NeuroImage 221 (2020) 116999

Contents lists available at ScienceDirect

NeuroImage

journal homepage: www.elsevier.com/locate/neuroimage

Reconstructing fine-grained cognition from brain activity

John R. Anderson [a,*], Shawn Betts [a], Jon M. Fincham [a], Ryan Hope [a], Mathew W. Walsh [b]

[a] Department of Psychology, Carnegie Mellon, United States
[b] The Rand Corporation, United States

ARTICLE INFO

Keywords:
Cognitive reconstruction
Cognitive modeling
EEG
Game playing

ABSTRACT

We describe the Sketch-and-Stitch method for bringing together a cognitive model and EEG to reconstruct the cognition of a subject. The method was tested in the context of a video game where the actions are highly interdependent and variable: simply changing whether a key was pressed or not for a 30th of a second can lead to a very different outcome. The Sketch level identifies the critical events in the game and the Stitch level fills in the detailed actions between these events. The critical events tend to produce robust EEG signals and the cognitive model provides probabilities of various transitions between critical events and the distribution of intervals between these events. This information can be combined in a hidden semi-Markov model that identifies the most probable sequence of critical events and when they happened. The Stitch level selects detailed actions from an extensive library of model games to produce these critical events. The decision about which sequence of actions to select from the library is made on the basis of how well they would produce weaker aspects of the EEG signal. The resulting approach can produce quite compelling replays of actual games from the EEG of a subject.

1. Introduction

The goal of this research is to track moment-by-moment what someone is thinking and doing over an extended period using the high temporal resolution of EEG. We will describe a method for achieving this goal that merges bottom-up information from classification of the EEG signal with top-down information from cognitive modeling. A great deal of research has studied classifying EEG signals and the results have been applied to a number of domains such as brain-computer interfaces (Lotte et al., 2018), emotion recognition (Kim et al., 2013), understanding human memory (Noh et al., 2014), estimating workload (Brouwer et al., 2012), among others. With few exceptions (e.g. Su et al., 2018), this research involves tasks where the experimenter has control over the presentation of stimuli and examines activity in predefined intervals, typically locked to the presentation of these stimuli. However, in many realistic situations such as driving a car one does not have such experimental control and the sequence of events emerges as an interaction between the subject and the environment. Furthermore, the intervals between actions in such situations can be much shorter than the typical intervals used in most classification efforts.

To explore the tracking of mental state in such a context we chose video game play. There has been work on EEG and video games (e.g. Kerous et al., 2018), but typically focused on using traditional BCI methods to serve as a controller for the game. These applications typically leverage three types of EEG signals: (1) Signals sensitive to the occurrence of rare events, such as the presentation of a letter that the individual is thinking of (i.e., the P300); (2) Signals sensitive to how objects that the individual is attending to are presented (i.e., the SSVEP); and (3) Signals sensitive to planned and imagined movement (i.e., the Mu rhythm). These studies demonstrate the potential of inferring control signals from EEG, yet they do not involve highly dynamic tasks with significant perceptual-motor demands.

There has been little focus on recognizing events that occur in free-flowing games. One exception is a study by Cavanagh and Castellanos (2016) who trained a neural classifier on controlled pre-game exemplar events. These events included the presentation of unexpected stimuli in an oddball detection task, and the presentation of positive and negative feedback in a gambling task. Both types of events—the occurrence of unexpected events and the delivery of rewards and punishment—also occur in many video games. Cavanagh and Castellanos found that classifiers trained using data from the control tasks could be used to categorize positive and negative events that occurred during Escape from Asteriod Axon, an 8-bit video game with continuous play. This demonstrates the transferability of EEG signals traditionally studied in simple laboratory tasks to a real-time game. A point of departure from the current study, however, is that Cavanagh and Castellanos directly provided

* Corresponding author. 5000 Forbes Ave., Pittsburgh, PA, 15213, United States.
E-mail address: ja@cmu.edu (J.R. Anderson).

https://doi.org/10.1016/j.neuroimage.2020.116999
Received 3 January 2020; Received in revised form 4 May 2020; Accepted 26 May 2020
Available online 1 June 2020

the classifier with subsets of epochs of video-game play that contained critical events. Thus, the classifier did not need to detect epochs that contained critical events, nor did it fill in the sequence of actions and states between those events.

Video games offer an excellent opportunity to test methods for tracking human cognition because one can collect a record of what the subject did and what the game did on each game tick. Additionally, video games offer an opportunity to bridge the gap between carefully controlled laboratory studies that seek to isolate one or a small number of EEG signals, and the far more complex tasks that people routinely perform like driving a car in traffic.

This paper describes a method that attempts to reconstruct the actual game play from the EEG signal. This is a high bar because even getting a couple of actions out of synch can lead to disastrous reconstruction that is not at all human-like. Nonetheless, we have had some success in achieving the goal of reconstructing video game play from EEG signals (for examples, see http://andersonlab.net/reconstruction/). This success requires more than just an EEG classification algorithm. No matter how good the classification method is, it will misclassify some things, leading to an incoherent reconstruction of the full game (i.e., improbable or impossible sequences of events). Rather than directly choosing actions from the classifier, we use the output of the classifier to select sequences of actions from a cognitive model (Anderson et al., 2019) that can play the game like actual players. The result is a reconstruction of the game that is coherent, human-like, and typically very similar to the game of the player whose EEG signal we are working from.

1.1. Space Fortress game

The video game we studied was a variant of Space Fortress. This game has a long history in the study of skill acquisition and training methods, first being used in the late 1980's by a wide consortium of researchers (e.g. Donchin, 1989; Frederiksen and White, 1989; Gopher et al., 1989). Part (a) of Fig. 1 illustrates the critical elements of the game. Players are instructed to fly a ship between the two hexagons. They are firing missiles at a fortress in the middle, while trying to avoid being hit by shells fired by the fortress. The ship flies in a frictionless space. To navigate, the player must combine thrusts in various directions to achieve a path around the fortress. Mastering navigation in the Space Fortress environment is challenging; while subjects are overwhelmingly video game players, most have no experience in navigating in a frictionless environment.

There have been EEG studies of Space Fortress. Maclin et al. (2011) recorded EEG from subjects as they played Space Fortress while concurrently performing a secondary task that involved counting rare auditory oddball stimuli. The amplitude of the P300 to rare stimuli in the oddball detection task increased following training on Space Fortress, while the amplitude of the P300 to stimuli in Space Fortress decreased. These results indicate that with training, the primary task of playing Space Fortress became less attentionally demanding, freeing resources for the secondary task. In subsequent work, Mathewson et al. (2012) found that event-related increases in frontal theta, an oscillation associated with attentional control, predicted individual differences in learning rate. Together, these results show the importance of attention in Space Fortress and that there is a reduction in attentional demands with practice.

We used the Autoturn version of the game introduced in Anderson et al. (2019). In this variant of the game, the ship is always aimed at the fortress and subjects do not have to turn it. The ship begins each game aimed at the fortress, at the position of the starting vector in Fig. 1a, and flying at a moderate speed in the direction of the vector. To avoid having their ship destroyed, subjects must avoid hitting the inner or outer hexagons, and they must fly fast enough to prevent the fortress from aiming, firing at, and hitting the ship. When subjects are successful the ship goes around the fortress in a clockwise direction. They can destroy the fortress by shooting missiles at it to build up its vulnerability and then destroying it with a "kill shot" (two shots in rapid succession). If the fortress is destroyed it leaves the screen for 1 s before respawning. If the ship is destroyed it respawns after 1 s in the starting position flying along the starting vector. Our version of the game eliminated much of the complexity of scoring in the original game and just kept three rules:

1. Subjects gained 100 points every time they destroyed the fortress.
2. Subjects lost 100 points every time the ship was destroyed
3. To reinforce accurate firing, every fire costs 2 points.

To keep subjects from being discouraged early on, their score never went negative. The replay site (http://andersonlab.net/reconstruction/) offers examples of game play.

Anderson et al. (2019) found that subjects can achieve relatively high and fairly stable performance within an hour of playing AutoTurn (much faster than in original Space Fortress where subjects are also responsible for turning their ship among other things). To maintain a constant challenge of game play, a staircase procedure decreased the separation between the inner and outer hexagons as subjects got better. Subjects played 1-min games. During the first 10 games the inner corners were 40 pixels from the center and the outer corners were 200 pixels from the center producing a width of 160 pixels. After the tenth game, the border width was reduced by 10 pixels if the subject had 0 or 1 deaths in the prior game and it was increased by 30 pixels (to a maximum width of 160

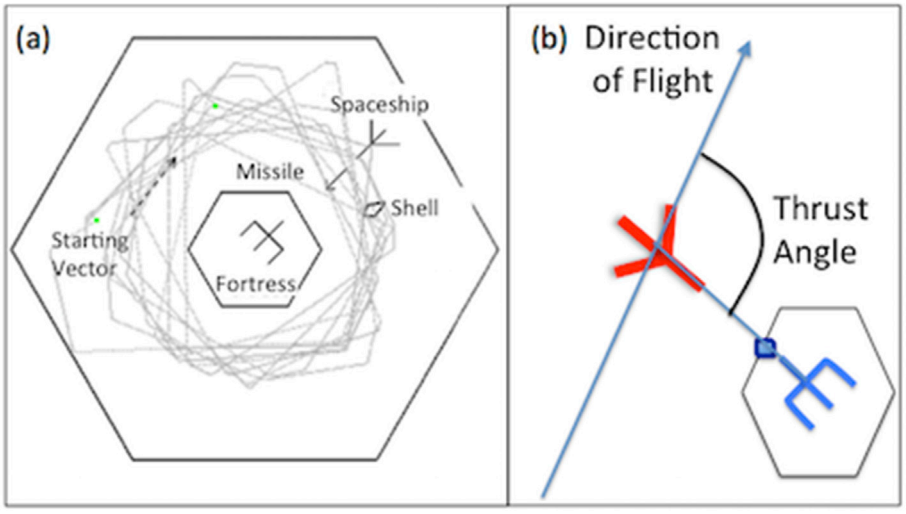

Fig. 1. (a) The Space Fortress screen, showing the inner and outer hexagon, a missile fired at the fortress, and a shell fired at the ship. The distance from the center (fortress) to the corners of the outer hexagon is 200 pixels and the distance to the corners of the inner hexagon is 40 pixels. The ship starts 120 pixels to the left of the center, flying at 30 pixels per second, parallel to the upper left side of the hexagon. The dotted lines illustrate an example path during one game. (b) A schematic representation of critical values for firing and flight control.

pixels) if they had 2 or more deaths. In this way the death rate in the game was maintained at about 1 death per 1-min game. For each 10 pixels the border is reduced, subjects get an additional 10 points for each fortress they destroy. Navigation becomes more difficult as one has to fly between narrower borders, with many deaths resulting from thrusting into the inner hexagon, a rare event with the original 160 pixel width.

The game advances at 30 ticks per second. Only two keys are pressed —a left-hand press of the W key to add thrust to the ship and a right-hand press of the space bar to fire at the fortress. Exactly when a player thrusts and fires is critical to performance. The difference of a single game tick can mean the difference between destroying the fortress and being destroyed. Critically, the impact of a key press depends on the past history of key presses as well: the consequence of a thrust depends on the ship's current position and flight path (determined by past thrusts) while the consequence of a fire depends on how preceding fires have affected the fortress's vulnerability.

