Where Should Researchers Look for Strategy Discoveries during the Acquisition of Complex Task Performance? The Case of Space Fortress

Marc Destefano and Wayne D. Gray
Rensselaer Polytechnic Institute

Abstract

In complex task domains, such as games, students may exceed their teachers. Such tasks afford diverse means to trade-off one type of performance for another, combining task elements in novel ways to yield method variations and strategy discoveries that, if mastered, might produce large or small leaps in performance. For the researcher interested in the development of extreme expertise *in the wild*, the problem posed by such tasks is "where to look" to capture the explorations, trials, errors, and successes that eventually lead to the invention of superior performance. In this paper, we present several successful discoveries of methods for superior performance. For these discoveries we used *Symbolic Aggregate Approximation* as our method of identifying *changepoints* within score progressions in the venerable game of *Space Fortress*. By decomposing performance at these changepoints, we find previously unknown strategies that even the designers of the task had not anticipated.

Keywords: expertise, performance, Space Fortress, SAX, changepoint analysis, skill acquisition, plateaus, dips, leaps, strategy discovery, method invention

Introduction

In studying the behavior of people who become expert performers, we must look beyond group measures, and focus on the behavior of individuals; that is, at their explorations, failures, and successes as they strive to become experts. Ours is a variant of Ericsson and Ward's (2007) expert performance approach with the twist that we wish to capture our performers in the act of discovering the methods that result in extreme expertise.

For example, we had people play the complex game of *Space Fortress* (Mané & Donchin, 1989; Donchin, 1995) across 31 sessions of 8 games per session, or 248 games total. Averaging the data across hours and across players produces the classic performance curve shown at the bottom of Figure 1, which shows few exceptions to the story (Fitts & Posner, 1967; Newell & Rosenbloom, 1981) of steady, incremental increases in performance with practice. Unfortunately for this story, as the upper plot shows, this smooth average represents none of players' actual performance. Although each of our 9 players show improvements with time, these improvements are not smooth and each individual curve is more notable for its plateaus, dips, and leaps (Gray & Lindstedt, 2016) than for smooth and steady improvement with practice.

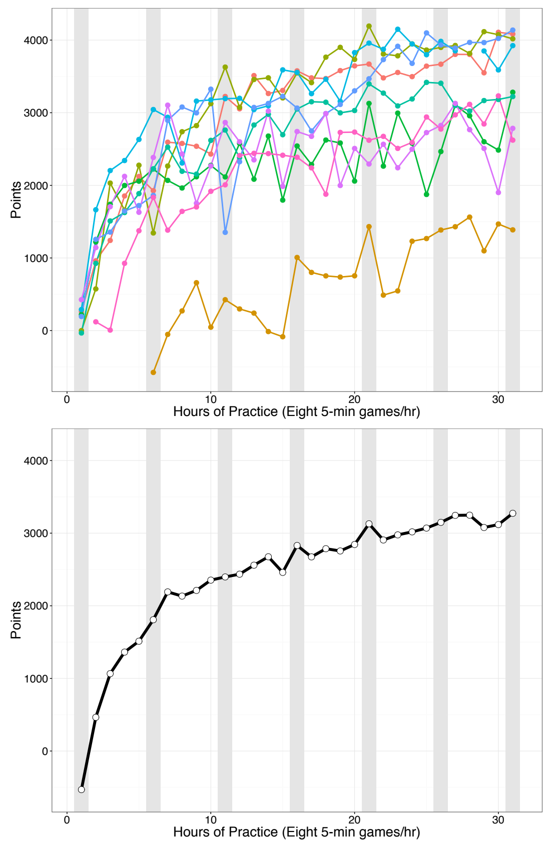

Figure 1. *Space Fortress Skill Acquisition Curves.* The top plot shows the scores of 9 Players who played Space Fortress for 31 hours each. To keep the plots compact, the early games for the lowest scoring players are truncated for hours 1-4. The bottom plot shows the average performance per hour for all 9 Players. The average (lower plot) is textbook perfect. As is clear, this average performance does not represent the progress of any of our 9 individual players.

