Tracking the Cognitive Band in an Open-Ended Task

John R. Anderson[a,*], Shawn Betts[a], Daniel Bothell[a], Cvetomir M. Dimov[b], Jon M. Fincham[a]

[a] *Department of Psychology, Carnegie Mellon, United States*
[b] *Department of Psychological Sciences, University of Geneva, Switzerland*

Abstract

Open-ended tasks can be decomposed into the three levels of Newell (1990)'s Cognitive Band: the Unit-Task level, the Operation level, and the Deliberate-Act level. We analyzed the video game Co-op Space Fortress at these levels, reporting both the match of a cognitive model to subject behavior and use of EEG to track subject cognition. The Unit Task level in this game involves coordinating with a partner to kill a fortress. At this highest level of the Cognitive Band, there is a good match between subject behavior and the model. The EEG signals were strong enough to track when Unit Tasks succeeded or failed. We were able to stitch together segments of model games to produce close matches to subject games at the Unit-Task level. The intermediate Operation level in this task involves legs of flight to achieve a kill. We were only able to stitch together games that have a modest overlap with subject behavior in terms of their leg structure, limited by the huge diversity of model and subject behavior at this level. The EEG signals associated with these operations are much weaker than the signals associated with the Unit Tasks, severely limiting tracking at this level. At the lowest Deliberate-Act level, there is a mismatch between subject and model behavior that prevented tracking. We conclude that the Unit-task level is the appropriate level for tracking open-ended tasks.

Keywords:
Cognitive Modeling, EEG, aModel Tracing, Video Games

*This work was supported by the Office of Naval Research Grant N00014-21-1-2586. We thank Henry Liu for his work in data collection. The data and analyses which were used to create the figures in the paper are available at

Email address: ja@andrew.cmu.edu (John R. Anderson)

Table 1: Newell's Time Scales of Human Action

Scale (sec)	Time Units	System	World (theory)
10^7	months		
10^6	weeks		Social Band
10^5	days		
10^4	hours	Task	
10^3	10 min	Task	Rational Band
10^2	minutes	Task	
10^1	10 sec	Unit Task	
10^0	1 sec	Operations	Cognitive Band
10^{-1}	100 msec	Deliberate Act	
10^{-2}	10 msec	Neural circuit	
10^{-3}	1 msec	Neuron	Biological Band
10^{-4}	100μs	Organelle	

1. Introduction

One goal of this research is to use EEG to track what is happening as someone performs a complex task. The advantage of EEG over other imaging modalities is its temporal resolution and it has been used in many controlled laboratory settings to track processes that happen in trials of no more than a few seconds. While such studies have some advantages, we are interested in tracking cognition in longer, more opened-ended tasks where events emerge as an interaction between an agent and the environment. This is what happens in more naturalistic and ecologically valid contexts such as driving, decision-making in complex social interactions, or problem-solving in uncontrolled settings. These are circumstances that allow us to study how the detailed processes revealed in the short-trial-based studies are combined to produce coherent behavior. Understanding how to use EEG in such tasks also has applications to development of brain-computer interfaces (for reviews, see Abiri et al., 2019; Lotte et al., 2018)

Given the time scale over which such tasks play out, one needs to consider the temporal detail to which one aspires in tracking cognition. It is useful to place such aspirations into Newell (1990)'s scales of cognition (see Table 1). The open-ended tasks reside in his

Rational Band, which ranges from minutes to hours. Below the Rational Band is the Cognitive Band, which he placed as spanning from Deliberate Acts which happen at the rate of around 100 msec to Unit Tasks which happen at the rate of 10 seconds. It is our belief (Anderson, 2002) that applications can be achieved by decomposing what is happening at the Rational Band into activities happening in the Cognitive Band. Given the temporal resolution of EEG, it might be possible to track events within the Cognitive Band.

The Cognitive Band is the focus of many cognitive architectures (Kotseruba & Tsotsos, 2020, for a review), which break Rational Band activities into performance of tasks at the Cognitive Band. We will be working with the ACT-R architecture (Anderson, 2007; Anderson et al., 2004) to find an interpretation of the subject's behavior on a specific task in terms of execution of activities in the Cognitive Band. This paper will report considerable success in doing so at the Unit-Task level of the Cognitive Band, limited success at the Operations level, and a significant roadblock at the Deliberate-Act level. While, in a certain sense, the results are specific to ACT-R (and many other details of our approach), they are representative of the issues in tracking cognition in open-ended tasks.

The open-ended task in this paper is a video game. Video games are becoming increasingly popular vehicles for the study of cognition (e.g. Altarelli et al., 2020; Boot, 2015; Gray, 2017). Many video games are excellent examples of open-ended tasks, in that behavior emerges as an interaction between the game software and the players. The video games we have studied have involved a high rate of action from the players. Such games combine the learning of complex strategies and perceptual-motor skills. Dealing with the demands of such games has led to advances in the ACT-R architecture (Anderson et al., 2019, 2021; Gianferrara et al., 2021). The high rate of action also provides a solid ground truth for judging the accuracy of tracking from EEG.

1.1. Using EEG to Track Cognition

While the more typical application of EEG for tracking cognition in open-ended tasks has been to identify relatively enduring states like fatigue in driving (e.g. Jap et al., 2009). or workload while managing a system (Baldwin & Penaranda, 2012), our concern is the identification of specific cognitive events localized in time. We have had success in localizing events like memory retrieval in short trials of a few seconds (Anderson et al., 2016; Borst & Anderson, 2021). In these well-defined tasks, a combination of multivariate pattern analysis (MVPA) and hidden semi-Markov analysis (HSMM) can localize

Deliberate Acts with fair precision. This approach treats the Deliberate Acts as forming a semi-Markov process where probability of the next act and the time to that act depend only on the previous act. The MVPA provides the probability that the EEG signal reflects a particular act at each point in the trial. The HSMM combines these probabilities with estimates of the temporal distribution of the acts to localize the acts in the trial.

We were not able to extend these methods directly to open-ended tasks. Part of the difficulty is that as tasks get longer, uncertainty increases about temporal location of the acts which overwhelms signal-to-noise ratio (Fincham et al., 2020). The other difficulty is that there are serious violations of the semi-Markov assumption in open-ended tasks. The next step along any path through the task depends not just on the last action but the state of the environment which reflects effects of prior actions. In most open-ended tasks that set of possible states is too large to represent in a practical HSMM.

Anderson et al. (2020) reported what they called the **sketch-and-stitch** method for localizing acts in such a task. It is built around **critical events** that happen relatively infrequently (at the Unit-Task level time scale) and whose distribution can often be given a semi-Markov treatment. Also, these critical events often produce strong EEG signals. An HSMM-MVPA can localize the critical events with relatively high levels of precision. The critical events so localized provide a **sketch** of what happened. While such a sketch is sufficient for some purposes, we can go on to **stitch** in a hypothesis about lower-level processes that led from one critical event to another. The steps at this lower level depend on much more than the prior step, seriously violating the Markov assumption. Rather, we use a computational model that represents how the next step depends on past steps. There tend to be many sequences of model actions that will produce the same transition between critical events. Capitalizing on the fact that cognitive acts are associated at least weakly with EEG activity, we can use EEG to judge the probability of different candidate sets of lower-level cognitive steps.

The video game in this paper is a version of the Space Fortress game (e.g. Mané & Donchin, 1989) which has been used for decades to test training regimes: subjects fly a ship around a central fortress trying to destroy it without being destroyed by the fortress. In the last decade, detailed modeling of Space Fortress has become the target of cognitive modeling because it reflects a complex mixture of perceptual, motor, and other skills, how they are integrated, and improve with practice. Anderson et al. (2020) demonstrated the sketch-and-stich method with a rather simple version of the game which is called Autoturn. While the choice of simplicity was strategic for the initial exploration, its simplicity had limitations for purposes of a strong evaluation. There were significant

constraints on key patterns and flight path, leaving the game at the low end of open-endedness.

In video games like Space Fortress, both subjects and models can show great variability in the detail of their behavior. Faced with this, the typical model evaluation is to compare statistics averaged from subjects with statistics averaged from the model. However, this method leaves open the question of whether a model that produces these averages corresponds to any specific subject (Siegler, 1987) There are methods for comparing simulation models like ACT-R with individual subject behavior(Fisher et al., 2020), but they cannot extend to open-ended tasks like video games. This paper will assess an ACT-R model by comparing average statistics, but also by addressing the question of whether it can produce the specific behavior that subjects show in specific games.

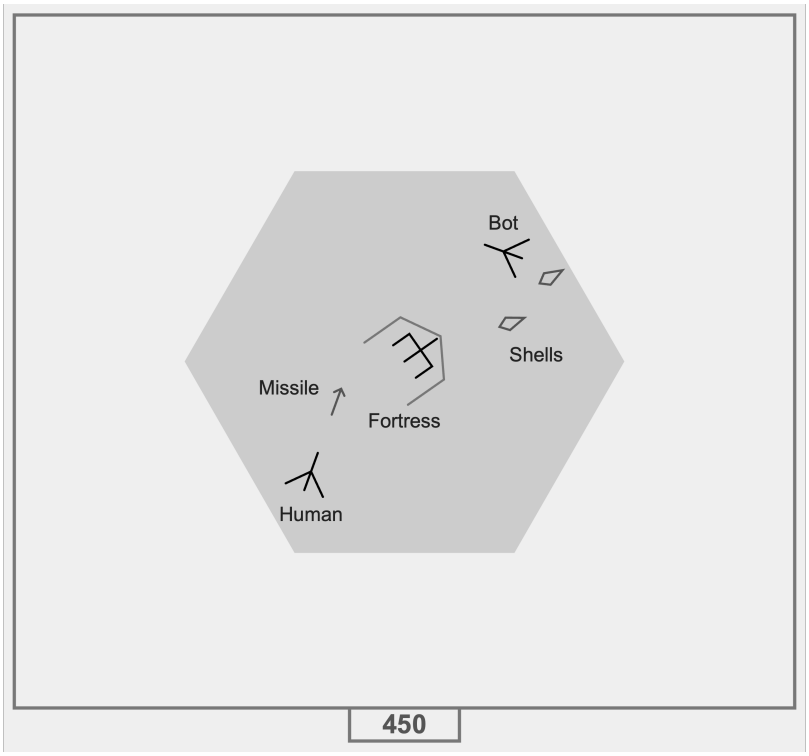

Figure 1: The Co-Op Space Fortress screen, showing the hexagonal battle area with the Fortress in the middle. The Fortress has turned and has shot shells at the bait. The shooter Is behind the Fortress and has shot a shell at the exposed rear of the fortress.

1.2. Co-op Space Fortress

The video game in this paper is a 2-player version of Space Fortress, called Co-Op Space Fortress[1], which has a recent ACT-R model (Dimov et al., 2023). The data in this paper provide a further test of the Dimov model. In Co-Op Space Fortress, depicted in Figure 1, two players control two identical, but differently colored spaceships and their goal is to destroy a fortress located in the middle of a grey hexagon. The distinctive feature of Co-Op Space Fortress is that the two players need to coordinate their actions to destroy the fortress. The fortress is surrounded by a shield, which is impenetrable to a ship's missiles, but the back portion of the shield disappears when the fortress fires shells at a ship. One player, the bait, tempts the fortress to shoot at it and so expose its rear. Then the other player, the shooter, can move behind the fortress and shoot it with a missile.