Good performance involved mastering two skills —destroying the fortress and flying the ship in the frictionless environment. To destroy the fortress one must build up the vulnerability of the fortress (displayed at the bottom of the screen). When the vulnerability reaches 11, subjects can destroy the fortress by quickly firing an additional missile at it. Each fire increases the fortress's vulnerability by one, provided the fires are paced at least 250 ms apart. If the inter-fire interval is less than 250 ms the vulnerability is reset to 0 and one must begin the build up of vulnerability anew. While subjects could easily make sure the fires building up vulnerability are at least 250 ms apart by putting long pauses between them, this would reduce the number of fortresses destroyed and points gained per game. Thus, subjects are motivated to pace the fires as close to 250 ms as they can without going below that 250 ms. threshold and producing a reset. In contrast to the fires that build up the vulnerability, the fire to destroy the fortress must be less than 250 ms from the last fire.

Since the ship is always aimed at the fortress, subjects do not need to turn their ship as in the original version of Space Fortress. To navigate around the fortress, they must press the thrust key at appropriate times and for appropriate durations. The direction of the ship after a thrust is determined by a vector sum of the current flight velocity and the acceleration they add in the direction of the fortress. The acceleration is determined by how long they hold the thrust key down. Subjects' average ship speed is a little over 1 pixel per game tick (the ship starts out flying at 1 pixel per game tick). Every game tick the thrust key is held down adds 0.3 units of speed in the current orientation of the ship (i.e., towards the fortress). As an example, suppose the ship is flying at 1.2 pixels per game tick, the angle between aim and ship direction (Thrust Angle in part b of Fig. 1) is 120°, and the thrust key is held down for 4 game ticks. The thrust will produce a force of 1.2 pixels in the direction the ship is aimed.[1] The resulting trajectory would still have a velocity of 1.2 pixels per game tick (more if the thrust angle was less than 120°, less if it was more), and would now be in a direction that bisected the thrust angle. Thrusts at the wrong time or for the wrong duration can lead to death of the ship, which happens if the ship hits the inner or outer hexagons or if the ship flies so slowly the fortress can shoot it.

1.2. Overview of Sketch-and-Stitch reconstruction

We developed the **Sketch-and-Stitch** method to infer a trace of the subject's cognition. While we apply the method here to a video game because that provides a demanding test, the underlying approach could be applied to any task. The method involves first developing a sketch of the critical mental events that happen occasionally during an extended task. To do this the method uses a combination of multivariate pattern analysis (MVPA) for identifying EEG patterns associated with the events and hidden semi-Markov models (HSMMs) for locating events in time. We have applied combinations of these MVPA and HSMMs to parsing of fMRI data (e.g. Anderson et al., 2010, 2012) and to the processing of EEG and MEG data (e.g. Anderson et al., 2016, 2018), but nothing as time-critical as reconstructing video game play. After describing the approach taken in this paper, we will highlight its key features and innovations relative to past applications that enabled it to succeed in the task.

Having produced a hypothesis about when the critical events happened in a game, the Sketch-and-Stitch method then stitches in a detailed reconstruction of the subject's cognition that led to these critical events. In this video game these detailed steps of cognition are directly associated with a detailed trace of actions providing a rigorous ground truth for judging the success of the effort. Stitching uses sequences of actions from runs of a simulation model that can produce human-like sequences of cognition. In our case, that simulation model is the ACT-R model described in Anderson et al. (2019) which produced a high-quality match to subject game play. Such a model (because it is stochastic like subjects) can be used to create a large library of candidate sequences for stitching between critical events. The Sketch-and-Stitch method selects among these candidate sequences according to how well they would produce EEG signals that match the subject.

2. Methods

This Methods section describes the source of all the information that goes into our application of the Sketch-and-Stitch procedure. The results section will describe and assess how well this information can be used to reconstruct game play.

2.1. Subjects

A total of 25 subjects were recruited from the CMU population of students and researchers between the ages of 18 and 40. 5 subjects were excluded because of poor performance (1 subject) and equipment problems (4 subjects), leaving 20 subjects (11 male, 9 female). All were right-handed. None reported a history of neurological impairment. Subjects were paid $75 for participation in the experiment that lasted less than 2 h.

All participants 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,[2] they played 60 1-min games choosing to move on to the next game at their own pace. The games varied in length from 1800 to 1820 ticks with a mean 1819 game ticks (each game tick a 30th of a second, making most games a little longer than 1 min). Our analysis will focus on the first 1800 game ticks, or exactly 1 min. The game records the state of the screen (where the ship is if alive, the direction and speed of movement, whether shells or missiles are on the screen, and whether a key is depressed) at each game tick. This serves as the ground truth both for training the decoder and for testing its predictions.

2.3. EEG analysis

The EEG was recorded from 128 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

[1] This is only a close approximation because the flight of the ship and its orientation update after each tick of thrust.

[2] Instructions were available for review between games.

mastoids. The EEG and EOG signals were filtered with a bandpass filter of .1–70.0 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 and 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.[3]

The EEG signal was recorded continuously for the entire experimental session and broken into 1-min games. There was also a complete record of what happened in each game. Portions of the game periods were identified as bad signals were excluded. Individual channels within an epoch were flagged based on having extreme values for mean absolute deviation, drift, or range. Flagged channels were interpolated. Epochs that still contained channels with extreme values after these steps were flagged and rejected. This resulted in loss of the signal for an average of 1.7 s per game for games used in the decoding (52.5% of the games had no lost signal; the worst game had 21.8 s of lost signal). This reflects a realistic complication in decoding where useful signal can be lost for some fraction of time.

After the processing above to produce the 512 Hz data, the EEG was down-sampled to 30 Hz to match the game ticks. This had the added benefit of making the data of a more manageable size. A 1-s window around each game tick (14 game ticks before, the game tick, and 15 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 30 $\times$ 128 = 3840 electrode readings, representing regional effects, frequency effects (below 30 Hz), and their interactions. Because the vector associated with a game tick requires a complete signal for 1 s, game ticks at the beginning and end of a game do not have corresponding vectors nor do game ticks in or near lost signal. Thus, 29 ticks at the beginning and end of the game have no vectors as well as an average of 71.6 ticks in the vicinity of lost signals, leaving an average of 1718.4 vectors per game. The available vectors for each game were z-scored to standardize them across games. To reduce dimensionality and filter out noise, the vectors for all games and subjects were subjected to a PCA analysis and the 1000 top dimensions were kept.[4] Thus, we had an average of 1718.4 1000-element vectors per game. These are what were used for all classification analyses.

2.4. Classification

All reconstructions efforts focused on the last 55 games where performance is relatively stable. We excluded an additional 20 games of the remaining 1100 games because of particularly bad EEG signal or low activity by the subjects (1 game without good signal throughout, 8 further games where subjects failed to destroy a fortress without resetting or being killed, and 11 games with 12 or fewer critical events). This left 1080 games, which will serve as the focus of analyses.

We performed 4 classification analyses using the same 1000-element vectors produced by the PCA for game ticks in the 1080 games —to identify key presses, to identify critical events, to identify ship position, and to identify vulnerability changes. In each case we used the same leave-one-game-out approach: For a given target game of one subject, the training was done with all remaining games for that subject and all games for all other subjects. A linear discriminant classifier was trained to label the vectors of EEG activity with the category associated with the game tick that the vector describes. To reflect the fact the sensor activity of the subject may be most relevant, that subject's other games are weighted 15 times more than the games of other subjects. This was repeated for each game to get results for all 1080 games. We have neither explored different weightings of that subject to other subjects nor different classification methods. Thus, while the classification results are quite good, they probably are not the best possible, but they are good enough to enable the Sketch-and-Stitch method to achieve fairly high performance.

2.5. Model

We used the same ACT-R model as described in Anderson et al. (2019). To summarize the model: it starts with a declarative representation of the instructions about when to do what. This produces slow performance initially, but over time the model builds action rules that directly perform the actions in the appropriate situations (bypassing the need for declarative retrievals). Critical to its performance in Autoturn are learning when to thrust and when to fire. A Controller module has been implemented within ACT-R that explores a range of values for when to fire and when to thrust and converges on appropriate settings, which it comes to exploit. The creation of action rules and the learning of control values for action underlie the improvement with practice in the model. The behavior of the model is similar to subjects because it uses established ACT-R settings (on the basis of prior experiments) for the timing and variability of mental steps and motor execution. While this model was developed only for the 160-wide pixel border separation, it generalizes to the narrower borders in this experiment because the model monitors for closeness to the borders.

We simulated 100 subjects by running the model 100 times for 60 games under the same game conditions as humans: As the model got better, the borders narrowed. If the model suffered more than one death in a game, the borders expanded. In addition, to collect enough games at each width to have a library for reconstruction, for each possible width 50 model runs of 60 games were executed at a fixed width. In all runs the model was learning and got better with later games. Since the first 5 games of subjects were excluded in the reconstruction efforts, we similarly excluded the first 5 games from each of these runs, yielding a library of 50 $\times$ 55 = 2750 games at each border width. There are 13 possible widths from 40 to 160 pixels, making for a library of 2750 $\times$ 13 = 35,750 model games to serve as a basis for reconstructing the 1080 subject games.

3. Results

3.1. Behavioral results: subjects and models

Fig. 2 shows how various measures changed over the course of the 60 games for the experiment participants and the 100 simulated subjects. Part (a) tracks the width of the space between the two hexagons. This is held constant at 160 pixels by the experiment for the first 10 games, after which the staircase process sets in. The width then decreases until it is bouncing around an average of about 100 pixels. Points and kills (Parts b and c) increase rapidly over the first 10 or so games and then increase more gradually. Ship deaths drop rapidly over the first 10 games before rising to about 1 death per game, which is the goal of the staircase procedure (Part d). Unlike human subjects, the model flies fairly safely from the beginning. Once the staircase procedure sets in both humans and the models show the expected rate of about 1 death per game.[5]

[3] We compared the time course of the activity of each ICA to the vertical EOG. ICAs that were correlated with the vertical EOG (r greater than 0.4) were flagged and examined by a researcher.

[4] This was a PCA performed on a 1,855,876 $\times$ 3840 matrix, producing a 1,855,876 $\times$ 1000 matrix. Because each row in the matrix has both spatial and temporal components, the PCA combines elements of a spatial and a temporal PCA. The first 1000 dimensions capture .990 of the variance. We reconstructed the data from these 1000 dimensions and compared the power spectra of the reconstructed data with the original data. The portion of the variance preserved shifts from 0.999 to 0.856 as we go from 1 to 15 Hz.

[5] The instructions provide information about the importance of thrust angle (see Fig. 1b) for flight —not too large or one will slow down and not too small or one will speed up too much. The model starts with a perfect encoding of this information whereas some subjects only gradually appreciate this.

