# Behavioral Valence - Modeling Cohesive Actions to Augment Human Interaction

***Abstract* — Recently, the use of learning technologies within digital game logic has been questioned as presenting an unnecessary level of control complexity, coupled with issues in reliably testing AI-based behavior and biases that are an emergent property of a learning process. These criticisms resonate in the context of game-side augmentation. Rather than strictly limiting a learning AI to an opposing, antagonist role, this paper proposes a complementary scheme wherein an automated planner / reinforcement learner aids in the decision-making process of the human player. This proposed system offers real-time, contextual objectives to an individual player, based on cohesive feedback provided by a runtime multi-agent system coordinating players towards common goals.**

**This paper outlines, investigates and proposes a series of improvements associated with decision making processes when faced with uncertainty in high speed reactive Finite State Machine (FSM) architectures, allowing for greater refinement through adaptive state transitions. Furthermore, this paper investigates recent proposals in Artificial Intelligence (AI) based decision making technology emerging from UC Berkeley's RiseLab that combines "slow" cloud-based behavior refinement techniques with "fast" reactive techniques operating at a network edge.**

**FSMs present issues for programmers principally due to limited standardization, testing infrastructure and adaptability. This paper examines current implementations of finite state behaviors specifically in the context of AI-applied digital games and proposes player augmentation improvements through the use of "fast" reinforcement learning coupled with "slow" policy planning.**

**Ultimately, AI augmented human interaction techniques have a broader scope beyond digital gaming and are emerging as a key capability for a wide variety of AI deployments in many fields.**

**Valence: ˈvāləns**

- **relating to or denoting electrons involved in or available for chemical bond formation**
- **the linguistic elements that make a discourse semantically coherent including the number of grammatical elements with which a particular word, especially a verb, combine in a sentence**
- **the emotional and social bonds among members of a community, organization and society**

***Index Terms* — machine learning, reinforcement learning, artificial intelligence, finite state machines, modeling, serious games, Monte Carlo simulation**

## I. Introduction

Alex Pentland [1] in his 2014 book "Social Physics — How Social Networks can Make Us Smarter" [2] sees social cohesion as the tendency to create structures that are cooperative, productive and creative. Klaus Jaffe in his 2006 paper "Simulations Show that Shame Drives Social Cohesion" [3] offers that "as far back as Plato (427-347 BCE) we have recognized that feelings of shame are fundamental for the maintenance of social cohesion," thereby suggesting there are more nuanced aspects to maintaining cohesive behaviors.

Pentland goes on to point out that "to understand our new [digital] world we must extend familiar economic and political ideas to include the effects of these millions of [online] people learning from each other and influencing each other's opinions," suggesting that our understanding of people as individual decision makers must be balanced by understanding of how behaviors aggregate across these emerging social networks at various scales, from local to global. Indeed, as Adam Smith understood well, it is the nature of our social fabric that guides the "invisible hand" of the market, not simply competition alone.

### *A. The Problem With People*

As Daniel Kahneman [4] routinely asks [5], "why do people regularly make poor decisions?" The answer, he asserts, lies in behavioral science, a field of study that, in part, believes human beings are unknowingly hamstrung by overconfidence, limited attention, cognitive biases and other psychological factors that inevitably result in errors of judgment.

Furthermore, Kahneman claims that much of human error is not attributable to a systematic cause, but rather to noise. "When people think about error, we tend to think about biases. But in fact, a lot of the errors that people make [is] simply noise, in the sense that it's random, unpredictable, it cannot be explained. You look at large organizations that are supposed to be optimally rational. And the amount of folly in the way these places are run is actually fairly troubling." Kahneman's prescription is for society to temper human judgment with "disciplined thinking" through the augmented use of algorithms. When it comes to decision-making, algorithms are superior to people. “Algorithms are noise-free. People are not,” he states. “When you put some data in front of an algorithm, you will always get the same response at the other end.”

In a seminal paper published in 1979, “Prospect Theory: An Analysis of Decision under Risk,” Daniel Kahneman and Amos Tversky [6] argue that the ways in which alternatives are framed — not simply their relative value — heavily influence the decisions people make.

Framing alternatives can, for example, substantially change people’s preferences regarding risk. Given AI is now beginning to play a dominant role in technology and McKinsey has recently identified a growing consensus [5], that future productivity demands much more creative and critical thinking skills from people. Since many today often assume that these algorithms will soon be able to make important decisions autonomously and fear these algorithms will eventually overwhelm them.

This latter extrapolation, Kahneman and Tversky assert, is not only wrong, it’s dangerous. A more accurate and reasonable view is that today's advancing cognitive computing tools are expected to remain relatively focused within specific domains. As such, AI should be used to amplify, not replace, human intelligence and intuitive skills.