Why so much fluctuation? Certainly, there are many external factors which could lead to such irregularity: time of day, stress, motivation, etc. Averaging across these factors has enabled the field to formulate general laws which capture what our subjects have in common. However, we argue that, especially for people at the limits of expertise, it would be a mistake to dismiss individual variations as "statistical noise."

We believe that something meaningful is going on within individuals during these variations, and that is it important to study them as individuals. Different subjects of different skill levels may discover or invent different strategies. Different methods for implementing the same strategy may be easier for some individuals than others, and the nature of such differences would not be apparent in aggregate data.

Extreme expertise in tasks such as Space Fortress requires highly complex strategies that demand tremendous cognitive resources. More specifically, they are not simply "reactive" – they require strategic choices as well as the invention of methods that can be executed in the time alloted by a fast paced game, situations in which, "even hesitating requires a decision"[1] and in which a temporary decrease in performance may be the cost of exploring a different way of doing things. Gray and Lindstedt tell us that:

> . . . dips in performance [are] not simply a concern due to their possibility of demotivating learners . . . but should be viewed as periods of experimentation, discovery, trial & error, and successive approximations to developing league-stepping habits. The capture and study of these dips may well be the most important and overlooked task of the past 120 years of Experimental Psychology. (Gray & Lindstedt, 2016)

Thus, we propose that some of these fluctuations in performance are caused by strategy discovery, testing, and exploitation. To measure these fluctuations, we settled on SAX (Lin, Keogh, Lonardi, & Chiu, 2003) as our Change Point Algorithm to identify time periods to target for further investigation. Here, we present a case study of an expert user discovering and adopting strategies in *Space Fortress*, a game that allows incremental progress in low-level perceptual-action skills, while also allowing for planning and strategy formation. In doing so, we have been able to determine specific moments when our focal players implemented radically advanced strategies to maximize their performance.

Space Fortress

Space Fortress (Mané & Donchin, 1989) is a cognitive-task-oriented video game with a long history in the Cognitive Science community. Perhaps the most efficient way to describe it is that it's similar to *Star Castle* (Cinematronics, 1980) combined with a Sternberg-style memory task (Sternberg, 1966) with an AX-CPT task thrown in (Cohen & Servan-Schreiber, 1992). There have been many different versions of the game developed over the past 30 years, with some evolution of the ruleset over that time.

In our version (Destefano, 2010) the player operates a small spaceship via second-order (acceleration-based) thrust

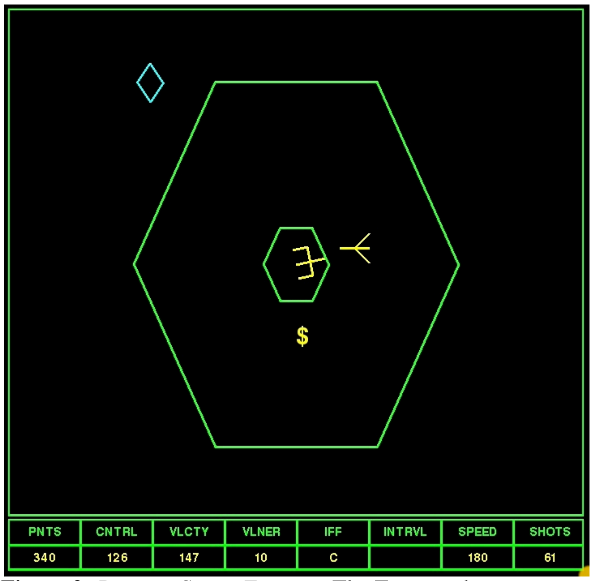

Figure 2. *Pygame Space Fortress*. The Fortress does not move from the center of the screen, but will rotate to face the player's ship (here at the right of the Fortress). The blue diamond is a mine, which is actively 'chasing' the player.

in frictionless space within a 2D plane, with first-order (velocity-based) rotation. If the ship exits the world space by flying off the "edge" of the world, it immediately reenters on the opposite side, giving the game world the topography of a torus. Within this plane are two nested hexagons: a small one to determine the "bounce zone" of the Fortress (if you pass its boundaries, you incur a control point penalty and your velocity vector is reversed), and a large one to encourage the strategy of keeping near the Fortress. By staying within the boundaries of the large hexagon, you gain a small control point bonus once per second. The goal of the game is to maximize *Total* score.