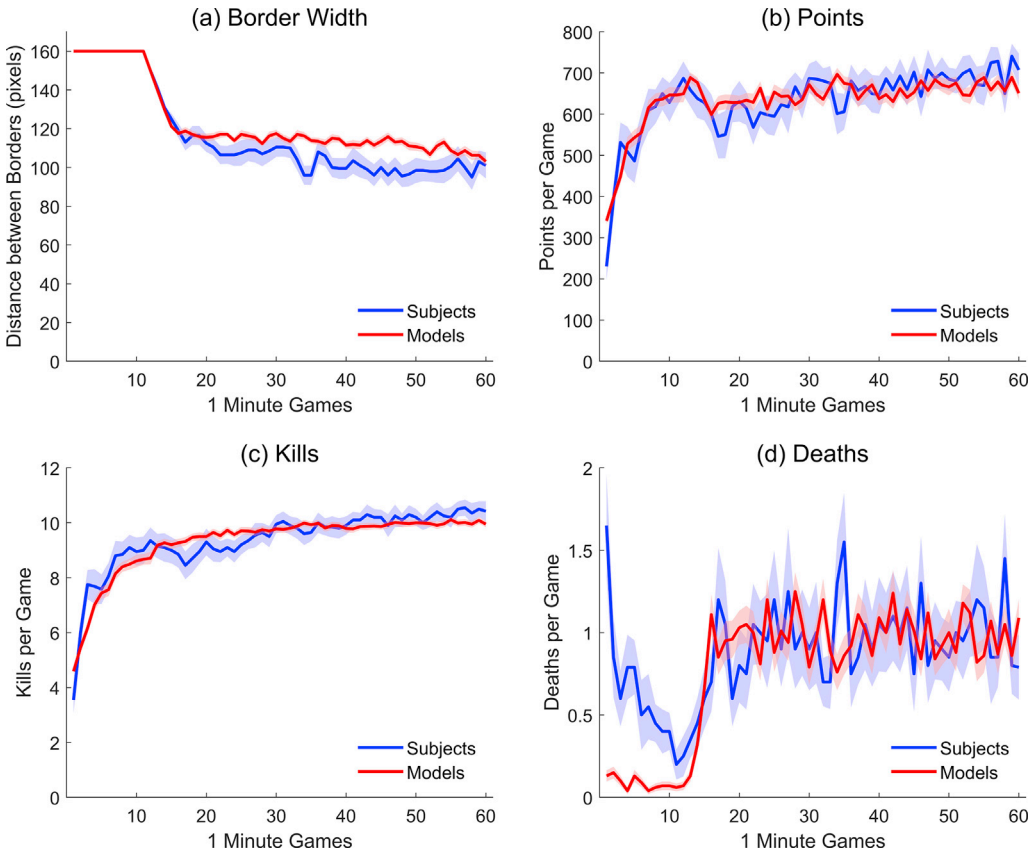

Fig. 2. Mean values (line) and standard errors (area around lines) per game for subjects and models as a function of game (a) border width; (b) points before bonuses for kills at narrow borders; (c) number of fortress destructions; (d) number of deaths.

In later games subjects fly at a range of widths. For instance, on the last game, different subjects are flying at ranges varying from 70 to 160 pixels. Subjects vary in how tight a space they manage to fly in, but 13 of the subjects manage to reach a width of 70 pixels at some point and all but 1 reach 90 pixels (the other reaching 110 pixels). The 100 simulated subjects show a similar range with 43 reaching 70 pixels, and all but 9 reaching 90 pixels. The best subject reached 40 pixels while 3 of the 100 simulated subjects reached 40 pixels. Fig. 3 shows how performance varies as a function of width (omitting the first 5 games where the rapid changes were taking place). Subjects earn somewhat more points with greater widths (Fig. 3a). There is relatively little effect on number of fortresses destroyed with width (Fig. 3b) but a large effect on number of deaths (Fig. 3c). Speed is somewhat greater with wider borders (Fig. 3d). The model also has these trends.

As Figs. 2 and 3 show and as is elaborated at length in Anderson et al. (2019), this model does a good job at capturing many aspects of human game play. The rest of this Results section has 4 further subsections (second through fifth subsections) concerned with using this model and the EEG data to reconstruct game play: The second subsection describes a classifier for identifying key presses from the EEG signal and assesses whether this classifier's output can be used to reconstruct game play. The third subsection describes how a classifier for critical events can be combined with the model to create a sketch of the critical events in the game. The fourth subsection describes how two other classifiers can be combined with the model to stitch in the key presses between the critical events. The fifth subsection will evaluate how well this Sketch-and-Stitch combination does at reconstructing game play.

3.2. Response-based classification

A direct model-free approach to reconstructing game play would be to try to recognize when the left (thrust) and right (fire) fingers are pressed. A fair amount of research has shown good discrimination between imagined left- and right-hand movements for application in BCI (Lotte et al., 2018) and classification of actual key presses by hand is also good (Krauledat et al., 2004). If we could correctly identify when the fingers are pressed we would be able to recreate the game play. Two features of the current task make it more challenging than the typical left-right discrimination. First, it requires not just a binary discrimination but rather a 4-way discrimination because the game includes ticks when neither finger is pressed (the majority: 68.2%) and when both fingers are pressed (rare: 0.2%). Second, the discrimination among these four categories must be made separately for every game tick (30th of a second).

Table 1 shows the classification performance on game ticks that have associated EEG vectors. Part (a) of the table shows the results when each game tick is assigned the label that makes the EEG vector most likely. Clearly, considerable discrimination can be achieved as indicated by the large values on the main diagonal. The overall accuracy is 47% and the average pair-wise area under the curve (AUC) is 0.78. A d-prime (Wickens, 2002) measure of discriminability is .90.[6] Only 25 of the 1080 games have negative d-primes. Excluding ticks with no or both keys pressed, a binary discrimination of ticks with fires from ticks with thrusts is better as would be expected (d-prime 1.54; 77.8% accuracy, 0.861 AUC). Part b of Table 1 shows the 4-way classification results using the posterior probability that weights the likelihood by the prior probability of the category. The d-prime measure in this case is 1.03.

The categorization uses a combination of spatial and frequency information over a 1-s interval surrounding a game tick and so the full

[6] The d-primes reported in this paper for an n-class categorization are the averages of n 2 × 2 classifications of the ability to discriminate each of the n categories from their absence. The average pairwise AUC are the averages of the AUCs from n(n-1)/2 AUC discriminations between pairs of categories.

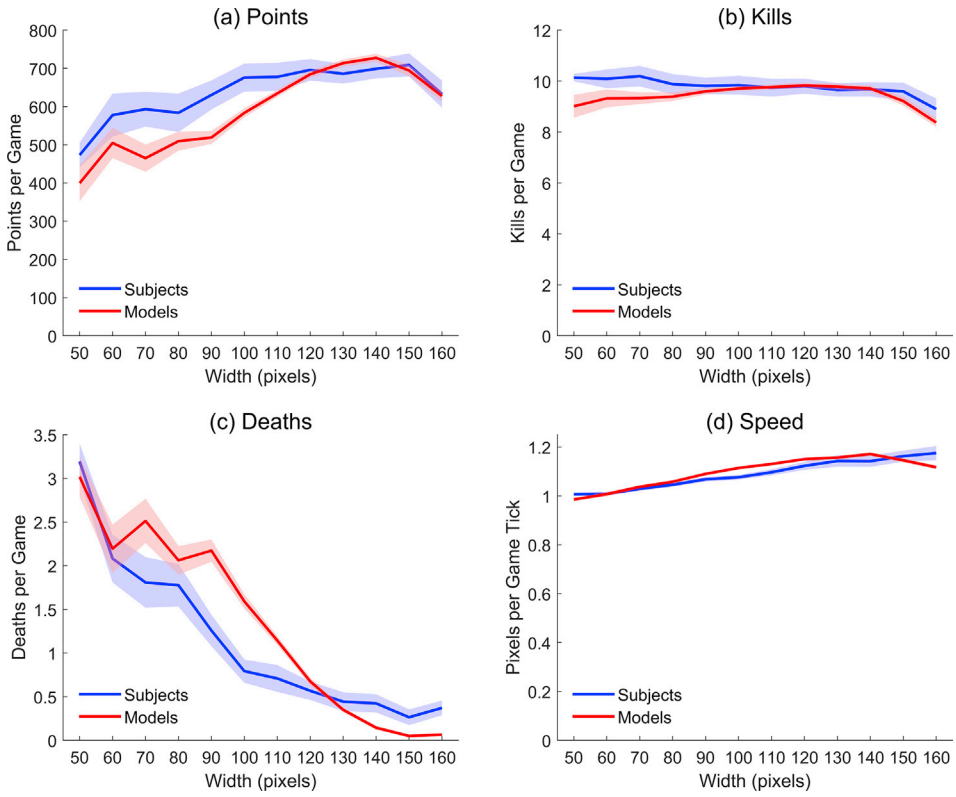

Fig. 3. Mean values (line) and standard errors (area around lines) per game for subjects and models as a function of border width (a) points before bonuses for kills at narrow borders; (b) number of fortress destructions; (c) number of deaths; (d) speed of ship. Because of the few cases of 40 pixels (1 game for subjects, 4 for models), these data are averaged in with 50 pixels (16 games for subjects, 20 games for models).

Table 1
Classification of game ticks according to key press activity.

(a) Classified by Likelihood		Game Tick Labeled as				
		None	Fire	Thrust	Both	Total
Keys Pressed during Game Tick	None	566,338	285,646	255,966	155,085	1,263,035
	Fire	105,799	261,162	66,504	82,210	515,675
	Thrust	10,169	7082	41,927	10,128	69,306
	Both	383	911	992	1652	3938
	Total	682,689	554,801	365,389	249,075	
(b) Classified by Posterior Probability		Game Tick Labeled as				
		None	Fire	Thrust	Both	Total
Keys Pressed during Game Tick	None	1,197,618	62,745	2613	59	1,263,035
	Fire	392,235	122,988	405	47	515,675
	Thrust	62,227	3023	4038	18	69,306
	Both	3023	874	33	8	3938
	Total	1,655,103	189,630	7089	132	

patterns are complex. Fig. 4 shows the average temporal pattern for three central electrodes and the full electrode patterns 5 ticks before and 10 ticks after the event.[7] At this level of aggregation the most prominent feature is that left frontal activation tends to drop after a fire and rise after a thrust. The difference between fire and thrust at the central electrode FZ 10 ticks after a thrust is highly significant ($t(19) = 6.04$, $p < .0001$). There is also greater right lateralized central negativity prior to a thrust and slightly greater left lateralized negativity prior to a fire, which would be the expected lateralized readiness potential opposite the hand pressing the key (e.g. Smulders et al., 2012). For instance, the difference between fire and thrust at the right central electrode F4 5 ticks before the thrust is highly significant ($t(19) = 6.30$, $p < .0001$). Notwithstanding these clear patterns that appear in average data, on single trials there is no electrode comparison on any game tick that provides even as much as 53% correct binary classification. The higher accuracy achieved by the classifier depends on more complex patterns.

How well could one use these key classifications to reconstruct the games? As a test we took the posterior most probable class for each game tick[8] and ran the resulting action sequence through the game. The average score was 13 points per game in contrast to 1218 points achieved by the subjects who generated the EEG signal. Could one do better with a

[7] The EEG signal in this and subsequent figures is the same as used for the PCA extraction of factors used in classification. Thus, there was no baseline correction.

[8] Game ticks without an EEG vector were assigned to the category of no key activity, which is the most common category.