[1]Available for demo at http://andersonlab.net/demos/coop-eeg-paper-2023/

As in Space Fortress, these spaceships fly in a frictionless environment and are controlled with four keys: W to thrust, D and A for clockwise and counterclockwise turns, respectively, and the space bar to launch a missile. The Appendix describes the detailed effects of these key presses as well as other details of the game. Navigating the ship without friction is counterintuitive. For example, the ship never stops unless the player turns 180 degrees relative to the flight direction and thrusts for just the right amount of time. Consequently, as in previous Space-Fortress-based games, learning to navigate has been a major challenge. This is even a greater challenge in this game because the needed flight paths are much more open, reflecting a dynamic game situation that is partly determined by the other player.

At the beginning of each game, the two players start outside (i.e., to the left and right of) the gray hexagon. They both need to enter the hexagon, because it defines the effective range of fortress shells and ship missiles. The fortress will not take aim at the bait while it is out of the hexagon and the shooter cannot destroy the fortress when it is outside of the hexagon. If the fortress is destroyed, 100 points are gained and both players must leave the hexagon before the fortress respawns and they can attack again. Players lose 100 points if a ship is shot by the fortress or if the ship crashes into the fortress or the outer boundary of the game. After a ship death, the ship will respawn 1 second later in the starting position. In addition to points gained by kills and lost by deaths, 10 points are lost for every shot that does not destroy the fortress.

Unlike in the original Co-Op game, one player is always the shooter and the other always the bait. In the current study the bait is an artificial agent developed in Dimov et al. (2023), the bot, to minimize variation in success due to the second player. The motor timing of the bot was based on the ACT-R model including randomness in exact motor execution. It basically behaves as a highly trained model and was better than most subjects or models.

The ACT-R model for Co-Op Space Fortress leveraged many of the features of past models to play other Space Fortress games (see Dimov et al., 2023, or a detailed description of the model). As other ACT-R models of skill acquisition, it starts with a declarative representation of how it should act. As role choice is fixed in this version of the game, this declarative representation is simpler than in the original Dimov model, because it skips information about role determination. The model transitions with practice from a beginning stage of retrieving this knowledge and deciding how to use it in the game to a final stage of **direct actions** which simply recognize what to do in a game state. Also, with practice the model learns control parameters that determine exactly how it performs

actions – e.g., how fast to fly or what an adequate shooting aim is. A major extension of this model over previous models was learning to predict where the ship would be in the future and where the other ship would be. This information is used to plan and adjust flight directory.

EEG data were collected after subjects (and models) had achieved a fairly stable and successful level of play. Subjects played a series of 30 3-minute games. The first 10 games were played without EEG and subjects had to average at least 200 points per game in games 8-10 to go onto the final 20 games which were played with EEG. Most of the learning for these successful subjects was complete by the 10th game.

2. Methods

2.1. Subjects

A total of 27 subjects were recruited from the CMU population of students and staff between the ages of 18 and 40. 5 subjects were excluded because they failed to meet the performance criterion on the first 10 games, leaving 22 subjects (13 male, 9 female, mean age 23.9 years). All were right-handed. None reported a history of neurological impairment. Subjects were paid $60 for participation in the experiment that lasted about 2 hours. In addition, they earned a bonus for performance (4 cents per 100 points, total bonus range $3.41 to$ 21.29 with an average of $13.43). All subjects signed informed consent and the experimental procedure and data handling was approved by the ethics committee of Carnegie Mellon University.

2.2. Game Play

After subjects studied game instructions, they played 10 3-minute games choosing to move on to the next game at their own pace. If they passed the criterion of an average of at least 200 points in the last 3 games, they went into the EEG lab and were fitted with the headset. Then they played 20 more games again at their own pace. The game records every 60th of a second the state of the game (e.g., whether the ships and fortress are alive, where the ships are and their direction and speed of movement, whether shells or missiles are on the screen, whether a key is pressed, and whether the fortress has opened its shield). This serves as the ground truth both for training the EEG decoder and for testing its predictions.

2.3. EEG Analysis and Classifier

The EEG was recorded from 64 Ag-AgCl sintered electrodes (10-20 system) using a Biosemi Active II System (Biosemi, Amsterdam, Netherlands). The EEG was re-referenced online to the combined common mode sense (CMS) and driven right leg (DRL) circuit. Electrodes were also placed on the right and left mastoids. Scalp recordings were algebraically re-referenced offline to the average of the right and left mastoids. The EEG and EOG signals were digitized at 512 Hz and were subsequently filtered with a bandpass filter of .1 to 40.0 Hz. and were digitized at 512 Hz. The vertical EOG was recorded as the potential between electrodes placed above and below the left eye, and the horizontal EOG was recorded as the potential between electrodes placed at the external canthi. The EEG recording was decomposed into independent components using the EEGLAB FastICA algorithm (Delorme & Makeig, 2004). Components associated with eye blinks were automatically identified and manually confirmed. All but the marked ICAs were projected back to the EEG signal.

The EEG signal was recorded continuously for the entire experimental session and broken into 3-minute games. To match the rate of game recording, the data were down-sampled to 60 Hz with default EEGLab anti-aliasing filtering applied (FIR low-pass filter, 45 Hz cutoff frequency (-6dB) and 10 Hz transition bandwidth). We excluded any 60 Hz sample that had an electrode with a value below -500 microvolts or above 500 microvolts. This resulted in loss of data for 60 of the 440 games (22 subjects x 20 games). For these cases the number of samples lost from the 10800 ticks (3 minutes x 60 seconds x 60) varied from 1 to 1591 with a mean of 244. A one-second window around each game tick (30 game ticks before, the game tick, and 30 game ticks after) was used to classify whether a game tick contained a critical event. This means that each game tick had associated with it a vector of 61*64=3904 electrode readings, representing regional effects, frequency effects (below 30 Hz), and their interactions. Because the vector associated with a game tick requires complete signal for 1 second, game ticks at the beginning and end of a game do not have corresponding vectors, nor do game ticks in or near periods of deleted signal. The available vectors for each game were z-scored to standardize them across games. To reduce dimensionality and filter out noise, the vectors for the games of each subject received principal component analysis (PCA) and the 1000 top dimensions were kept. Depending on how many points were excluded, the result was 8,572 to 10,740 (mean 10,689) 1000-element vectors associated with game ticks. These are what were used for all classification analyses. The PCA and classifications were performed on a per subject basis, which resulted in better classification results than combining data over

subjects in Anderson et al. (2020). Not surprisingly, EEG patterns are somewhat distinct among subjects, perhaps reflecting true subject differences or perhaps differences in exact locations and recording properties of the electrodes for different subjects. Nonetheless, the EEG patterns are somewhat similar across subjects, again not surprising.

Two linear classifiers were trained, one for identification of events at the Unit-Task level and one for identification at the Operation level. These classifiers learned the conditional probabilities of the EEG patterns associated with different categories of tick events. Assuming normal distributions of the 1000 PCA features for any category, the classifier estimated the parameters of the distribution and so could map any vector onto conditional probabilities for each category. For any game that the subject played, the parameters were estimated from the other 19 games of that subject and used to classify the vectors of the target game. The Unit-Task classifier learned 4 categories: the patterns associated with kills, deaths, and missed shots, plus the pattern of other ticks called **Null** events. The Operations classifier learned 3 categories: ticks where the subject began a new flight path, ticks were the flight path ended, and Null events. Most of the game ticks are Null events (an average of 99.8% of the ticks for the Unit-Task classifier and 98.6% of the ticks for the Operation classifier). Anderson et al. (2020) only used a subset of the Null events for training. However, using all Null events leads to better estimates of the covariance matrix used by the linear classifier.

2.4. Model Data

We ran the mode[2] using the same combination of 25 individual-difference parameters as in Dimov et al. (2023). Even within a particular individual difference setting, there was substantial variability in the games the model played, reflecting randomness both in the model and the bot. We ran 300 models for each of the 25 parameter settings for a total of 7500 model runs. However, 206 of these models failed to meet the criterion of an average of 200 points over games 8-10 that was used with subjects. Those were removed, which resulted in 7294 model runs of 30 games each, creating a library of 218,820 games which were used at the stitch level of the analysis.

3. Results

Figure 2a shows the growth in points by subjects and models who met the selection criterion. The EEG data come from games 11 through 30. Most of the learning seems