Kahneman goes on to observe [5] that psychologists have long realized that the mind is comprised of two distinct systems:

1) *System 1 (Fast Autonomic Nervous System)* operates automatically, intuitively and quickly with little or no effort and no sense of voluntary control. These automatic operations generate surprisingly complex patterns of ideas and behaviors and include innate skills humans share with other animals including perception, recognition, orientation and emotions. Many other activities can, in fact, be made automatic and fast through prolonged practice — in effect, programming our spinal cord rather that our cerebral cortex to react.
2) *System 2 (Slow Conscious Nervous System)* allocates attention to the effortful mental activities that demand it, including complex reasoning and calculations, and is often associated with the subjective experiences of identity, choice and concentration. Equally, it is these purposeful mental activities that construct thoughts in an orderly series of steps. These highly diverse activities have one feature in common — they all require attention and are disrupted when attention is drawn away.

The impact on our choices due to various biases include loss aversion, over confidence in achieving goals, difficulties in predicting what will make us happy, and the challenges of properly framing risks. All of these can only be understood in terms of how these two mental systems shape our judgements and decisions.

### *B. The Problem With Machines*

Among several increasingly popular AI techniques, deep learning, mimics the way interconnected layers of neurons fire in a brain as it learns to make sense of new information. An example of which, Google’s DeepMind Alpha Go program, taught itself through hours of automated practice to beat the former Go world champion Lee Sedol in March 2016 [7]. Through gradually honing an internal sense of strategy, Alpha Go’s ability to beat one of the world’s best Go players represents a true milestone in machine intelligence and AI.

Another aspect of machine intelligence was highlighted recently by UC Berkeley’s RiseLab [8], which claims that for all the vast amounts of information being collected, this “data is only as good as the [human] decisions its enables.” From this, they suggest we need machines to:

1) “make increasingly rapid complex decisions correctly in uncertain environments”;
2) “be robust and resilient by handling noisy and unforeseen inputs and failures”; and
3) “have the ability for the AI to trace and explain its own decision making”.

This highlights, for RiseLab, a concern with the way in which AI infrastructure has emerged to date, specifically, the latencies associated with the existing cloud-based statistical and neural learning techniques compared to decision speeds needed locally at the network edge. Given Daniel Kahneman’s similarly entitled book, RiseLab refers to this as “inferencing, fast and slow,” which focuses attention on the evolving role of AI from its deployment and performance expectations to achieving effective human augmentation.

From this, we expect to see a separation of AI-based cognitive functions into two distinct implementations of Fast and Slow AI. (see Figure-1a and Figure-1b)

## II. Architecture

### *C. What is AI?*

All too often, the publicity surrounding AI today is woefully misunderstood and subject to many interpretations across a wide spectrum of fields. Without a clear definition of AI, the tendency is to simply offload menial, tedious tasks to a machine and refer to this automated handling of the task as “AI”. As a result, individuals outside the field (and even some AI practitioners themselves) appear confused as to what AI actually is.

Historically, after several decades of research and development efforts, classical AI techniques were facing intractable issues (e.g. combinatorial explosion of modelling artifacts) when AI was confronted with real-world planning problems. Rodney Brooks [11] began disrupting this established approach in the 1980s by innovating a "situated AI" that focused on building agents that behave effectively (simply) in their environment. From this Brooks argued [12][13] that "situated AI" required designing AI agents "from the bottom-up not top-down" by focusing on basic perceptual and motor skills required to survive.

This "situated AI" approach gives much lower priority to abstract reasoning and theorem proving about plans, in preference to developing basic behaviors to explore an environment, discover food, threats and sanctuaries while developing resources.

Ultimately, the question of what is AI does not have a simple answer. As the capabilities of AI have proven themselves practical for a wide variety of subject areas, the question of AI heavily relies of the requirements of the researchers involved.

In the case of this paper, AI is defined as the production of predictive policies at various scales derived from environmental evidence, which collectively drives the adaptation of subsumptive behaviors of individual agents in-situ. (see Figure-2a and Figure-2b)

## III. AI IN DIGITAL GAMES

As software development within the video game industry has evolved over time, so too have various techniques employed by game developers to minimize the associated software development effort and costs. For instance, reusable game engines provide a significant reduction in cost for studios. Rather than constantly re-developing the core components of a game (i.e. 2D/3D rendering engine, physics engine, network management, sound, animations, etc.) a reusable game engine provides developers with a more economic approach, via reusable core engine components.

A key factor in the development of modern digital games is the inclusion of non-player characters (referred to as NPCs or agents). These entities are strictly computer controller characters whose behaviors are executed as reactions to human player choices and surrounding environmental cues. When developing game agents, a significant amount of time is spent reimplementing NPC behaviors despite strong similarities shared with other completed game agents. As a result, a sizeable portion of software development cost can be attributed to the overall number of these discrete game agents.

Christopher Dragert in his 2015 PhD thesis entitled "Model-Driven Development of AI for Digital Games" [15] suggests that a lack of a suitably generic AI representation that is both formalized and flexible has constrained the streamlining of the AI development process. A key aim of Dragert's work is to utilize model-driven development to improve the software engineering process of modern game AI. Dragert demonstrates that the creation of a reusable tool (an external core engine component) for AI development enables the formation of modular artificial intelligence and a new formalism of layered statecharts for the development of NPC AI behaviors.

