ACT-R Workshop

9:00am Session: Learning and Transfer in Complex Environments
John Anderson – Transfer of Cognitive Skills
Frank Ritter – Predictions and Test of a Model of Learning and Retention
Christian Lebiere and Edward Cranford – Decision Making in the Presence of Deceptive Signals
10:20am Break
10:40am Session: Neural and Perceptual Embodiments
John Lindstedt and Michael Byrne – Simple Agglomerative Visual Grouping for ACT-R
Patrick Rice and Andrea Stocco - Using TMS to Test the Associations between ACT-R Modules and Cortical Regions
Andrea Stocco - ACT-R as a Model for the Brain's Functional Connectivity: Insights from the Human Connectome Data
12:00pm Lunch
1:30pm Session: Human Machine Interaction
Greg Trafton – Two Models of Social Influence
Sterling Somers – CogXAI: Cognitively eXplainable Artificial Intelligence
Nele Russwinkel - Developing a Concept of an Active Self through Natural Interaction
2:50pm Break
3:10pm Session: Future of ACT-R
Dan Bothell – Updates
Everyone – Open Discussion
4:30pm Adjourn

ACT-R is Not Monolithic

1. While ACT-R may be maintained from CMU it no longer resides at CMU. The community motto is "Let a thousand flowers grow".
2. While ACT-R is originated as LISP software for purposes of simulation, it is no longer tied to LISP. There are many theoretically-motivated extensions and alternative practicality-motivated alternative implementations. There are full-functioning Python and Java versions.
3. The ACT-R community shares a commitment to the "No Magic" Principle" -- cognitive theory has to run and it has to **predict** data
4. The theory includes assumptions that are more core, shared by most in the community, and others that are more peripheral and often the subject of active experimentation.

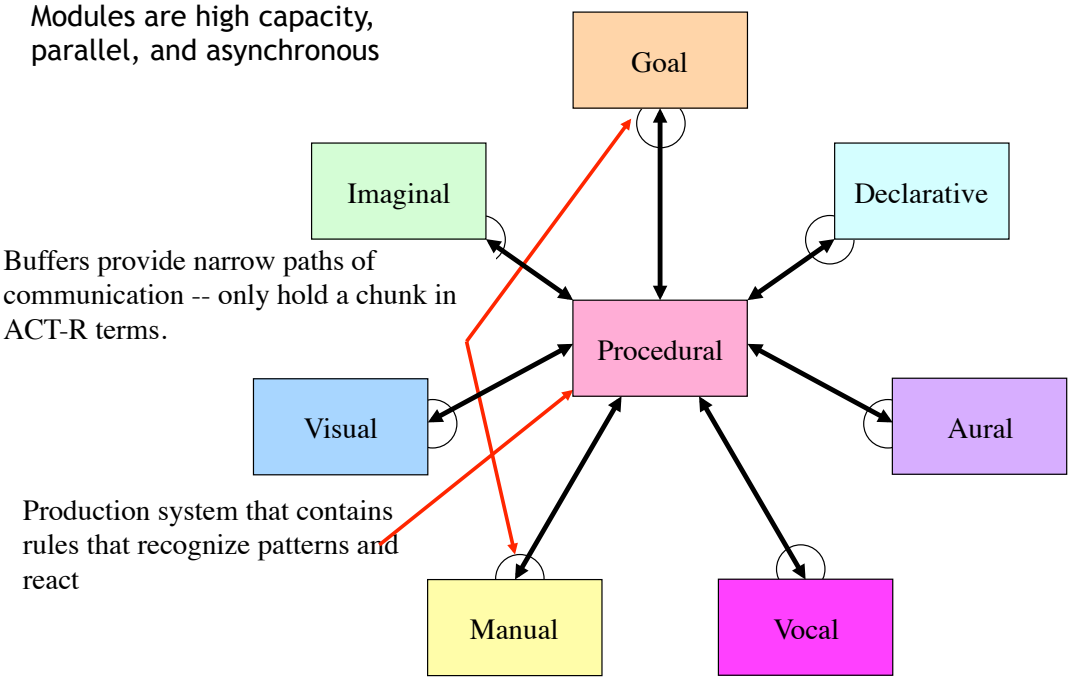
ACT-R: The Oldest Core Principles

1. The Procedural-Declarative Distinction
   a. The declarative component originated in Anderson & Bower (1973) HAM network representation of memory.
   b. The procedural component originated in Newell's (1973) production system theory of cognitive control.
   c. Both the procedural and declarative components have evolved far from these origins.
2. The Symbolic-Subsymbolic Distinction
   a. In addition to the symbolic level that represented knowledge there is a subsymbolic level that controls access to that knowledge.
   b. The subsymbolic level was initially designed to reflect the 1970s & 1980s ideas about neural processing.
   c. Guided by rational analysis the subsymbolic level was updated in 1993 to reflected the likelihood that the information was useful. This was the birth of ACT-R.

