THE UNIVERSITY OF ARIZONA® | Arizona's First University.

When can a UAV get smart with its operator, and say 'NO!'?

Jerry Ding**, Jonathan Sprinkle*,
Claire J. Tomlin**, S. Shankar Sastry**

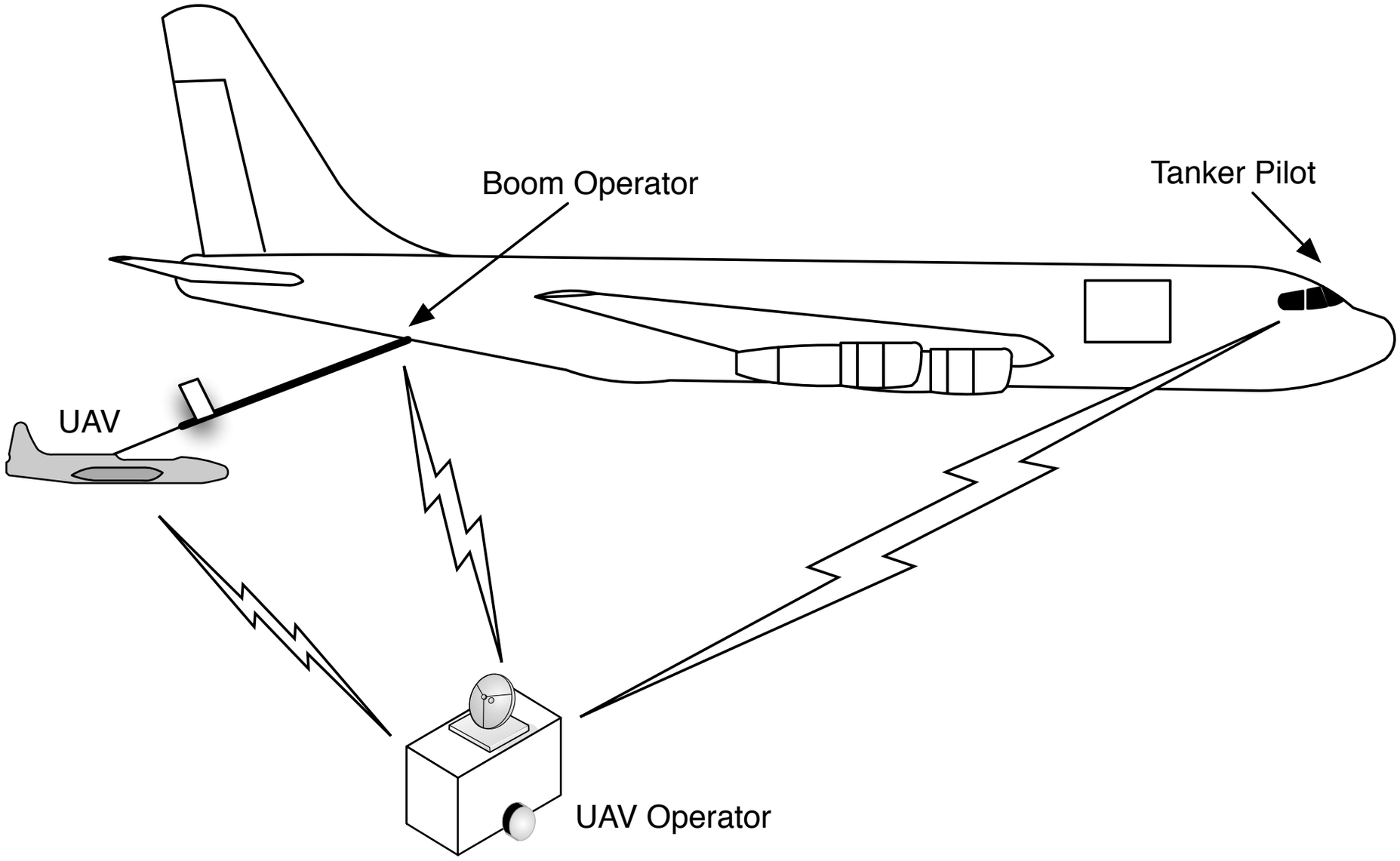
* University of Arizona, Tucson
** University of California, Berkeley

This work is supported in part by the "Certification Technologies for Flight Critical Systems (CerTA FCS)" project, through a subcontract with Boeing Phantom Works"