[2]Available at

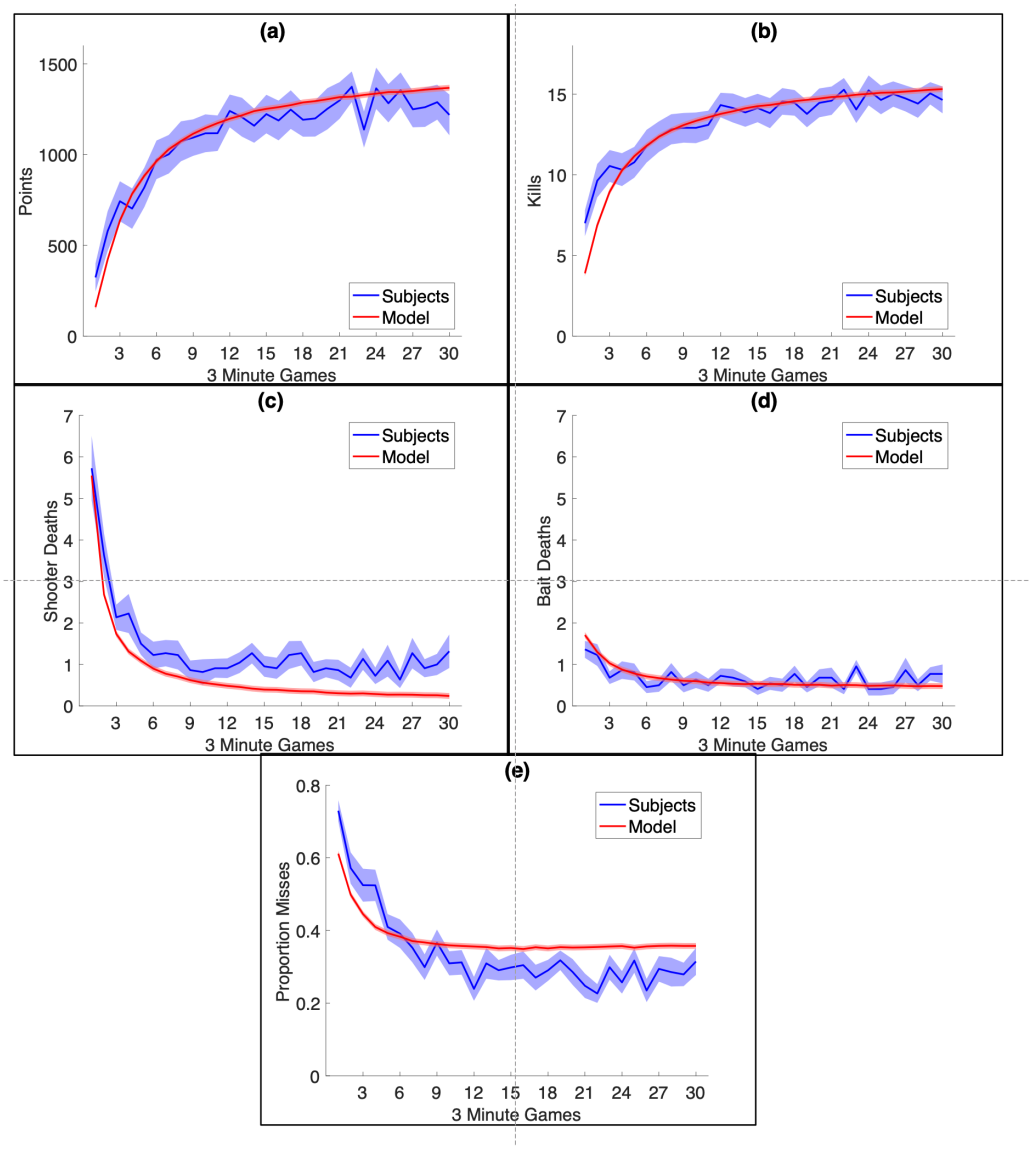

Figure 2: Mean values (line) and standard errors (area around lines) per game for subjects and models as a function of game (a) Points; (b) number of fortress destructions; (c) number of shooter (human) deaths; (d) number of bait (bot) deaths; (e) Proportion of shots that are misses.

complete by game 11, although there is a weak but significant increase in performance after the 10th game (6 points per game, t(21)=2.97, p ¡ .01). Points reflect kills (+100 points) minus deaths (-100 for either shooter (human) or bait (bot) deaths) minus misses (-10). The majority of deaths (66.5%) are deaths of the human shooter. Figures 2b-2e display the data for these subcomponents of the score. The figures also show the data from the model, which generally match up, although the model reaches a lower death rate in Figure 2c.

The remainder of this Results section will consist of a more detailed discussion of this task at the three levels in Newell's Cognitive Band: first the Unit-Task level, then the Operation level, and then more briefly the Cognitive-Act level. We will be concerned with games 11-30 where EEG was collected.

3.1. Unit-Tasks: Critical Events

The Unit tasks in this game are attempts to achieve a kill that can either end in a success (the kill) or a failure (miss or death). The deaths of interest are those of the human subject, who always played the role of shooter (Figure 2c). Because our EEG analysis requires 30 ticks around the event we excluded misses that were within 30 ticks of a kill or death (reducing their mean number per game from 5.8 to 3.8). Continuing the terminology of Anderson et al. (2020) we will refer to these three outcomes as **critical events**. These events are rare – on average only 19.2 of the 10800 ticks in a game are associated with these events.

3.1.1. EEG analysis

Figure 3 shows the activity of the three central electrodes for a second around the events plus the scalp profiles 19 ticks (.32 seconds) after the event. Figure 3d shows the average pattern for the remaining ticks which is essentially 0, reflecting the normalizing of the EEG signal. Figure 3 shows activity averaged over subjects, but note that the classifier estimates patterns subject by subject and individual subjects may show different patterns and the classification will exploit the subject-specific patterns. Nonetheless the relatively narrow standard errors displayed in Figure 3 indicate somewhat consistent patterns.

These patterns associated with the critical events in Figure 3 seem be variants of the P300 event-related potential (Verleger, 2020). They vary in the timing and magnitude of the positive deflection, with the largest response to deaths and the weakest to misses. While the peaks are the most striking feature, the classifier is based on the activity of all 64 electrodes over all 61 ticks. Essentially the 4 patterns in Figure 3 can be conceived of as

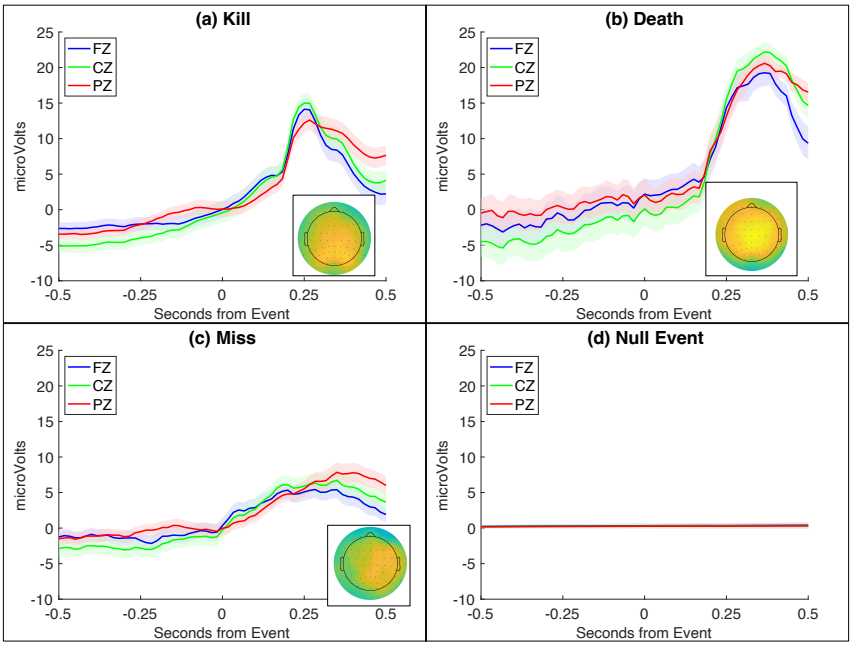

Figure 3: EEG activity (mean values and standard errors) of 3 central electrodes from a half second before a critical event to a half second afterwards. The scalp profiles illustrate activity at .32 seconds after the event. (a) Kill: scalp profile is scaled to range between -15 and +15 microvolts; (b) Death: scalp profile is scaled to range between -20 and +20 microvolts; (c) Miss: scalp profile is scaled to range between -10 and +10 microvolts; (d) Null Event.

occupying points in a 64x61=3904-dimensional space (or a 1000-dimensional space after the PCA compression). These four points in high-dimensional space can be projected down into a 3-dimensional subspace, and as it turns out, that 3-dimensional subspace can be projected down into 2 dimensions with little loss. Figure 4a illustrates the location of the points in a 2-D space with an attempt at an informative choice of axes and scale. The 0-0 point reflects the zero activity that the Null pattern approximates. The horizontal dimension reflects the magnitude of game points and the vertical axis represents loss. In terms of EEG activity, as one moves up one adds proportionally more of the pattern in Figure 4b and as one moves to the right one adds proportionally more of the pattern in Figure 4c. A death which is at the 1-1 point can be reconstructed as the sum of these two patterns. The two patterns 4b and 4c reflect 2 peaks of activity at somewhat different points in time – vertical dimension peaking 26 ticks after the event (.43 seconds) and the horizontal dimension peaking 14 ticks after the event (.23 seconds). The activity across the 64 electrodes at the two peaks are strongly correlated and no electrode shows a significant difference between the two peaks (maximum t for a difference is t(21)=1.53).

While Figures 3 and 4 display average patterns, the challenge for the classifier is to identify individual game ticks using signals which may depart substantially from the subject's average. Table 2 summarizes the success of the classifier using patterns from other games of that subject to classify a particular game. Tables 2a and 2b give the counts with which various types of ticks were assigned various labels. They use different thresholds for assigning a label to a tick. Table 2a assigns it to the category with the highest conditional probability while Table 2b assigns it to the category which is most likely given the different frequencies of the categories. Our preferred d-prime measure of discriminability (Wickens, 2002) is 2.63 for Table 2a and 2.59 for Table 2b[3]. Overall accuracy is 97.6% and 99.6% for the two tables. Table 2c gives the pair-wise d-primes reflecting discriminability of the pairs of categories and Table 2d gives the pair-wise area under the curve (AUC), which does not depend on threshold. The discriminability is quite high and better than in Anderson et al. (2020), which reflects, at least in part, the refinements in the procedure (training only using subject data, using all Null events in estimating covariance matrix). Despite their striking appearance in Figure 3, deaths are poorly identified because of their rarity. Classification is done on a per subject basis. The number of deaths per subject varies from 3 to 54 with a mean of 19.5. There is a

[3]This is calculated by collapsing the n x n matrix into n 2x2 matrices representing presence versus absence of the category and averaging the d-primes for the n matrices.

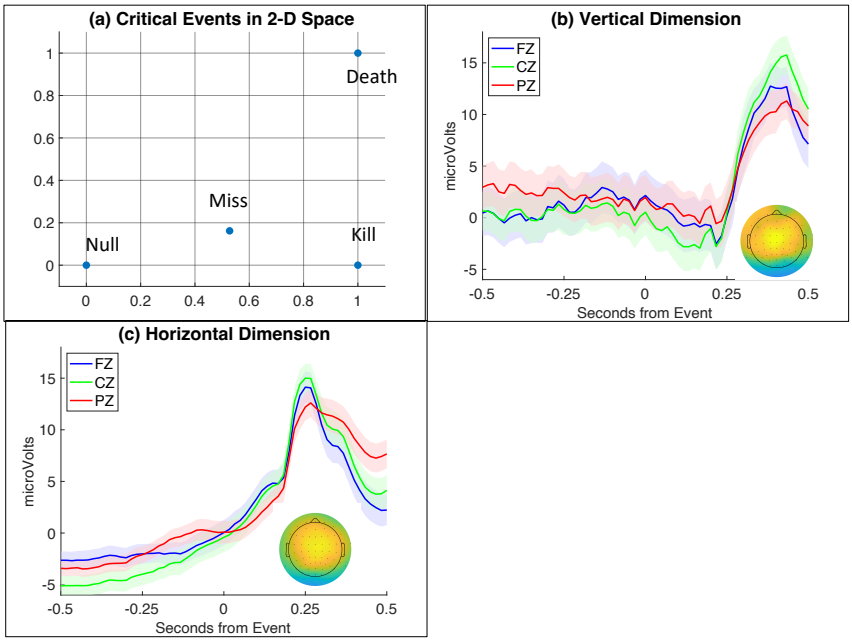

Figure 4: (a) A representation of the 4 patterns in Figure 3 in a 2-dimensional space. (b) A representation of the pattern associated with the vertical dimension – the scalp profile is at .43 seconds and is scaled to range between -15 and +15 microvolts. (c) A representation of the pattern associated with the horizontal dimension – the scalp profile is at .23 seconds and is scaled to range between -15 and +15 microvolts.