Evolution from ACT-R 2.0 (1993) to ACT-R 7.x (2018)

1. There have been three driving forces:
   a. The emergence of a user community around the publicly available ACT-R 2.0 (this is when Christian Lebiere's influence began).
   b. The realization that the "No Magic" principle required that we be able to model the processing all the way from input to output.
   c. The insistence on not making assumptions that could not be cashed out into neurally plausible computations.
2. This converged in the modular architecture of ACT-R 6.0 (2005):
   a. The allowed community members to try variations on existing ideas and extensions but keep what they wanted.
   b. We borrowed the modular organization of EPIC for the perceptual-motor modules.
   c. There was growing evidence that, while the brain was a complex parallel machine, different regions had their specializations.
3. **The current ACT-R 7.x (2017) largely reflects software refinements, some of which are significant as Dan Bothell will describe.**

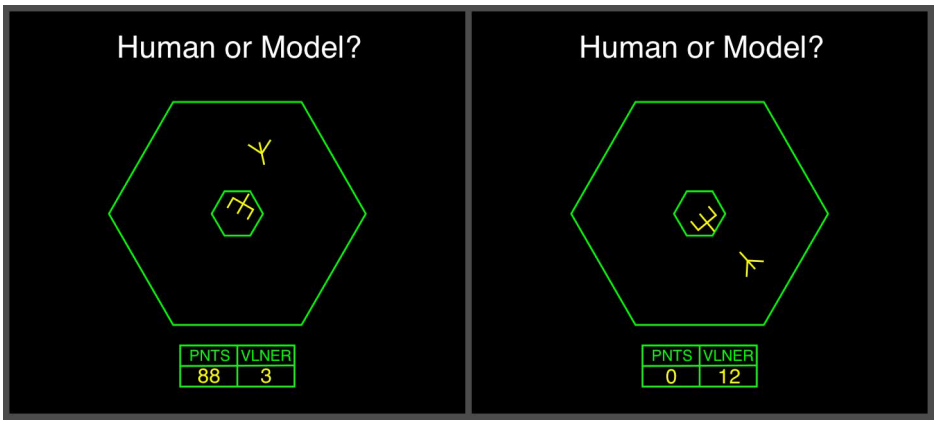
Module Structure of Current ACT-R



Transfer of Cognitive Skills

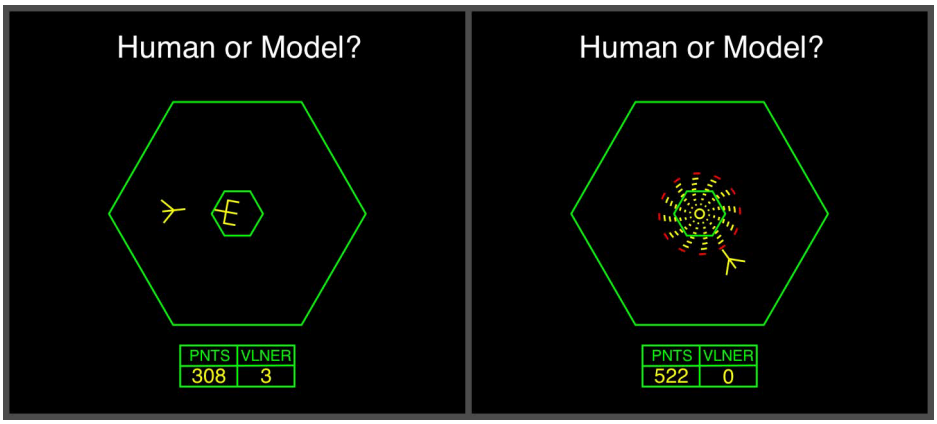
John R. Anderson
Shawn Betts
Daniel Bothell

Space Fortress Game 1



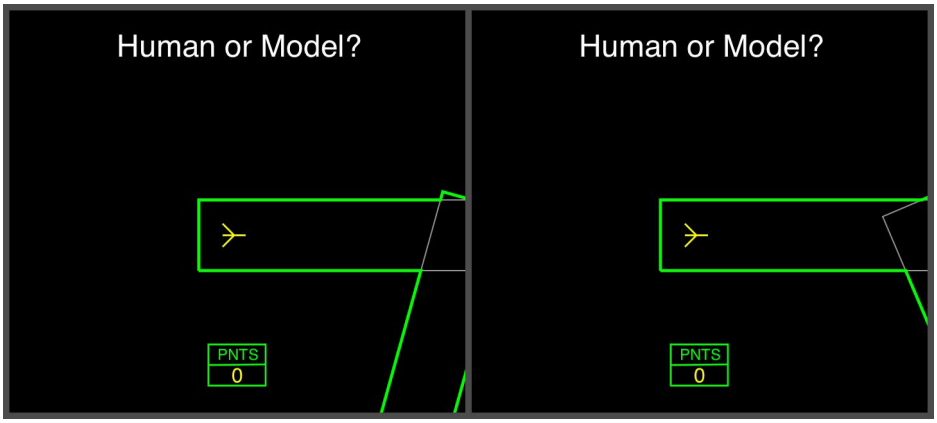
- 1 minute of play at 2x real time
- Navigate a ship between two hexagons, shooting at fortress while avoiding be shot by it.
- **One major challenge is flying in a frictionless space.**