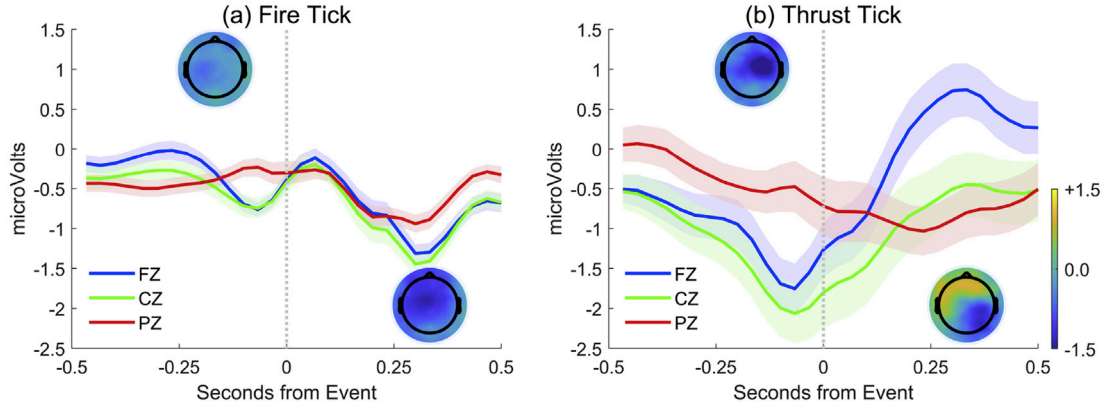

Fig. 4. (a) EEG activity around a tick with the fire key (right hand) depressed. (b) EEG activity around a tick with the thrust key (left hand) depressed. There are 30 game ticks per second. Shaded areas represent a standard error of the mean calculated from the standard deviation of the subject means. Scalp profiles are for −133 ms and +333 ms (−4 and +10 game ticks).

better classifier? To simulate a classifier that was "10 times" more accurate we generated a sequence of key activity that chose the true key presses for each game tick with probability .9 and otherwise the keys of the current classifier. The activity generated in this manner (which matches the actual press pattern 97.2% of the game ticks) averages 92 points a game. We created a classifier that was "100 times" better by using the true key presses 99% of the time. The sequence of key activity generated in this manner (which matches the actual press pattern 99.7% of the game ticks, unrealistically good) still fell short of human performance with 818 points per game. Although only about 5 or 6 game ticks (out of 1800 ticks in a 1-min game) were being missed, these mistakes could include fires that created resets and threw the vulnerability count off. They could also include extra or lost thrusts that changed the direction of the ship. Once the vulnerability or direction was off, later actions did not have the same effect in the reconstructed game as in the original game. In conclusion, even with improvement in accuracy greater than seems possible with improved classification, it does not seem possible to reconstruct game play from direct action classification.

3.3. *Creating the critical sketch*

Higher classification performance can be achieved if rather than trying to classify every action one just tries to classify the critical events that happen during the game. Five critical events could occur in the course of game play:

1. **Kills.** The fortress is destroyed and the player gains 100 points.
2. **Fortress Respawns.** The fortress is respawned after a second and the player can resume firing.
3. **Deaths.** The player's ship is destroyed and the player looses 100 points.
4. **Ship Respawns.** The ship is respawned a second after death and the player can resume thrusting and firing.
5. **Resets.** If the interval between fires is less than 250 ms and the vulnerability is less than 11, the fortress vulnerability will be set back to zero and the subject must begin building vulnerability anew.[9]

The 1080 1-min games in the pool averaged 9.38 kills, 0.86 deaths, and 1.15 resets. The numbers of critical events with EEG vectors are 9742 kills, 9731 fortress respawns, 819 deaths, 805 ship respawns, and 1159 vulnerability resets. In total, these are far less numerous than the thrust key and fire key events (row sums in Table 1).

Fig. 5 shows the EEG activity around the critical events, which produce much stronger signal changes than the key presses in Fig. 4. Part (a) shows the 1-s around a kill. There is a posterior positivity that reaches a maximum 100 ms before the kill, then a general negativity 100 ms after the kill, and then an anterior positivity about 300 ms after the kill. Part (b), which is a continuation of Part (a) shows the activity around the respawn of the fortress. There is return to an anterior positivity about 300 ms after the fortress reappearance. Parts (c) and (d) show the activity around a death and the respawn of the ship. There is negativity in anticipation of the death, which switches to strong central positivity peaking 400 ms after the death. The positivity remains but becomes left lateralized after the ship respawns. Part (e) shows the response to a reset where there is a strong central negativity 300 ms after the reset followed by an even stronger central positivity 500 ms after the reset. One feature common to kills, deaths, and resets (Parts a, c, and e) is a post-event positivity, although that positivity varies somewhat in its timing and distribution across the scalp. The magnitude of this positivity varies with the rareness of the event, with the most common kills showing the smallest response and the least frequent deaths showing the largest response, as we would expect from a P300 (Polich, 2012). The delay in the positivity for resets relative to the other two events may reflect the fact that resets are not anticipated until they occur.

The same classification method described earlier was used to distinguish these 5 events from other game ticks as used for response classification. Given that critical events are rare, to prevent the classifier from being overwhelmed by non-events we limited the number of non-events in training the classifier by randomly choosing two non-critical game ticks for each critical game tick in the game. Once trained on other games the classifier was applied to all game ticks in the target game. Part (a) of Table 2 shows classification of all game vectors by likelihood of the data (d-prime = 2.00) and Table 2b shows classification by posterior probability (d-prime = 2.18). The average accuracy in Table 2a is 59.6% and in Table 2b, which is thresholded to favor the majority null category, the average accuracy is 98.5%. The average pairwise AUC, which does not depend on threshold choice, is 0.942.

While this is considerably better than classification of key presses (Table 1), these classification results by themselves would not produce particularly good game reconstructions. In Part (a) of Table 2, 97.5% of all labels are false alarms to game-ticks that do not involve the ascribed event. In Part (b) of Table 2 the false labels are reduced to 70%, but now 84.4% of all critical events are missed. In addition to these problems, identification of critical events alone does not provide the detailed behavior of the subject that produced them.

While the classifier is not adequate in itself, it can be combined with statistical information from model runs to reconstruct fairly accurate critical sketches from the EEG signal as illustrated in Fig. 6. From the

[9] If the vulnerability were 11 or above, the same interval between fires would produce a fortress destruction.

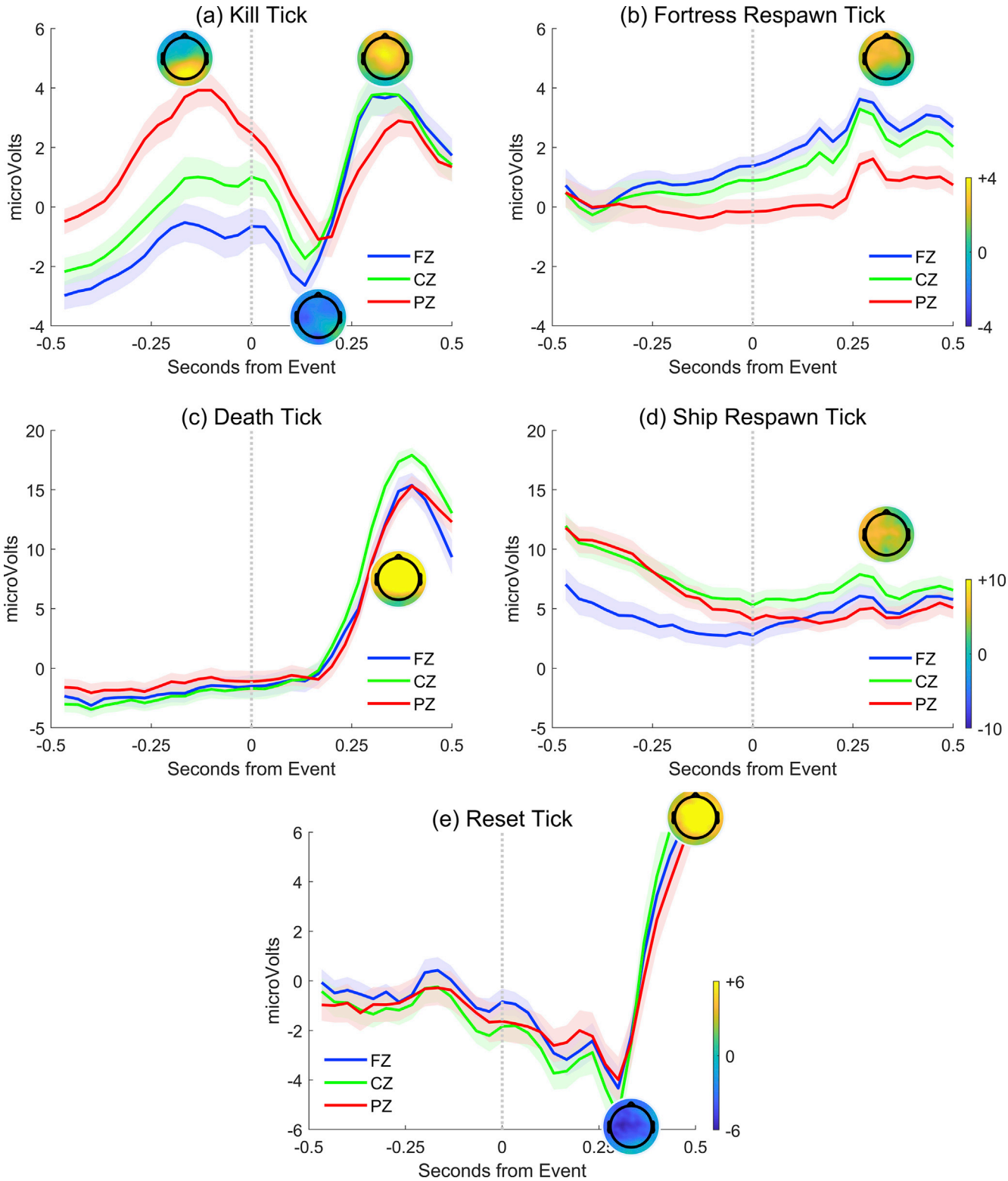

Fig. 5. (a, b) EEG activity around the destruction of the fortress and its respawn with the scalp profiles ranging from −4 to 4 μV. (c, d) EEG activity around the death of the ship and its respawn with the scalp profiles ranging from −10 to 10 μV. (e) EEG activity around a reset of the vulnerability with the scalp profiles ranging from −6 to 6 μV. Shaded areas represent a standard error of the mean calculated from the standard deviation of the subject means.

large library of model games it is possible to obtain reliable estimates of the probabilities of one event following another and how far apart these events are. These transition probabilities and event distributions can be used to parameterize a HSMM. Fig. 7 shows the distributions in the model between other events and the probability that one event will follow another. These distributions and probabilities also vary with the width between the borders (see Fig. 3), with the most dramatic effect being on the probability of a death.

We used an HSMM to efficiently combine these model-based statistics and the conditional probabilities from the EEG classifier to estimate the

Table 2
Classification of Game Ticks according to Critical Events (F refers to Fortress and S refers to ship).

