

Investigation of Utility-Based Alerting

Lee Winder

Jim Kuchar

International Center for Air Transportation

MIT

FAA/NASA Joint University Program

FAA Technical Center

January 9-10, 2003

Alerting System

An alerting system is automation that monitors some human-operated system and issues guidance to operators when needed to avoid an incident.

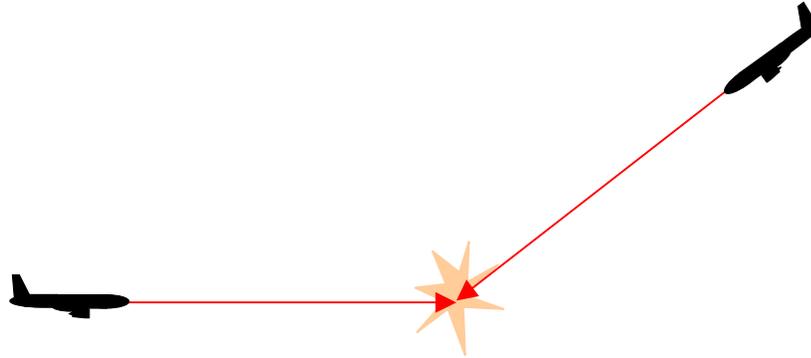
Examples: terrain, traffic, windshear, internal aircraft systems

Have been very successful in preventing accidents, but also occasionally contribute to accidents

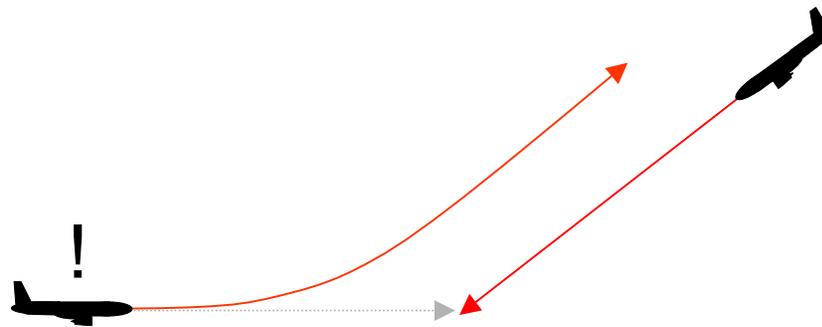
Systems being pushed toward increasing capability and complexity -- current design methods may not be able to keep up
(e.g., TCAS took approx 10 years to develop, another 10 years of tweaking in operation)