### *D. Statechart Modeling*

Statecharts were introduced in 1987 by David Harel, in his paper entitled "Statecharts: A Visual Formalism for Complex Systems" [16] as a means to visually model behaviors in reactive systems. The core component of statecharts is the concept of discrete states and their associated transitional connections. Statecharts take a sequence of discrete events as inputs that cause transitions to (timed) states, which in turn produces a sequence of discrete output events. While statecharts are similar to previous FSM development techniques, statecharts extend FSMs by providing improved encapsulation [17], in addition to supporting visual development and animated program execution.

### *E. Improvements to Model-Driven AI Development*

Dragert demonstrates the modular capabilities of layered statechart-based AI in the form of "Scythe AI" [15], a tool designed to manage modular reuse of game AI. By supporting the creation and manipulation of AI interfaces, the visual engineering of new AI logic, and ensuring correctness through an error and warning system, "Scythe AI", provides a complete workflow for reuse. However, while "Scythe AI" facilitates reuse through modular interfaces within layered statechart-based AI, the tool could be further enhanced by providing support for dynamically changing AI behaviors at runtime.

Example: Within a combat simulation, an AI game coordinator releases a squad of agents. The objective of this squad is to engage and destroy a human opponent's game character. Once engaged, the squad of AI is eliminated when running from cover to cover during sustained enemy fire.

Utilizing a Fast / Slow AI comprised of a multi-agent reinforcement learning system directed by a secondary policy coordinator analyzes the cost of the defeat. The AI coordinator establishes an event chain leading up to the death of each squad agent, and determines an appropriate modification to the resulting behavior queue. Strictly using prebuilt behavior states, the policy coordinator may modify the agent behavior rewards and costs to include a higher reward value to the "crouch" action when running from cover to cover, and in turn a higher cost value to the "running while standing" behavior.

Once the battle is complete (e.g., opponent agents are destroyed or all NPC agents are eliminated), the primary multi-agent system feeds the secondary policy coordinator with summary data regarding the performance of each individual agent, as well as the surrounding environmental conditions over time. The coordinator then iteratively re-evaluates the decisions over the timeline, providing further policy based suggestions for improvement.

In order to establish refined, dynamically changing AI behaviors at runtime, a subset of Machine Learning, known as Reinforcement Learning (RL), should be employed by the game's agents at the network edge. As noted in "A brief introduction to reinforcement learning" by Kevin Murphy [18], "reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards". Rewards are typically numerical signals used to direct an RL agent in order execute an ideal behavior within a specific context, while maximizing its reward value. Furthermore, Reinforcement Learning enables goal-oriented learning based on a software agent's interaction with its surrounding environment.

Lionhead Studio's 2001 game "Black and White" is a particularly prominent example of early adoption of reinforcement learning in a digital game [19]. Players begin the game by selecting a creature based on a Belief-Desire-Intent (BDI) model [20]. As the game proceeds, the player can teach the creature how to perform various tasks (fighting, eating, etc). Unlike the approach used in Q-Learning, Lionhead Studios leveraged a modification of the BDI framework to reinforce the creatures' decision making process. Specifically, when a player evaluated a proposed action to be performed by an agent, a positive reinforcement could be applied through a 'stroking' response, while undesired actions could be dissuade through a 'slapping' response.

Implementation of RL algorithms vary. In a 1989 PhD thesis paper "Learning From Delayed Rewards", Christopher Watkins [21] introduced a method of RL called Q-Learning. The distinctive feature of Q-Learning is its capacity to choose between immediate rewards and delayed rewards. At each step of time **t**, an agent observes its current state **xt**, then chooses and applies an action **ut**. As the process moves to state **xt+1**, the agent receives a reinforcement or reward signal **r(xt, ut)**. The goal of this training is to find the sequential order of actions that maximizes the sum of the future reinforcements, thus leading to the shortest path from start to finish.

The relation between states, actions and rewards are typically derived through some variation of the Bellman optimality equation (either through an optimal state-value or action-value function). Resolving these equations is typically done through the use of either dynamic programming methods, such as value or policy iteration, or through Monte Carlo based simulations. Dynamic programming based value or policy iteration require a complete and accurate model of the environment. Typically the surrounding environmental model is often unknown to a digital game agent, and therefore the agent must interact with its environment to obtain further information to help produce an optimal policy.

Common approaches to learning an optimal policy are 1) model-free or 2) model-based. In a model-free approach the agent learns a policy without the need to learn the model of the environment. The previously mentioned Q-Learning is a well known, model-free approach based on a value iteration algorithm. The model-based approach revolves around an agent that learns a model of the environment, then uses that to determine an optimal policy.

The Dyna framework is a common model-based approach, which integrates RL and planning (policy coordination) into a single process operating alternately on the world and on a learned model of the world. The Dyna architecture has historically been used in combining RL and planning. Notably, Richard Sutton in 1990 presented two approaches to Dyna, showcasing their use in dynamic environments [22]. Sutton's primary example utilized dynamic programming's policy iteration, while the other leveraged RL based Q-Learning. While Sutton's paper presented a number of limitations with Dyna (limited search controls, explicit knowledge of world state, and quality/resolution of state estimations), criticism of Q-Learning has been raised, referring to the process as often very slow, and in situations of high complexity, difficult to determine an optimal policy [16]. Implementation of state planners, in tandem with RL, provide the potential to address this optimization issue.