		Game Tick Labeled as					
(a) Classified by Likelihood		Fortress Kill	Fortress Respawn	Ship Death	Ship Respawn	Reset	Null
Event During Game Tick	F. Kill	7695	425	132	136	195	1159
	F. Respawn	475	6687	63	451	235	1820
	S. Death	49	40	591	12	68	59
	S. Respawn	31	123	1	569	11	70
	Reset	74	73	25	17	749	221
	Null	164,756	390,063	13,312	76,326	97,403	1,087,838
		Game Tick Labeled as					
(b) Classified by Posterior Probability		Fortress Kill	Fortress Respawn	Ship Death	Ship Respawn	Reset	Null
Event During Game Tick	F. Kill	2470	1	26	0	4	7241
	F. Respawn	2	103	2	12	0	9612
	S. Death	8	0	457	0	20	352
	S. Respawn	1	0	0	141	0	663
	Reset	3	0	5	0	227	924
	Null	3275	153	1792	1644	1271	1,821,545

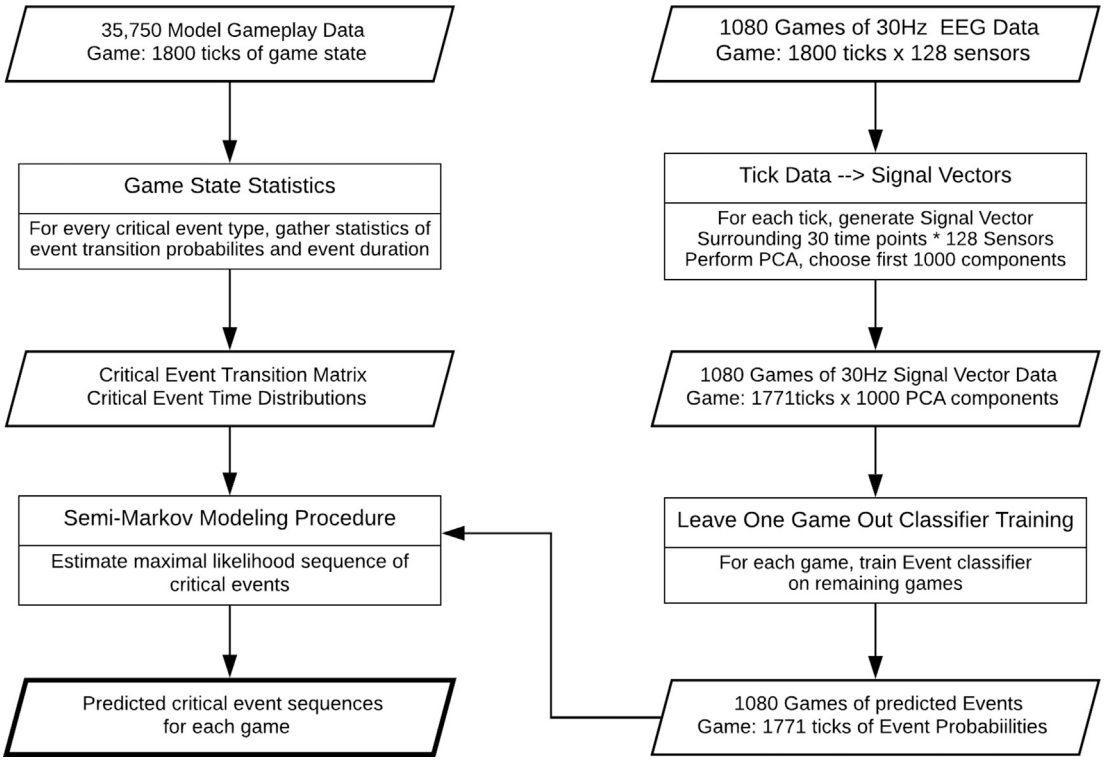

Fig. 6. Information from model and subject combined to create the critical sketch. On the left, runs of the model generate large numbers of game records from which critical sketches are produced. The statistics from these critical events are used to parameterize a semi-Markov model. On right, signal vectors derived from an actual games EEG are passed through a classifier trained on other games to determine the likelihood of critical events in a game. These probabilities are passed through the semi-Markov model to produce a critical sketch.

most likely sequence of critical events in a game. Any sequence of events can be denoted $a_1, a_2, \ldots, a_n$ occurring at game ticks $t_1, t_2, \ldots, t_n$ where a_1 is the start of the game (hence t_1 is the first game tick), a_n is the end (hence t_n is the last game tick), and the rest are fortress kills and respawns, ship deaths and respawns, and resets. The following proportionality describes the probability of any such sequence relative to the probability of other sequences:

$$Prob(a_1, a_2, \ldots, a_n) \approx \prod_{i=1}^{n-1} trans(a_i, a_{i+1}) \times f(t_{i+1} - t_i | a_i, a_{i+1}) \times P(EEG(t_i + 1 : t_{i+1}) | a_{i+1}) \quad (1)$$

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}. The conditional probabilities come from the classifier. Their use can be made much more efficient and robust by using the fact that, if the signals at different ticks were independent:

$$P(EEG(t_i + 1 : t_{i+1}) | a_{i+1}) = P(EEG(t_{i+1}) | a_{i+1}) \times \prod_{x=t_i+1}^{a} P(EEG(x) | Null) = \frac{P(EEG(t_{i+1}) | a_{i+1})}{P(EEG(t_{i+1}) | Null)} \times \prod_{x=t_i+1}^{a} t_{i+1} P(EEG(x) | Null) \quad (2)$$

The product involving the conditional probabilities $P(EEG(x)|Null)$

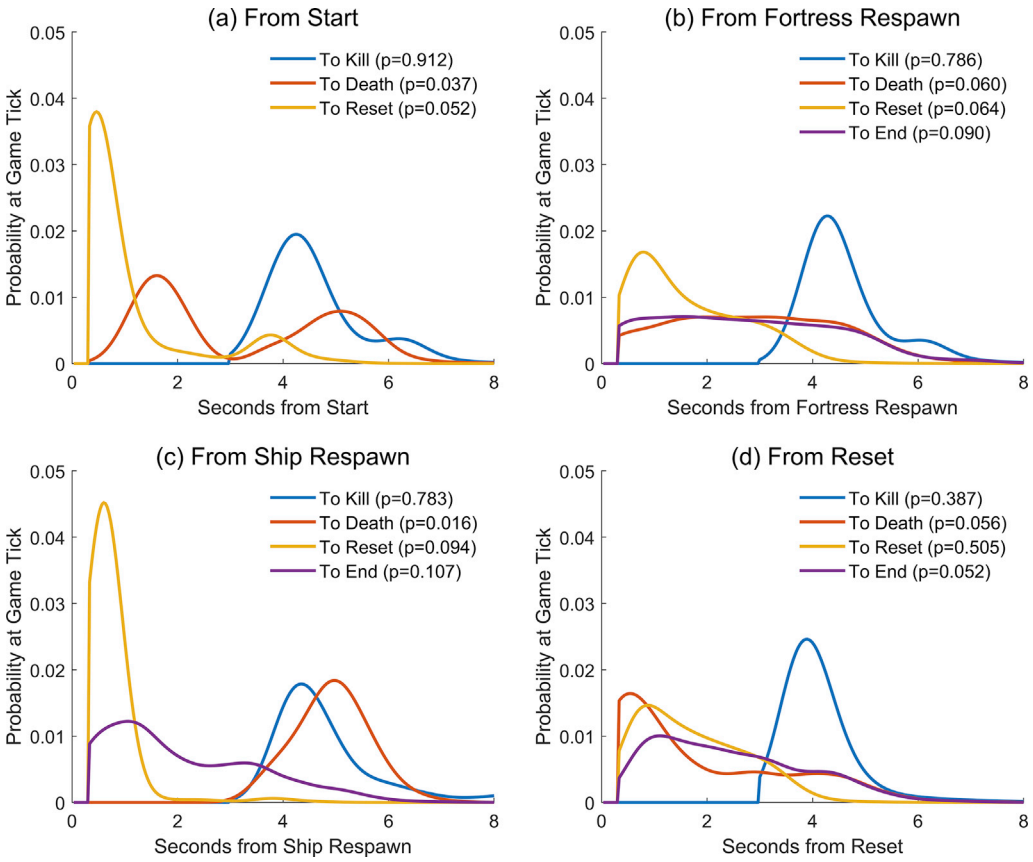

Fig. 7. Probabilities of transitions (given as "p = ") between events and distribution of intervals between events in the simulations. The statistics displayed are averaged over border widths from 40 to 160 pixels (the actual sketching procedure uses statistics specific to a width).

will be the same for all sequences of critical events and can be ignored in determining which sequence has the highest proportionality. The only thing that matters is the conditional probability of the EEG signal on game tick t_{i+1} if critical event a_{i+1} happened, relative to the probability of the signal if there were no critical event. Therefore we can rewrite Proportionality 1 as

$$Prob(a_1, a_2, \ldots, a_n) \approx \prod_{i=1}^{n-1} trans(a_i, a_{i+1}) \times f(t_{i+1} - t_i | a_i, a_{i+1}) \times \frac{P(EEG(t_{i+1}) | a_{i+1})}{P(EEG(t_{i+1}) | Null)} \quad (3)$$

The probability of any particular sequence will only involve about 20 critical events and thus only 20 ratios of conditional probabilities of the EEG on particular game ticks. These ratios are many game ticks apart (on average about 100 game ticks or about 3 s). While the ratios for adjacent game ticks are highly correlated due the temporal correlation of the EEG signal, the ratios at these distances are not. Thus, the independence assumption underlying a HSMM will be approximately correct. The HSMM-MVPA applies to all game ticks, including the 4.7% that do not have a corresponding signal vector. For these game ticks the ratios were 1 for all critical events. Thus, for these stretches of time without signal, the only information about possible events comes from the transition probabilities and distribution of delays between events.

We used the Viterbi algorithm (Rabiner, 1989) for hidden semi-Markov models to find the assignment of events that maximized $Prob(a_1, a_2, \ldots, a_n)$. This produced for each game a set of inferred events and the game ticks at which they occurred. For each game the match between the assigned events and the actual events was calculated as a sum of a recall and a precision measure (Buckland and Gey, 1994) calculated from locations of kills, deaths, and resets (since the respawns of the fortress and ship were tied to the kills and deaths). The measure of recall focused on the events that occurred in the game and identified the closest predicted event. If that event type matched and was within 2.5 s it was scored according to how many game ticks it was away —thus the maximum score was 75. If the closest event was further away or failed to match it was also scored 75. The average of these recall scores for a game can vary from 0 (perfect match of all events) to 75 (worst possible). The measure of precision applied the same scoring procedure but now started with all predicted events and found the closest actual event. These recall and precision measures were identical for 379 of the 1080 games, but were somewhat different for the rest because of differences between the actual game versus the reconstruction in number critical of events or their timing. Even when they are different they do tend to be correlated (r = 0.624). The average measure of recall was 14.1 and the average measure of precision was 11.8 for a total average match rating of 25.9 out of 150.

Fig. 8 shows the distribution of the recall and precision scores. Summing recall and precision the average match rating is 25.9. To provide a chance measure one can use the average rating between actual events and predicted events of other games, which is 95.2. 847 games are best matched by the game constructed from their EEG signal rather than any other reconstructed game. 9 games have all critical events predicted perfectly to the game tick. The mean rank of the reconstruction of a game out of the 1080 reconstructions is 7.7. The range of possible rankings is 1–1080 —if reconstructions were randomly assigned to games the expected ranking (chance) would be 540.5.