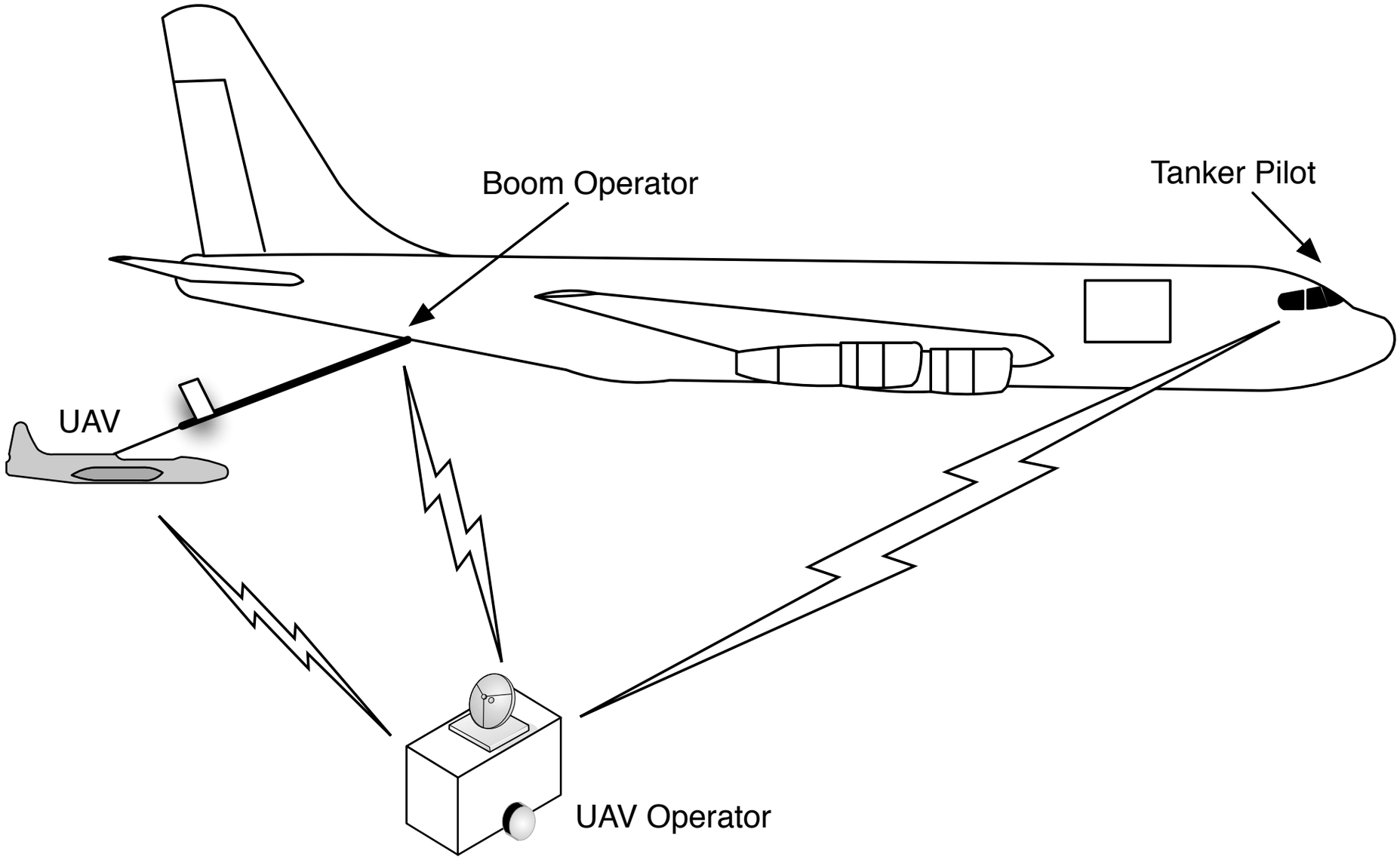
Free *hoc*!!*†

* While supplies last, see store for details.
† No interest, guaranteed!

- Historical complaints about embedded systems design
  - ad hoc
  - Post-design analysis (not formative analysis)
  - Simulation key part of acceptance
  - Unknown latency requirements
- Design strategy for mixed-initiative avionics software:
  - Simulation/human testing key part of acceptance
  - Large human factors impact
  - Best-practices influence future designs



- Questions. What happens:
  - in the event of high latency (human or network)?
  - when directed specifications are incorrectly entered?
  - when an event arrives after a state change?
  - when the vehicle is entering an unsafe state



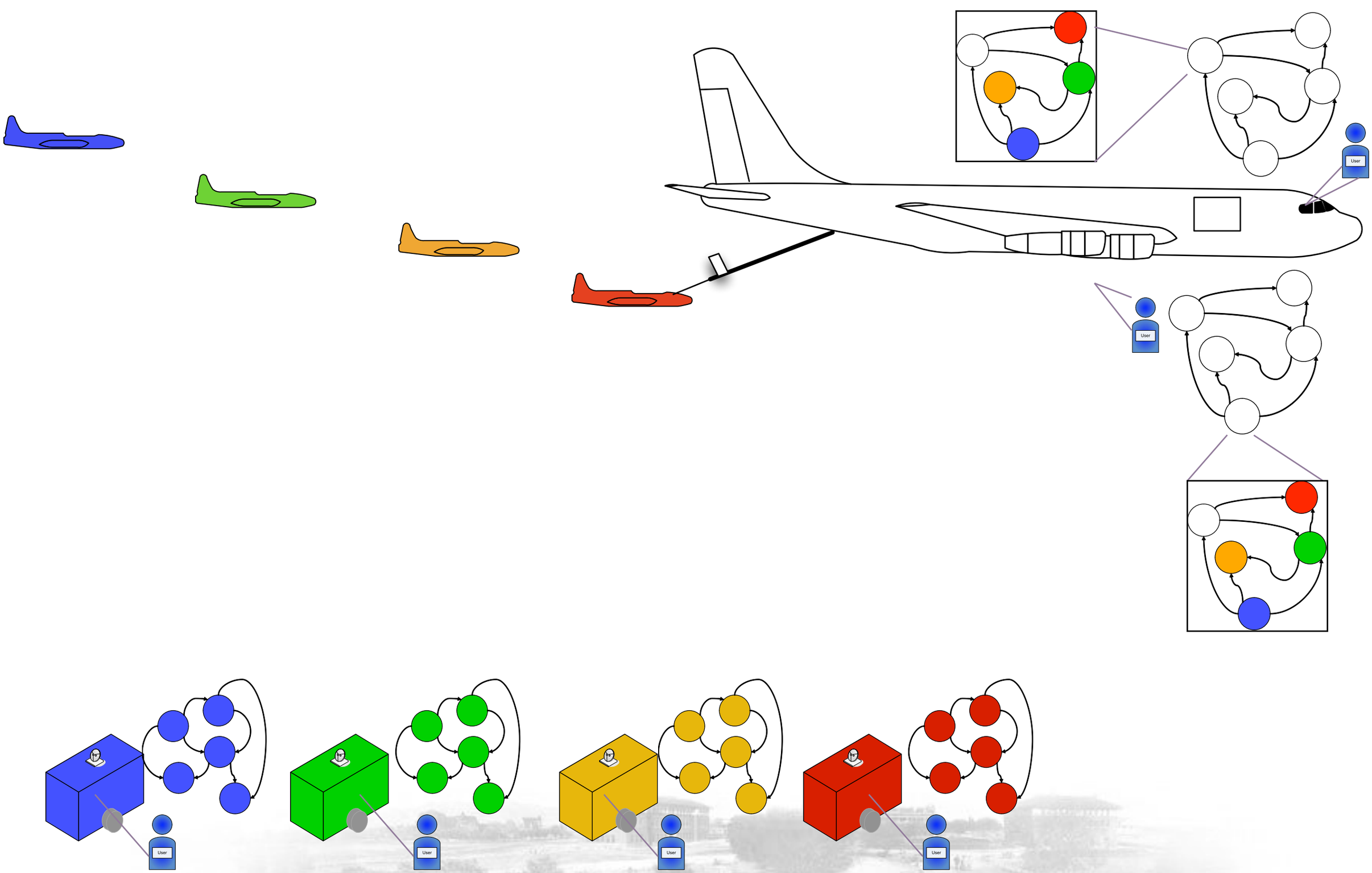
- When can the UAV disregard operator commands? ☆ Ongoing
- How are models of such behavior expressed? ☆ Ongoing
- In what way would such protocols be certified? ☆ Future
- What existing technology do we need to answer these questions? ★ Focus