Table 2: Classification of Game Ticks according to Critical Events

(a) Classification by Likelihood

True Class	Null	Kill	Death	Miss
Null	4,632,704	34,824	2,225	73,801
Kill	603	5,664	30	43
Death	124	203	95	8
Miss	1,168	59	0	449

(b) Classification by Posterior

True Class	Null	Kill	Death	Miss
Null	4,726,963	13,004	695	2,892
Kill	1,539	4,782	15	4
Death	236	137	55	2
Miss	1,601	11	0	64

(c) Pairwise dprimes

Class 1	Kill	Death	Miss
Null	3.71	2.89	1.55
Kill		1.89	3.01
Death			2.12

(d) Pairwise AUC

Class 1	Kill	Death	Miss
Null	0.994	0.885	0.832
Kill		0.851	0.984
Death			0.894

(e) Three-way Classification

True Class	Kill	Death	Miss
Kill	6,175	36	129
Death	281	110	39
Miss	272	5	1,399

strong correlation (r = .809) between number of deaths for a subject and the proportion correctly classified.

Looking at average patterns in Figure 3, it is not surprising that these critical events can be discriminated with high accuracy from other Null game ticks. What may seem surprising is that they are quite discriminable from one another. Part (e) of Table 2 shows the three-way classification by likelihood which has d-prime of 2.48. Discrimination between the 3 critical events depends on capitalizing on their distinctive features as illustrated in Figure 4.

While good, the classification results by themselves do not provide a useful basis for identifying critical events. Many events are erroneously labelled as critical events in Table 2a. While Table 2b has somewhat fewer such false positives, it is at the cost of missing many critical events. The HSMM below will not use any particular assignment of ticks to categories, but rather will use the conditional probabilities of the tick pattern for each category.

3.1.2. Using an HSMM to Localize Critical Events

The function of the HSMM is to combine the probabilities from the classifier with information about the probability of these events happening at different time points. Figure 5 shows fitted distributions[4] between events and the probabilities of one event following another. Figure 5a includes both starts from the beginning of the game and

[4]Using the Matlab function ksdensity.

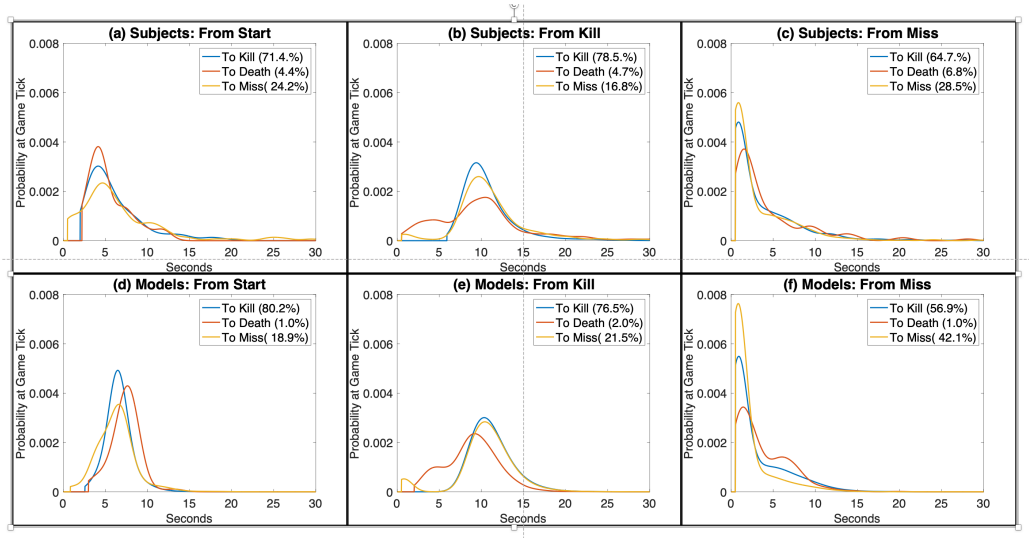

Figure 5: Probabilities of transitions (given in parentheses) between events and distribution of intervals between events in the simulations. (a)–(c) give the data from the subjects and (d)–(f) the data from the model.

from when the ship reappears outside the hexagon after a death. The ships take a fair amount of time (Figure 5a) to fly in from the start position and be in position to destroy the fortress. It takes even longer to achieve another kill after a kill (Figure 5b) because the two ships must first fly out of the battle zone and then back in. In other cases, one event can rapidly follow another (although the HSMM was set to impose a .5 second minimum interval between events). Figures 5d-5f show the corresponding distributions from the model. There are both similarities and differences in the distributions, but there is generally overlap which will enable the stitching efforts to follow.

While the distributional information shown in Figures 5a-5c is averaged overall subjects, the HSMM used distributions estimated from other games of the subject in fitting the data for any game from the subject. The HSMM also used conditional probabilities of the EEG that were estimated from other games of that subject. Such per subject estimation allows for tuning to the specifics of a particular subject's game play. The Viterbi algorithm (Rabiner, 1989) was used with the HSMM to identify the maximum likelihood placement of events in the target game. The Appendix explains the probability function that is being optimized by the Viterbi algorithm

In a semi-Markov model, the probability of the next event depends on the time since the last event. However, it remains Markov in that what happens after an event only depends on time from that event and not prior events or their timing. This assumption is

reasonable for the timing between kills and deaths. After a death the ship respawns at its starting position and carries with it no effect of what happened before the death. After a kill, critical events prior to the kill have little if any influence on what the next event will be or how long it will be. However, the Markov assumption becomes problematical when misses are included. For instance, suppose a miss intervenes between one kill and another kill. To achieve the next kill, the ship must fly outside the hexagon, wait for the fortress to respawn, and re-enter. It matters little where on this journey the ship has its miss. The time of the next kill depends on the time of an event 2 back, the prior kill, and not the event one back, the miss.

We used a hierarchical HSMM (Fine et al., 1998; Johnson & Willsky, 2013) to avoid violating the semi-Markov assumption. The first HSMM found the most probable locations of kills and deaths ignoring misses. Then with these anchored, another HSMM placed misses in the intervals using information about the distribution of these misses relative to the other events. This two-pass HSMM identified a mean of 14.62 kills per game compared to a mean of 14.41 in the actual games; a mean of 0.44 deaths per game compared to 0.98 actual deaths, and 1.49 misses compared to 3.81 actual misses. Maximum likelihood estimation is biased against rarer events.

For each game, as in Anderson et al. (2020), the match between the assigned events and the actual events was calculated as a sum of a recall and a precision measure (Buckland & Gey, 1994) calculated from locations of kills, deaths, and misses. The measure of recall focused on the events that occurred in the game and identified the closest predicted event. If that predicted event was the same category as the actual event and was within 1.5 seconds (90 game ticks), it was scored according to how many game ticks it was away –thus the maximum score was 90. If the closest event was further away or the closest event was a different category, the match for that event was also scored 90. The average of these recall scores for a game can vary from 0 (perfect match to all actual events) to 90 (worst possible). The measure of precision applied the same scoring procedure but now started with all predicted events and found the closest actual event. Figure 6 shows the distribution of the recall and precision scores. The recall score tends to be larger because the reconstructed games tended to have fewer events, leaving fewer potential matches to the actual events.

The average measure of recall was 24.9 (median 22.8) and the average measure of precision was 15.3 (median 12.0) for an average match rating of 20.2 (median 17.0). This amounts to being about one-third of a second off on average. The average match rating of 20.2 between a game and its reconstruction compares to an average match of

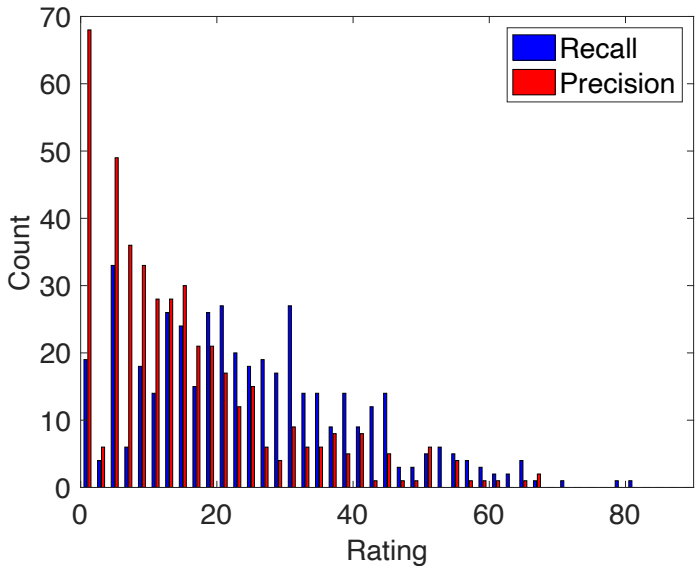

Figure 6: Distributions of recall and precision ratings on a scale where 0 is perfect match between game and its inferred sketch and 90 is the maximum possible mismatch score.

79.6 to reconstructions of other games. All but two games were best matched by their own reconstructions over reconstructions from other games. The remaining two games' second best match was their own reconstruction. The accuracy of the reconstructions is considerably better than Anderson et al. (2020), even though that attempt was only localizing events in a 1-minute period rather than a 3-minute period. The improvement reflects two refinements in procedure – performing these analyses on a per subject basis and dealing with the violation of the Markov assumption. Aspiring to predict critical events to a 60th of a second is a high standard. Still 49.6% of all events are being identified to the game tick and 76.2% of the events are being identified to within a tenth of a second (6 ticks).

Nonetheless, the reconstructions themselves are not perfect. Even the best reconstruction has a non-zero match rating (.11 reflecting all 18 events predicted but two off by 1 game tick). Figure 7 illustrates the reconstruction of games whose ratings span from the 5th percentile (better scores) to the 95th percentile (worse scores). As seen in these examples, the best performance is achieved for kills, which are the most frequent events.

Figure 8 compares the scores subjects would have gotten for their games and the score their game reconstructions would have gotten (ignoring bait deaths in both cases). The correlation is strong although the HSMM tends to overpredict subject scores (mean

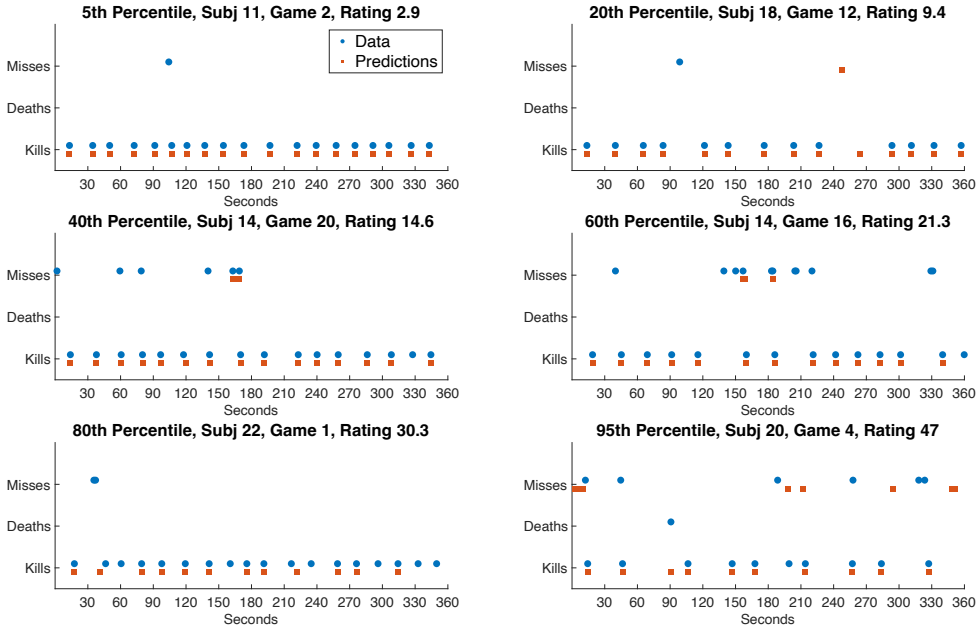

Figure 7: Examples of differential success in reconstructing the critical events of a game.