## IV. Criticism of Learning in Digital Games

RL provides the potential to greatly enhance behaviors found in a multi-agent system through refinement, although digital games leveraging learning have been frequently criticized. When outlining additional architectural approaches to AI decision making, Dragert provides a proposal in support of AAA titles incorporating learning into their game AI [15].

Dragert explains that while historically, some games have attempted to adopt a learning AI, the nature of learning, in combination with the variation in possible play styles, could result in a game that is easy for some players, and impossible for others. Furthermore, his paper extends the critique to AI debugging, where Dragert states that leveraging a learning AI "complicates testing, since the behavior of the AI is an emergent property of the learning process itself". The criticism of fluctuating ranges in difficulty, as well as complications in testing, is a side effect that Dragert noted, particularly with Lionhead Studio's Black and White.

This paper acknowledges the criticism of previously developed learning AI. The criticism of fluctuating difficulty can be directly associated to AI that typically plays an opposing role against a human player. However, the inclusion of a complementary system, where an automated planner / reinforcement learner aids in the decision making process of the human player, could address the latter critique. This proposed system offers realtime, contextual objectives to an individual player, based on cohesive environmental feedback provided by the runtime multi-agent system.

## V. Dynamic Agent Behavior Implementation

An article by Ioannis Partalas, Dimitris Vrakas and Ioannis Vlahavas entitled "Reinforcement Learning and Automated Planning: A Survey" [23], notes that planners (or policy coordinators) combined with learners offer similar operational opportunities. By equipping an agent with a planner (coordinator) and a learner that cooperate either sequentially or in a concurrent way, improvements can be made progressively when guiding an agent to achieve a set of longer term goals (especially when situated in a dynamic environment). With this in mind, there are three possible approaches in merging both planning (policy coordinators) and RL to achieve layered-statechart AI:

1) **Plan First then Learn**
In this method, planning (coordination) is first utilized to construct a fixed sequence of actions that may then be used to speed up the learning process. This procedure can be demonstrated as a tree, whereby the null node is the goal state, and all other nodes are behavior states. Furthermore, the actions executed by the RL agent are represented as connections between the discrete nodes. When an action is performed in a node, then the conditions of the parent node are achieved.

The three authors propose that the simplest means to implement a planning and learning merge is to pre-compile compound actions from simple ones, thereby generating hierarchies to be used by an RL agent, done either manually or automatically as a part of the planning process. Finally, the agent learns a policy for choosing optimal paths in the plan that was previously produced from the planner. This combination of planning and learning is further described in Malcolm Ryan and Mark Pendrith's work on "Reinforcement Learning Teleo-operators" in 1998 [24].

2) **Learn First then Plan**
Unlike the previous approach, this method prioritizes an agent to learn first, then plan. During the first step an RL algorithm is used to estimate the value function for each problem state. A value function is defined as "the expected discounted return for executing action a in state s and thereafter following policy π" [21].

In the second phase, the agent planner produces an event hierarchy according to the traditional GOAP formulations. The produced event list consists of an explicit sequence of potential actions that require little to no environmental feedback when executed. Partalas et al state that the advantage of this approach is to improve the usability of the results. Specifically, RL relies on moment-to-moment sensing, while the sequences of actions produced by planners are very useful in situations where continuous sensing is difficult or have the potential to be computationally intensive.

3) **Fuse Learning and Planning Together**
Finally, Partalas et al [23] propose a third hybrid option as a merger between both learning and planning methodologies. Planners search a state space in order to construct a solution. By incorporating learning into the planning process, they are able to bias the search in directions that make it possible to create high quality, highly optimized plans.

## VI. Implementation: Fast Reinforcement Learning Through Slow Policy Creation

To test, validate and ultimately evolve the theory of "fast", "slow" behavioural AI, we need to implement these agents within a constrained finite yet largely random game environment utilizing electro-mechanical actuators under programmatic control to effect outcomes while still allowing for both human input and remote behavioural coordination. The gaming environment we elected to build as a test platform is a *robotic FoosTable*.

Specifically, this paper proposes a split solution, comprised of co-dependent “fast” and “slow” inferencing agents, including one set of up to two (2) “fast” agents executing on the *FoosTable’s* embed computer at the network edge, where each agent controls a set of four (4) electro-mechanical servo *FoosMen* actuators from eight (8) in total (1 x Goalie; 2 x defensemen; 5 x midfielders; 3 x forwards) that can effect left / right translation and 360° rotational high speed motions.

Each agent’s software effects the aggregate behaviours of a given FoosBall player in a match, as captured and synthesized from repeated live human play over time by a “behaviour coordination agent”. Each of a player’s numerous “behavioural fragments” (and associated triggering events) are aggregated within a “behavioural tree” by this “behavioural coordinating agent” at the end of each match. Within each “behavioural fragment” is a set of operational instructions used to drive.

These “fast” player agents, in turn, utilize Reinforcement Learning (RL) adaptation techniques within a “slow” behaviour coordination agent executing within the cloud.

This configuration has the potential to create more dynamic NPC actions in a MAS through the combination of a “fast” and “slow” learning, and planning system. In addition, this proposed configuration provides an opportunity for more sophisticated AI opponents to emerge from within the genres of digital as well as physical gaming.