Example: Alerting for Aircraft Collision Avoidance

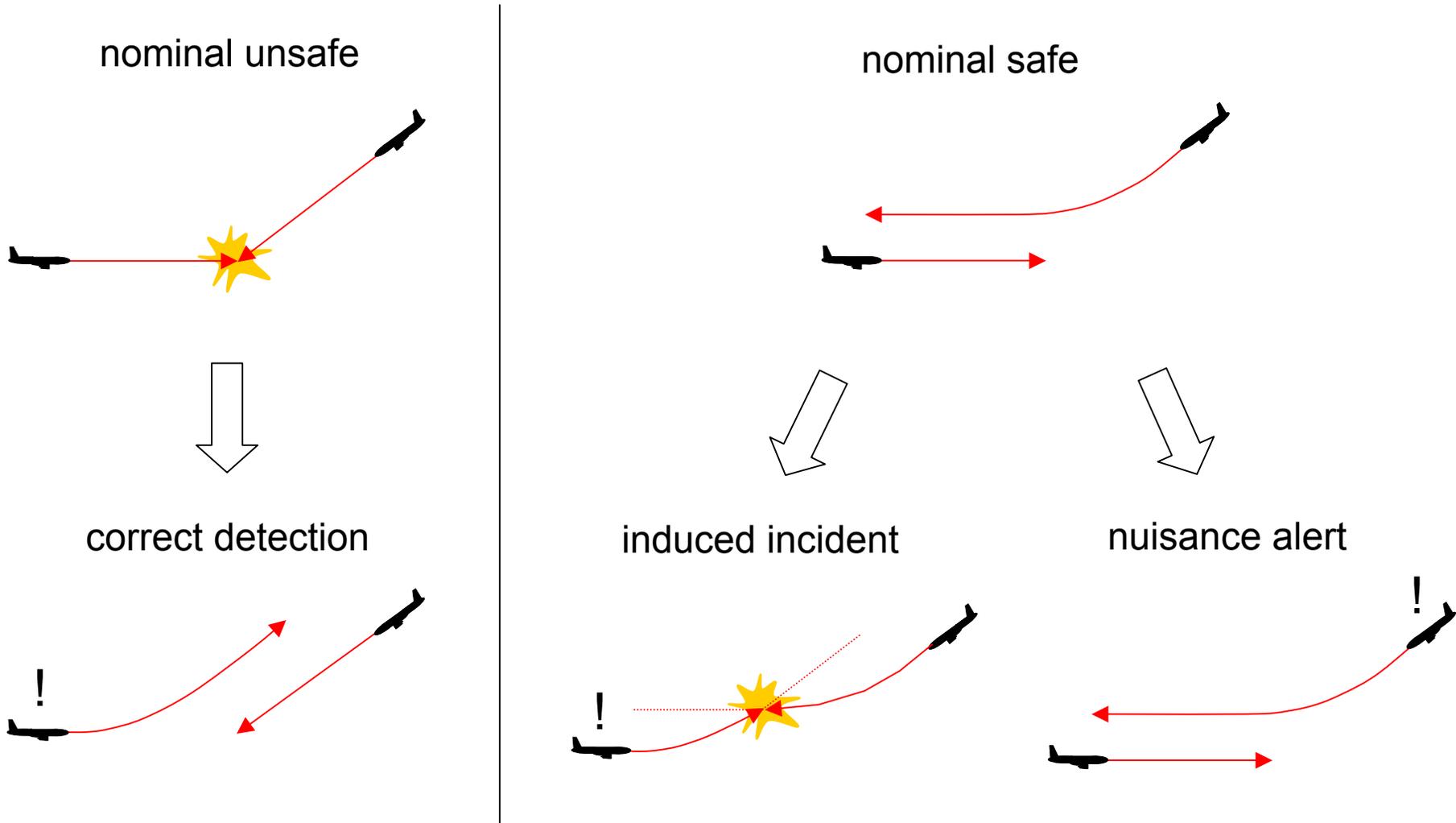
No alert



Alert triggers evasion guidance

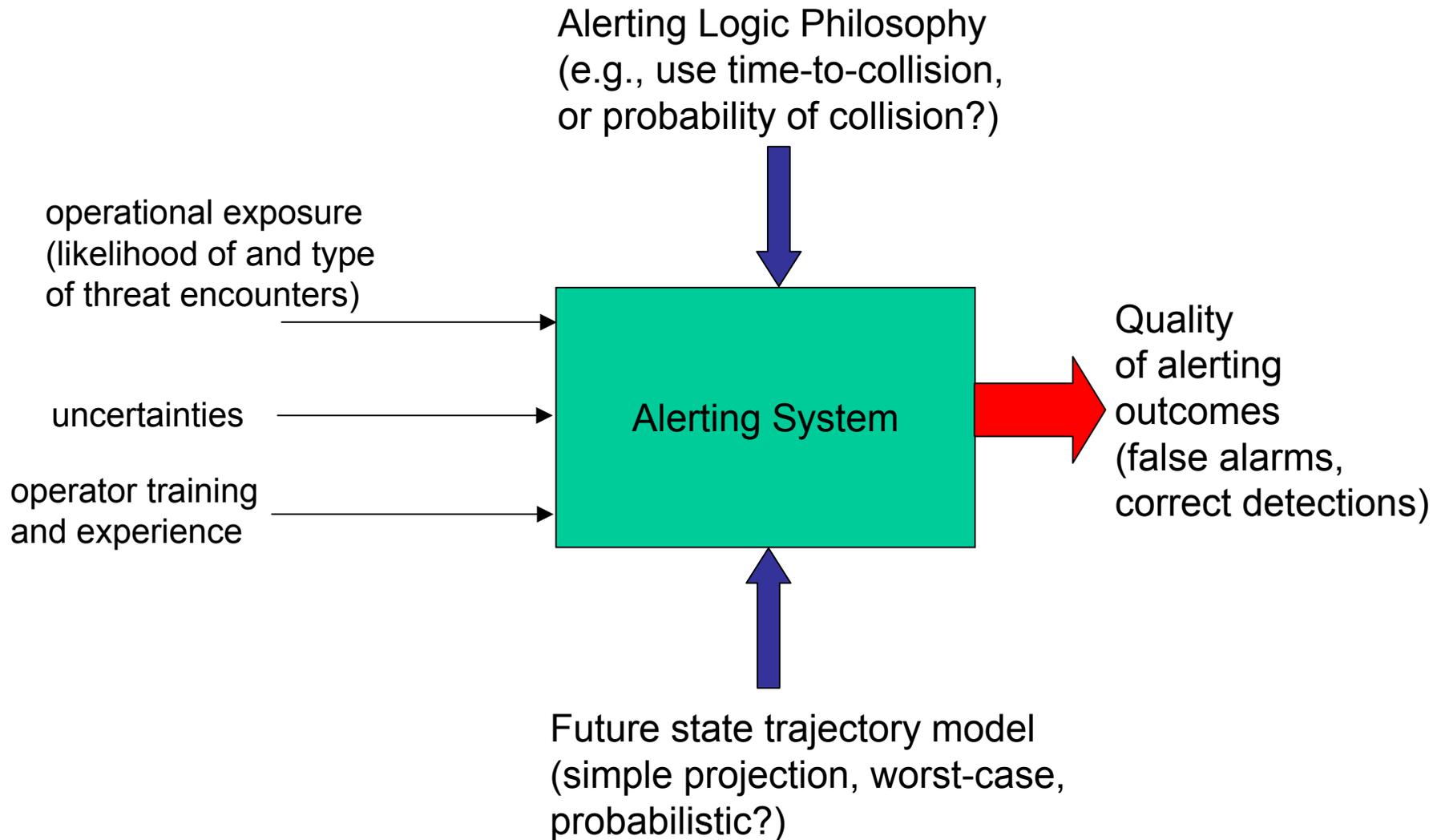


Uncertainty Causes Alerting Failures



Alerting system design requires trade-off between good and bad outcomes

Drivers of Alerting System Performance



Research Objectives

We currently only have a qualitative understanding of how the inputs to alerting system design affect alerting quality

Goal: generate quantitative understanding of alerting system behavior so that system performance can be improved

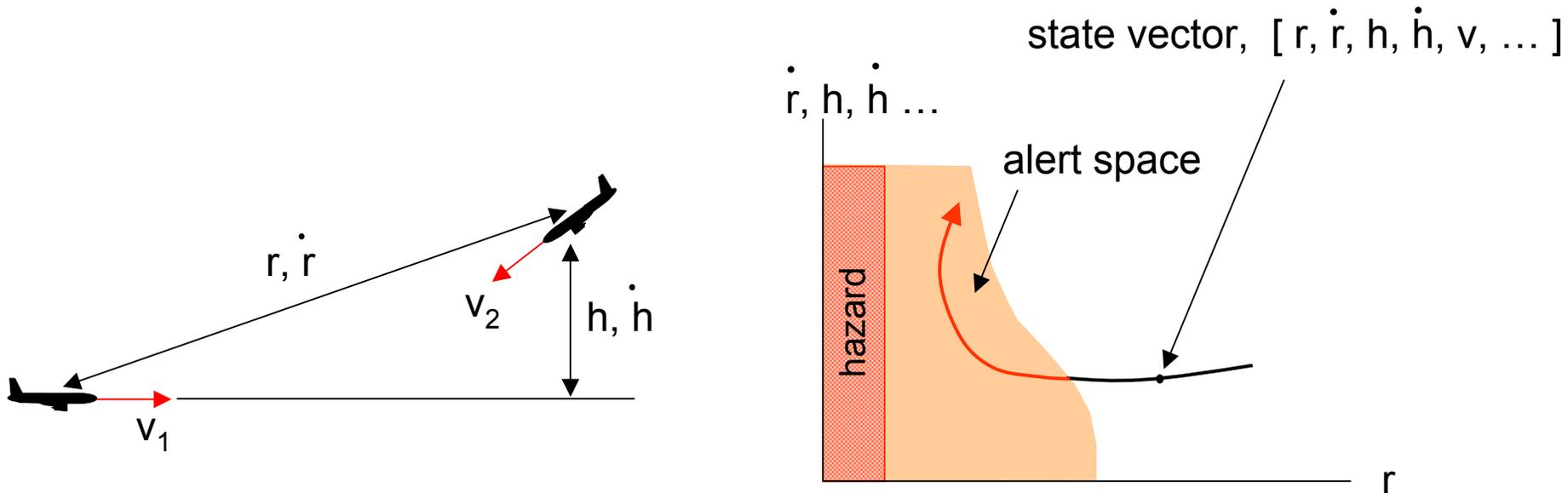
- Extend prior work on modeling alerting system behavior
- Develop principled methods for design of alerting systems
 - New methods for selecting alerting thresholds
 - More accurate models of future trajectories
 - Articulate benefit of different design methods as function of the problem being addressed (may be better to use approach A in high-certainty situation but approach B in low-certainty situation)