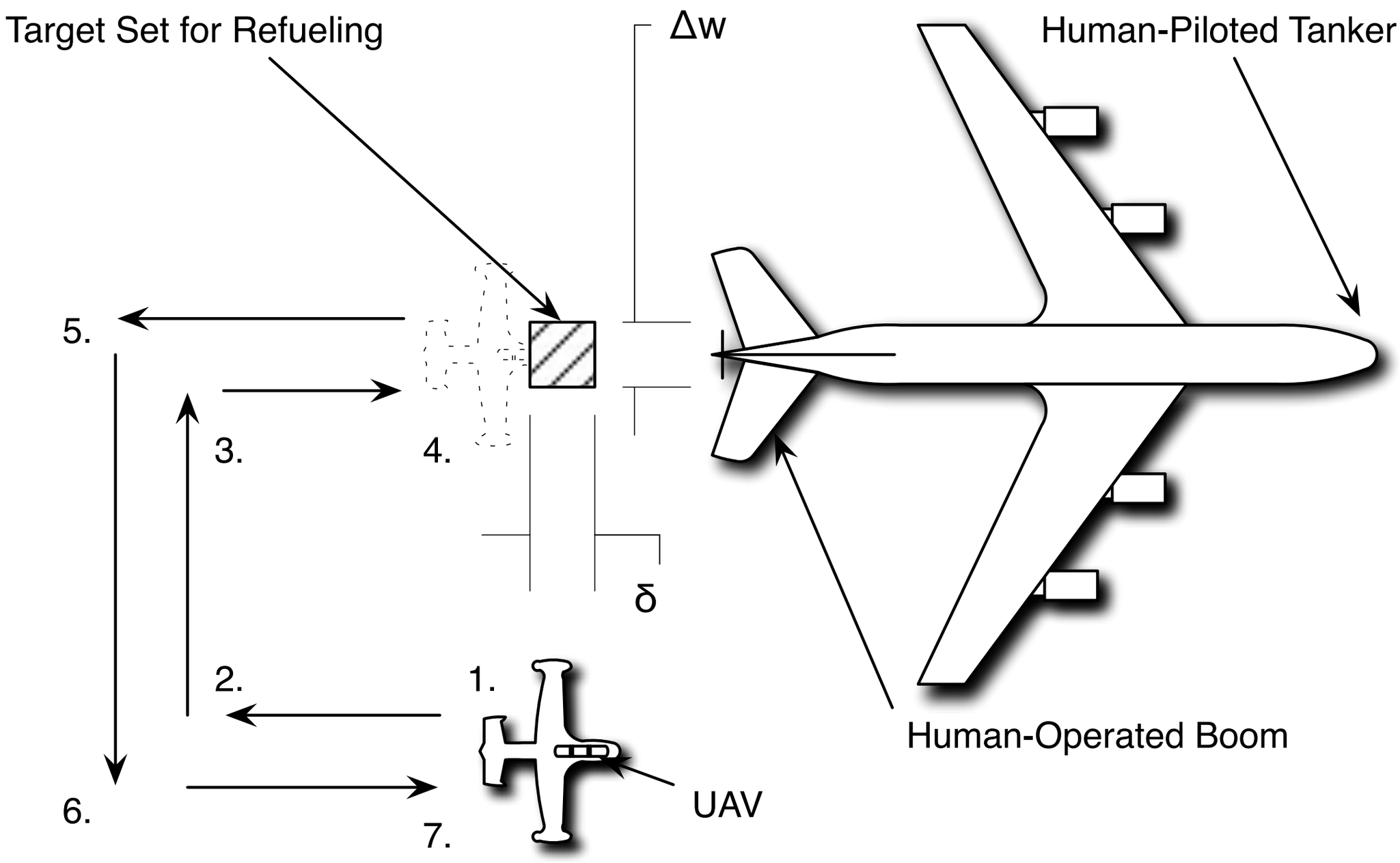
Pilot

Aircraft response

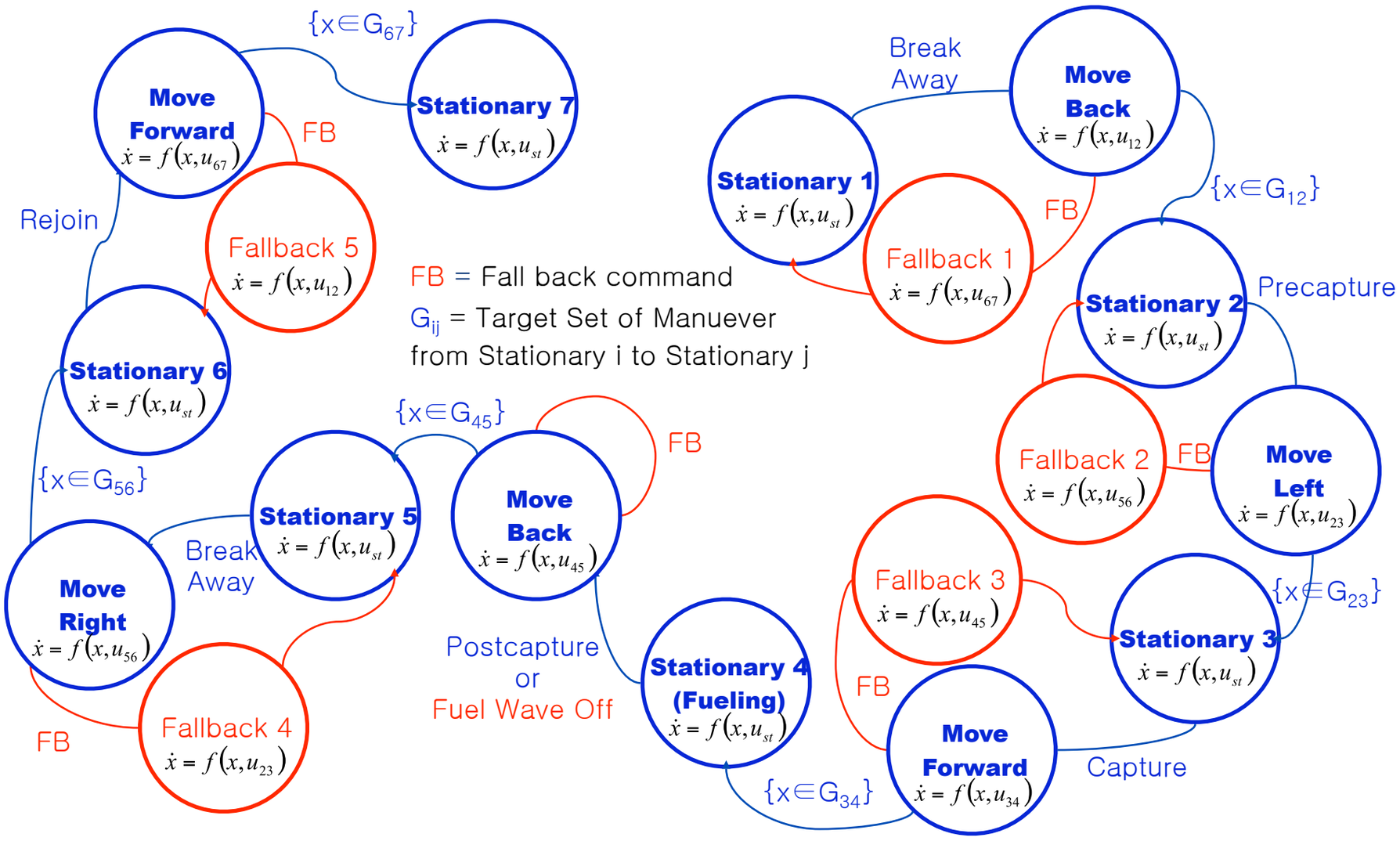
Pilot feedback

Scenario: Mixed-Initiative UAVs (e.g., AAR)