There are many ways of scoring points, each of which entails complex methods involving some combination of memory, timing, and perceptual-motor coordination. For example, *killing* the Fortress requires staying alive while shooting at least ten times with a delay of at least 250 msec between shots. Once the Fortress has been hit 10 times, it is *vulnerable* and can be destroyed by firing two shots within 250 msec of each other. However, if the inter-hit-interval is > 250 msec, then the Fortress is not killed rather, its vulnerability is reset from 10 to zero.

While the Fortress is spinning around trying to shoot the player's ship, mines are appearing every few seconds at random locations and actively chase the ship. The ship cannot destroy the Fortress while a mine is on the screen. However, the Fortress can continue trying to destroy the ship. To make things more difficult, there are two types of mines, friends or

[1]Lec (1962): Auch zum Zögern muß man sich entschließen – "Even the hesitation you have to decide."

foes. When a mine appears, a 3-letter sequence appears in the *IFF* (identify friend or foe) slot at the bottom of the screen (see Figure 2). If these three letters are one of the three 3-letter sequences memorized before the game started, the mine is a "foe mine." If not, then the mine is a "friend mine." When either mine is on the screen the Fortress cannot be damaged and the player runs the risk of having their ship destroyed by bumping into the mine. Killing a friend mine is easy. Just shoot at it. Killing a foe mine is harder as, before it can be shot, it first must be tagged as a "foe" by tapping a special key twice at a interkeystroke interval between 250–400 ms.

While the above is going on, *bonus symbols* appear and disappear below the Fortress at regular intervals. Whenever two '$' symbols appear one after the other, the player has the option of collecting a bonus. If the player presses the "bonus missiles" key, the number of available missiles increases. Alternatively, if the player presses the "bonus points" key, a score increase is received.

With this scoring system, the optimal strategy is thus to always keep flying (but not too fast), shoot the Fortress as often as possible (but not too quickly), stay within the hexagons at all times, always memorize the three 3-digit sequences that appear before each game, deal with mines as quickly as possible, and be sure to grab all the bonuses you can (either points or missiles, as needed).

Pygame Space Fortress (PSF) (Destefano, 2010), which allows for a finer-grained resolution of logged data than prior versions, was used in a longitudinal study to measure the performance of players as they progressed from novices to high-scoring experts. Specifically, Rensselaer students were recruited to play PSF for 31 'hours' each. As in previous research, a *Space Fortress* hour is defined as eight 5-min games, with arbitrary break times between each game. (*Space Fortress* has never had victory conditions — skill was determined by how many points a player could achieve in a fixed amount of time).

In the first session, players played an hour of PSF on a lab computer and had a copy of PSF installed on their laptop. They then played the next five hours, at home, in 1-hr (8 game) increments. At the end of the fifth hour, the home version of the game would lock up. This required the player to return to the lab to play an 'hour', while the experimenter collected log files from the last 5 "home" hours and integrated them into the timeline of data for each player. Prior to leaving the lab, the player's laptop was reset to play and collect data for an additional 5 hours. Hence, each of our 9 players, had a total of 31 hours of play divided into 6 lab and 25 home sessions.

Symbolic Aggregate Approximation

As we were looking for plateaus, dip, and leaps in individual performance, we needed a *Change Point Algorithm* (CPA) to identify points at which scores increased or decreased faster than would be expected by chance. Although many types of CPAs have been developed, we use the Symbolic Aggregate approXimation method (SAX) (Lin et al., 2003). SAX discretizes time-series data by first dividing the time in question into equal segments (in our case, 31 – one for each 'hour' of play), then normalizes the dependent variable and divides it into an arbitrary number of quantiles along the normal distribution function. We chose four quantiles as an appropriate balance to demonstrate possible effects of plateaus, dips, and leaps in performance, without cluttering our analysis with an oversensitive measure. See Figure 3 for an example of applying the SAX algorithm to notional data.