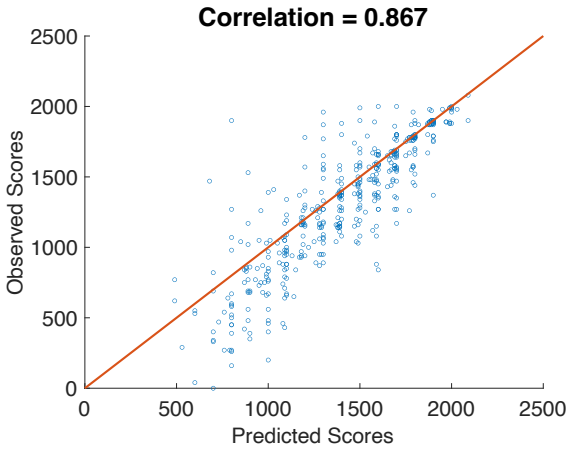

Figure 8: Relationship between points earned in the actual game versus the EEG-based reconstruction of the game.

score is 1404 points for HSMM compared to 1315 for subjects). The overestimation reflects the fact the HSMM underestimates the infrequent categories of deaths and misses. Nonetheless, Figure 8 gives testimony to the ability of the HSMM analysis to identify how well a subject is doing in a particular game.

3.1.3. Stitching in Kills and Deaths.

While the critical events represent highlights of the game, they do not represent anything like the details of the subject's thinking that resulted in these critical events. The detailed cognitive steps contain major violations of Markov assumptions in that the next step of cognition depends on much more than the last action. Thus, there is no hope of extending the Markov approach to produce more detailed hypotheses about subject cognition. Rather, we use the ACT-R model to find a sequence of cognitive steps that would produce these critical events.

The following analyses involve the placement of three sets of critical events: those in the subject games, those in the HSMM reconstructions, and those in the model games. Matching subject critical events to model critical events addresses the question of whether the model is like subjects. Matching HSMM critical events to model critical events addresses the question of reconstructing behavior from EEG. As it turns out, the conclusions will not prove very different for these two goals, reflecting how close the HSMM critical events are to the actual critical events.

A major challenge is the variability in subject and model games, even at the coarsest time scale of the critical events level. No model or subject game came close to matching any other model or subject game in terms of the placement of critical events. Using the same assessment scale where HSMM reconstructions averaged 20.2 in their match to the subject games (and the best match was .11), the best match score between 2 model games was 22.5 (from over 20 billion pairwise comparisons). The best matches to the 440 subject games selected from the 200,000+ model games averaged 49.1[5].

Rather than trying to find a complete model game that matched a target set of critical events, the Sketch-and-Stitch approach tries to stitch together segments from different model runs to achieve a match. A segment goes from one critical event to another. We focused on stitching in segments that bridge between kills and deaths because these place navigational constraints on the flight. Usually, there were segments in model runs that exactly matched the length of segments in subject games (or as defined by the HSMM)

[5]The best matches to the 440 subject games selected from the other 439 subject games averaged 60.3.

but there were a few missed cases. In the case of subject critical events 24 segments (out of 7207) are not matched in length by any segment from the model. In the case of the HSMM reconstructions 3 segments are not matched. For both subjects and HSMM, the missing segments were all very brief or very long. Very brief and very long intervals are rare and even with 200,000+ model runs some possibilities are not covered. The model can generate these extreme duration critical events, but they depend on such an improbable sequence of model steps that they did not show up in the 200,000+ games[6].

In each of its steps the stitching algorithm takes a game **fragment** that it has stitched together as a match to the first n critical events and adds in a new segment to get a now longer fragment that will match the first n+1 critical events. The Appendix describes the details of the stitching algorithm, which used segments that were close when perfect matches could not be found. A constraint on this stitching is that the resulting flight path must be viable. Many stitched paths are not **viable** because when the actions in the segment are added to the fragment they send the ship crashing into the fortress or the outer border before the next critical event. When the segment is supposed to end in a kill, the resulting path might not put the ship into a position where it can achieve a kill. When the segment is supposed to end in a death, the resulting path might not end in a ship death. The constraint of viable paths eliminates many combinations but usually not all. Typically, there are too many viable stitches to practically compute them all, and so the algorithm uses a beam search carrying forward a select subset of the fragments at each iteration of stitching. The algorithm did not always produce perfect matches to the critical events of subjects or games, but the best matches were close. The average mismatch between event boundaries and stitched boundaries was .58 ticks for subject critical events and .07 ticks for HSMM critical events.

In conclusion, for all intents and purposes, stitching can produce model runs that match the critical events in the game play of subjects. Also stitching in games to match the HSMM critical events produces model runs that are as close to the data as the HSMM critical events were. These model runs offer hypotheses as to the detailed cognitive events that were happening during the game. The next subsections addresses whether these model details match subject details.

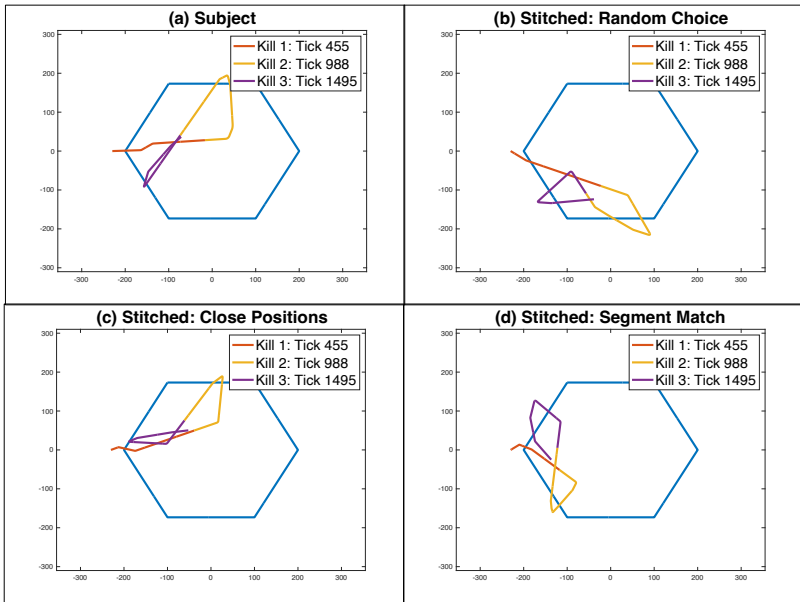

Figure 9: Example beginning flight paths: (a) a subject game; (b) – (d) examples of matching stitched games.

3.2. Operations: Legs of Flight

Figure 9a illustrates the flight path that a subject took in one game while achieving the first three kills. The three segments (from start to first kill, from kill to kill) are color coded. The individual segments contain linear legs of flight, connected by brief periods of turning which are too short to discern in the flight path when examined on the scale of Figure 9a. The Appendix describes the algorithm for identifying legs of a flight, but they are often apparent in the flight path. In the model, and we assume in subjects, a switch from one leg to another reflects a change in subgoal: there are legs that get the subject into the hexagon, legs to get behind the fortress, legs to get out of the hexagon, legs to avoid crashing, and legs to correct these legs when they turn out not to achieve their subgoal (sometimes because of unexpected actions of the bait). These legs correspond at least approximately to Newell's Operation level within the Cognitive Band (Table 1).

3.2.1. Leg Structure

The analysis of leg structure produces an alternating sequence of legs and turns. Figure 10 compares subjects and models in terms of several properties of these legs and

[6]For perspective, we note that there are 5388 extreme duration critical events from the model runs that are not found in any subject game or any other model game.

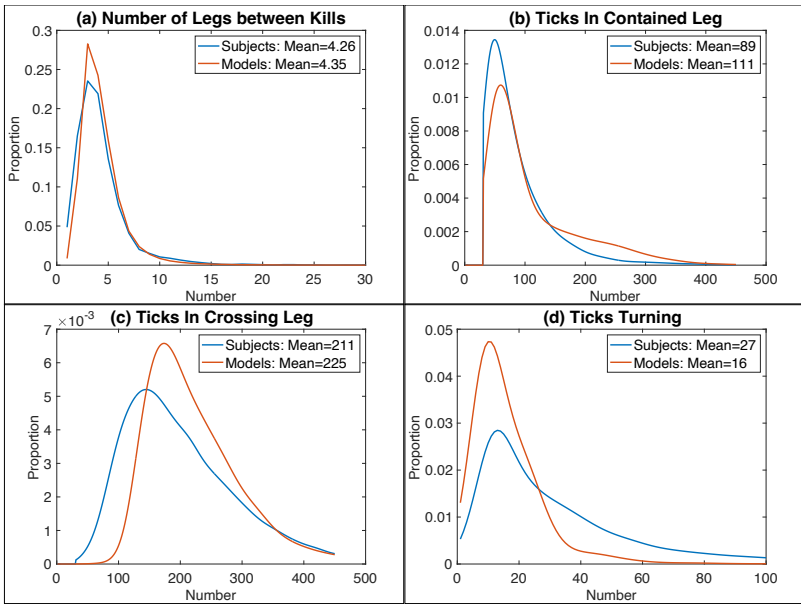

Figure 10: Comparison of subjects and models in terms of their leg structures.

turns. Figure 10a displays the number of legs contained inside kill-to-kill game segments. Both subjects and models average a little more than 4 legs per kill-to-kill segment. Figure 10c shows durations of legs that start before a kill-to-kill segment and end after the initial kill of the segment. These segment-crossing legs are much longer in both subjects and models. The models (and we assume the subjects) have typically achieved the goal of being on a trajectory that will put the ship behind the fortress well before the kill and are drifting in position to perform the kill. The models (and we assume the subjects) wait until they determine that they have achieved the kill and only then determine how to best exit before ending the long crossing leg. Figure 10d shows the distributions of the briefer times engaged in the turns between legs. While the model was designed before these data on legs, the distributions in Figure 10 display much overlap. However, the model does not show the very brief crossing legs that subjects occasionally (Figure 10c) have nor the very long turns that subjects occasionally have(Figure 10d). The later section on Cognitive Acts will discuss the source of these mismatches.