Traditional Alerting Threshold Design

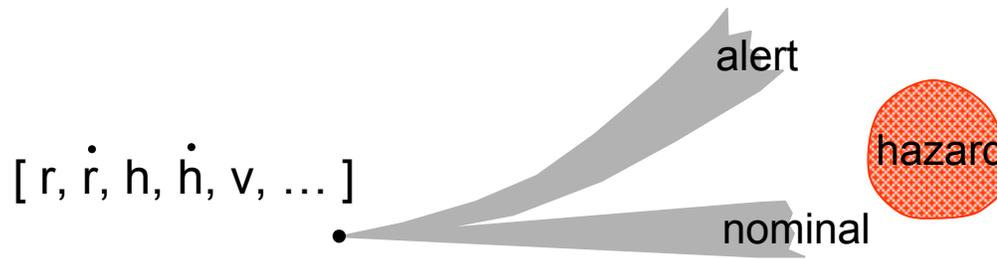
Traditional methodology: Build threshold directly in physical state space (used in every current on-board alerting system, most other alerting systems)

Iterative trajectory simulations aid choosing and tuning threshold parameters for best performance

Simple, but may not really capture the actual design tradeoffs



Probabilistic Alert Analysis

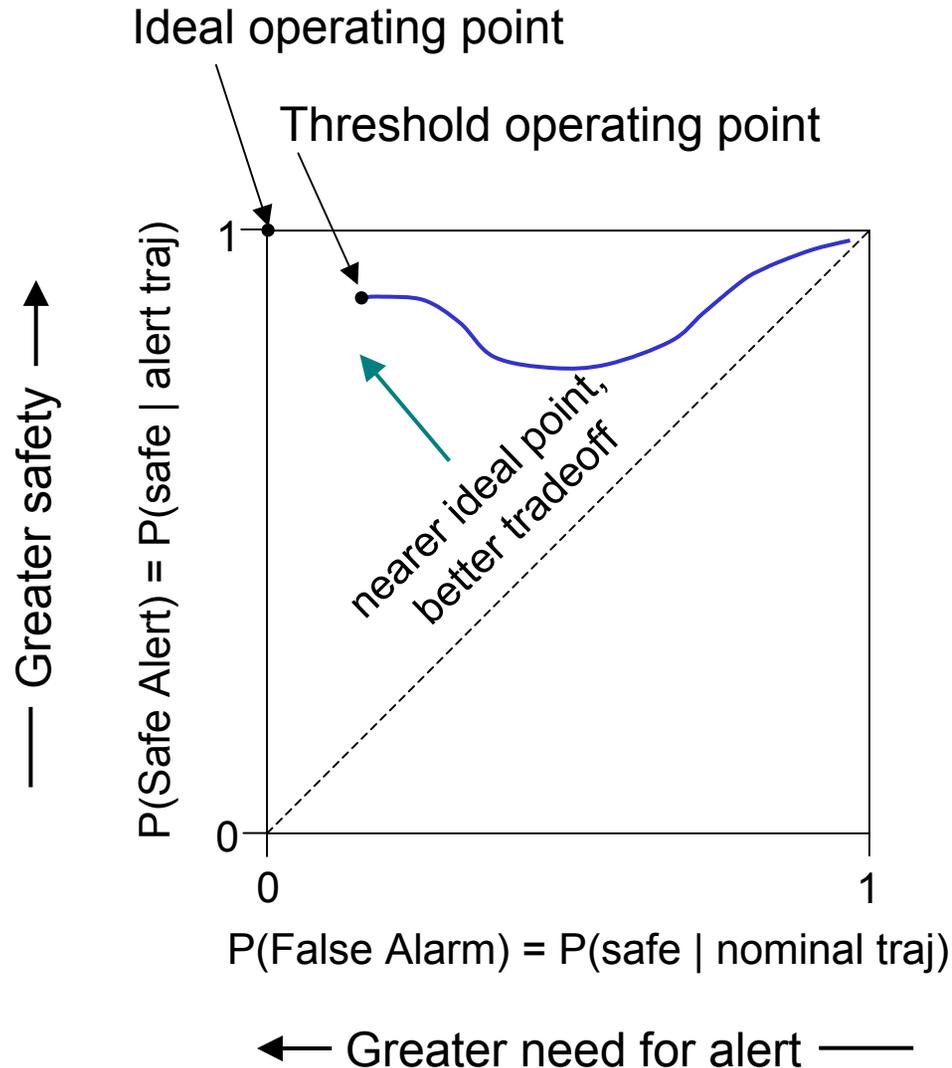


<u>State vector</u>		<u>Probabilistic metrics</u>	
$[r, \dot{r}, h, \dot{h}, v, \dots]$	⇒	$P(\text{False Alarm} \mid \text{nominal traj})$	Measure of need to act
		$P(\text{Safe Alert} \mid \text{alert traj})$	Measure of alert safety

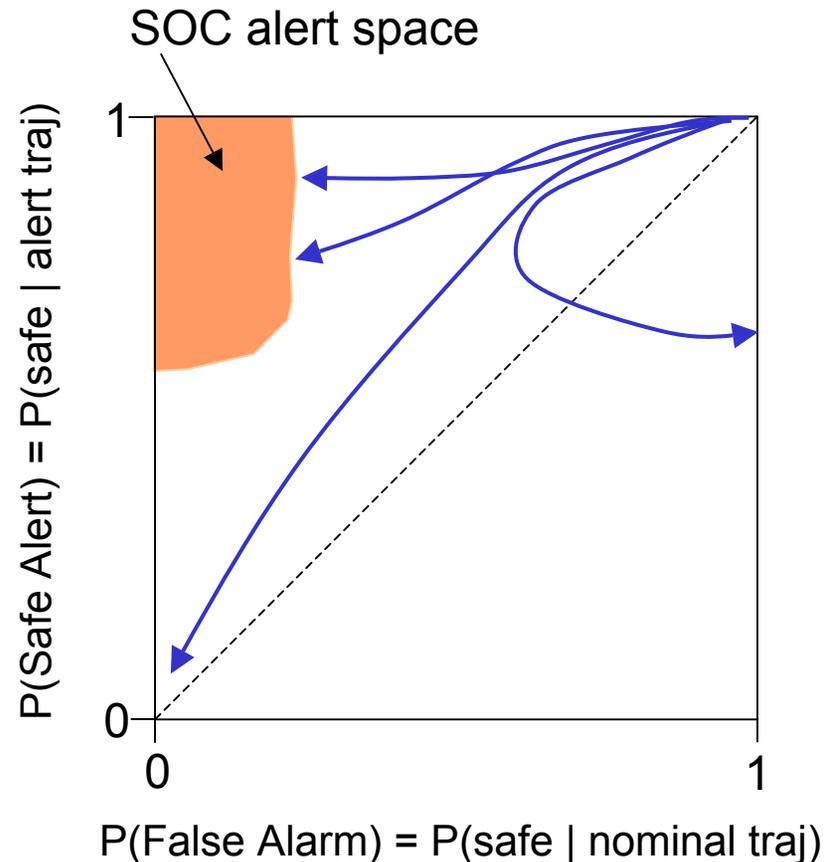
Idea: $P(\text{SA})$, $P(\text{FA})$ represent the fundamental performance types to trade off

Beginning to be introduced, e.g., CTAS and URET conflict probes

Probabilistic Alert Analysis ("System Operating Characteristic")



Threshold Design in SOC Design Space



Achieve desired performance trade-off by defining threshold in SOC space

Unresolved issue: just how should the threshold be defined in SOC space?