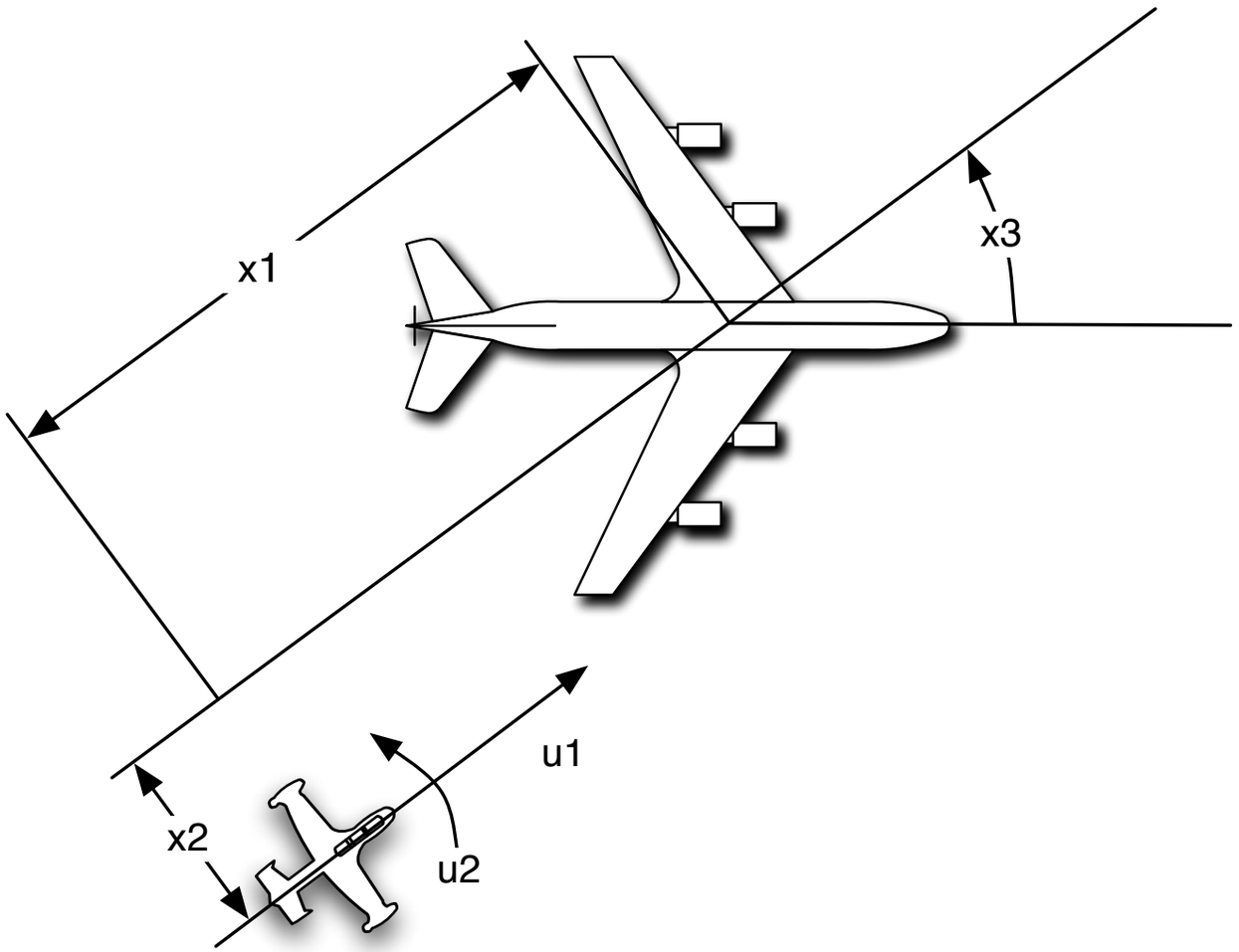



Target Set for Refueling
Δw
Human-Piloted Tanker
5.
3.
4.
δ
2.
1.
6.
7.
UAV
Human-Operated Boom

Which is very nicely expressed as a state machine



- 2D Kinematics Model used in aircraft conflict resolution research (Tomlin, et al. 2003)
  - UAV States:
    - x1 = horizontal distance to tanker
    - x2 = vertical distance to tanker
    - x3 = heading relative to tanker
  - Controlled inputs:
    - u1 = linear velocity of UAV
    - u2 = angular velocity of UAV
  - Disturbance inputs:
    - d1 = linear velocity of Tanker
    - d2 = angular velocity of Tanker



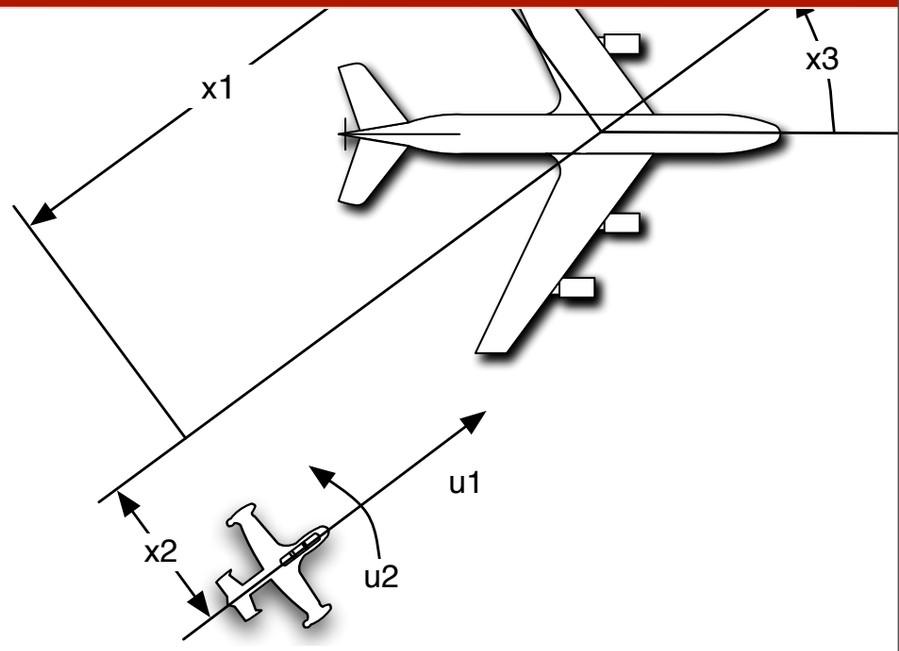
$$\dot{x} = \frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -u_1 + d_1 \cos x_3 + u_2 x_2 \\ d_1 \sin x_3 - u_2 x_1 \\ d_2 - u_2 \end{bmatrix} = f(x,u,d)$$