Fig. 9 shows a pair of games to illustrate the range of prediction. Part a is the 276th best-matched game with a rating of 10.6. All events but the last kill are closely predicted – the model does not complete its last kill by game's end. Part b is 787th best-matched game with a score of 36.4. 8 of the actual events (6 kills and 2 deaths) are identified relatively

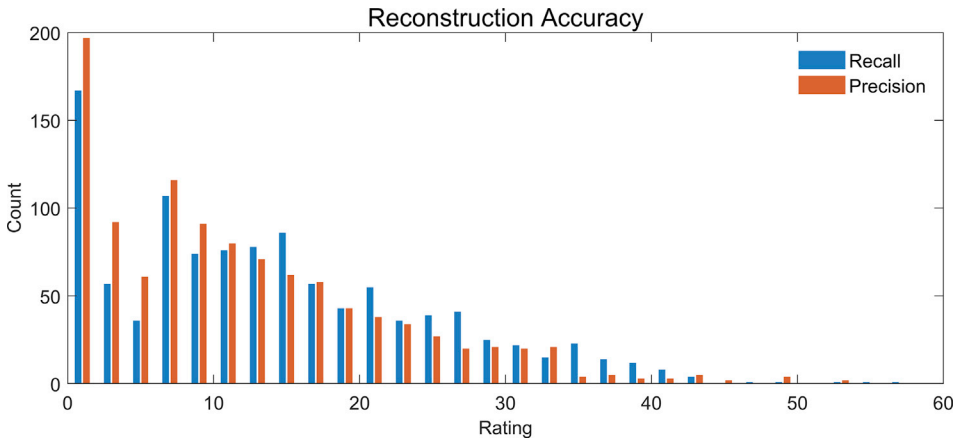

Fig. 8. Distributions of recall and precision ratings on a scale where 0 is perfect match between game and reconstructed critical events and 75 is the maximum possible mismatch score.

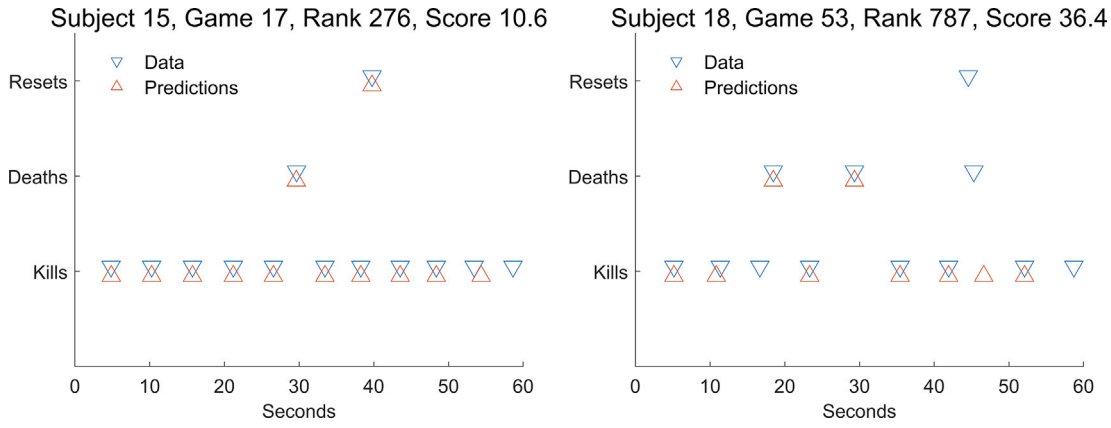

Fig. 9. Reconstructions of critical events for two games.

accurately. However, one death, one reset, and 2 kills are missed while one kill and one death are predicted that did not occur.

Table 2 reported how well the classifier labeled game ticks. We constructed a similar matrix from the results of the Viterbi algorithm looking at how well it classifies each game tick. Table 3 presents those results, which have a d-prime of 2.75, superior to both matrices in Table 2 [10]. All 20 subjects are classified better using the Viterbi algorithm than just the classifier (either method in Table 2). Moreover, the Viterbi algorithm results include classification of game ticks without a signal vector (the 1000 PCA values which are not available when there is lost EEG). Also, Fig. 9 shows that even if the Viterbi algorithm does not identify the exact game tick it often identifies a nearby game tick (e.g. the cases in the figure where the predicted and observed kills are slightly offset). Most important and not reflected in Table 3, the resulting positioning of events render a coherent interpretation of game play, which is to say that one can create sequences of key presses that would produce these critical events. While this is a better result, it leaves us short of being able to reconstruct the detailed behavior of the subject. For that we need stitching, which we will describe next.

3.4. *Stitching in detailed game play*

Stitching involves finding sequences of key presses (fires and thrusts) in simulated games that would produce the same timing of critical events as in reconstructed sketches like those in Fig. 9. It is unlikely to find a complete simulated game whose critical sketch perfectly matched a reconstructed sketch. Even though there were 2750 simulated games for each border width, no simulated game had a critical sketch matched the critical sketch of any other simulated game or actual subject game (which makes the fact that 9 of the Viterbi-reconstructed critical sketches perfectly matched the actual games quite remarkable). Rather than looking for complete games that matched, the stitching procedure just searched for segments of simulated games that could reproduce segments of the reconstructed sketch. Given that the effects of firing and thrusting are nearly independent,[11] the procedure selected thrust patterns and fire patterns from the simulated games independently.

Thrusts determine the flight path of the ship. The overall path in a game can be divided into a number of segments that begin with the ship flying in a starting configuration and ending either in a death or terminating with the end of the game. Fig. 10 illustrates how thrusts are chosen from the model games for a segment in a critical sketch. For critical sketches of games with no deaths, there is no shortage of simulated games without deaths for that border width that can serve as candidate segments (which will all span the full game). Beginning segments of these games also provide thrust sequences for segments in critical sketches that go from respawning of the ship after a death to the end of the game. The

[10] Table 3 is most comparable to Table 2b, which incorporates information about the frequency of the events.

[11] There are rare possibilities for an interaction: A fire sequence can change the consequences of a thrust sequence when the fires destroy the fortress and saves the ship from being shot when it is flying too slowly. A thrust sequence may change the consequences of a fire sequence because it controls the distance of the ship, which determines when a shot hits the fortress, potentially turning a vulnerability reset into an increment or vice versa.

Table 3
HSMM-MVPA Labeling of Critical Events (F refers to Fortress and S refers to ship).

		Game Tick Labeled as					
Classified by Viterbi Algorithm		Fortress Kill	Fortress Respawn	Ship Death	Ship Respawn	Reset	Null
Event During Game Tick	F. Kill	4937	11	13	1	2	5164
	F. Respawn	7	4937	3	13	2	5166
	S. Death	13	2	348	1	1	542
	S. Respawn	1	13	0	348	0	545
	Reset	5	3	0	0	184	1050
	Null	6211	6213	766	763	957	1,908,989

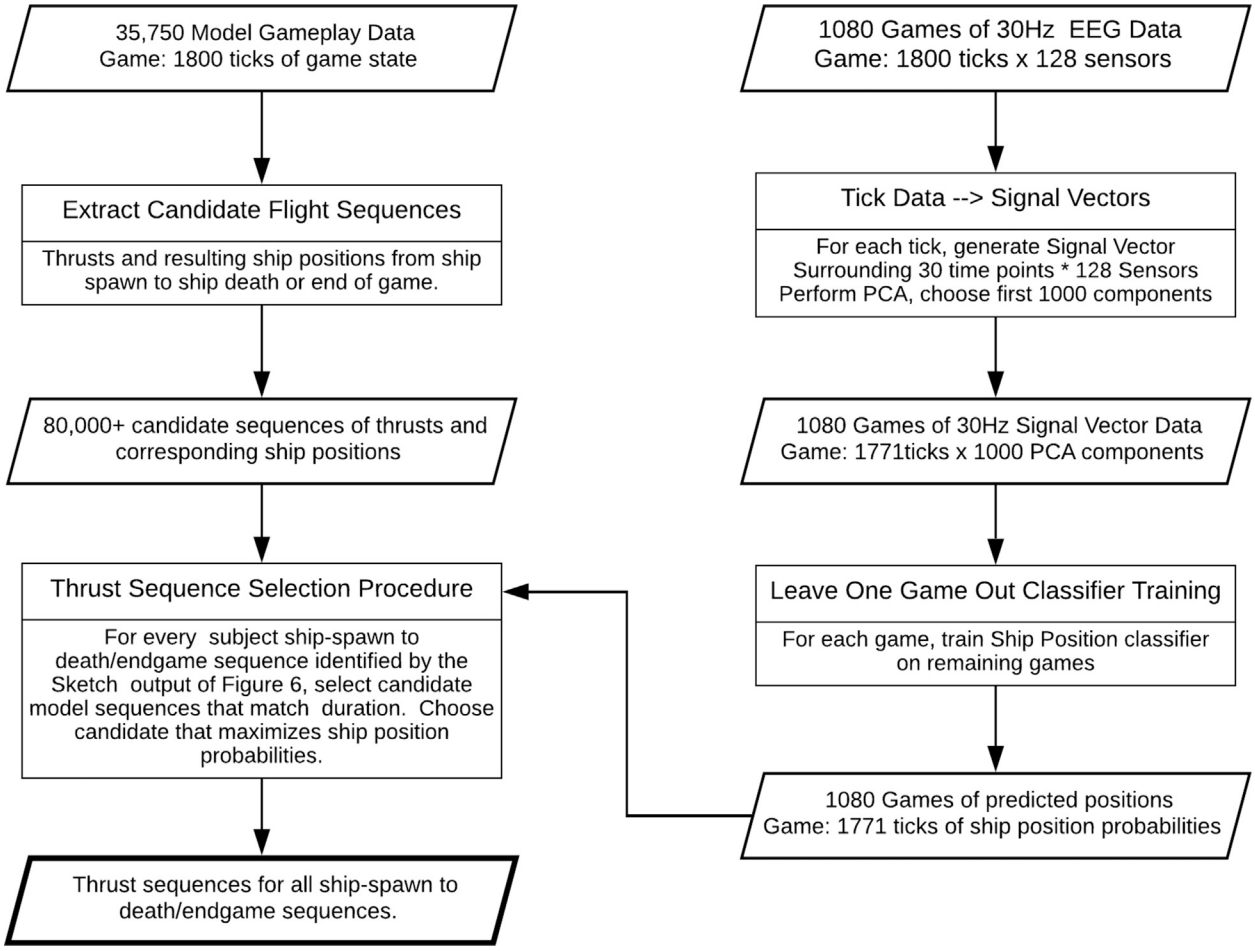

Fig. 10. Illustration of how a thrust sequence is selected for a predicted sequence of critical events that goes from a ship spawn to a death or game end. On the left, runs of the model generate large numbers of flight segments. On right, a classifier trained on other games assigns probabilities to flight positions in actual subject games. Thrust sequences from the model are chosen for segments of the critical sketches that are associated with the most probable positions.

challenge is to find segments from the model that match segments in critical sketches that go from the appearance of the ship to a death. There are nearly 1800 possible numbers of game ticks that can pass until death (the case illustrated in Fig. 10). Even if they were equally likely (which they are not), ten times more simulations than the current 35,750 simulated games in the library would be required to come close to guaranteeing model runs with a death at each game tick for each width. As it was, model ship deaths did not occur at 32% of possible game ticks in the inferred critical sketches. We relaxed the criterion and selected any deaths within 5 ticks of the desired duration. This reduced the number of missing cases to 2%. 42 cases were still not covered and for these we expanded the difference until there was a case. With these relaxations, there were on average 21 candidate segments ending in death for each death in a critical sketch. Unlike the case with the earlier action classifier, these segments involved plausible sequences of thrusts that produced a plausible flight pattern even if it ended in a death.