Space Fortress Game 40



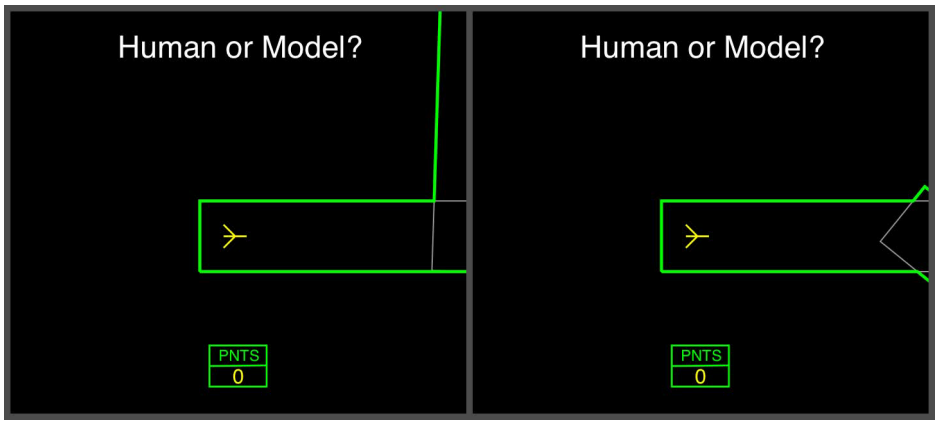
- 1 minute of play at 2x real time
- Navigate a ship between two hexagons, shooting at fortress while avoiding be shot by it.
- **The other challenge is timing shots to be be just above 250 ms.**

Space Track Game 1



- 1 minute of play at 2x real time
- Fly down rectangles as fast has you can without hitting the sides.
- **Again one major challenge is flying in a frictionless space.**

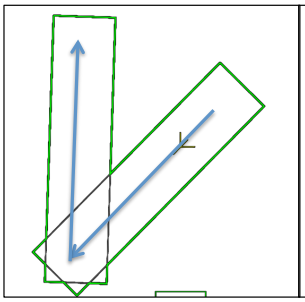
Space Track Game 40



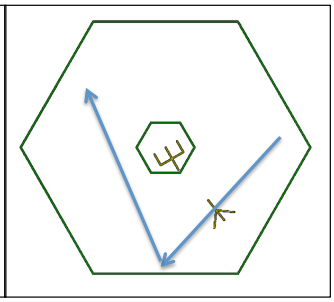
- 1 minute of play at 2x real time
- Fly down rectangles as fast has you can without hitting the sides.
- **Most deaths occur on hard turn at end of the rectangle.**

Can Navigation Skill Transfer between Games?

Space Track



Space Fortress

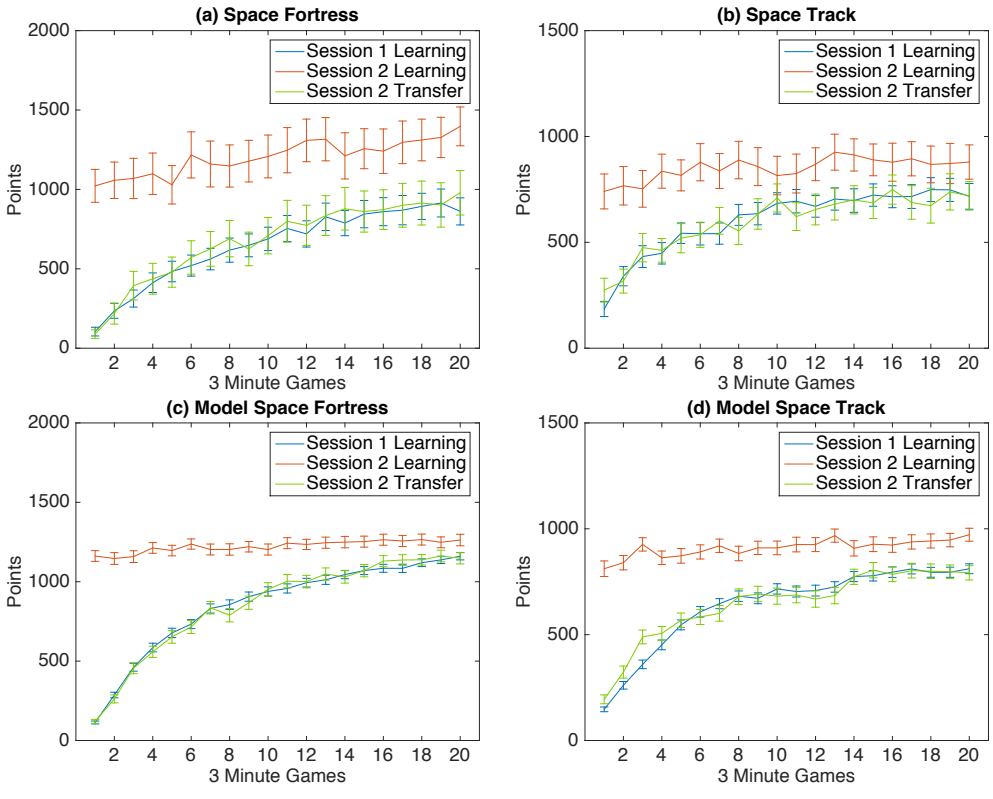


Transfer Design

Session 1 : (20 games)	**Session 2: (20 games)**
Track	Track
Track	Fortress
Fortress	Track
Fortress	Fortress

Hint: Race car drivers do not have better off-track records than normal drivers (Williams & O'Neill, 1974).