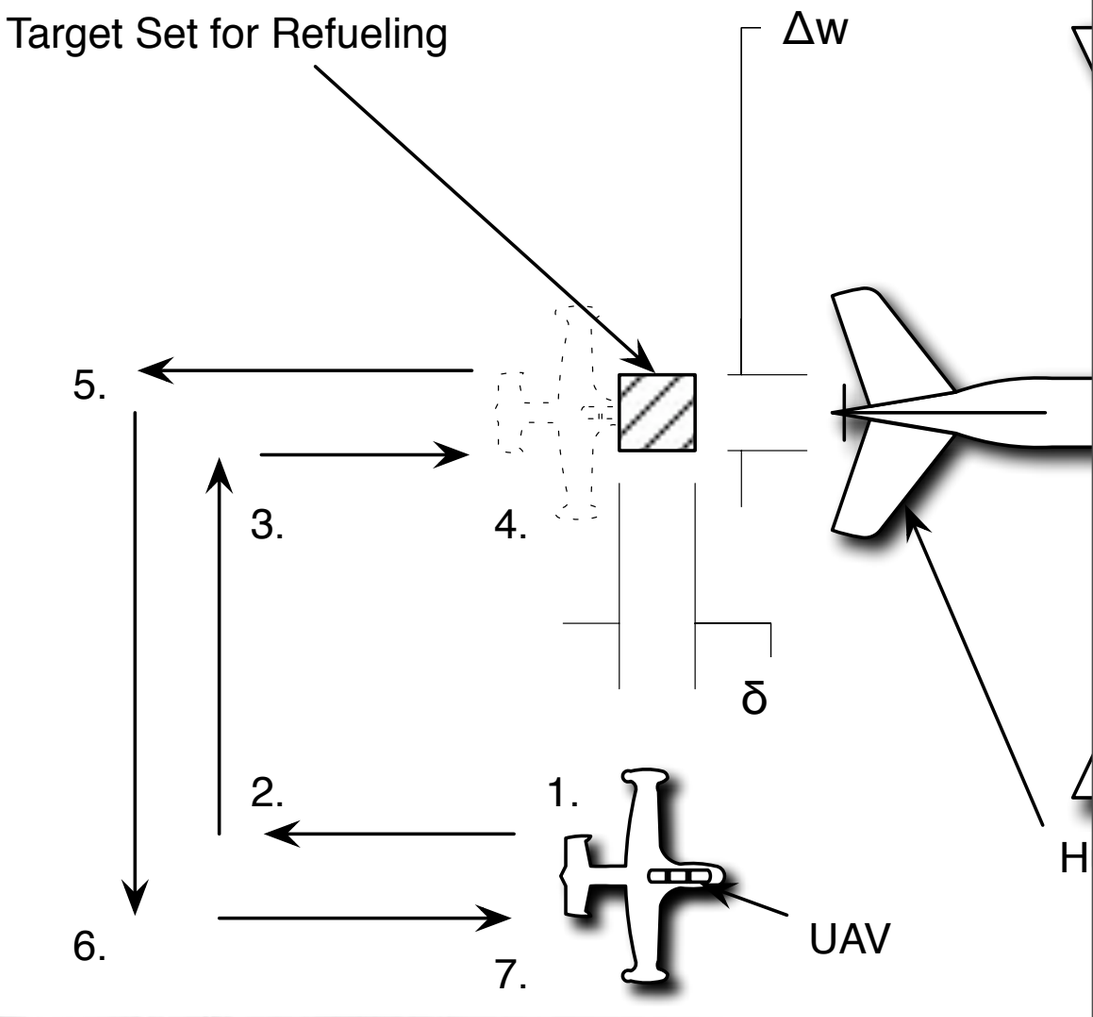
- Feedback control laws applied through inputs u1 and u2
- Assumptions:
  - Tanker velocity: d1 = v0 (constant velocity)
  - Tanker heading: d2 = 0 (constant heading)
- To control UAV to stations 1 through 7:
  - Use proportional control
  - Velocity input limits: [0.05, 0.5]
  - Angular velocity limits: [-pi/6, pi/6].

$$u_1 = \max\{\min\{k_1(x_1 - x_{1f}) + v_0, 0.5\}, 0.05\}$$

$$u_2 = \max\{\min\{k_2(x_2 - x_{2f}), \pi/6\}, -\pi/6\}$$

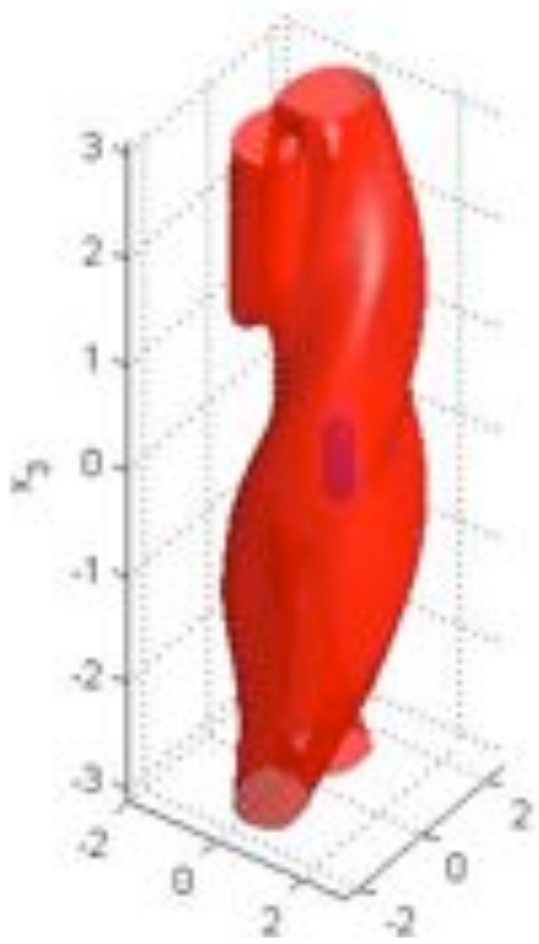
Maneuver	k_1	k_2	x_{1f}	x_{2f}
Breakaway 1	5	5	1	1
Precapture	0.15	5	1	0
Capture	5	5	0.25	0
Postcapture	5	5	1	0
Breakaway 2	0.15	5	1	1
Rejoin	5	5	0.25	1





- For transitions in refuel sequence, want target set to be small set of states around the way points
- For example, the target set for “Precapture” would be

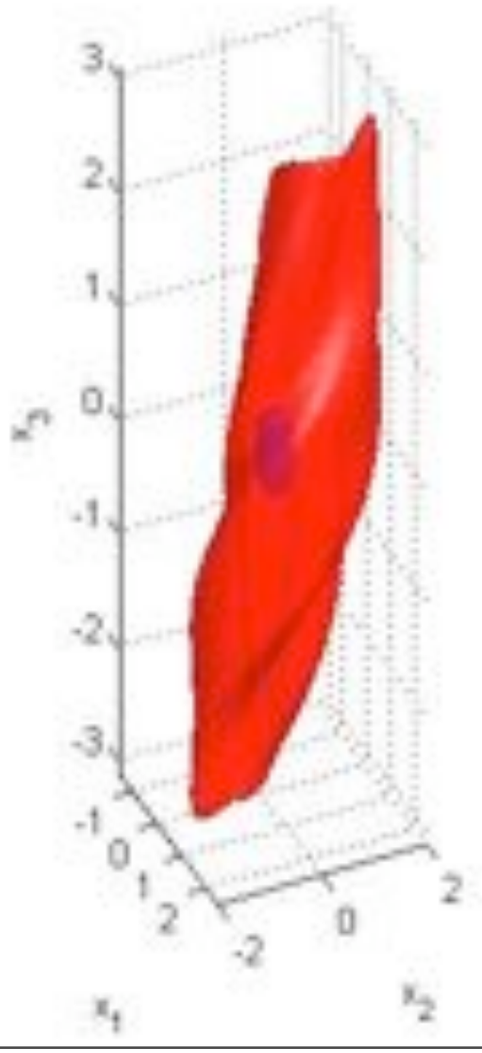
$$G_0 = \begin{cases} x_1 \in [0.75, 1.25] \\ x_2 \in [-0.25, 0.25] \\ x_3 \in [-\pi/9, \pi/9] \end{cases}$$