The rules of the primary MAS system are as follows, and apply to all agents:

- All independent agents contain a suite of identical known behavior fragments (In the case of a physical *FoosTable* implementation, ball juggling behavior, offensive push shot behavior or 5-bar to 3bar pass behavior, etc).
    - These behavior states assigned to each agent constrain the range of possible executable actions the agent can perform.
    - These scoped behavior ranges construct a behavior tree, whereby the MAS traverses the behavior tree with the objective of reaching the final end goal state (win or loss).
- Each agent within the primary system must act independently of one another. In the case of a physical game such as *FoosBall*, each rod containing a row of variable *FoosMen* represents a single agent.
- The paths connecting each state in the behavior tree contain a weighted value. These weighted links are the behavioral valence values upon which MAS decides future actions and records successful and unsuccessful traversal paths.
- Consistent with standard RL practices, the decision for an agent to act upon any given behavior state is directed by the weighted behavioral valence value linking each state in the behavior tree. Therefore, each agent must be provided with and directed by a reward goal.
- Each agent attempts to discover which traversal path yields the greatest reward return.
    - Agents must learn, over discrete time steps, which combination of behavioral actions yields the highest current reward. This per-agent discovery process is executed using event triggered scheduling within the agent’s logic.

Upon completion of the “fast” learning MAS, the secondary “slow” cloud based planning system takes over. In order for the secondary system to process the game data from the primary system, the following rules must be met:

- When a game is completed, the MAS must package recorded game metadata into a behavioral timeline.
    - Game metadata contains all recorded game data, including but not limited to:
        - Executed behavior trees containing exact state paths and the corresponding epoch time, scores etc.
- A behavioral timeline must then be validated by a consensus of nodes on the network. Upon validation, the behavioral timeline is transmitted to the cloud policy manager.
    - Individual node validation is achieved through a successful replay of the behavioral timeline.
- Only upon a valid consensus of the network nodes, will the recorded behavioral timeline be uploaded to the cloud

Each independent agent found within the primary system acts using a suite of known behaviors in order to maximize this numerical reward signal. The reward signal directing the behavior selection is provided by the secondary policy planner coordinator. Directing the primary MAS is the secondary, cloud-based, policy coordinator. This coordinator evaluates the decision path contained in the provided behavioral ~~fragment~~ timeline.

The influx of each new game’s behavioural metadata, when added to the behaviours already captured, in aggregate, by the existing behaviour tree, is then used to synthesize a new behavioural tree. By modifying the valence of each behavioural fragment, as well as the hierarchy or order of behavioural fragments in the behavior tree, the policy system is able to correct current behavior and synthesize future behavior responses. The modifications (referred to as policies) made to the original behavior tree are repackaged as a new behavior fragment and provided back to the MAS. Through the creation of policies generated by the coordinator, the secondary system is in turn, influencing the behaviors of the agents in the primary system.

It is important to note that as the coordinator acts independently of the MAS, one can imagine these two mechanisms operating with a distinct "fast" and "slow" execution cycle. The "fast" execution process acts continuously in real time. The MAS determines which behavior states to act upon in order to yield a reward, while the policy coordinator, the "slow" execution process, operates on a need-to-update basis, providing refinement when needed. As agents from the primary "fast" system feed data to the secondary "slow" system, the policy coordinator updates only when conditions dictate. In other words, if a critical mean threshold for all agents is exceeded, such as hunger or health, or if an environmental condition is triggered (i.e., an area is being attacked by an enemy population), then an offensive behavior is recommended.

### *F. FoosTable Multi-Agent System Implementation*

Execution of a network edge MAS comprised of stochastic agents leveraging reinforcement learning, requires a powerful embedded computational device dedicated to managing the regarding core components. Nvidia's Jetson TX2 module provides the foundation for accurate environmental inferencing, as well as the platform from which the MAS operates.

Operating as the primary layer on top of the TX2 board, The Robot Operating System (ROS) offers a message passing interface providing inter-process communication. As the MAS iterates through the previously described operational loop, traversing through a behavioral tree and executing corresponding behavior states. The MAS passes a message set to the servo controller, which in turn drives the servo controls operating the physical *FoosAgents*, as outlined by the MAS's current behavior state.

In addition to providing standardized message definitions and libraries, ROS's utilization of an anonymous, asynchronous message passing service readily allows for data to be captured and replayed. Additionally, the ROS acts as the foundational services layer to provide the basic secure Internet protocol stack, enabling a baseline implementation of a ledger to record and convey fine-grained per-game data back to the policy coordination system for ongoing cloud based behavior analysis.

In order to properly direct the MAS layer, an object tracking system is needed to maintain a 3D map of the current location of the *FoosBall* on the *FoosTable's* playing surface in real time. In turn, this object tracking system must be capable of tracking its target object despite visual occlusion. Reliant on external cameras looking "inward" towards the constrained playing surface, the OpenCV SLAM software module used to gather information regarding its surroundings.