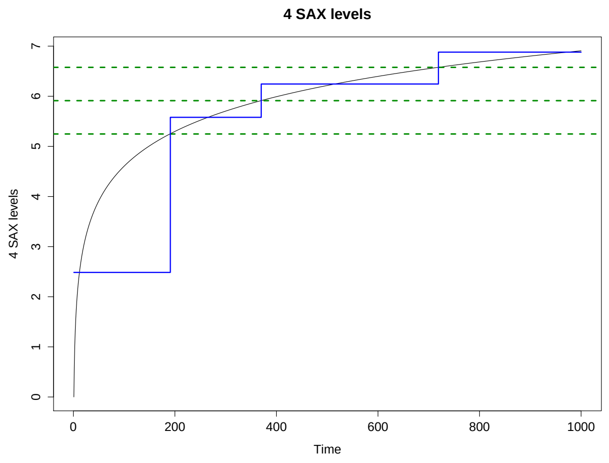

Figure 3. The application of the SAX algorithm to notional data that increases logarithmically over time. The figure superimposes a "perfect" performance curve with the result of analyzing the curve through SAX using an "alphabet size" of 4: The dashed green lines separate the 4 quantiles, or bands of categorization, and the solid blue line indicates which band the current data point falls into.

Player 3534

We begin our dive into the relationship between performance plateaus, dips, and leaps with strategy discovery with a close look at data from Player 3534. Figure 4 shows that player's total score over 31 clusters of eight 5-minute PSF games. For this example, we focus on the transition from the plateau between hours 15 through 18, to the dip in hour 19, to the leap in game 20 (indicated in the figure by the gray bar). Is this merely statistical noise, the player having an "off-day"? A cursory look at the data suggests an affirmative answer, but closer investigation reveals a strategy discovery.

Many measures of performance drop in hour 19 — not only the total score, but *all* of the individual scores: PNTS, VLCTY, CNTRL, and SPEED. Not only that, but bonus captures drop in that hour, and even the accuracy of the player's shots decreases into a lower SAX band. Seeing all these measures of performance falter in this particular hour, it is tempt-

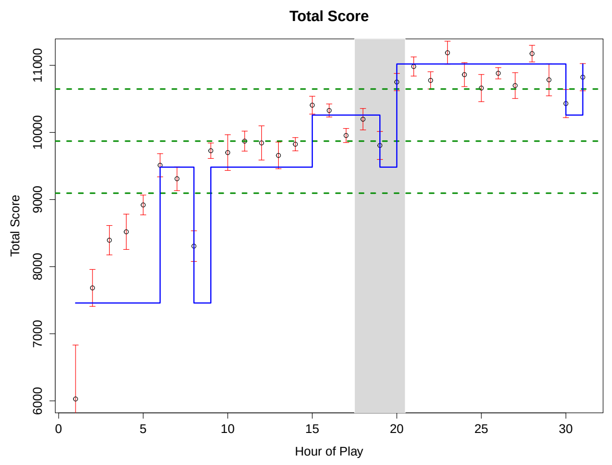

Figure 4. Player 3534's total score over 31 nonconsecutive hours of Space Fortress play. The figure shows a general trend of logarithmic growth, to which we apply SAX analysis to help us identify plateaus, dips, and leaps. As shown in 3 we then subdivide the figure into 4 quantiles. The blue line shows the band that each data point falls into, and the vertical gray bar highlights the three hours of play that is the focus of subject 3534's strategy implementation — a plateau transitioning into another plateau, via a dip and a leap in performance.

ing to dismiss these results as a simple consequence of a poor night's sleep, low motivation, or a distracting environment. Indeed, were we to cease the investigation at this point, we would write off this situation as an anomaly, a simple random fluctuation in performance. However, there are other measures of skill that complicate this simple conclusion.

As shown in Figure 5, Player 3534 was just as adept at destroying the Fortress and shooting mines in hour 19 as in hour 18. This might seem contradictory — given that these are the primary goals of the game, how could it be that the player flew more poorly and was less accurate in shooting, yet still maintained high counts in destroying both mines and Fortresses? The answer is connected to an often overlooked rule of *Space Fortress*:

> The mine appears five seconds after the destruction of either the Fortress or another mine.

Another way of phrasing this rule is that the mine does not and will not appear within five seconds of destroying the Fortress or another mine. This being the case, what happens when the player becomes skilled enough at the game to shoot the Fortress ten times (with at least 250 milliseconds between each shot!), and then double-shoot it to destroy it, *all within five seconds*? The player would then be able to destroy the Fortress over and over again, constantly resetting the internal timer that controls the mine's appearance, and thus *preventing the mine from ever appearing*.