Reachable Set for Precapture
Time Horizon: 10s

- Target set for “Capture”:

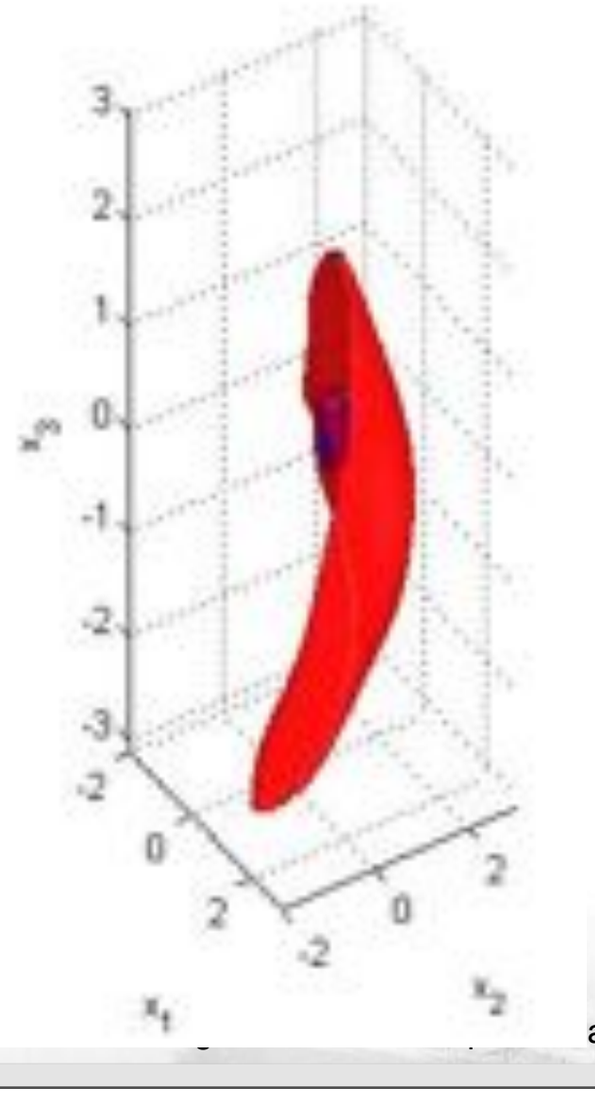
$$G_0 = \begin{cases} x_1 \in [0.25, 0.75] \\ x_2 \in [-0.25, 0.25] \\ x_3 \in [-\pi/9, \pi/9] \end{cases}$$



Reachable Set for
Capture
Time Horizon: 5s

- Target set for “Rejoin”

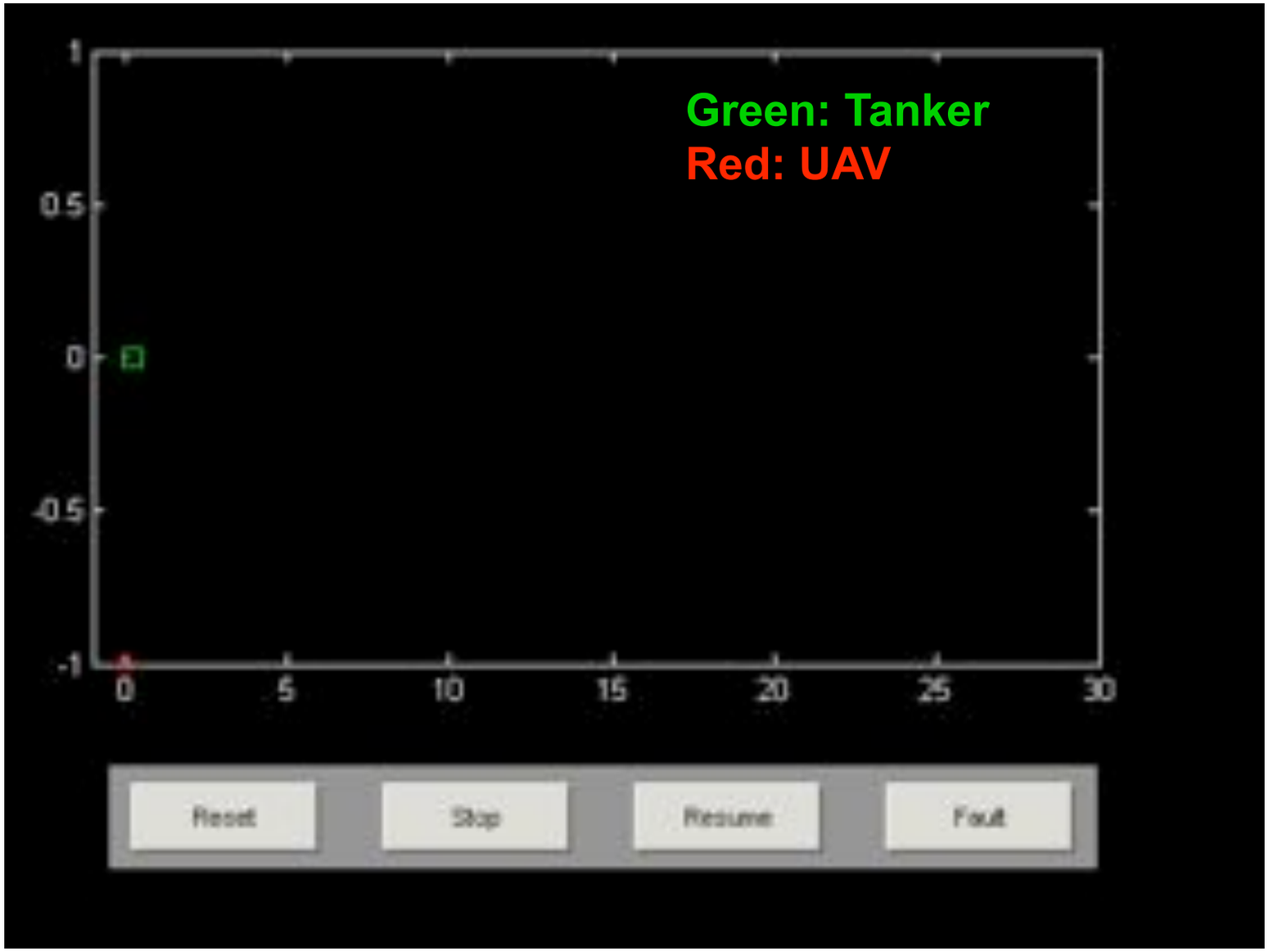
$$G_0 = \begin{cases} x_1 \in [0.5, 0] \\ x_2 \in [0.75, 1.25] \\ x_3 \in [-\pi/9, \pi/9] \end{cases}$$