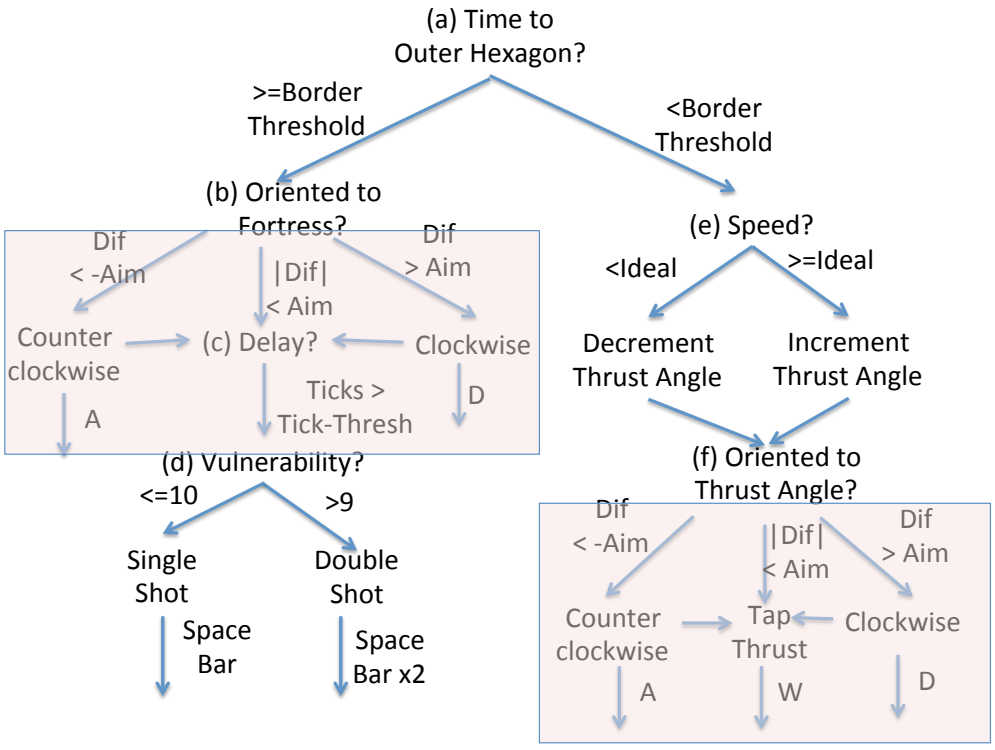
Points



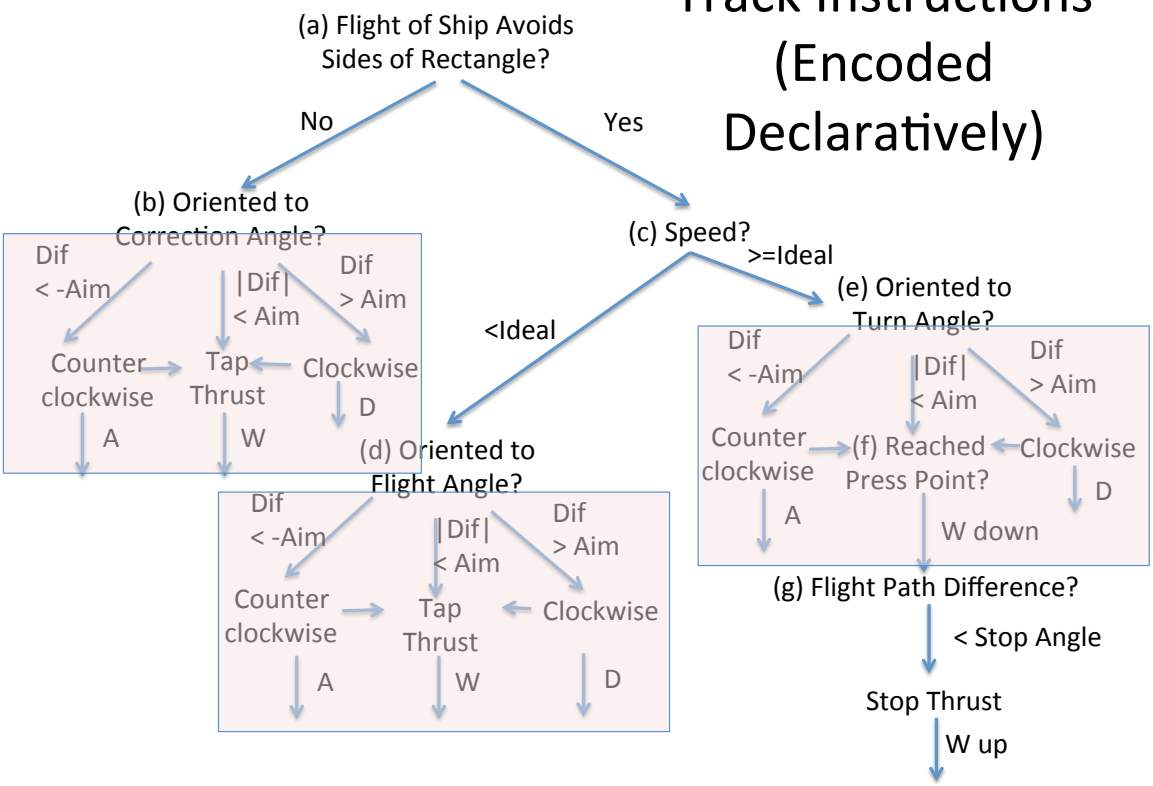
Four Key Features of How ACT-R Models Performance and Learning