3.2.2. Stitching to Match Leg Structure

While segments can be stitched together to match the critical events, the detailed model behavior can be quite different from the subject. Figure 9 illustrates this for a subject game that began with 3 kills. Part (a) illustrates a subject's trajectory while

Table 3: Matching Leg Structure

Using Actual Critical Segments

(a) Best Possible Stitching

	Beginnings	Ends
Hits	69.3%	68.3%
False Alarms	23.2%	23.9%
d-prime	1.24	1.19

(b) Random Stitching

	Beginnings	Ends
Hits	46.7%	44.2%
False Alarms	37.6%	39.3%
d-prime	0.23	0.12

(c) Stitching Informed by EEG

	Beginnings	Ends
Hits	40.8%	38.9%
False Alarms	30.2%	31.5%
d-prime	0.28	0.20

Using HSMM Critical Segments

(d) Best Possible Stitching

	Beginnings	Ends
Hits	65.7%	67.3%
False Alarms	26.1%	25.0%
d-prime	1.04	1.12

(e) Random Stitching

	Beginnings	Ends
Hits	46.8%	44.5%
False Alarms	38.2%	39.9%
d-prime	0.22	0.12

(f) Stitching Informed by EEG

	Beginnings	Ends
Hits	42.1%	40.8%
False Alarms	32.5%	33.5%
d-prime	0.26	0.19

parts (b) through (d) illustrates three obtained by stitching segments from model games to match the timing of these critical events. Typically, there are many stitched flight paths that are equivalent or nearly equivalent in terms of matching when the critical events happened for the subject. The stitched flights in Figures 9b-9d are all perfect matches in the temporal placement of the three kills. The first stitched path in (b) was a random choice from the many available with matching segment lengths, while the paths in (c) and (d) were chosen because they were similar to the subject path, but on different criteria.

The path in Figure 9c was chosen to be physically close to the subject path. However, differences in the position of the ship may not reflect significant differences in what the subject is trying to do. Small differences in timing of keystrokes can accumulate and result in the ship being in quite different locations. Two segments can be in opposite sides of the space and reflect identical intentions and even involve identical keystrokes.

Figures 9a and 9d are most similar in terms of the timing of their leg structure, which taps subject intentions in flight. It is not apparent from viewing the flight paths, but all 9 legs in Figure 9a begin and end within 30 ticks of a beginning or end of the legs in Figure 9d (although 9d has 3 extra legs). By this criterion Figures 9b and 9c only match 5 of the beginning of legs and 5 of the ends of legs in Figure 9a.

We developed a measure of how well the beginnings/ends in two games matched using

hits and false alarms. A hit was a beginning or end of a leg in the stitched game that had a beginning or end within half a second (30 game ticks) in actual games. Otherwise, the beginning/end was classified as a false alarm. Proportion hits were calculated as hits divided by the number of beginnings/ends in the actual game. Proportion false alarms were number of false alarms divided by the number of blank periods in the actual game. From these proportions we calculated a d-prime measure of match. Table 3a shows the result when stitchings are selected to optimize this measure.

While these results are far from perfect, they reflect the best that can be obtained with the current library of 200,000+ games.. That library usually provided a good number of segments (an average of over 3,000) that matched the length of actual game segments. However, only some of these would be viable when stitched in a game. Most severely, only 0.65% of the kill-to-kill segments were viable when added to a fragment. There is a relationship between how many model segments there are to match segments in the game and how well we can reproduce the leg structure of the game. The correlation between the average number of matching segments in a game and the d-prime measure of leg correspondence was .81. If we had run even more model games, the results would have improved somewhat.

Table 3b shows leg matches for random stitching. The d-primes are much lower, but 19 of the 22 subjects have positive d-primes. These small but positive d-primes show the advantage of just matching the position of the critical events . The results in Tables 3a-3c contain the leg matches using the actual critical events to identify segments. Tables 3d-3e contain the corresponding results when using the critical structures as identified by the HSMM. The results are quite similar, reflecting the fact that the critical segments identified by the HSMM are quite close to the actual critical elements.

3.2.3. Using EEG to Identify Leg Structure

There was a modest level of correspondence when model games were stitched to best match the leg structure of the target game. However, this stitching used knowledge of the target leg structure. How well could we identify the leg structure using only the EEG signal? Figures 11a and 11b show the patterns associated with the beginning and end of legs. These patterns are much weaker than the patterns associated with the critical events (Figure 3). Table 4 shows the results using the same classification methodology that was used for critical events (Table 2). Despite the weaker signal, there is still some discriminability among the categories. The classifier only uses other games of a subject to classify a target game and these may have some distinctive properties not in the

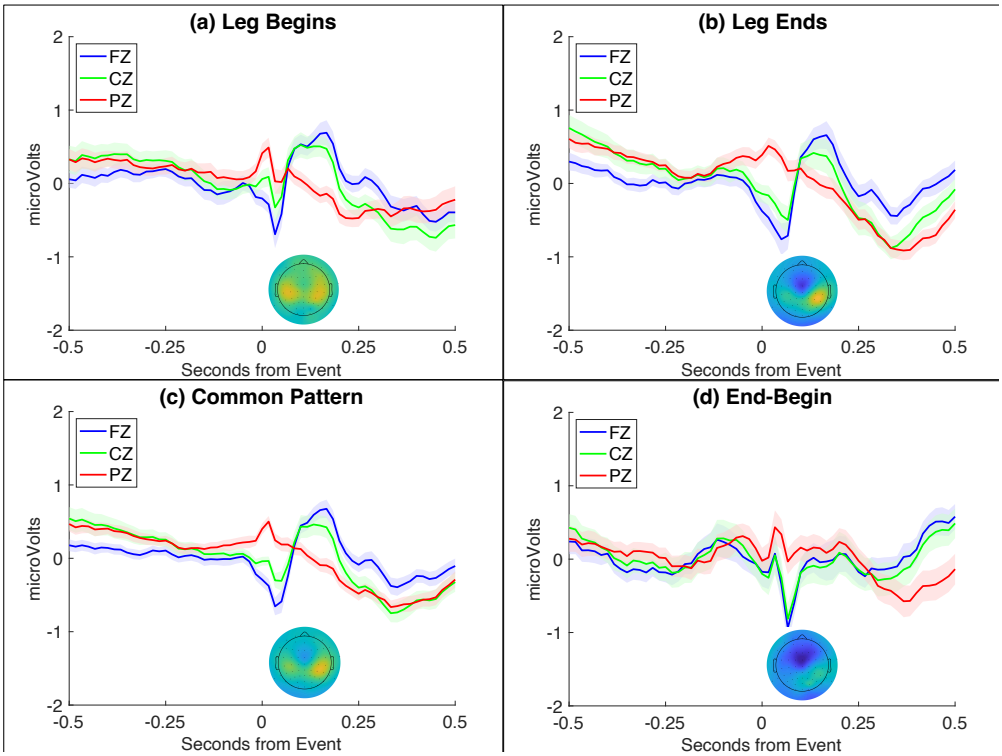

Figure 11: (a) EEG activity (mean values and standard errors) of 3 central electrodes from a half second before the beginning of a leg; (b) EEG activity (mean values and standard errors) of 3 central electrodes from a half second before the end of a leg; (c) Common Pattern (c) Difference between ends and beginnings. The scalp profiles in all cases illustrates activity at .1 seconds after the event and are scaled to range between -1 and +1 microvolts.

Table 4: Classification of Leg Boundaries

(a) Classification by Likelihood

True Class \ Labelled by Classifier	Null	Begin	End
Null	3,069,314	831,599	736,596
Begin	10,288	18,102	4,341
End	8,829	4,343	19,202

(b) Classification by Posterior

True Class \ Labelled by Classifier	Null	Begin	End
Null	4,617,154	9,473	10,882
Begin	30,587	2,086	58
End	29,166	59	3,149

(c) Pairwise dprimes

Class 1 \ Class 2	Begin	End
Null	1.08	1.27
Begin		1.46

(d) Pairwise AUC

Class 1 \ Class 2	Begin	End
Null	0.783	0.822
Begin		0.858

averages displayed in Figure 11. Figures 11c and 11d attempt to display the basis for the discriminability in the average data. Figure 11c shows the average pattern displayed by EEG activity across both begin-leg and end-leg events, which is part of the basis for discriminating these points from other Null points. It is somewhat surprising that there is a common pattern as the beginning of a leg reflects the end of the navigational keying that set the ship on its new path while the end of a leg reflects the onset of a new keying to set the ship on a new path. If anything, one might have expected the pattern prior to the beginning of the leg (x = 0) in Figure 11a to match the pattern posterior to the end of the leg in Figure 11b. Figure 11d displays the difference between the activity patterns for the beginning and end of legs. The activity in frontal electrodes is more negative 100 msec. after the end of a leg than 100 msec. after the beginning of a leg. The negative dip for electrode FZ is significant across subjects ($t(21)=3.59$, $p < .005$).

While the discriminability is not as high for the markers of legs as it was for critical events, there are many legs in a game which in combination can be quite discriminative. We calculated the probability of the EEG signals for each of the 440 games given each of the 440 leg structures of the games (a 440 x 440 matrix of probabilities)[7]. In 421 cases the EEG signal from that game is most probable given the leg structure of that game. The average rank of the game's reconstruction among the 440 was 1.7 and the lowest rank was 88. Thus, even though Table 4 shows limited ability to identify where individual legs begin or end, combined over a 3-minute game the leg structures result in

[7]This probability is conceptually the product of all the probabilities of the labels assigned to the 10,800 ticks by the leg structure. In implementation, we more efficiently calculate the log of a quantity that is proportional to this product.

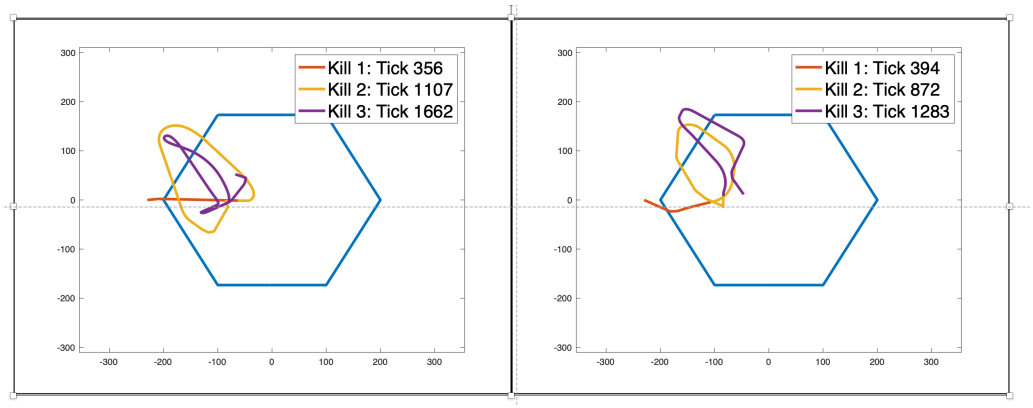

Figure 12: Two examples of curved flight from a subject with a high tendency to overlap turn and thrust keys.

highly discriminable EEG signatures.