The EEG signal was used to select among the candidate model thrust segments, exploiting information that the signal provides about where the ship is on the screen. Part (a) of Fig. 11 shows the average scalp activity when the ship is in different 30-degree sectors. We trained a classifier to recognize which sector the ship was in plus when the ship was off the screen due to a death. We then assigned each game tick to the category that maximized the probability of the EEG signal associated with that tick. This resulted in a d-prime for discrimination among the 13 categories of 0.57, an accuracy in choosing among the 13 categories of 17.2%, and an average pairwise AUC of 0.738. Part (a) of Fig. 11 appears to show greater left-right discrimination than top-down, although the classification uses richer 1-s patterns than these single snapshots. A binary discrimination between the two left sectors from the two right sectors shows a d-prime of 1.33 (74.7% accuracy; AUC of 0.825) while a

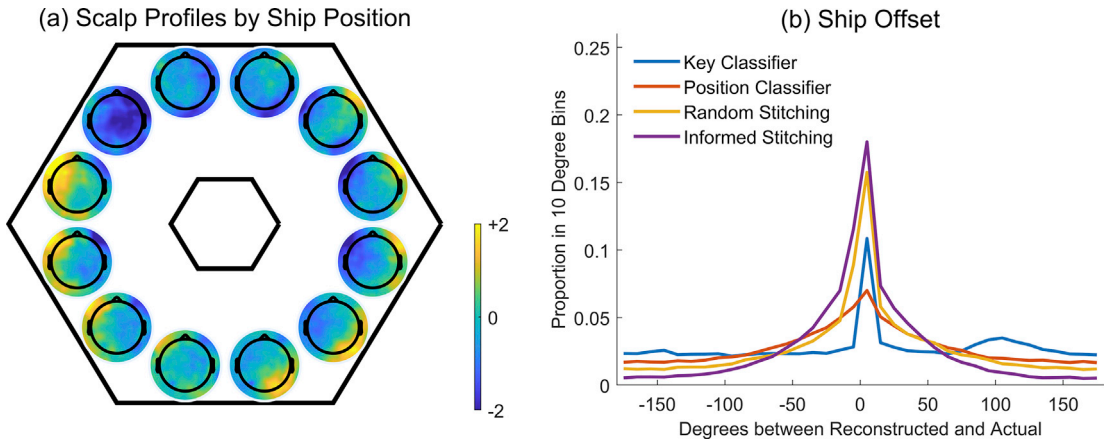

Fig. 11. Scalp profiles when ship is in various 30-degree sectors of the hexagon space. (b)Disparity between assigned angle and actual angle using different criteria for reconstructing ship position.

binary discrimination between two top sectors from the two bottom sectors shows a d-prime of 0.87 (65.3% accuracy; AUC of 0.75).

We used this classifier for ship position to select among different sequences of game thrusts produced by the model. We fed the candidate sequences of thrusts in these segments into the game engine to produce flight paths and chose the sequence whose flight path had the greatest summed log probability from the classifier. Part (b) of Fig. 11 shows the distribution of offsets between the angle of the reconstructed flight and the actual angle in the line called **Informed Stitching**. For comparison, the figure also shows the results of three other bases for positioning the ship:

1. **Key Classifier.** We took the thrusts and fires predicted by the classifier using the posterior criterion (Part b of Table 1) and fed them into the game engine as if they came from the subject. The game engine then flew the plane and we used the resulting ship positions given by the game engine.
2. **Position Classifier.** The posterior most probable ship positions according the position classifier (Part a of Fig. 11). Note that, because these positions are not produced by the game engine, they typically do not correspond to a possible flight pattern.
3. **Random Stitching.** Feeding into the game engine thrusts from segments of the right length without regard to the EEG signal of the subject.

For Informed Stitching the average angular disparity between reconstructed ship and actual ship is 40°; for Random Stitching it is 55°, for the Position Classifier it is 68°, for the Key Classifier it is 71° (randomly placing the ship on the screen would produce an average angular disparity 90°).

The stitching procedure found sequences of fires in the same way as illustrated in Fig. 10 for thrusts. However, the required fired sequences were typically shorter than the thrust sequences: sequences that would span the time in the critical sketch from either a fortress respawn or a reset (either starting the vulnerability at 0) to a kill or a reset. There tended to be many candidates from the model for reconstruction of fire sequences. The stitching procedure selected a model fire sequence from these candidates that would produce a similar pattern of EEG activity as the subject. Part (a) of Fig. 12 illustrates how the EEG signal changes with the build up of vulnerability.

Another classifier was constructed to discriminate among 17 categories of game ticks:

1-11 Game ticks with fires that raised the vulnerability to 1 through 11. In the averaged Fig. 10a these would be the game ticks associated with the first 11 vertical lines.

12 Game ticks with fires that raised the vulnerability to more than 11 but were too slow to destroy the fortress (not shown in part a of Fig. 12).

13 Game ticks with fast fires that killed the fortress after reaching a vulnerability of at least 11 (the last vertical line numbered 12 in part a of Fig. 12).

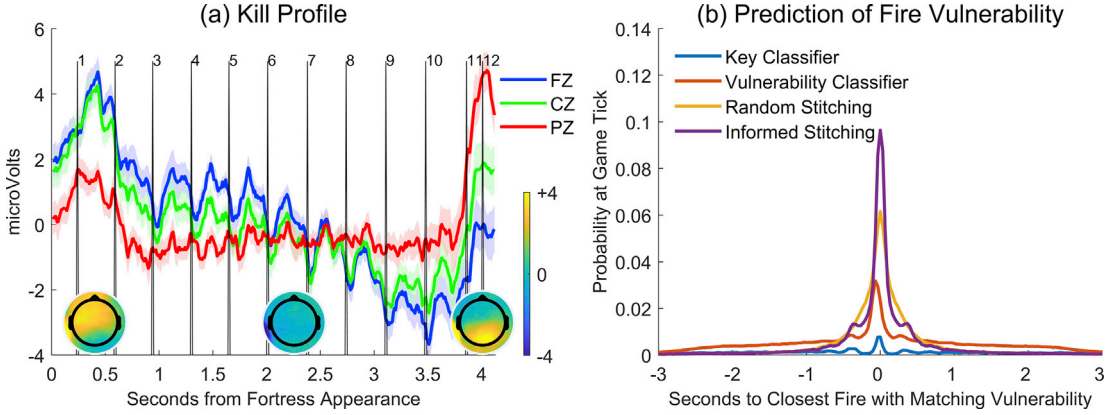

Fig. 12. (a) EEG warped to the period of an average kill (4.12 s – only games with exactly 12 fires included). The black lines show the fires that increase the vulnerability to 11 with the last fire being the fire (12) that destroys the fortress. They are numbered with the vulnerability they produce. The EEG profiles between vulnerability changes have been warped to the average vulnerability intervals. The scalp profiles are for 0.4 s, 2.25 s, and 4.0 s. (b) Distance from an assigned fire to a fire with a matching vulnerability in the actual game. Negative values mean that the fire came before the matching actual fire and positive values mean it came after.

14 Game ticks with fast fires that reset the vulnerability to 0 before reaching a vulnerability of at least 11 (not shown in part a of Fig. 12 —typically the fire will be about 150 ms before the reset which is time 0 in part e of Fig. 5).
15 Game ticks with both the ship and fortress present without a fire (almost 75% of all ticks are in this category).
16 Game ticks with the fortress absent because of a kill. (ticks covering 0–0.5 s in part a of Fig. 5 and -0.5 to 0 sec. in part b of Fig. 5).
17 Game ticks with the ship absent because of a death (ticks covering 0–0.5 s in part c of Fig. 5 and -0.5 to 0 sec. in part d of Fig. 5).

Focusing only on the ability to discriminate among the first 15 categories, which are the game ticks that occur in a fire sequence, the d-prime for discriminability is 0.91, the accuracy in choosing among the 15 categories is 24.4%, and the average pairwise AUC is 0.787. The reconstruction procedure selected the fire sequence from the model library with the greatest summed log probability of the EEG signal for its game ticks.

Part (b) of Fig. 12 illustrates the accuracy in shot placement. It plots the time between a shot with an inferred effect on vulnerability and the closest actual shot that had the same effect on vulnerability. Part (b) of Fig. 12 plots the distribution of offsets for:

1. **Key Classifier.** We took the thrusts and fires predicted by the classifier using the posterior criterion (Part b of Table 1) and fed them into the game engine as if they came from the subject. We used the resulting vulnerabilities produced by the game engine.
2. **Vulnerability Classifier.** Game ticks assigned that vulnerability by the classifier. As this was not run through the game engine the sequence of vulnerabilities often was not a possible sequence.
3. **Random Stitching.** Feeding into the game engine fires from segments of the right length to match the critical sketch without regard to the EEG signal of the subject.
4. **Informed Stitching.** Selecting the segment that produced the best match to the EEG signal of the subject and feeding that into the game engine.

The Key Classifier locates 3% of the fires within 0.1 s of a fire with the same vulnerability change, Vulnerability Classifier 15% of the fires within a 0.1 s of a fire with the same vulnerability change, Random Stitching 32%, and Informed Stitching 43%.

While Informed Stitching, which uses both the game library and the classifier, does best of the 4 alternatives at both at identifying ship position (Part b of Fig. 11) and vulnerability (Part b of Fig. 12), it is more important to have a coherent segment of the right length (Random Stitching) than it is the use the results of the stitching classifiers alone (Position and Vulnerability Classifiers). This reflects the importance of aligning the thrusts and fires with the identification of the critical events. If the reconstructed deaths are aligned with the actual deaths, the ships will be set back to the starting position at the same time. If the reconstructed kills are aligned with the actual kills, the vulnerability buildups will start at the same points.

3.5. *Summary evaluation of reconstruction accuracy*

The combination of EEG classification and the model can do well at reconstructing the critical events (Figs. 8 and 9) and capturing where the ship is in space (Fig. 11) and what the fortress vulnerability is (Fig. 12). To provide a summary measure of the overall correspondence between actual games and their reconstructions we used an equally weighted sum of three factors:

1. The z-score of the rating of the critical events (Figs. 8 and 9). As noted earlier, the reconstruction scores well on this measure with 818 of the 1080 games best predicted by their reconstruction and a mean ranking of the reconstruction is 8.4.[12]
2. The z-score of the angular disparity between the position of reconstructed ship and the actual ship averaged across the game ticks. While much better than chance this detail is less well reconstructed, with only 106 of the 1080 games best predicted by their reconstruction and a mean rank of 176.2 out of 1080.
3. The z-score of the difference between the vulnerability of the reconstructed ship and the vulnerability of the actual ship averaged across the game ticks. 803 of the games are best matched by their reconstruction and the mean rank is 14.2.