Reachable Set for
Rejoin
Time Horizon: 5s

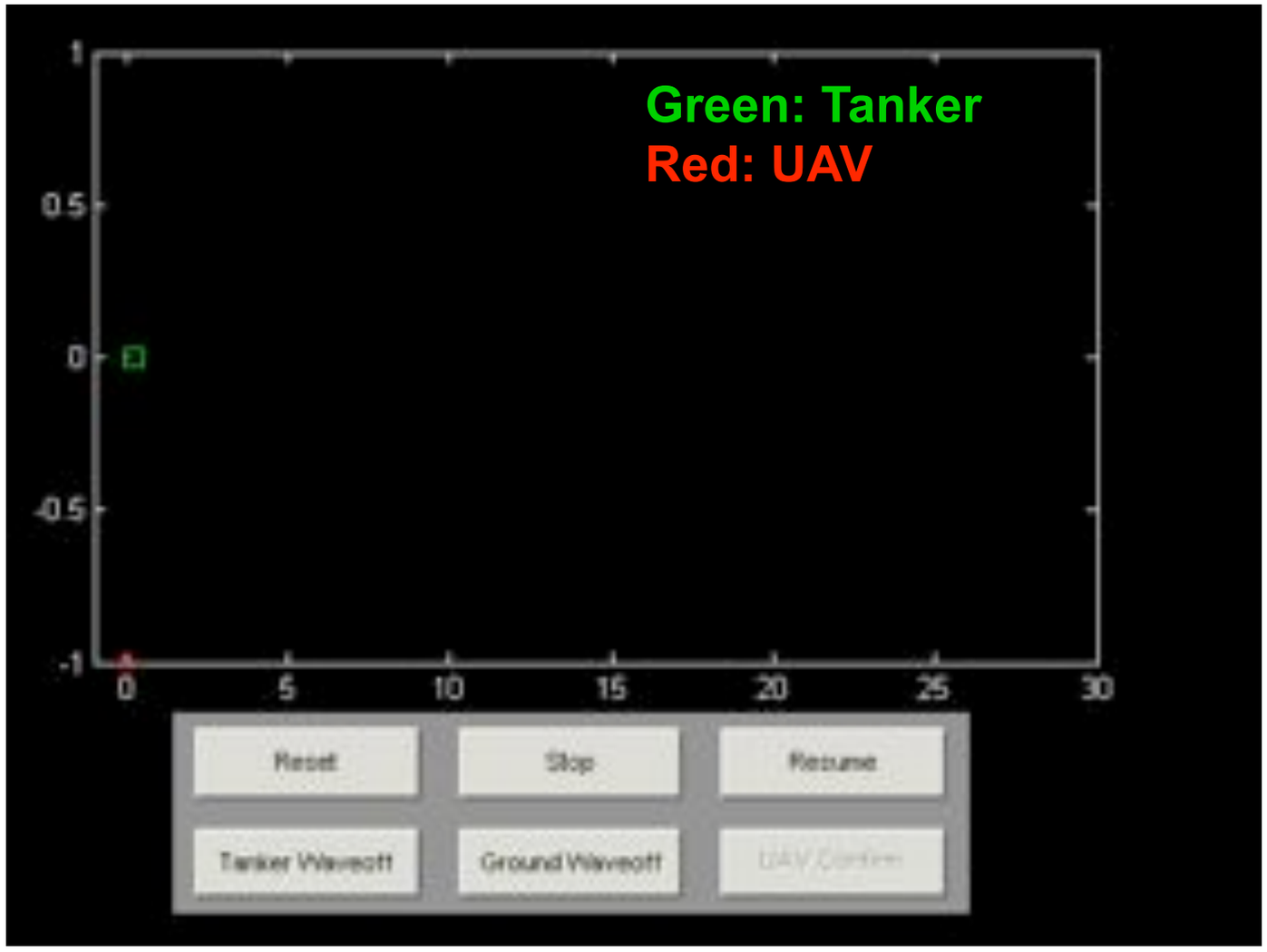
- MATLAB simulation environment
- Plots trajectories of tanker and UAV
- Updated in real-time at 1 second intervals
- Allows fault injection by user
- UAV executes fallback immediately upon fault

1. Regular run, without faults

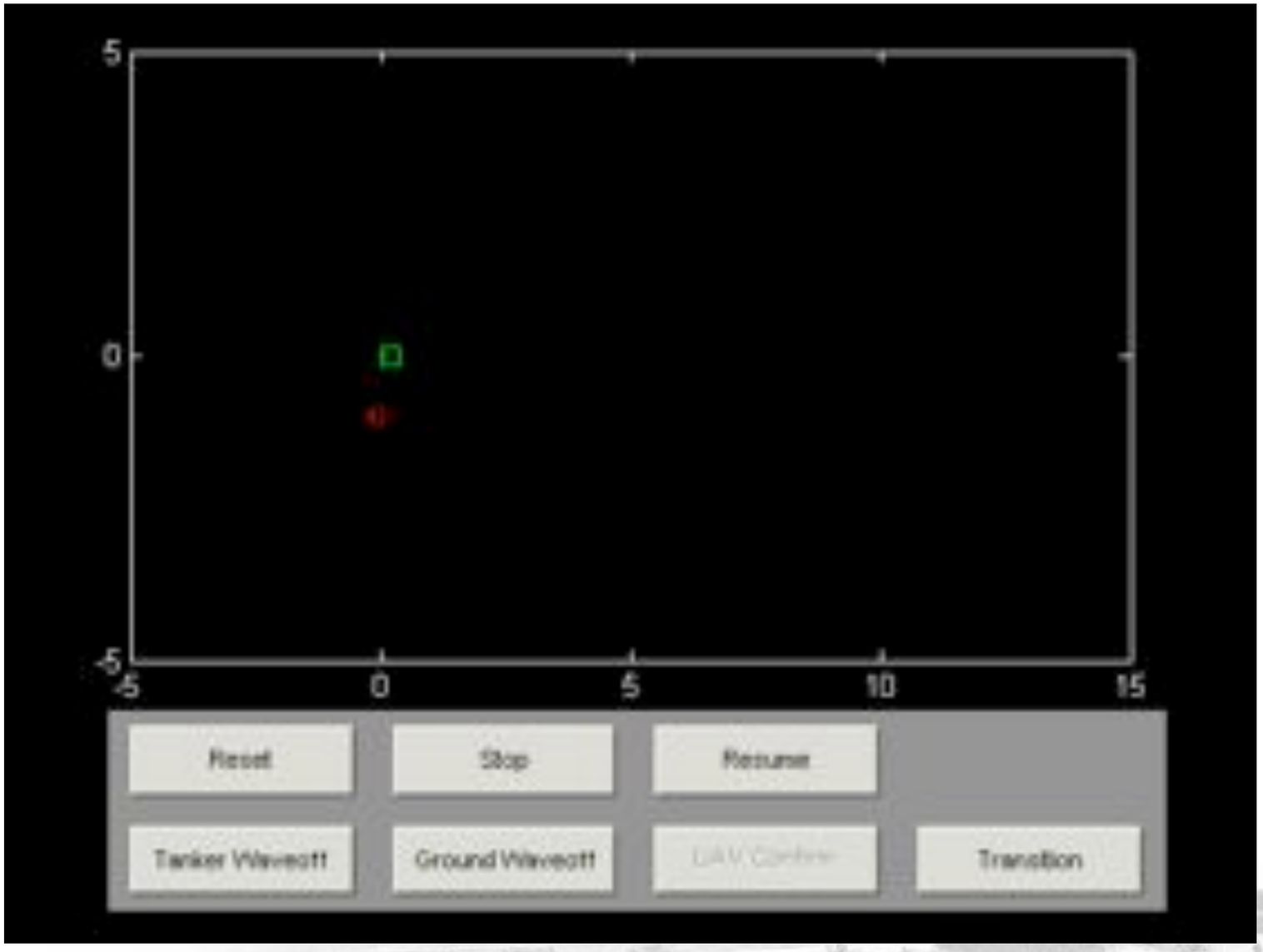


- Improved simulation environment with separate waveoff for tanker and ground operators
- Latencies simulated as delay between waveoff and appearance of UAV confirm
- Fallback executed only when UAV confirms waveoff
- Latencies currently hard coded