Utility-based Alerting Philosophy

- Possible decision metrics
 - Physical metrics (time, distance...)
 - SOC metrics
 - Utility
- Previous research (e.g. Pritchett): “nuisance alerts” and nonconformance indicate operator/pilot has a preferred internal evasion strategy for the hazard.
- Utility decision theory: A “rational” agent acts to maximize its expected outcome utility.
- Notion: Design alerting system as a rational utility-based agent.
 1. Assign numerical utility to different trajectory outcomes.
 2. At each moment compute expected utility for different alert options (alert now, defer alert).
 3. Defer alerting until it has greater expected utility than not alerting.

Experimental Testbed

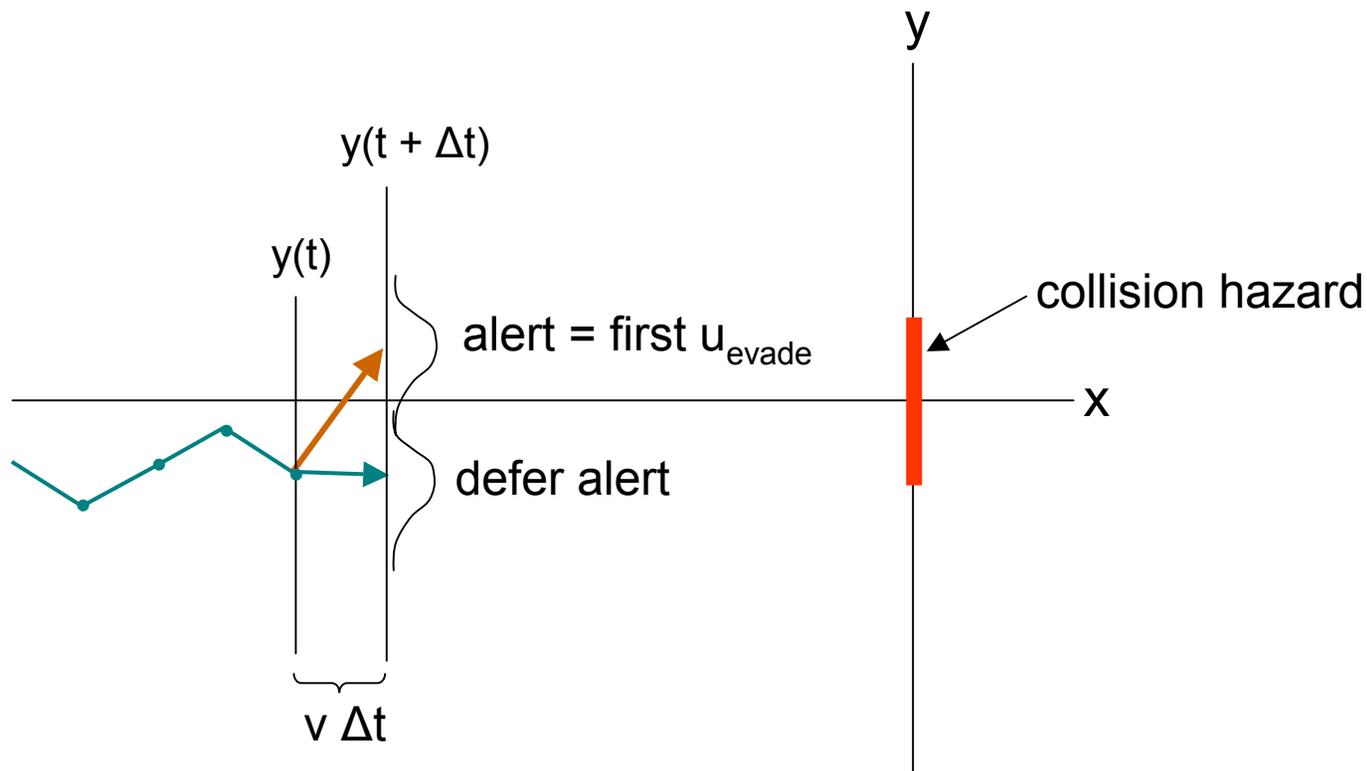
$$x(t + \Delta t) = v \Delta t$$

w = Gaussian white sequence

$$y(t + \Delta t) = y(t) + u + w$$

u = Alert input = $\begin{cases} 0, & \text{nominal (defer alert)} \\ u_{\text{evade}}, & \text{constant bias} \end{cases}$