While we weighted these three equally in coming up with a combined measure, they seem ordered as above in how compelling the reconstruction is as a match to the original game. By this combined measure 846 of the games are best predicted by their reconstruction and the mean ranking of the reconstruction is 6.1. The highest ranked seem quite compelling as reconstructions of the original games. Only one was worse as a reconstruction of the game than the majority of the reconstructions of other games.

The focus in this paper has been reproduction of the details of game play. A less detailed reproduction may be adequate for many purposes. For instance, one might just want to assess how well the person is playing the game. One can used these detailed reconstructions for such summary evaluations. Fig. 13 shows the correspondence between the final scores of reconstructed and actual games, for which the correlation is 0.848. For comparison we investigated the correlation with the games produced with the Random Stitching and Key Classifier (used in Figs. 11 and 12). The correlations were 0.853 with the output of Random Stitching and 0.019 with the output of the Key Classifier. The fact that there is little difference between informed and random stitching indicates that the correlation in Fig. 13 really depends on getting the kills, deaths, and reset right and not on the details of the actions in between. Given the poor performance of the Key Classifier in playing the game, its low correlation with participant score is not surprising. Earlier we described hypothetical Key Classifiers that were 10 and 100 times more accurate than our classifier. Their correlations were 0.243 and 0.687, still less that the product of stitching a sequence of action to match the critical sketch.

4. Discussion

The success of the Sketch-and-Stitch approach depends critically on both having the EEG signal from the subject and a good model of human behavior. The variability in game play is enormous and no two games are identical, eliminating any chance that a model by itself could predict what subjects are going to do. Even though we had a library of over 35,000 games none matched a subject game nor did any two subject games match. On the other hand, while the EEG signal allows for much better than chance classification of the different aspects of game play, that information by itself is also not good enough to enable successful reconstruction.

The Sketch-and-Stitch method uses information from classification and the model at both the Sketch and the Stitch level. The Sketch level takes advantage of both the strong signals associated with critical events and statistical information about the order and timing of those events from the model. The Stitch level required the model to provide sequences of actions of the appropriate length to stitch in and selected among these according to their EEG signatures. Even if the procedure failed to choose the right detail for a period between two critical events, the reconstructed game stayed on a path that was consistent with the sketch. Thus, even after a period of significant mismatch between reconstruction and

[12] These are not exactly the same as the values reported with respect to Fig. 7 because slight differences in the timing of the deaths and occasional unintended interactions between fire and thrust sequences in the reconstructed games.

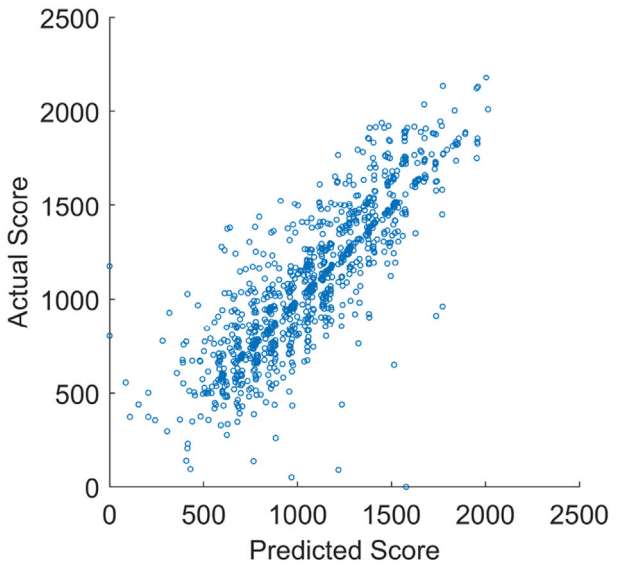

Fig. 13. Relationship between the score of the 1080 actual games and their reconstructions.

subject, the two would often come back to close correspondence.

At an abstract level, the approach in this paper bears similarities to BCI efforts for merging statistical regularities in language with EEG signals (Speier et al., 2016). These regularities can be leveraged to tailor the presentation of letters on an external device based on their base rates and posterior probabilities given the currently typed word. Likewise, these regularities can be used to support inference of intended spelling using EEG signals (Mora-Cortes et al., 2014). Our cognitive model effectively provides a "grammar" of gameplay and action. More generally, using such cognitive models could enable successful application of EEG decoding to a vastly wider range of tasks.

While we have focused on reproducing the subjects' actions because that is where we have the ground truth of the game record, the models also make commitments about the cognitive processes giving rise to these actions. For instance, as described in Anderson et al. (2019) decisions about how to pace fires is made by an evolving sense of the appropriate pacing. One could infer the threshold a subject is currently using to time fires by considering the threshold in the model segment that has been stitched in. Similarly, one could infer the subject's threshold for thrusting from the threshold in the inferred model segment. If there were alternative strategies for game play, one could infer the subject's strategy by which strategy provided the matching segments for stitching. One could use these inferences to provide feedback on what the subject needed to do to improve their performance.

The Space Fortress game has provided a good testbed for judging reconstruction from EEG. The domain is one where small differences in the timing of events can greatly affect the course of subsequent events. There is a strong sequential dependence where the earlier actions can change the consequences of later ones. The approach described in this paper handles these challenges well, and is capable of creating compelling reconstructions of game play. The approach combines training of classifiers to detect events and using a computational model to provide the statistical information about the distribution of events and the details about how these events where achieved. While reconstruction performance is good, it might be improved by better approaches to classification or tuning cognitive models to individual player's style. However, the current results do establish the potential of combining classification and cognitive models to reconstruct mental activity.

The complete game record of Space Fortress provides a reliable ground truth for training classifiers and for judging the success of reconstructions. In other work (e.g. Anderson et al., 2016) we have had similar success in parsing EEG signals using the HSMM-MVPA method without such training events. Rather, HSMM-MVPA discovered critical events in the absence of labeled training events. However, in the earlier work the intervals were much briefer and the EEG signal was constrained to be a simple ERP-like bump. Although those conditions accurately represent the majority of ERP studies with brief trials, this unsupervised method did not scale to the current situation with much longer intervals. Error in identifying events without supervision increases as one moves from the beginning or end of an interval such that replicable ERP-like bumps are no longer found. The current approach is appropriate for longer intervals provided that one has access to ground truth from some data to train the classifiers for other data.

CRediT authorship contribution statement

John R. Anderson: Writing - original draft, Writing - review & editing, Formal analysis. **Shawn Betts:** Writing - original draft. **Jon M. Fincham:** Writing - review & editing, Formal analysis. **Ryan Hope:** Formal analysis. **Mathew W. Walsh:** Writing - review & editing, Formal analysis.

Acknowledgements

This work was supported by the Office of Naval Research Grant N00014-15-1-2151. We thank Cvetomir Dimov for his comments on the paper. The data and analyses which were used to create used to create the figures in the paper are available at http://act-r.psy.cmu.edu/?post_type=publications&p=31867.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.neuroimage.2020.116999.

References

Anderson, J.R., Betts, S., Bothell, D., Hope, R., Lebiere, C., 2019. Learning Rapid and Precise Skills. Psychological Review.
Anderson, J.R., Betts, S., Ferris, J.L., Fincham, J.M., 2010. Neural imaging to track mental states while using an intelligent tutoring system. Proc. Natl. Acad. Sci. Unit. States Am. 107, 7018–7023.
Anderson, J.R., Borst, J.P., Fincham, J.M., Ghuman, A.S., Tenison, C., Zhang, Q., 2018. The common time course of memory processes revealed. Psychol. Sci. 29, 1463–1474.
Anderson, J.R., Fincham, J.M., Schneider, D.W., Yang, J., 2012. Using brain imaging to track problem solving in a complex state space. Neuroimage 60, 633–643.
Anderson, J.R., Zhang, Q., Borst, J.P., Walsh, M.M., 2016. The discovery of processing stages: extension of Sternberg's method. Psychol. Rev. 123, 481.
Brouwer, A.-M., Hogervorst, M.A., Van Erp, J.B., Heffelaar, T., Zimmerman, P.H., Oostenveld, R., 2012. Estimating workload using EEG spectral power and ERPs in the n-back task. J. Neural. Eng. 9, 045008.
Buckland, M., Gey, F., 1994. The relationship between recall and precision. J. Am. Soc. Inf. Sci. 45, 12–19.
Cavanagh, J.F., Castellanos, J., 2016. Identification of canonical neural events during continuous gameplay of an 8-bit style video game. Neuroimage 133, 1–13.
Delorme, A., Makeig, S., 2004. Eeglab: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. J. Neurosci. Methods 134, 9–21.
Donchin, E., 1989. The learning strategies project: introductory remarks. Acta Psychol. 71, 1–15.
Frederiksen, J.R., White, B.Y., 1989. An approach to training based upon principled task decomposition. Acta Psychol. 71, 89–146.
Gopher, D., Weil, M., Siegel, D., 1989. Practice under changing priorities: an approach to the training of complex skills. Acta Psychol. 71, 147–177.
Kerous, B., Skola, F., Liarokapis, F., 2018. EEG-based BCI and video games: a progress report. Virtual Real. 22, 119–135.
Kim, M.-K., Kim, M., Oh, E., Kim, S.-P., 2013. A review on the computational methods for emotional state estimation from the human eeg. Comput. Math. Methods Med. 2013.
Krauledat, M., Dornhege, G., Blankertz, B., Losch, F., Curio, G., Muller, K.-R., 2004. Improving speed and accuracy of brain-computer interfaces using readiness potential features. In: The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, vol. 2. IEEE, pp. 4511–4515.
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. J. Neural. Eng. 15, 031005.

Maclin, E.L., Mathewson, K.E., Low, K.A., Boot, W.R., Kramer, A.F., Fabiani, M., Gratton, G., 2011. Learning to multitask: effects of video game practice on electrophysiological indices of attention and resource allocation. Psychophysiology 48, 1173–1183.
Mathewson, K.E., Basak, C., Maclin, E.L., Low, K.A., Boot, W.R., Kramer, A.F., Fabiani, M., Gratton, G., 2012. Different slopes for different folks: alpha and delta EEG power predict subsequent video game learning rate and improvements in cognitive control tasks. Psychophysiology 49, 1558–1570.
Mora-Cortes, A., Manyakov, N.V., Chumerin, N., Van Hulle, M.M., 2014. Language model applications to spelling with brain-computer interfaces. Sensors 14, 5967–5993.
Noh, E., Herzmann, G., Curran, T., de Sa, V.R., 2014. Using single-trial EEG to predict and analyze subsequent memory. Neuroimage 84, 712–723.
Polich, J., 2012. Neuropsychology of p300. In: Oxford Handbook of Event-Related Potential Components, vol. 159, p. 88.
Rabiner, L.R., 1989. A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE 77, 257–286.
Smulders, F.T., Miller, J.O., Luck, S., et al., 2012. The lateralized readiness potential. In: The Oxford Handbook of Event-Related Potential Components, pp. 209–229.
Speier, W., Arnold, C., Pouratian, N., 2016. Integrating language models into classifiers for bci communication: a review. J. Neural. Eng. 13, 031002.
Su, K.-m., Hairston, W.D., Robbins, K., 2018. EEG-Annotate: automated identification and labeling of events in continuous signals with applications to EEG. J. Neurosci. Methods 293, 359–374.
Wickens, T.D., 2002. Elementary Signal Detection Theory. Oxford University Press, USA.