1. The model needs to represent realistic performance limitations including perceptual-motor skills and not assume super human response times.
   - ACT-R architecture captures these human limitations.
2. The system needs to deploy task knowledge – without this it would take many more trials to master what participants master in 20 games.
   - ACT-R instruction following can capture this.
3. Participants start out deploying that knowledge slowly but speed up.
   - ACT-R production compilation can capture this.
4. Successful performance requires learning the parameters that define successful actions (e.g. when to thrust, how to pace shots).
   - In such a fast-paced game, there is not enough time for ACT-R to perform the task and monitor these parameters within its cognitive cycle.
   - Therefore, we have developed a new Tracking Module that can be informed by the cognitive cycle but monitors performance off-cycle.

Fortress Instructions (Encoded Declaratively)

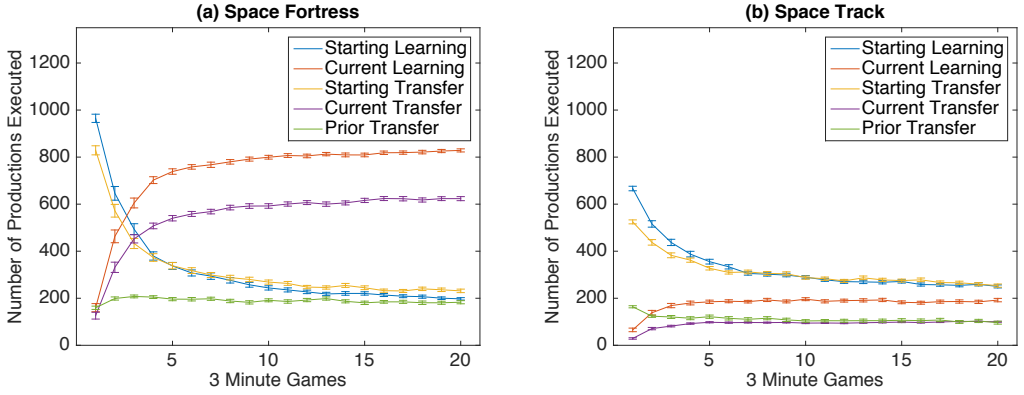


Track Instructions (Encoded Declaratively)



Production Rule Learning

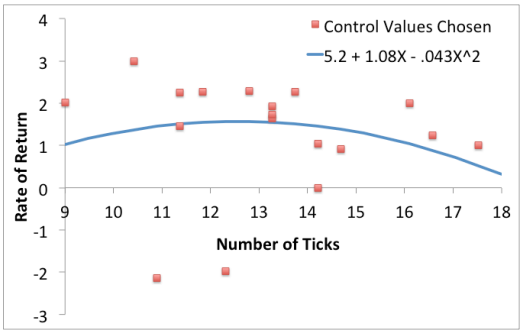
- Initially the model performs the task by retrieving declarative representations of the instructions and using its instruction following production rules to make the tests specified and perform the actions.
- It takes about 500 ms. to retrieve and interpret the instructions leading to an action in the game (the action will add in its own motor time).
- Production compilation combines initial steps and eventually produces direct action rules that respond to game situations with motor requests in 50 ms.
- Since the two games share the same knowledge about navigation some of these learned rules can transfer between games.



The Tracker Module

- Learns values on continuous dimensions to control action such as what delay to place between shots or how fast to fly in Space Track.
- **Informed Learning:** starts with a plausible range for the control value – e.g., it has a range of time ticks that it thinks might correspond to 250 ms.
- Experiments with different settings for random intervals.
- **Informed Learning:** evaluate settings according to relevant dimensions not simply points earned – e.g., increasing vs resetting vulnerability.
- Estimates a quadratic function giving rate of return for control values

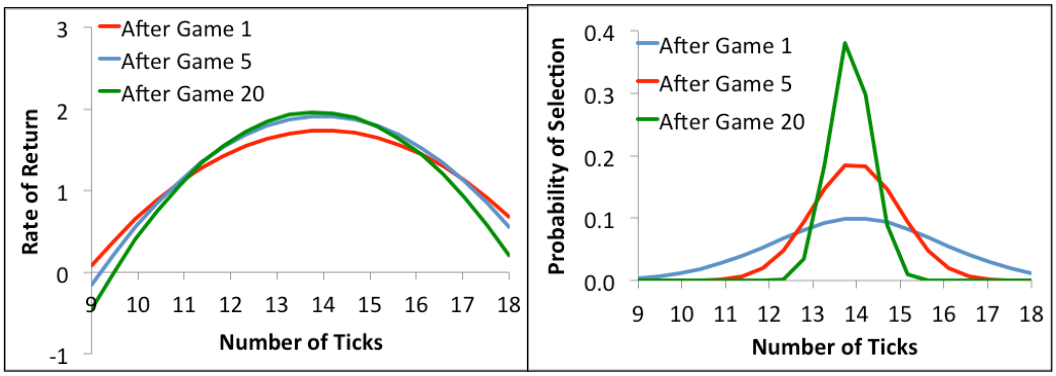
- Function estimated after one 3-minute game.
- Note: Points are weighted by the duration of the interval over which they were sampled.