Any input sequence allowed



Experimental System Outcome Utilities

Outcome utility definitions

Safe outcome and no alert occurred

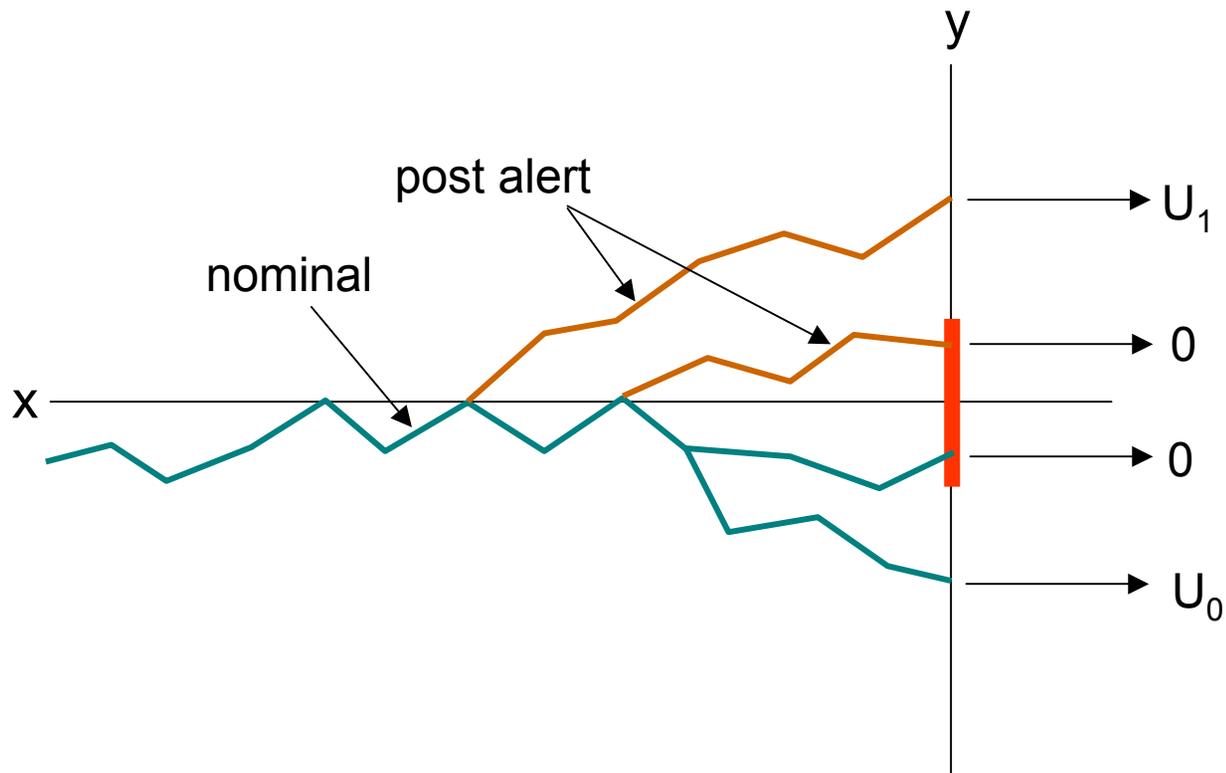
Safe outcome and alert occurred

Collision with or without alert

$+U_0$

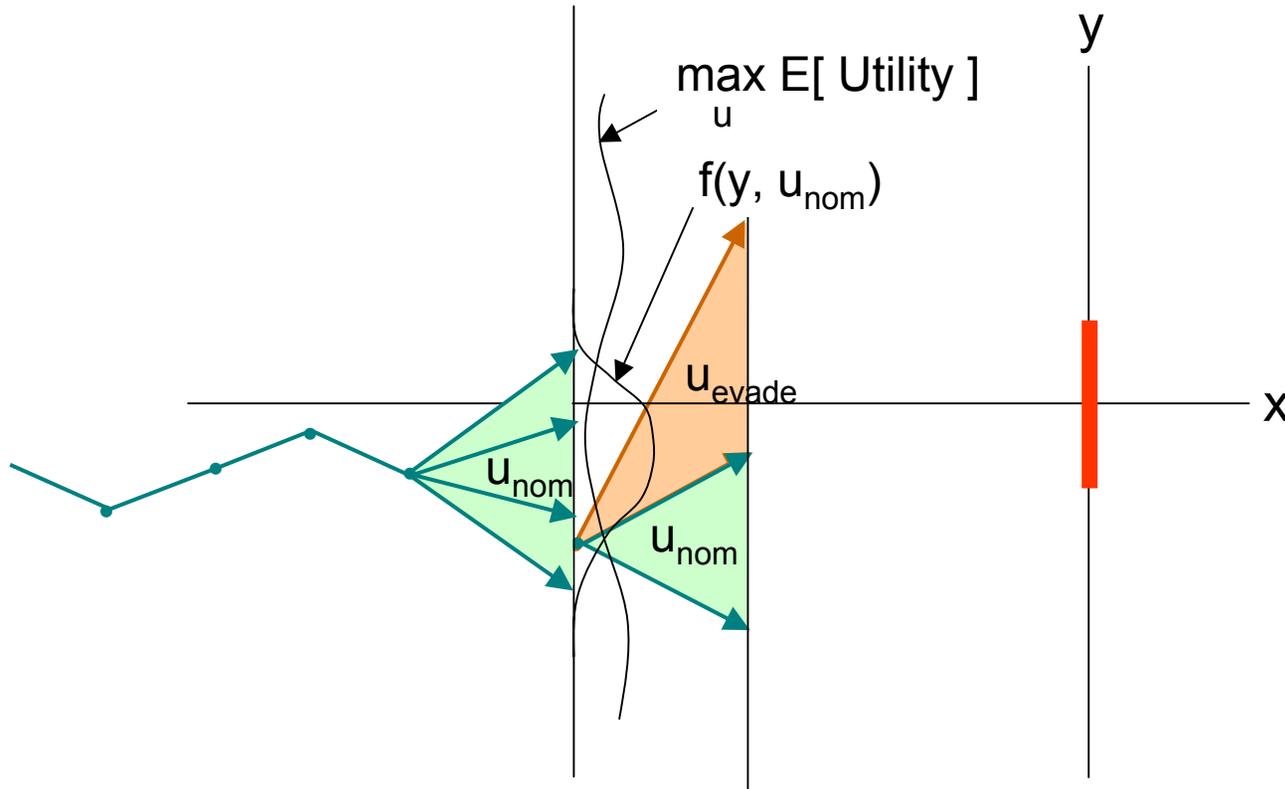
$+U_1 < U_0$

0



Utility Computation

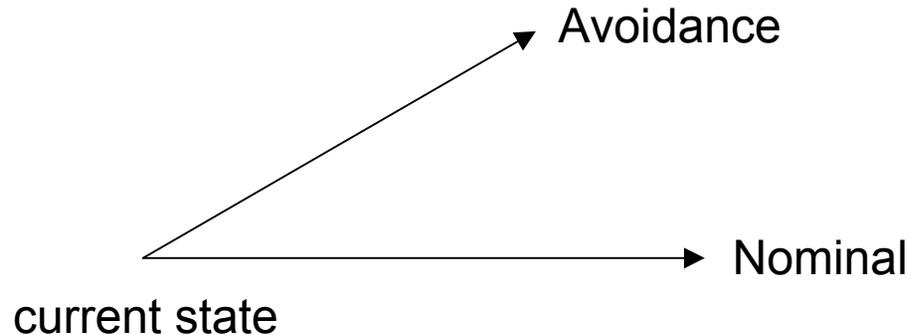
$$E[\text{Utility} | u_{\text{nom}}] = \int \max_u E[\text{Utility}] f(y, u_{\text{nom}}) dy$$



A recursive algorithm can yield an approximate $E[\text{Utility}]$ for each point in time for each input option (alert or defer).

For this simple system, can also propagate $E[\text{Utility}]$ backward from the end states.