This is a legitimate strategy, and one that might even seem optimal at first glance — after all, it takes a high degree of

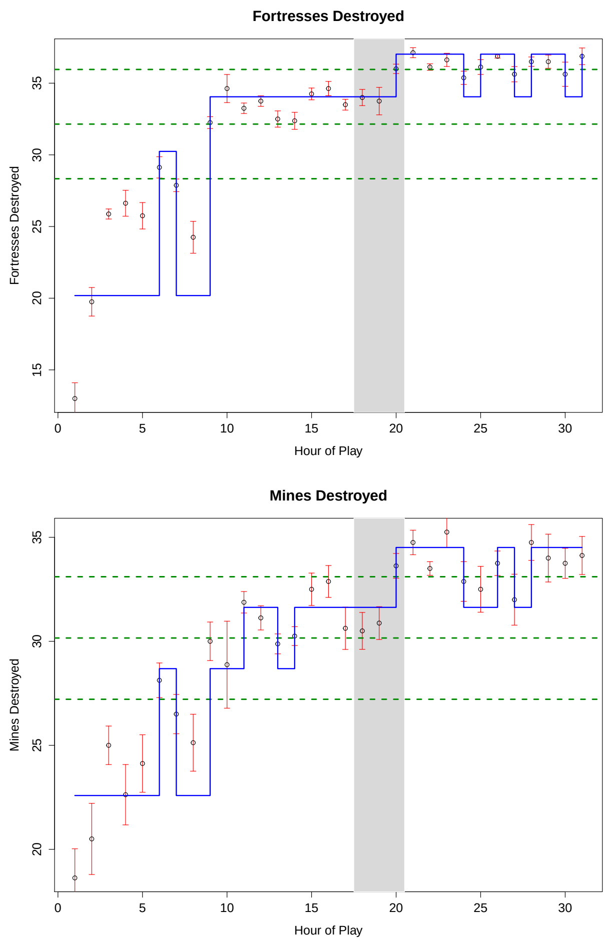

Figure 5. Player 3534's performance over time in destroying Fortresses (top figure) and mines (bottom figure). Unlike other measures of performance (score, accuracy, etc.), 3534's performance in these measures did not drop in hour 19.

skill to pull it off, and the player can now spend all their time and effort interleaving just two subtasks (the Fortress and the bonus symbol) instead of three. Indeed, Player 3534 experimented with it briefly, as shown in Figure 6 (note, in Figure 6, the unit of the x-axis is individual games, not hours). By preventing the mines from appearing, the player misses out on getting points in both the PNTS and SPEED scores, and the trade-off turns out to not be worth it.

What then? It is still desirable to destroy the Fortress as often as possible, but experts will also allow mines to appear. The best strategy our players have found, including 3534, is the following:

1. Fly slowly and as closely to the Fortress as possible without bouncing into the small hexagon.
2. Shoot the Fortress as quickly as possible to increase its vulnerability to the point where it becomes destructible, without shooting it so quickly that its vulnerability resets.

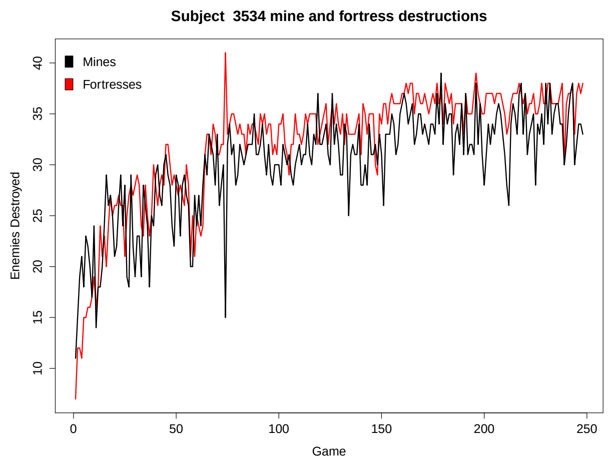

Figure 6. Number of mines and Fortresses destroyed by subject 3534 for each of their 248 games. Note the spike in Fortress kills in Game 74, with its corresponding drop in mine kills. In this particular game, 3534's score dropped by approximately 2000 points (roughly a 20% drop from recent previous games), which was harsh enough for 3534 to abandon the strategy immediately.