A key component of the object tracking system is the *FoosTable's* OpenCV based Synchronous Localization And Mapping system (SLAM) software. Intel's 3D RealSense infrared cameras are used to capture the position of the *FoosBall* with this location and trajectory data being fed into a Kalman filter to project ball trajectory based on recorded ball position and velocity. **[REFERENCE Joan Sol`a paper**] Operation procedure of the SLAM system consists of three core tasks executed iteratively through a fixed time step.

1. A *motion model* is generated upon movement of the agent. The agent **A** moves according to a control signal u and a perturbation **n** and updates its state, **A ← f(A, u, n)**. The *motion model* acts as a mathematical model to reduce uncertainty on the robot's localization.
2. An *inverse observation model* is used to record and map key features (known as landmarks) throughout the play space. The *inverse observation model*, a mathematical model, is used as an automated solution to determine the position of landmarks in the scene. The position of these landmarks is provided by the data obtained by the exteroceptive and proprioceptive sensors and can be computed as **Li = g(A, S, yi)**, where **Li** is an observed landmark mapped by a sensor **S**.
3. As the agent **A** travels throughout the play space, prior mapped landmarks **Li** are observed by means of a sensor **S**, where **yi = h(A, S, Li)**. A *direct observation model* is used in updating both the localization of the agent, as well as the localization of all landmarks, ultimately producing an updated measurement of **yi**.

Each of these three operations, in combination with an estimator engine, can be used to build an automated solution to the SLAM system. In the case of *FoosBall* implementation, a Kalman filter is used as the estimator engine.

### G. *FoosTable Cloud-based Policy System Implementation*

As the number of *FoosTables* grows, so too will the corresponding behavioral fragments generated by each game played. These discrete fragments contain specific player motion / controller / ball-state sequences captured as metadata before being used in the behavioral ~~fragment~~ refinement process by the central cloud based policy manager. These refined behavioral fragments provided by the policy manager are used in turn to optimize the corresponding AI's player persona. Not only will a secure database need to be utilized in this effort to record all games played, but so too will a system be required which enables secure transmission of the corresponding data between the reinforcement based "fast" AI found on each *FoosTable* and a central cloud based "slow" policy manager.

Deployment of a Blockchain system is used to fulfill these requirements. The Blockchain operates as a private game ledger, not only functioning as a scalable database but providing a means to transmit, and consequently, validate the block of data sent. In the case of the fast AI based *FoosBall* table, each table acts as a node in a larger network. Once a game is completed, that table transmits the game metadata through the network connecting all *FoosTables*. Once a supermajority of nodes agrees on the validity of the game played, the game is captured and the corresponding metadata transmitted to the cloud.

Game validation is a critical component when recording games played. As a private ledger, the identity of all parties is known to all nodes on the network. This in fact adds a layer of security as, in the case of a rouge/ malicious actor, attempting to tamper with the table can be pursued and removed from the network.

Once the cloud based policy manager has determined the next course of action, the Blockchain is used to securely pass the refined behavioral fragment data back down to the originating *foostable's* MAS.

## VII. Conclusions

Behavioral Valence represents a new component enabling AI augmented Human Interaction through sophisticated policy coordination in a wide variety of game or digital contexts. For instance, players operating within a real time strategy game are "influenced" by an AI Coordinator to focus on: farming policy, defensive posture, expanding the housing stock, expanding the transportation network, etc. Specifically, a system of human driven players operating at the network edge are augmented by a cloud-based policy coordinator (planner), who evaluates and responds to situational metadata provided by the multi-agents.

Future work regarding the implementation of a policy planner is needed. For example, the question arrises whether a planner must be based on RL, or if other variations of planning algorithms are more appropriate. Future experimental work is also needed to extend and refine the realtime FSM implementation at the network edge with synchronized metadata and reward coordination.

Implementing agent behavior through fast learning, augmented by slow policy coordination, offers developers the ability to refine the performance of more traditional decision-making mechanisms. Principally, this is achieved through individual agents acting through stochastic Reinforcement Learning techniques, coupled with a coordinator generating near-term policy goals to also influence these per-agent behaviors. In turn, an MAS sends situational feedback data to the coordinator in order to inform the next epoch's policy choices. While individual agents remain driven by a statechart / subsumption-based architecture, the addition of rewards driven policy directives further enhances the behavioral outcomes.

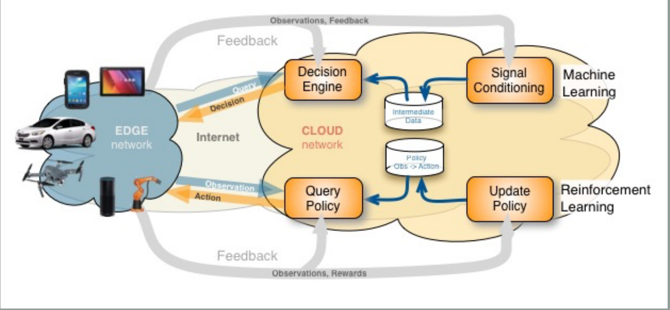

Figure-1a: Berkeley's RiseLab current AI worldview for both Machine & Reinforcement Learning algorithms which are located dominantly in the cloud