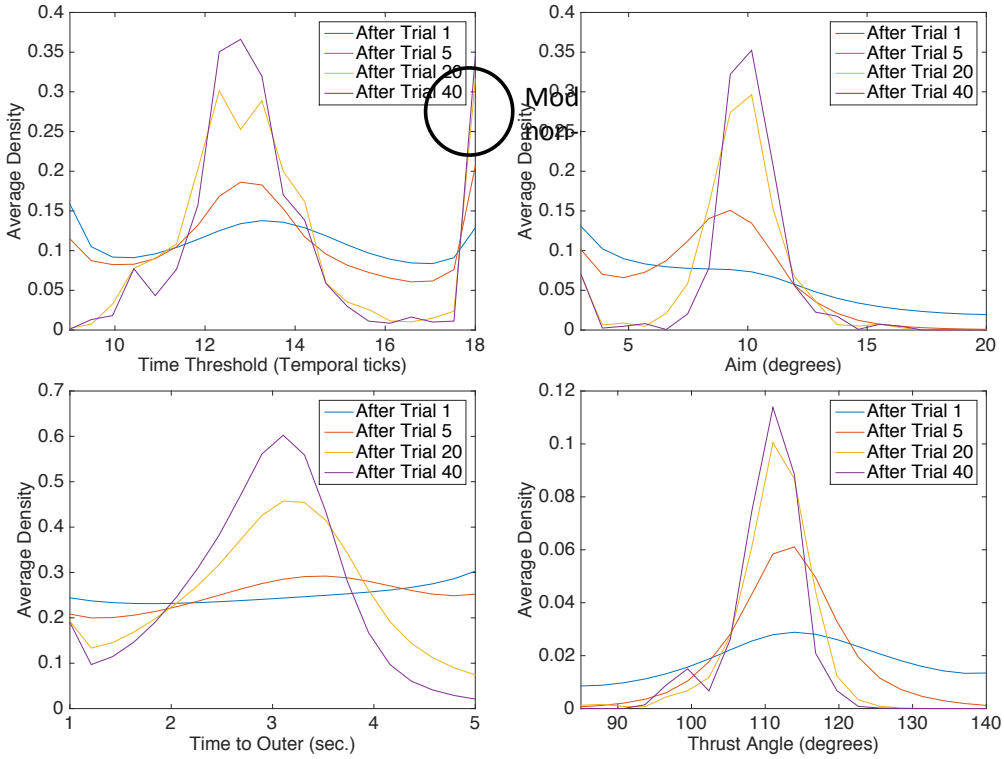
Selecting a New Control Value

- Use a softmax on quadratic function V with a temperature T.
- T=A/(1+B*time) -- A defaults to 1, B defaults to 1/180 s.

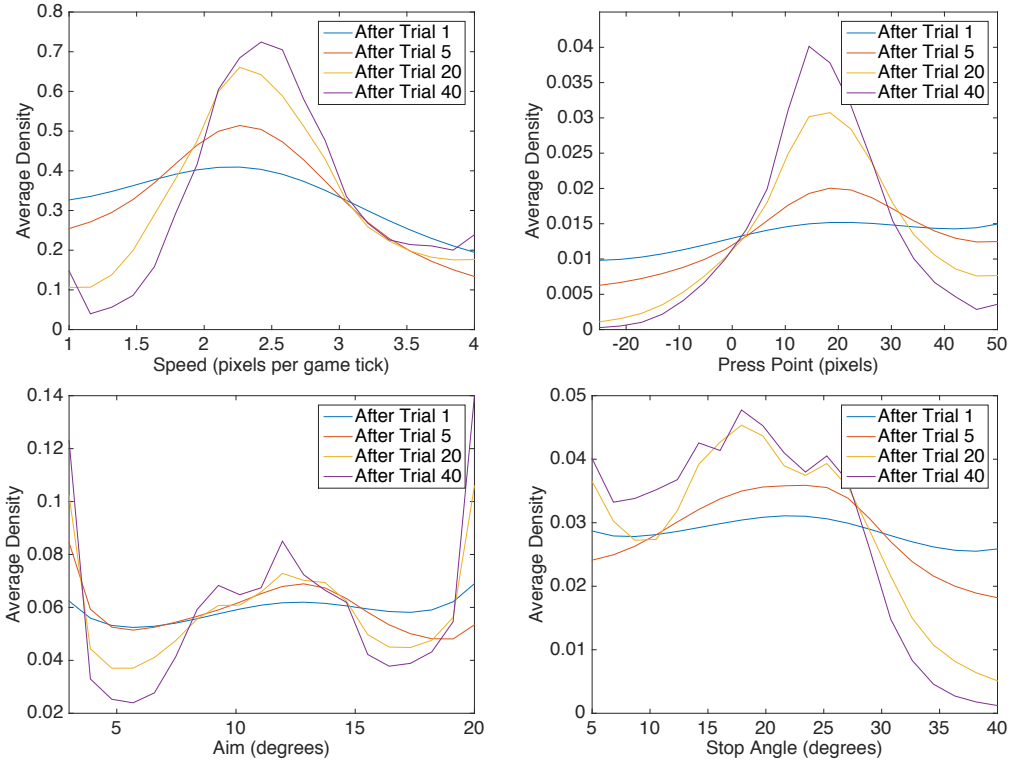
$$P(i) = \frac{e^{V(i)/T}}{\sum_j e^{V(j)/T}}$$