While the EEG signal associated with key presses provides a good basis for identifying a subject's game, how well can it do in guiding how to stitch together a game? To address this question, we selected segments for stitching whose leg structure made the EEG signal associated with that game most probable. The results (Table 4c) prove to be only modestly better than random stitching. 19 of the 22 subjects do have better d-prime leg matches with the EEG signal than with random stitching, indicating that the modest gain is still quite significant ($p < .001$ by a sign test). The stitchings we found were the best given the library of model games and the beam search algorithm for finding a stitching. We would have found better matches to the EEG signal had we a larger library and searched with a larger beam. However, already the leg structure of 37 of the stitched games makes the EEG signal more probable than the leg structure of the actual games. The average leg d-primes for these 37 stitched games is not better than for the remaining 407 (.234 vs .236). Thus, while the EEG signal for the leg structure of a game tends to identify it from other actual games, it is quite possible to stitch together games that make the EEG even more probable and still have only a weak correspondence to the leg structure of the real game. The EEG signal does not provide enough guidance in this context to identify a close match to the leg structure.

3.3. Deliberate Acts: Individual Keying Actions

Flight legs, the topic of the previous section, correspond to the Operations level in Newell's Cognitive Band. The individual keying actions reflect steps at Newell's Deliber-

ate Act level[8]. When we examined the individual keying actions, we found a discrepancy between subjects and models. Subjects sometimes hold down a turn key (either clockwise or counterclockwise) while they are holding down the thrust key. While all subjects show some key overlap, 8 subjects showed such overlap less than 1% of the time that they were pressing the thrust key. On the other hand, 4 subjects held a turn key down while thrusting more than 10% of the time[9] . While the ACT-R architecture is capable of producing such overlapping key presses, the model kept the keying strategy of earlier models that did not overlap keys. These were models of games where subjects rarely if ever overlapped keys, probably reflecting the different navigational demands of these games. Figures 12a and b display the flight paths for the first 3 kills in two games of the subject with the highest proportion overlap, 47%. As can be seen these flight paths contain long curved trajectories, although there are also linear legs. The algorithm, trying to parse the curved portions of subject paths into linear legs, produces the short crossing legs in Figure 10c that the model does not produce. It also produces the long turns in subject data that are not found in model data (Figure 10d)[10] . Overlapping keys is part of the reason for the less-than-perfect match between model stitches and subject games in Table 3a.

Given that such overlap is not present in the model, we decided not to pursue stitching games that matched subject behavior at the level of individual key presses. Given these cases of key overlap, it is hard to always define an EEG signature associated with the pressing or release of a key. Therefore, we also did not pursue an EEG analysis.

4. Conclusions

These results indicate both success and limitation with respect to the goal of tracking open-ended tasks at the various levels of Newell's Cognitive Band. The success involved identifying critical events at the highest Unit-Task level. Killing a fortress has the characteristics of a Unit Task (Card et al., 1983). Misses and deaths can be conceived of as

[8]There are other steps at this level that have no marker in the game record. Prominent among these in the model, and we assume subjects, involve deciding where to fly next. These decisions are then followed by executing the keystrokes

[9]It is also the case that subjects vary in how long they tend to hold both keys down. The 8 subjects with low overlap also hold the keys briefly down when they do (an average overlap of 2.4 ticks) suggesting this is just an accidental overlap where one finger fails to come up before another goes down. The 4 subjects with high overlap tend to hold the keys down for long periods (an average of 10.3 ticks) indicating deliberate intent to achieve curved flight.

[10]The correlation between proportion key overlap and proportion of crossing legs less than 100 ticks is .89. The correlation between proportion key overlap and proportion of turns longer than 60 ticks is .74.

failed Unit Tasks. The EEG-based MVPA did well identifying the critical events and we could stitch together model games that matched these critical events. The HSMM-MVPA analysis takes advantage of both the constraints on the timing of these critical events and the strong EEG signals associated with their appearance. The improved success over Anderson et al. (2020) reflects focusing on subject-unique patterns in classification and satisfying the semi-Markov assumption.

There is reason to believe that one can pursue such HSMM-MVPA analysis at the Unit-Task level for many open-ended tasks. If the task decomposes into Unit Tasks (like the classic Card, Moran, and Newell text editing), each Unit Task will result in either success or failure which should produce a strong EEG signal. Often the end of the next Unit Task depends only on the last Unit Task and how long it has been since that last Unit Task, satisfying the semi-Markov assumption. However, the independence of Unit Tasks may only be partial, leaving violations of the semi-Markov assumption as in this task. At least, sometimes these violations can be dealt with by going beyond a single HSMM as we did here with a hierarchical HSMM. Recently, Anderson et al. (2023) have extended HSMM-MVPA approach to identify critical events in the Mat-B task (Miller et al., 2014) focusing on the correction of faults which appear as Unit Tasks in a 6-minute task. The structure of that experiment creates long-term dependencies between different types of faults. Rather than a hierarchical HSMM like here, we used parallel HSMMs tracking different faults to remove these dependencies.

The EEG signals used to identify critical events are variants of the P300 ERP (Figure 3) which often used in BCI applications ((Fazel-Rezai et al., 2012). As the critical events reflect positive or negative outcomes, one can ask how they relate to the ERN and FRN (error-related negativity and feedback-related negativity, (Holroyd & Coles, 2002; Walsh & Anderson, 2012) which appears as a negative frontocentral deflection 80-150 msec after an error and 200-350 msec after negative feedback. Such a negative deflection can be found in the vertical dimension that separates negative from positive events (Figure 4c). There is approximately a 2.5 negative microvolt deflection in the FZ and CZ electrodes about .25 seconds after the event. This is somewhat dwarfed by the following large positive deflection reflect that dimension's P300.

Having a sequence of critical events, either from the subject or the HSMM we investigated whether the ACT-R model could produce these sequences. The ACT-R model not only provides a good match to the statistics of these critical events (Figure 5), but it was also possible to stitch together near perfect matches to the critical events. However, the model was typically matching the critical events through different sequences of lower-level

actions. There are many sequences of actions that result in the same critical event and indeed the stitching algorithm was able to find many ways tthat he model could match the critical events in a game (Figure 9).

We used the leg structure of games as a way of judging how well the subjects' intentions matched model intentions. A leg corresponds to the Operation level in Newell's Cognitive Band, in that it achieves a subgoal along the way to the goal of a kill. Even 200,000 games is not enough to provide an adequate library of segments. Less than 50% of subject critical segments had even approximate matches in the model games in terms of their leg structures. In many cases where there was a model segment that produced a good match to the leg structure of a subject segment, it could not be stitched with other segments to produce viable game play. The best stitchings (Table 3a and 3c) had only about 70% of their legs placed close in time to the actual legs in the game. These results point to practical limitations in trying to match subject behavior below the Unit-Task level.

The match to leg structure was much worse if we used only the EEG signal to guide the stitching rather than knowledge of the target leg structure. While there are EEG correlates of the leg structure, they are much weaker than the correlates of the critical events. They do well at distinguishing one game from another, but they provide poor guidance for constructing a stitching.

The presence of overlapping keys in subjects' behavior probably contributed to the difficulties at the leg level. The model did not produce curved legs dooming any attempt to perfectly match the leg structure of a game that contained a substantial number of overlapping keys. The model captured some but not all of the ways subjects could achieve their navigation intentions. It depends on one's goal in modeling, whether it is a reasonable aspiration to try to achieve complete coverage at this level of detail.

The difficulties we faced in getting model correspondences at both the Operation level and Deliberate Act level point to the challenges of modeling open-ended tasks at this level of detail. The combinatorics are overwhelming: there are a great many ways that models like ours can put these small pieces can be put together to achieve the same end. It is not feasible to create a library that holds all the possibilities. Subjects show even greater variability than the model with some subjects adopting styles that are not in the model. While detailed modeling of such details may be feasible for brief, controlled laboratory tasks, when it comes to open-ended tasks we have to be content with comparisons of average behavior like Figure 10.

The unit-task level seems the right level of aspiration for detailed reconstruction of open-ended tasks. Completion of one unit-task frees the performance of the next unit-

task from the details of past history. When the stitching algorithm considered whether a segment could be stitched into the fragment it was growing, it only had to consider where that fragment had gotten to the ship, not how it got there. Similarly, we were able to use semi-Markov methods in interpreting the EEG signal, which require independence from past history. In addition, the EEG signal associated with such tasks tends to be sufficiently robust to stand out against the noise. It is worth noting that while unit-tasks occupy seconds of task, EEG allowed us to identify them of with an accuracy that matched the temporal grain size of deliberate acts.

References

Abiri, R., Borhani, S., Sellers, E. W., Jiang, Y., & Zhao, X. (2019). A comprehensive review of eeg-based brain–computer interface paradigms. *Journal of neural engineering*, *16*, 011001.

Altarelli, I., Green, C. S., & Bavelier, D. (2020). Action video games: From effects on cognition and the brain to potential educational applications. In *Educational Neuroscience* (pp. 273–297). Routledge.

Anderson, J. R. (2002). Spanning seven orders of magnitude: A challenge for cognitive modeling. *Cognitive Science*, *26*, 85–112.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?*. Oxford University Press.

Anderson, J. R., Betts, S., Bothell, D., Hope, R., & Lebiere, C. (2019). Learning rapid and precise skills. *Psychological review*, *126*, 727–760.

Anderson, J. R., Betts, S., Bothell, D., & Lebiere, C. (2021). Discovering skill. *Cognitive Psychology*, *129*, 101410.

Anderson, J. R., Betts, S., Fincham, J. M., Hope, R., & Walsh, M. W. (2020). Reconstructing fine-grained cognition from brain activity. *NeuroImage*, *221*, 116999.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological review*, *111*, 1036.

Anderson, J. R., Fincham, J. M., Fox, E. L., Stevens, C. A., & Swan, G. (2023). Using eeg to understand multitasking performance. In *Annual Meeting of the Society for Mathematical Psychology, July, 2023*.

Anderson, J. R., Zhang, Q., Borst, J. P., & Walsh, M. M. (2016). The discovery of processing stages: Extension of Sternberg's method. *Psychological Review*, *123*, 481.

Baldwin, C. L., & Penaranda, B. (2012). Adaptive training using an artificial neural network and eeg metrics for within-and cross-task workload classification. *NeuroImage*, *59*, 48–56.

Boot, W. R. (2015). Video games as tools to achieve insight into cognitive processes.

Borst, J. P., & Anderson, J. R. (2021). Discovering cognitive stages in m/eeg data to inform cognitive models. *An Introduction to Model-Based Cognitive Neuroscience*, .

Buckland, M., & Gey, F. (1994). The relationship between recall and precision. *Journal of the American society for information science*, *45*, 12–19.

Card, S., Moran, T., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale [etc.]: Lawrence Erlbaum.

Delorme, A., & Makeig, S. (2004). Eeglab: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Journal of neuroscience methods*, *134*, 9–21.