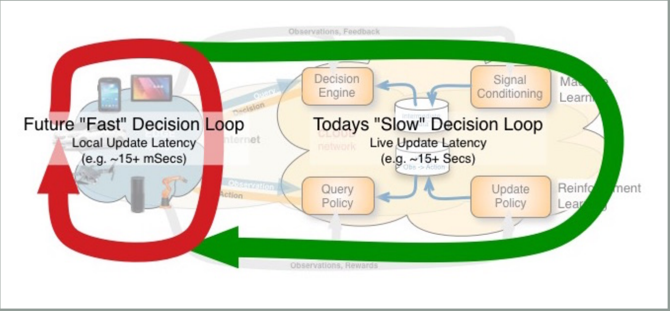

Figure-1b: Berkeley's RiseLab foresees AI infrastructure becoming more local to the network edge, in part, to address an order magnitude increase needed in its performance

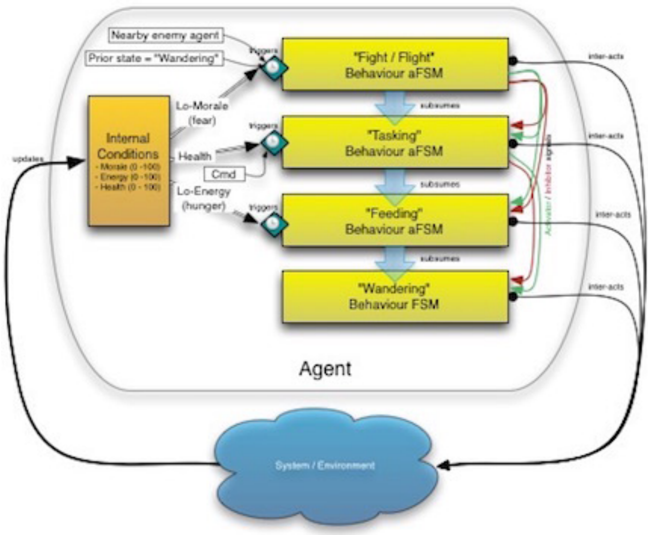

Figure-2a: Example of per-agent Brooksian subsumption layering in the game environment *Valence* [14] with Activator / Inhibitor inter-layer signalling

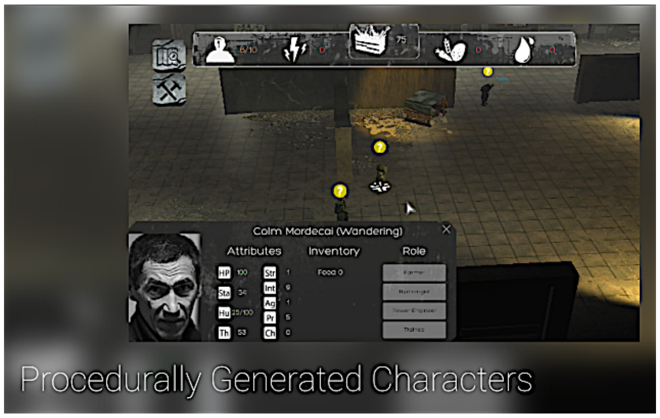

Figure-2b: Screen captures from the game environment *Valence* [14] showing a Procedurally Generated Agent (left) and Reactive AI Agents (right) in-situ