3. Wait for the mine to appear, while aiming your ship towards the most probable location for that to happen (typically away from the closest screen edges to the ship).
4. Manage the mine as normal, either shooting it right away if it's a friend, or tagging it as a foe and then shooting it.
5. Double-shoot the Fortress as quickly as possible.
6. Repeat, while also interleaving in the bonus symbol task when appropriate.

Step 3 here is the insight that allows the best players to pull ahead of the rest of the pack. This strategy:

- Requires the player to be fast and accurate enough to be able to shoot Fortresses continuously, *but choose not to do so.*
- Requires a different flight pattern than the one used to follow the strategy of destroying the Fortress as quickly and as often as possible.

To further investigate this phenomenon, we developed a new performance measure: the average number of mines per game that were destroyed *within one second of shooting the Fortress*, shown for player 3534 in Figure 7. Critically, this measure does not drop in hour 19, and then skyrockets into hours 20 and 21. We are now faced with the following collection of premises for our player in regards to hour 19, the dip that precedes the leap:

1. An equivalent number of Fortresses and mines were destroyed.

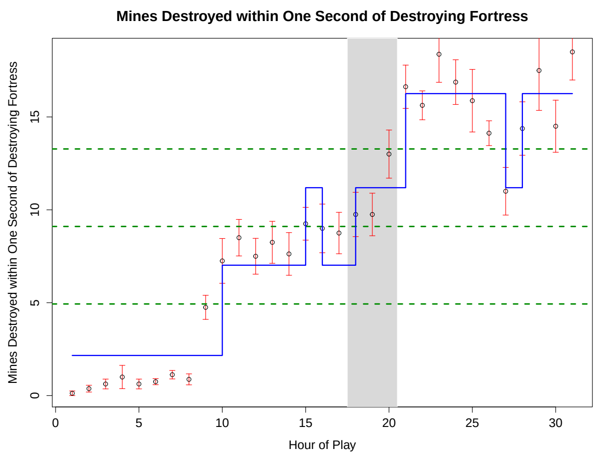

Figure 7. The average number of mines per game that player 3534 destroyed within one second of destroying the Fortress. This is an advanced strategy that takes skill and coordination to perform effectively.

2. The Fortress was destroyed within one second of destroying the mine an equivalent number of times.
3. The player flew less well (lower VLCTY and CNTRL scores).
4. Accuracy dropped — more targets were missed.
5. Then in hour 20, the player shows a dramatic increase in the use of the one-second strategy, with a corresponding leap in total score.

We thus infer from this collection of premises that our player *invented, tested, and implemented* a new flight pattern that allowed them to better prepare for mine onset once the Fortress was vulnerable, and with this new pattern the player was able to exploit the one-second strategy more often.

VLNER to 9

There is an even-more-advanced addendum to this strategy that was discovered by Player 4171. This strategy requires discovering another minor rule of *Space Fortress*:

> Shooting a friendly mine will increase the vulnerability of the Fortress by 1

This rule becomes relevant when combined with two other pieces of information. First, once the initial stock of 100 shots is depleted, firing a shot will incur a 3-point penalty. It can thus be reasoned that each shot is worth some small number of points (slightly less than 3). Second, most mines (70% of them, in fact) are friendly. With these three facts, Player 4171 discovered that approximately three points could be saved with every mine appearance by implementing the following strategy: Shoot the Fortress to a vulnerability of 9 (instead of the typical 10), wait for the mine, and the 70% of the time that the mine is friendly, shooting it will increase