Dimov, C. M., Anderson, J. R., Betts, S. A., & Bothell, D. (2023). An integrated model of collaborative skill acquisition: Anticipation, control tuning, and role adoption. *Cognitive Science*, *47*, e13303.

Fazel-Rezai, R., Allison, B. Z., Guger, C., Sellers, E. W., Kleih, S. C., & Kübler, A. (2012). P300 brain computer interface: current challenges and emerging trends. *Frontiers in neuroengineering*, (p. 14).

Fincham, J. M., Lee, H. S., & Anderson, J. R. (2020). *Spatiotemporal analysis of event-related fMRI to reveal cognitive states*. Technical Report Wiley Online Library.

Fine, S., Singer, Y., & Tishby, N. (1998). The hierarchical hidden markov model: Analysis and applications. *Machine learning*, *32*, 41–62.

Fisher, C. R., Houpt, J. W., & Gunzelmann, G. (2020). Developing memory-based models of act-r within a statistical framework. *Journal of Mathematical Psychology*, *98*, 102416.

Gianferrara, P. G., Betts, S., Bothell, D., & Anderson, J. R. (2021). Simulating human periodic tapping and implications for cognitive models. In *Proceedings of the 19th international conference on cognitive modeling*.

Gray, W. D. (2017). Game-xp: Action games as experimental paradigms for cognitive science.

Holroyd, C. B., & Coles, M. G. (2002). The neural basis of human error processing: reinforcement learning, dopamine, and the error-related negativity. *Psychological review*, *109*, 679.

Jap, B. T., Lal, S., Fischer, P., & Bekiaris, E. (2009). Using eeg spectral components to assess algorithms for detecting fatigue. *Expert Systems with Applications*, *36*, 2352–2359.

Johnson, M. J., & Willsky, A. S. (2013). Bayesian nonparametric hidden semi-markov models, .

Kotseruba, I., & Tsotsos, J. K. (2020). 40 years of cognitive architectures: core cognitive abilities and practical applications. *Artificial Intelligence Review*, *53*, 17–94.

Lotte, F., Bougrain, L., Cichocki, A., Clerc, M., Congedo, M., Rakotomamonjy, A., & Yger, F. (2018). A review of classification algorithms for eeg-based brain–computer interfaces: a 10 year update. *Journal of neural engineering*, *15*, 031005.

Mané, A., & Donchin, E. (1989). The space fortress game. *Acta psychologica*, *71*, 17–22.

Miller, W., Schmidt, K. D., Estepp, J. R., Bowers, M., & Davis, I. (2014). An updated version of the us air force multi-attribute task battery (af-matb). *DTIC Document*, .

Newell, A. (1990). *Unified theories of cognition*. Harvard University Press.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*, 257–286.

Siegler, R. S. (1987). The perils of averaging data over strategies: An example from children's addition. *Journal of Experimental Psychology: General*, *116*, 250.

Verleger, R. (2020). Effects of relevance and response frequency on p3b amplitudes: Review of findings and comparison of hypotheses about the process reflected by p3b. *Psychophysiology*, *57*, e13542.

Walsh, M. M., & Anderson, J. R. (2012). Learning from experience: event-related potential correlates of reward processing, neural adaptation, and behavioral choice. *Neuroscience & Biobehavioral Reviews*, *36*, 1870–1884.

Wickens, T. D. (2002). *Elementary signal detection theory*. Oxford University Press, USA.

Appendix A. Co-Op Space Fortress

The game is implemented in JavaScript and runs in NodeJS. It uses the SDL2 and Cairo libraries for graphics output. The game updates 60 times a second. The game is played in an area 710 pixels wide by 620 pixels tall. The gray hexagon is centered in the middle of the game area at x=355, y=310 with a radius of 200. The fortress is also centered in the middle of the game area at x=355, y=310. Surrounding the fortress is its shield, a green hexagon of radius 40 that rotates with the fortress. The shooter ship starts 230 pixels to the left of the fortress at x=125, y=310 and pointing up. The bait ship starts 230 pixels to the right of the fortress at x=585, y=310 and pointing up. While the thrust key is pressed, the velocity increases by 0.05 in whatever direction the ship is pointing. While a turn key is pressed, the ship will turn 4 degrees per tick.

When the fire key is pressed, a missile is created at the ship's position. It points in the same direction as the ship and each game tick advances 10 pixels in that direction. After 15 ticks the missile disappears. This limits a missile's range such that the shooter must be inside the gray hexagon in order to hit the fortress. Another missile will not be fired until the fire key is released and pressed again. Only the shooter may fire missiles.

When the bait ship is inside the grey hexagon, the fortress turns to point at the bait at a rate of 1 degree per tick. When the bait is outside the gray hexagon, the fortress turns clockwise at a rate of 0.25 degrees per tick. A gap opens in the fortress' shield, making it vulnerable to a kill shot, when the bait is inside the gray hexagon and there is at least one shell visible.

On each game tick where the bait is inside the gray hexagon, the fortress' orientation in degrees is divided by 10 and rounded to the nearest whole number. A shell is fired if this value has not changed in 60 ticks, the bait is inside the gray hexagon, and it has been at least 60 ticks since a shell has been fired. This means the bait must traverse at least 10 degrees of arc around the fortress in one second or the fortress will fire a shell at it. Shells are fired pointing directly at the bait ship. They start 30 pixels from the fortress' position and travel at 0.8 pixels per tick. After 180 ticks, the shell disappears. This means shells disappear roughly at the edge of the gray hexagon.

For the purposes of collision detection, ships are considered a single point located where the wings meet the body. A collision occurs between a ship and a shell if the ship's collision point is inside the shell's diamond shape. Relative to the position of the shell, the shape is defined by the following x-y pairs: 8,0; 0,-6; 16,0; and 0,6. A collision between a ship and the edge of the game area occurs if the ship's collision point is outside the game

area. A collision occurs between the ship and the fortress' green hexagonal shield if the ship's collision point is inside the hexagon.

For the purposes of collision detection, missiles are considered a single point located at the center of the missile. A collision occurs between a missile and the fortress' shield if the ship's collision point is inside the hexagon. A collision occurs between a missile and the fortress shield if the ship's collision point is inside the fortress' collision box. The box is formed from 4 points on the rear section of the fortress. Relative to the fortress' position they are the following x-y pairs: 18,18; 18,-18; 0,-18; and 0,18.

The results of a collision are as follows:

- A collision between missile and fortress results in a fortress kill.
- A collision between missile and fortress shield results in the missile disappearing.
- A collision between missile and game boundary results in the missile disappearing.
- A collision between ship and fortress results in the ship exploding.
- A collision between ship and shell results in the ship exploding.
- A collision between ship and game boundary results in the ship exploding.

Appendix A.1. Probability Maximized by the Viterbi Algorithm

Any sequence of events (kills, deaths, misses) can be denoted as events $a_1, a_2, ..., a_n$ occurring at game ticks $t_1, t_2, ..., t_n$ where a_1 is the start of the game (hence t_1 is the first game tick) and a_n is the last event before the last game tick 10,800. The following gives the probability of the sequence assuming it satisfies the semi-Markov assumption.:

$$Prob(a_1, a_2, ..., a_n) \approx \left(\prod_{i=1}^{n-1} (trans(a_i, a_{i+1}) \times f(t_{i+1} - t_i | a_i, a_{i+1}) \times P(EEG(t_i + 1 : t_{i+1}) | a_{i+1})) \right) \times S(10800 - t_n | a_n) \times P(EEG(t_n + 1 : 10800 | Null)$$

where $trans(a_i, a_{i+1})$ is the probability of transition between the events a_i and a_{i+1}, $f(t_{i+1} - t_i | a_i, a_{i+1})$ is the probability of the $t_{i+1} - t_i$ game ticks between the events a_i and a_{i+1}, $P(EEG(t_i + 1, t_{i+1}) | a_{i+1})$ is the conditional probability of the EEG signal for this period if it ends in a_{i+1}, and $S(10800 - t_n | a_n)$ iis the probability that there are no critical events in the remaining game ticks if the last event is a_n. While calculating this quantity for all possible sequences is prohibitive, the Viterbi algorithm uses dynamic programming to efficiently identify the most probable sequence. Anderson et al. (2020)

describe a further efficiency that allows us to ignore the probability of the EEG signal for Null ticks and deals with the non-independence of the EEG signal on adjacent ticks.

Appendix A.2. Stitching Algorithm

Segments start from either a kill or a starting position (the ship is in a starting position at the beginning of a game and after a death) and end in a kill, death, or end of the game. Thus, there are 3x2=6 types of segments. The model runs provide 3,283,423 segments. The majority, 78.470% are between two kills. The least frequent (from a starting position to game end), 0.35% still have 11,595 cases.

Since each death in a game returns the ship to the starting position. The stitching of segments after a death does not depend on the stitching of segments before a death. Therefore, we run the algorithm separately on every **stretch** of a game that goes from a starting position to a death or to game end. For any such stretch the algorithm initializes its beam width to 100. If it fails with that width, it will double the width until it is able to stitch segments for that stretch. With the beam set to n, the algorithm starts with n segments from start to the first critical event. If there are more than n segments that match the length of the critical event, it selects the first n if choice is random, the n segments with highest d-prime leg match if trying to match leg match, or the n segments with the highest EEG match probability if trying to maximize that. If there are not n segments of the target length it expands its set to be the n that are closest in length.

The segments selected for the first critical event will be the starting set of fragments. Then it repeatedly performs the following steps until it reaches the end of the game stretch:

1. Find n segments to the next critical event that are closest in length to the target segment. If there are more than n matching length exactly, it selects according to same criterion it used for selecting the initial segments.

2. Create a new set of fragments by combining the segments with current fragments and keeping those fragments that have viable flight paths.

3. If there are more than n new fragments, select those that are best by the chosen metric.

The number of fragments that are produced in step 2 can be quite variable. If there are m existing fragments and n new fragments it can be anywhere from 0 to $m \times n$. Stitching fails if there are 0 and a new search is started with n doubled.

The determination of a viable path depends on how the next segment began and ended. If it ended in a kill a viable path had to end inside the hexagon and be aiming at the fortress. If it also began with a kill, it had to fly outside the hexagon and back in. A final criterion for kill segments was that the speed never exceeded 2.5 pixels per time tick because few successful subject kills happened at such speed. If the next segment ended in a death, the flight path also had to end either within the fortress or outside of the border. Of such segments the algorithm choses the ones that crashed closest to when the target segment crashed. This could mean that some of the stitched deaths were shorter that the target segment. If the next segment ended the game, flight paths were accepted that did not have a death before that point.

Appendix A.3. Legs of Flight

A thrust will send the ship on a new path. Therefore, legs of flight begin when the thrust key is lifted up, leaving flight direction unchanged, and they end when the thrust key is depressed, changing flight direction. Turns are often achieved by thrusting, then turning the ship some more, and then thrusting again to achieve a new path. This can result in a short period of constant flight direction between the two thrusts. To avoid treating these as true legs, we required that legs be at least 30 ticks (half a second long) and considered the shorter legs part of a turn.