## References

1. Alex "Sandy" Pentland is the Toshiba Professor at MIT where he directs the Human Dynamics Lab. Pentland's research focuses on “social physics”, big data, and privacy. His research helps people better understand the "physics" of their social environments.
2. *Social Physics — How Social Networks can Make Us Smarter,* A. Pentland, MIT Penguin Books, 2014 [Online]. Available: https://www.amazon.ca/Social-Physics-Networks-Make-Smarter/dp/0143126334.
3. “*Simulations Show that Shame Drives Social Cohesion,*” Klaus Jaffe, Simon Bolivar University, Advances in Artificial Intelligence Conference January 2006 [Online]. Available: https://www.researchgate.net/publication/220942816_Simulations_Show_That_Shame_Drives_Social_Cohesion
4. Daniel Kahneman is the 2002 Nobel Laureate in Economic Sciences where as a psychologist in collaboration with Amos Tversky they challenged the assumptions of human rationality embedded in the decision-making prevalent in the economic theory at that time.
5. *Thinking, Fast and Slow*, D. Kahneman 2011 [Online]. Available: https://www.amazon.ca/Thinking-Fast-Slow-Daniel-Kahneman/dp/0385676530
6. *Prospect Theory: An Analysis of Decision under Risk*, Daniel Kahneman and Amos Tversky; Econometrica, March 1979 [Online]. Available: http://www.jstor.org/stable/1914185
7. *AlphaGo Beats Go World Champion*, Christopher Moyer, The Atlantic, March 28, 2016 [Online]. Available: https://www.theatlantic.com/technology/archive/2016/03/the-invisible-opponent/475611
8. University of California, Berkeley, Department of Electrical Engineering and Computer Sciences, RISE lab (Real-time, Intelligent, Secure Execution) for Low Latency Inferencing, 2016 [Online]. Available: https://rise.cs.berkeley.edu/
9. R. C. Schank, “*What Is AI, Anyway?,*” AI Magazine vol 8 no. 4, 1987 [Online]. Available: https://aaai.org/ojs/index.php/aimagazine/article/view/623/556
10. *Machines Who Think*, P. McCorduck, A K Peters, Ltd., 2004 [Online]. Available: http://www.pamelamc.com/html/machines_who_think.html
11. Rodney Brooks is the Panasonic Professor of Robotics (emeritus) at MIT. He is a robotics entrepreneur and Founder, Chairman and CTO of Rethink Robotics. He is also a Founder and former CTO (1990-2008) of iRobot Corp. Dr. Brooks is the former Director (1997-2007) of the MIT Artificial Intelligence Laboratory and then the MIT Computer Science & Artificial Intelligence Laboratory (CSAIL). He has published many papers in computer vision, artificial intelligence, robotics, and artificial life.
12. R. A. Brooks, “*A Robust Layered Control System for a Mobile Robot,*” Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1986 [Online]. Available: http://publications.ai.mit.edu/ai-publications/pdf/AIM-864.pdf
13. *Fast, Cheap and Out of Control*, Errol Morris, Sept 24, 2002 [Online]. Available: https://www.amazon.ca/Fast-Cheap-Out-Control-Widescreen/dp/B00003CX9Z
14. Jake Deugo, Tyler Dinardo, Laryssa Ribeiro Da Silva, Weri Sin, Zachary Sullivan and Vishesh Thanki, “Valence - A Subterranean Game,” Undergraduate thesis, School of Information Technology, Carleton Univ., Ottawa, ON, Canada 2016 [Online]. Available: https://www.zacharysullivan.net/valence
15. Christopher Dragert, “Model-Driven Development of AI for Digital Games,” Ph.D dissertation, School of Computer Science, McGill Univ., Montreal, QC, Canada 2015 [Online]. Available: http://gram.cs.mcgill.ca/theses/dragert-15-model.pdf
16. D. Harel, “*Statecharts: A Visual Formalism for Complex Systems*”, The Weizmann Institute of Science, 1987 [Online]. Available: https://www.inf.ed.ac.uk/teaching/courses/seoc/2005_2006/resources/statecharts.pdf
17. Steve Rabin, “*AI Game Programming Wisdom 4,”* Boston, MA Charles River Media, 2008 [Online]. Available: https://www.amazon.ca/AI-Game-Programming-Wisdom-4/dp/1584505230
18. K. Murphy, “*A Brief Introduction to Reinforcement Learning,*” Department of Computer Science, University of British Columbia, 1998 [Online]. Available: https://www.cs.ubc.ca/~murphyk/Bayes/pomdp.html
19. Peter Molyneux, “*Postmortem: Lionhead Studios' Black & White*”, Gamasutra, 2001 [Online]. Available: https://www.gamasutra.com/view/feature/131476/postmortem_lionhead_studios_.php
20. *The Belief-Desire-Intention Model of Agency,* Michael P. Georgeff, Barney Pell, Martha E. Pollack, Milind Tambe and Michael Wooldridge, International Workshop on Agent Theories, Architectures, and Languages, pp. 1-10. Springer, Berlin, Heidelberg, 1998 [Online]. Available: http://www.cs.ox.ac.uk/people/michael.wooldridge/pubs/atal98b.pdf
21. Christopher J.C.H. Watkins, “*Learning From Delayed Rewards,*” PhD dissertation, Department of Psychology, University of Cambridge, UK 1989 [Online]. Available: http://www.cs.rhul.ac.uk/~chrisw/thesis.html
22. *Integrated architectures for learning, planning and reacting,* Richard Sutton, Proceedings of the Seventh International Conference on Machine Learning, pp. 216-224, 1990 [Online]. Available: https://pdfs.semanticscholar.org/b5f8/a0858fb82ce0e50b55446577a70e40137aaf.pdf
23. Ioannis Partalas, Dimitris Vrakas and Ioannis Vlahavas, “Reinforcement Learning and Automated Planning: A Survey,” Department of Informatics, Aristotle University of Thessaloniki, 2008 [Online]. Available: http://lpis.csd.auth.gr/publications/rlplan.pdf
24. *RL-TOPs: An Architecture for Modularity and Reuse in Reinforcement Learning*, Ryan M. R. K, Pendrith M. D, 15th International Conference of Machine Learning, pp.481-487, 1998 [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.37.491&rep=rep1&type=pdf

**Zachary Sullivan** received a Bachelor of Information Technology degree from Carleton University, Ottawa, ON in 2016.

From 2013 to 2015, he was a Research Assistant at Carleton’s School of Information Technology working on a range of projects including Robotic Hand Articulation utilizing *InMoov* 3D printed components, as well as a PTSD treatment strategy developed in conjunction with the Department of Psychology that utilized *Unity3D*™ immersive game technology to facilitate coping strategies, using simulation environments, for patients who experienced traumatic events.

Mr. Sullivan also has micro-simulation software experience (*Modgen*) at Statistics Canada through a Research Assistant role via the University of Ottawa and has extensive development experience in web based UI / UX. He is an avid gamer with a particularly strong interest in the application of AI to behaviors in game simulations. He was the project lead of the six person team that developed *Valence: A Subterranean Game* in 2016.