Trajectory Modeling

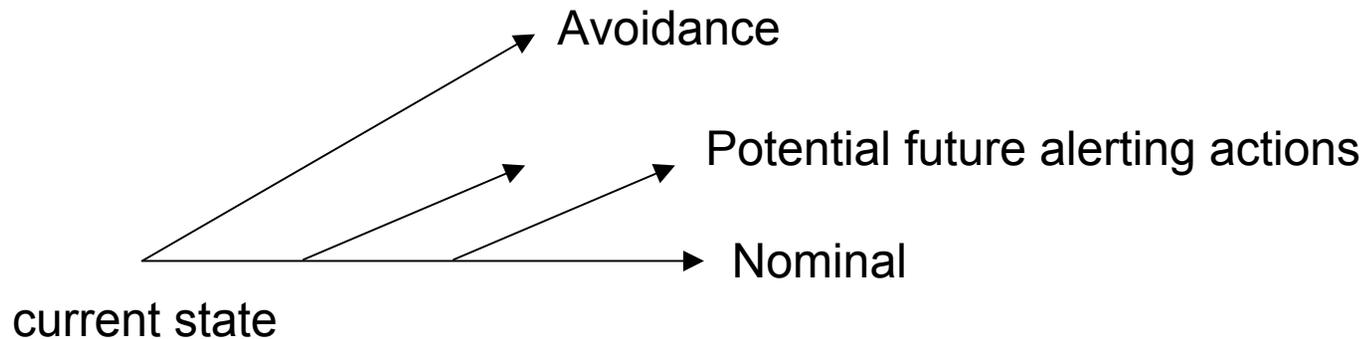


In traditional trajectory models (in fact, in all models in current use) the future state trajectory is propagated assuming a single control behavior is used (e.g., if no alert is issued, state will continue along Nominal trajectory forever).

Simple, but the safety along the Nominal branch is actually probably higher than predicted because we could divert from that branch later.

Trajectory Modeling

More accurate modeling is possible by including potential for future alerts:



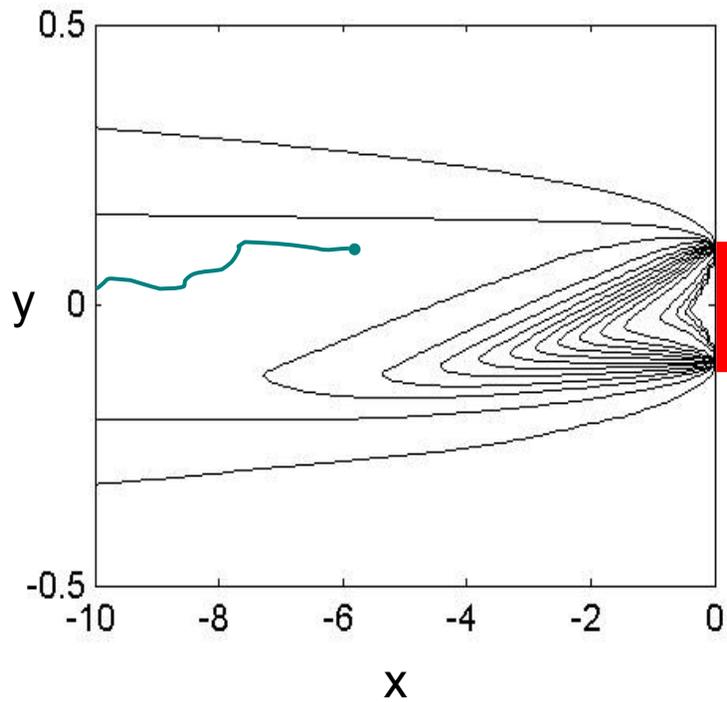
This provides a more accurate estimate of safety (or utility) along each branch.

This type of multiple-branch model was implemented in the experimental testbed.

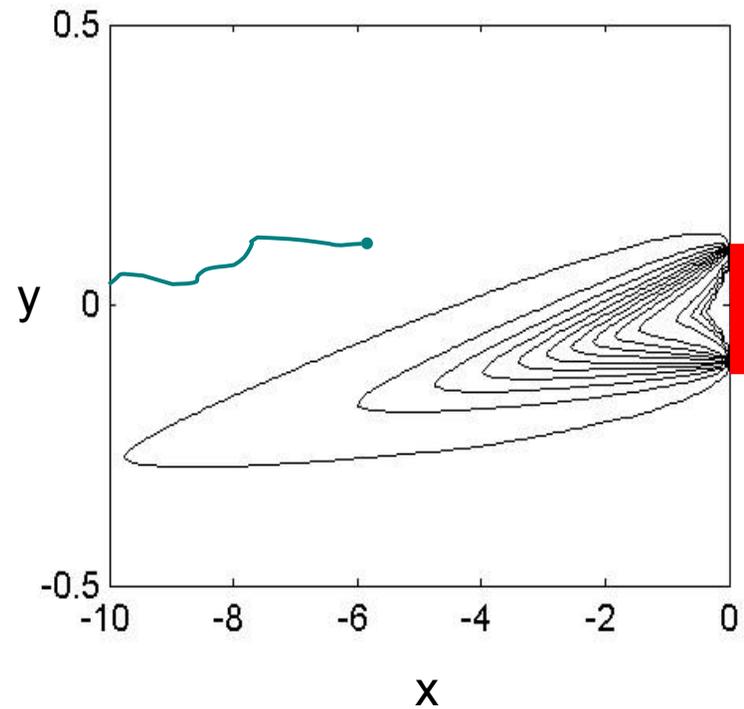
This modeling comes at a cost of complexity -- the tradeoff between complexity and performance is one aspect we will be investigating further.

