one of a small set of combat movement options that are available to the bot. Such sensors were built to make evolution of mimicry possible should it prove useful in combat. Such mimicry should be useful since evolved behavior is generally not inherently human-like.

Human-like tendencies are enforced in how the outputs of the evolved network are interpreted. As with `UT^2-2010`, the combat actions available to `UT^2` are defined relative to its current opponent, which was selected via a scripted routine. The available actions are: `Approach`, `Retreat`, `Strafe` (left or right), stand `Still`, and `Go To Item` which is nearest. The bot always looks at the opponent while performing these actions, and thus seems to be focused in a human-like manner. The mimicry sensors indicate if the bot's opponent is performing any of these actions. During these actions, the bot has the option of jumping and/or shooting. `UT^2-2010` evolved the decision of when to shoot, but this year's version simply favors shooting whenever a target is available. Many new filters and restrictions on when these actions can be performed have also been added. For example, the bot is not allowed to strafe into walls, and if it is not being attacked and has the high ground, it will favor standing still to snipe. In terms of weapon usage, specific subroutines have been implemented to make weapon-specific tricks work, such as the BioRifle's secondary charge attack. These and other filters on the available actions were added based on the authors' knowledge of what humans consider to be human-like/bot-like behavior in UT2004.

Another important change in how the `Battle Controller` is used relates to use of the judging gun. `UT^2-2010`'s `Judging Controller` made use of the `Battle Controller` to define bot movement while attempting to judge opponents. However, competition experience has shown that the power and importance of judging makes getting a successful judgement important enough that human players are less concerned with maneuvering to avoid damage than usual. Therefore, `UT^2`'s `Judging Controller` does not use the `Battle Controller` to select from all available combat movement actions, but instead simply chooses the `Approach` command every time.

3 Human Trace Replay

`UT^2-2010` made use of human traces purely for the purpose of getting the bot unstuck whenever one of several stuck triggers fired, thus indicating that navigation had somehow failed. The current version of `UT^2` still uses human traces for this purpose, but also uses scripted actions for getting unstuck under specific circumstances.

The new approach to getting unstuck uses human traces as only one component. The scripted responses to getting stuck are to `Move Forward` if standing still, `Move Away` from walls and agents that the bot is colliding with, and to `Dodge` away from obstacles if

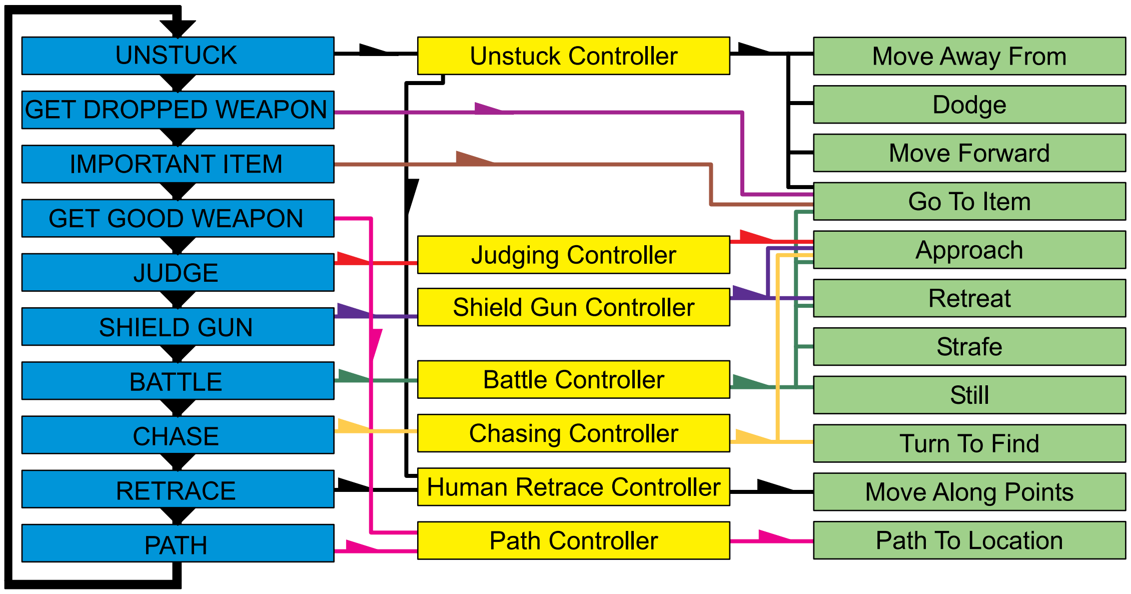

Figure 1: `UT^2` Architecture. Control cycles through the list of modules on the left once every logic cycle. If a reached module's trigger fires, then that module will define the bot's next action. Most modules have an associated controller (middle column) that further arbitrates between several available actions, or otherwise aggregates and makes use of information relevant to the actions performed by that module. All actions available to the controllers are in the right column. One of these actions is executed each logic cycle. Some control modules are simple enough that they do not need controllers: they carry out a specific action directly. Most of the control in this diagram flows from left to right, but note that the `Unstuck Controller` can actually make use of the `Human Retrace Controller` to define its action.

collisions are occurring with high frequency. These scripted responses usually work well in these situations, but if these responses repeatedly fail, or if the bot is near the same navpoint for too long, then the bot will try using human traces to get unstuck.

There is a separate database of human traces corresponding to each level in UT2004. Each database consists of a collection of agent locations stored along with the game time that the player was at that position. The sequence of locations for one player ordered by time represents a trace of how a human player moved through a given level. The locations within the traces are indexed by their nearest navpoint within the level. This indexing scheme speeds up the operation of finding the nearest point of the nearest trace when needed. Whenever a trace is retrieved for replay, the bot picks points along the trace starting from near its current location, and uses the `Move Along Points` action to move directly to one point while planning ahead to the next point in the sequence. Such advance planning results in smoother, more human-like movement.

If there is no reasonably close human trace available, or if the human traces have repeatedly failed to get the bot unstuck, then the bot resorts to random unstuck actions, which include `Move Forward`, `Move Away From`, `Dodge`, and `Go To Item`.

In addition to using human traces to get unstuck, the new control module `RETRACE` is entirely based on the prolonged playback of human traces, for the sake of smooth, human-like navigation throughout the level. Playing back human traces via the `RETRACE` module results in smoother movement than the lower priority `PATH` module, which is a slightly improved version of the path navigation module used by `UT^2-2010`. The human data used by `RETRACE` plays back smoothly because it was created in a synthetic manner: individual players ran around levels by themselves with no enemies, with the purpose of collecting items while exploring the level as much as possible. Such synthetic data is free of the erratic movement which is characteristic of combat, but which would look strange of replayed in the absence of an opponent.

4 Conclusion

More extensive use of human traces, extra filters on the evolved combat behavior, and numerous other small changes to the bot have made this year's version of `UT^2` more human than before. These improvements should give `UT^2` an advantage in the upcoming competition.

References

[1] Hingston, P. A New Design for a Turing Test for Bots. In *Computational Intelligence and Games* (2010).

[2] Schrum, J., and Miikkulainen, R. Evolving agent behavior in multiobjective domains using fitness-based shaping. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Portland, Oregon, July 2010), pp. 439–446.