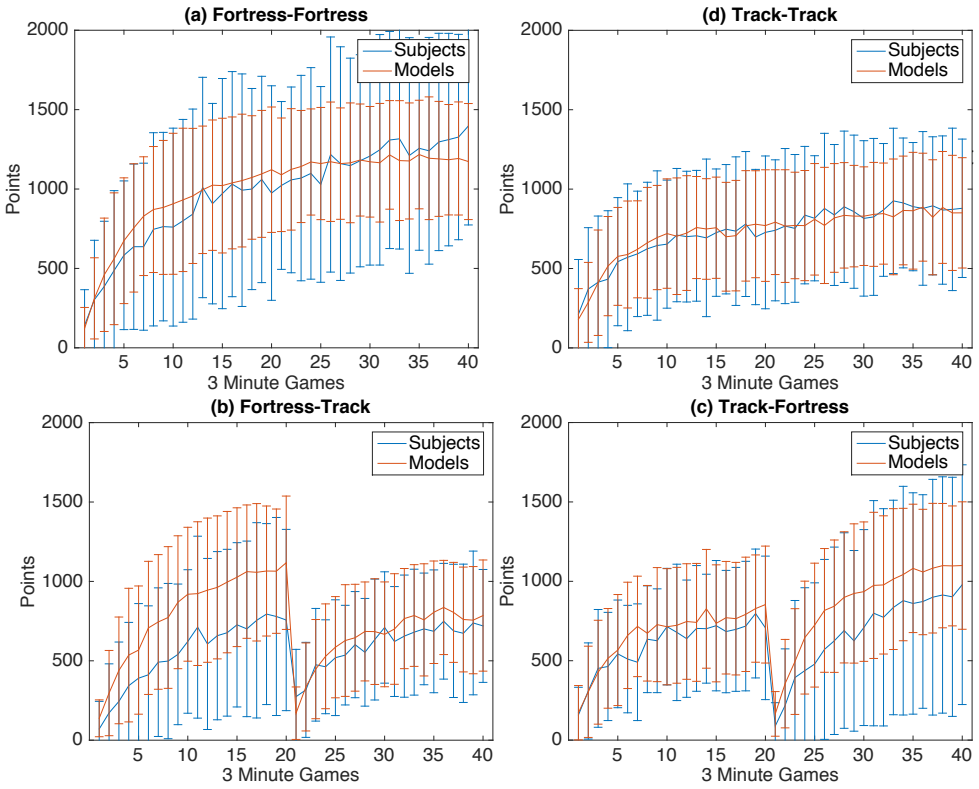
Fortress: Choices Averaged over 96 Models