Expected Utility for Different Actions

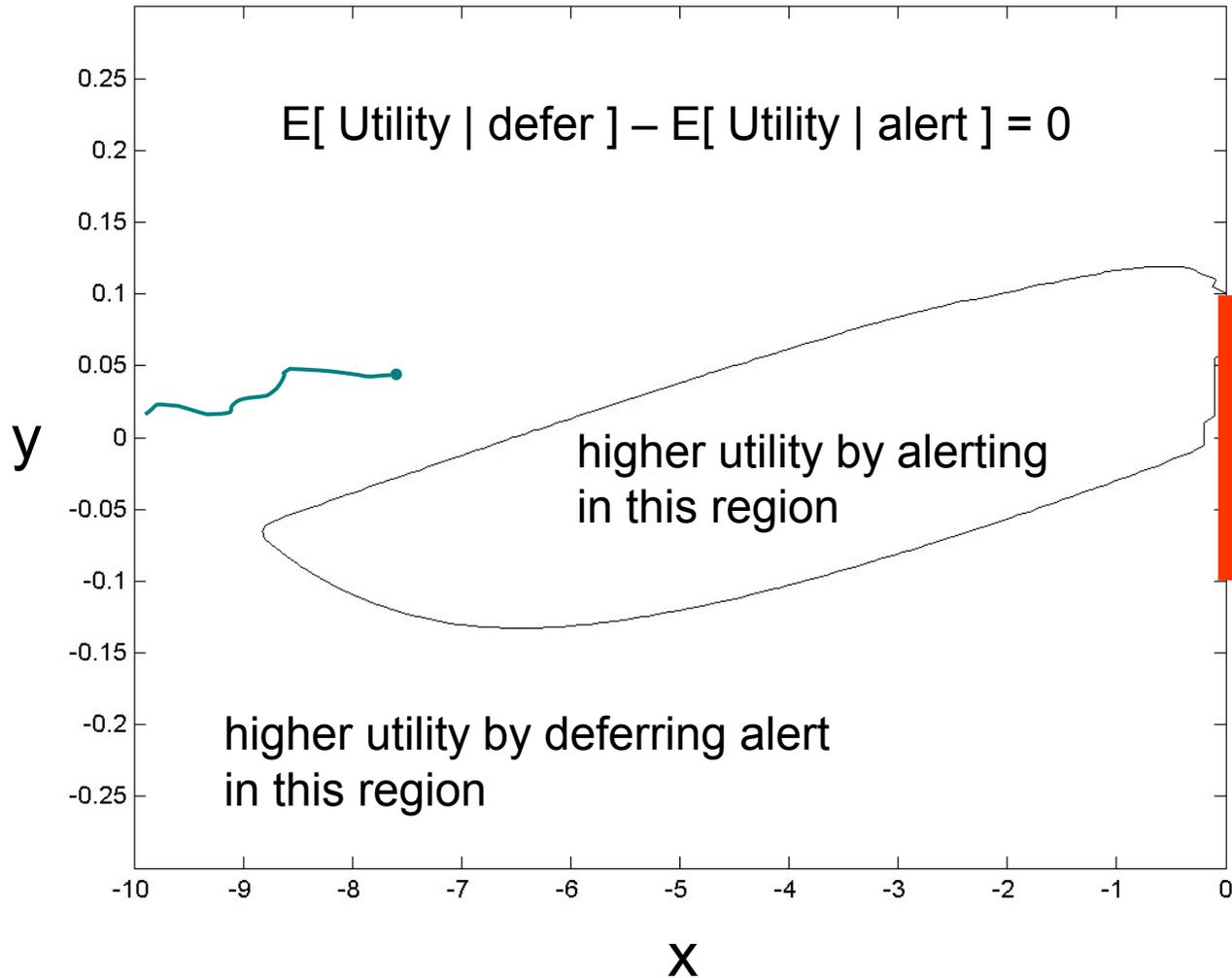
E[Utility] Nominal Action
(Deferred Alert)