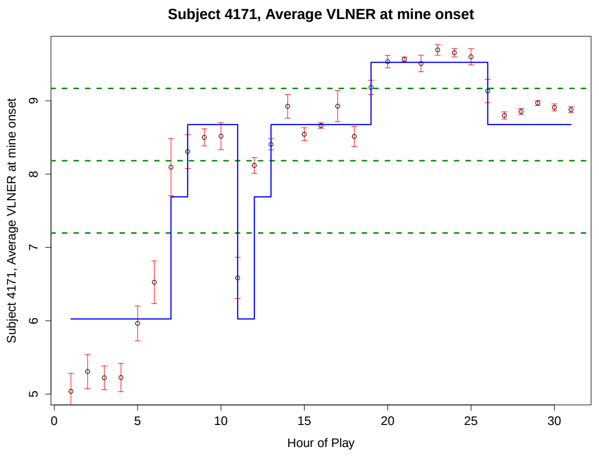

Figure 8. The average vulnerability rating of the Fortress when the mine appears for Player 4171. Note the clear transition from a plateau of 10 to a plateau of 9 once the Player has logged 25 "hours" playing the game. This is a hyper-advanced strategy, one which is well above and beyond the wait-for-the-mine-to-appear strategy detailed previously.

the vulnerability of the Fortress to 10, meaning that it can be immediately destroyed. For the remaining 30% of mines that appear, once they are destroyed it is then necessary to shoot the Fortress once more before destroying it with a double-shot. Figure 8 shows Player 4171 adopting this strategy 25 hours into the study.

Conclusion

We addressed the problem of "where to look" to capture the explorations, trials, errors, and successes that result in new strategies and methods for complex, task performance. This work should provide a strong complement to recent efforts to rethink the study of skill acquisition (e.g., see Anglim & Wynton, 2015; Tenison & Anderson, 2015).

Strategy discovery and method invention become "blurred out" in averaged data, and subtle but critically important signals are lost, such as the signs of strategy invention and the markers for strategy shift that we see by applying SAX analysis to individual *Space Fortress* players.

The plateau, dips, and leaps approach (Gray & Lindstedt, 2016) maintains that extreme expertise involves both slow accretion of low-level skill and bursts of exploration which sometimes leads to new strategies or methods. Such explorations are often punished by drops in performance and abandoned. Other times, as in the cases documented here, success may leap performance to new levels of proficiency.

Acknowledgements

Address correspondence to Wayne Gray – grayw@rpi.edu. The work was supported, in part, by grant N000141310252 to Wayne D. Gray from the Office of Naval Research, Dr. Ray Perez, Project Officer.

References

Anglim, J. & Wynton, S. K. A. (2015). Hierarchical bayesian models of subtask learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *41*(4), 957–974.

Cinematronics. (1980). Star Castle. Arcade.

Cohen, J. D. & Servan-Schreiber, D. (1992). Context, cortex, and dopamine: a connectionist approach to behavior and biology in schizophrenia. *Psychological Review*, *99*, 45–77.

Destefano, M. (2010). *The mechanics of multitasking: the choreography of perception, action, and cognition over 7.05 orders of magnitude* (Doctoral dissertation).

Donchin, E. (1995). Video games as research tools - the space fortress game. *Behavior Research Methods Instruments & Computers*, *27*(2), 217–223.

Ericsson, K. A. & Ward, P. (2007). Capturing the naturally occurring superior performance of experts in the laboratory toward a science of expert and exceptional performance. *Current Directions in Psychological Science*, *16*(6), 346–350.

Fitts, P. M. & Posner, M. I. (1967). *Human performance*. Belmont, CA: Brooks Cole.

Gray, W. D. & Lindstedt, J. K. (2016). League-stepping habits as an escape route (with plateaus, dips, and leaps) from stable suboptimal performance. *Cognitive Science [accepted pending revisions]*.

Lec, S. J. (1962). *Unfrisierte gedanken (unkempt thoughts)* (J. G. (translator), Ed.). New York: St. Martin's Press.

Lin, J., Keogh, E., Lonardi, S., & Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on research issues in data mining and knowledge discovery* (pp. 2–11). DMKD '03. San Diego, California: ACM.

Mané, A. & Donchin, E. (1989). The Space Fortress Game. *Acta Psychologica*, *71*(1-3), 17–22.

Newell, A. & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition* (pp. 1–55). Hillsdale, NJ: Lawrence Erlbaum Associates.

Sternberg, S. (1966). High-speed scanning in human memory. *Science*, *153*, 652–654.

Tenison, C. & Anderson, J. R. (2015). Modeling the distinct phases of skill acquisition. *Journal of Experimental Psychology: Learning, Memory, and Cognition*.