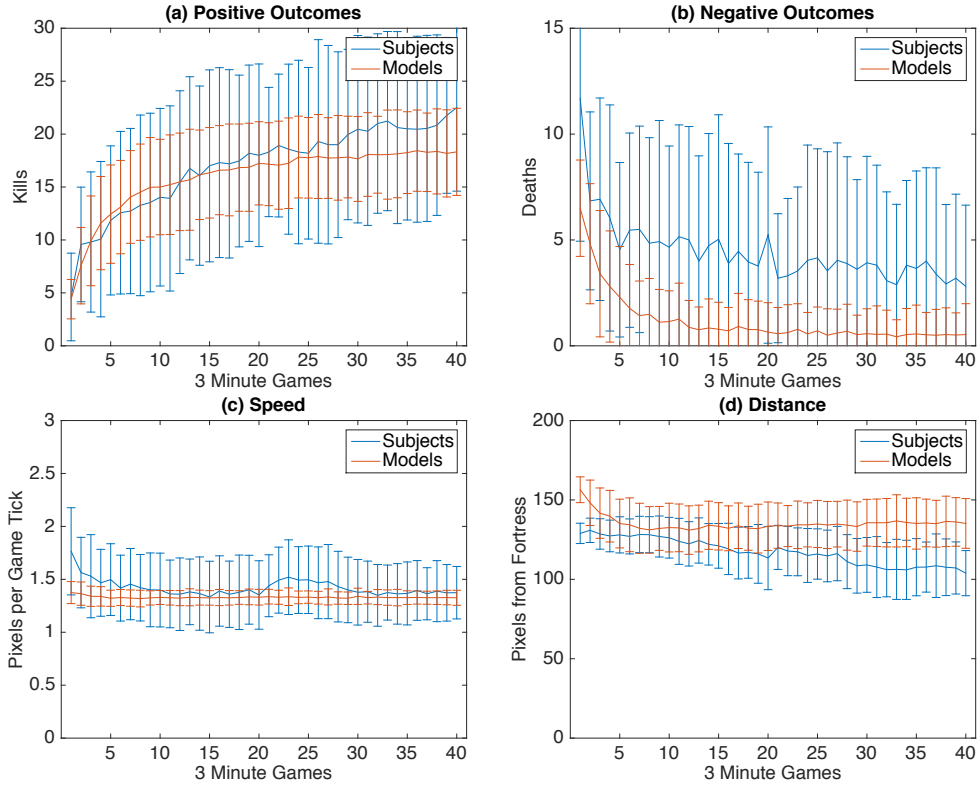
Track: Choices Averaged over 96 Models



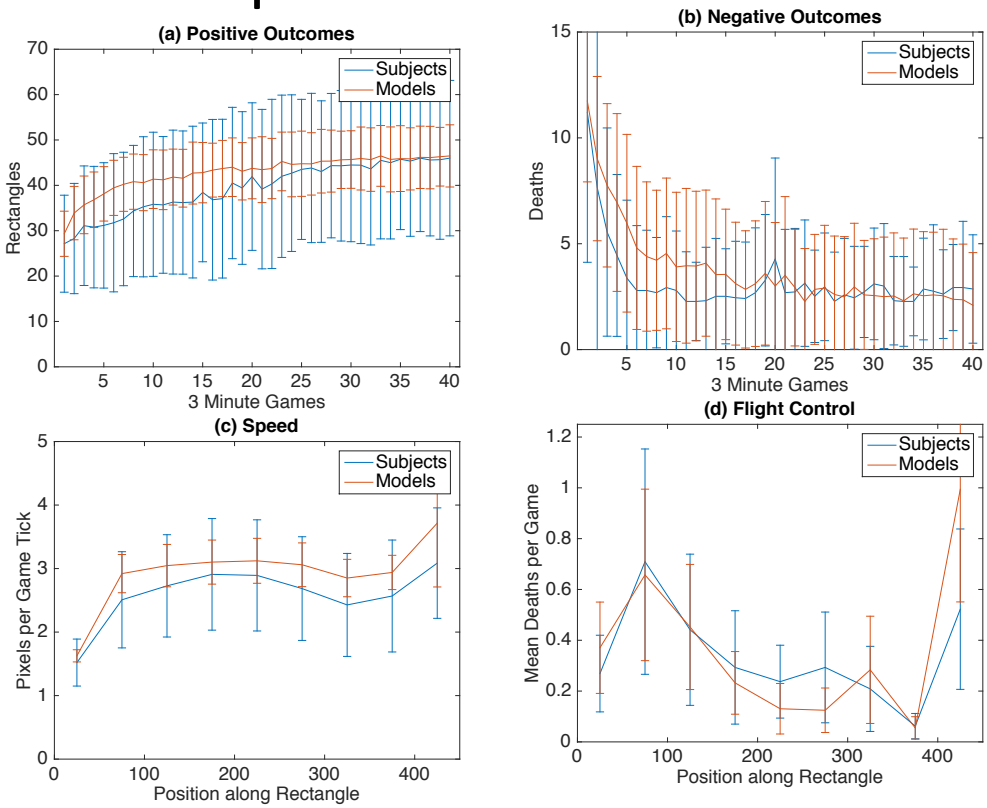
Four Conditions



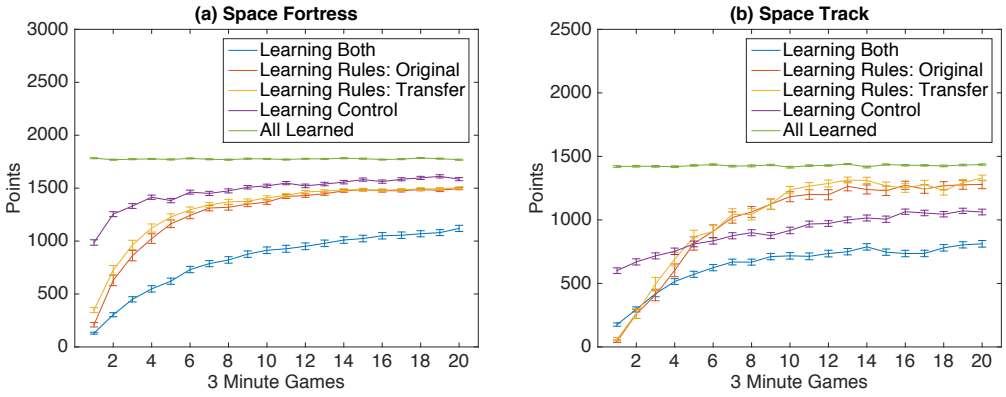
Space Fortress Details



Space Track Details



Contribution of Production Compilation and Control Tuning



- One of the reasons there is not transfer is that instructions give the common declarative knowledge. Results would have been different if subjects had to discover the principles of flight.
- Subjects are learning how to convert that declarative knowledge into high performance:
  - The control knowledge is entirely unique.
  - The shared compiled rules are relatively few (20%).

Conclusion

- These results are not not unique to these two games or to video games.
- They should generalize to any task that requires taking rapid and highly tuned actions -- driving race cars or driving a car on a city street.
- High levels of performance depend on the acquisition of highly specific action rules and control parameters that are unlikely to transfer to other situations.