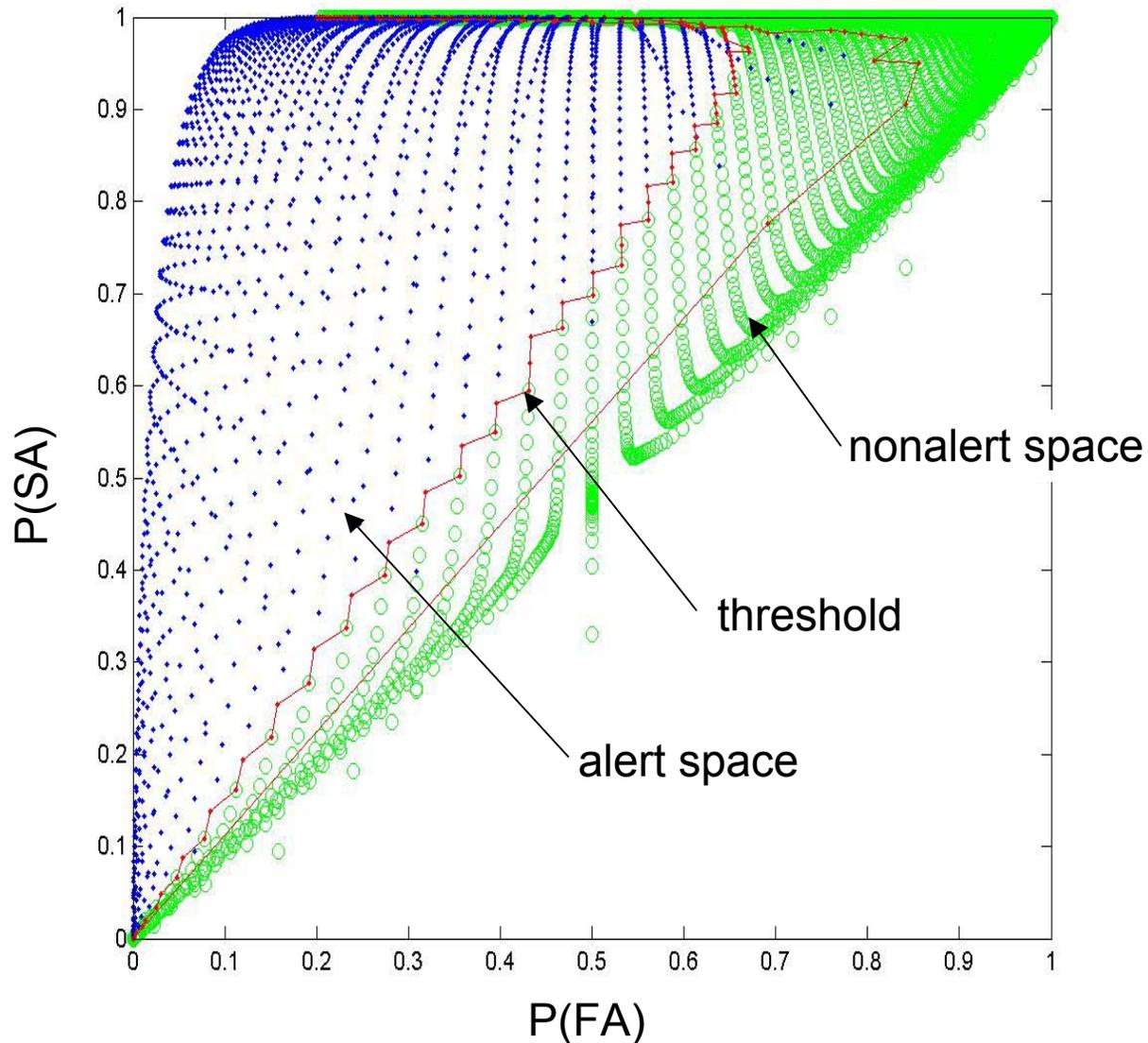
E[Utility] Alert Action



Utility-based Alert Threshold



Mapping of Utility-based Alert Regions into SOC Space

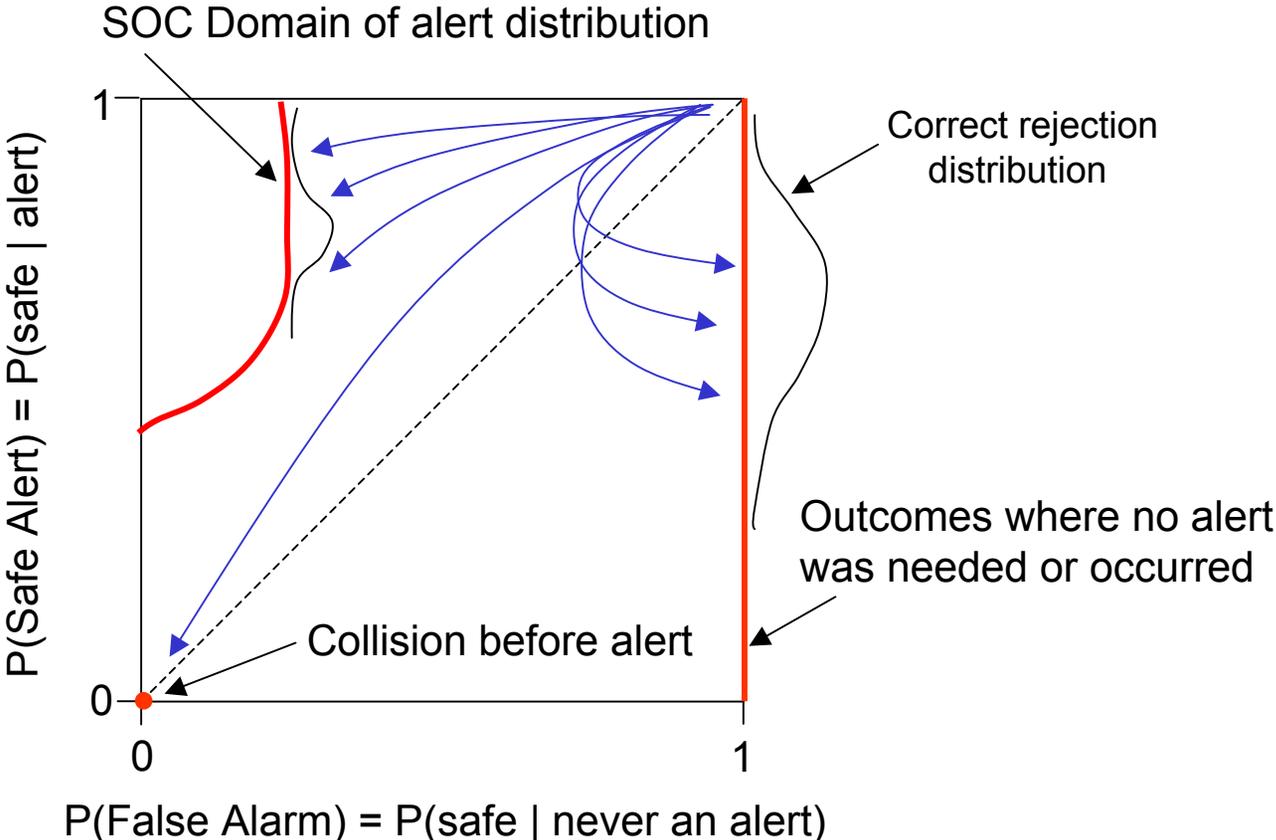


Next Steps

- Use experimental testbed to explore alerting threshold design methods
 - Overall expected utility of the system as function of
 - Different threshold design methods
 - Different utility assignments
 - Different modeling complexity (single-branch vs. multiple branch)
 - Effect of varying process uncertainty on performance
- Ultimate goal: articulate the relationships between alerting system inputs and system quality (ref earlier figure)

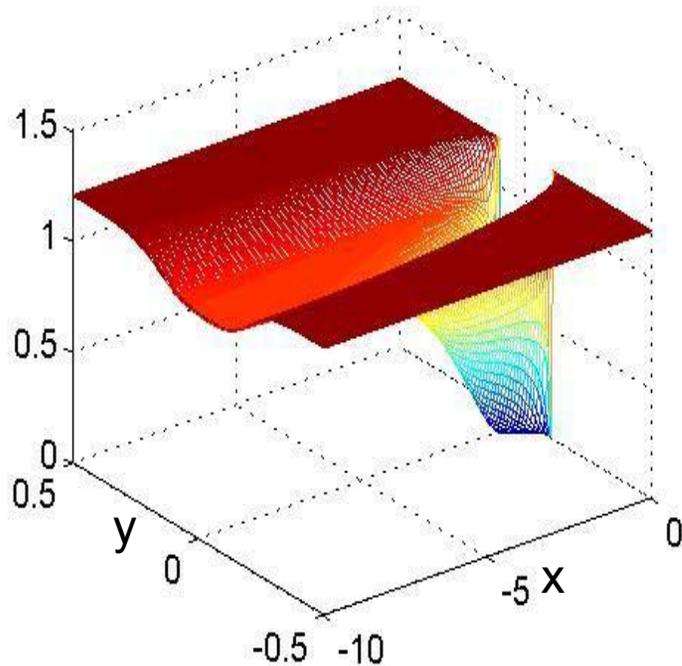
The End

SOC Distribution of Alert Outcomes

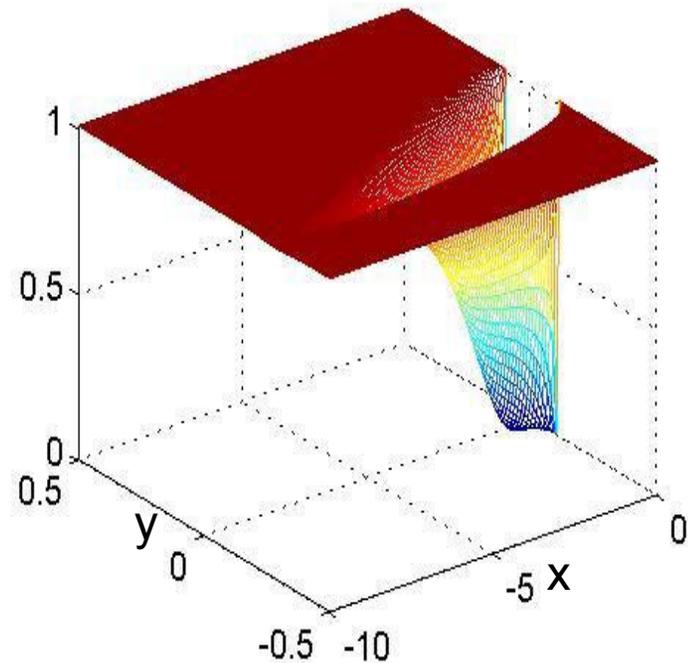


Expected Utility Surfaces

E[Utility] Nominal Action
(Deferred Alert)



E[Utility] Alert Action



these would be better as simple 2-D contour plots I think, but you probably don't have time to do that now. Just be sure to walk people through them -- where is the hazard, which way does the state move through the plot, why is it shaped the way it is...