2. Tanker waveoff during "precapture"



- Complete refuel sequence with capture sets for all maneuvers
- User input specifies transitions between waypoints
- Capture sets can be used to minimize allotted time for each maneuver

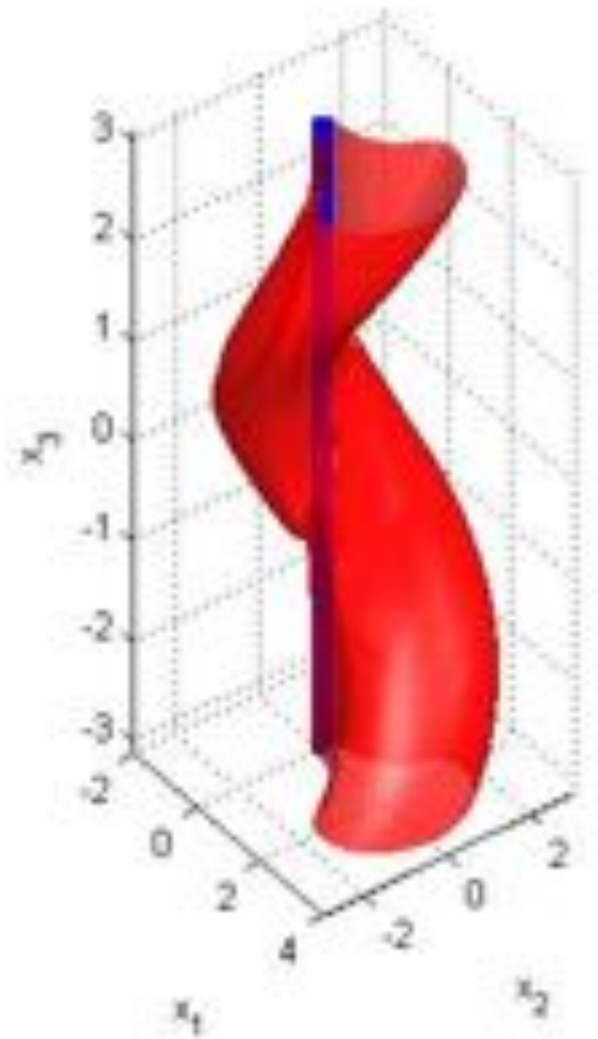


- During any formation transition, need to prevent UAV from entering into collision with tanker
- Unsafe set is set of states that can reach an unsafe zone within a given time horizon

- Unsafe zone is set of locations within a certain radius of the tanker; e.g.

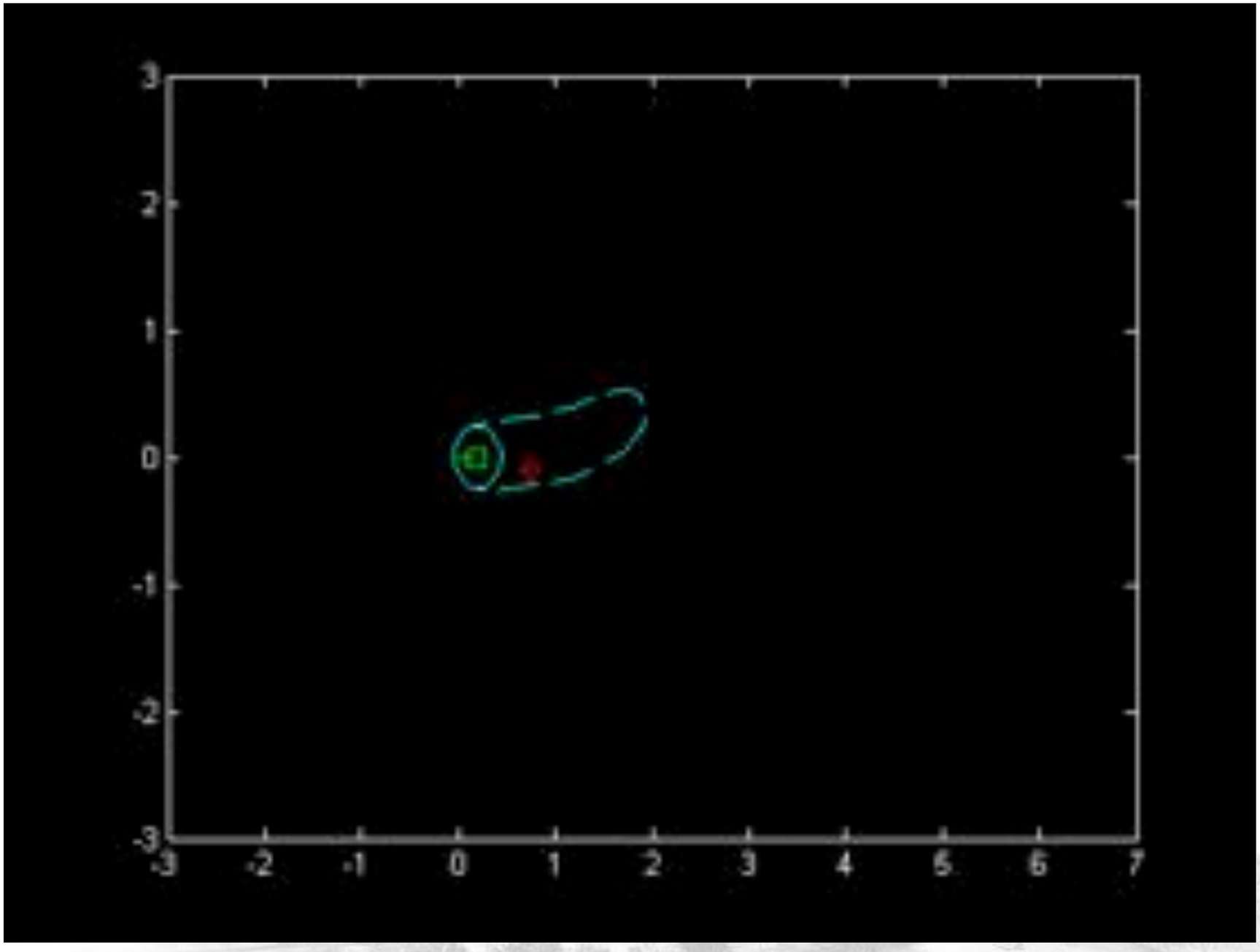
$$G_0 = \begin{cases} \sqrt{x_1^2 + x_1^2} \le 0.25 \\ x_3 \in [-\pi, \pi) \end{cases}$$

- Provides information on which maneuver should be executed to prevent collision

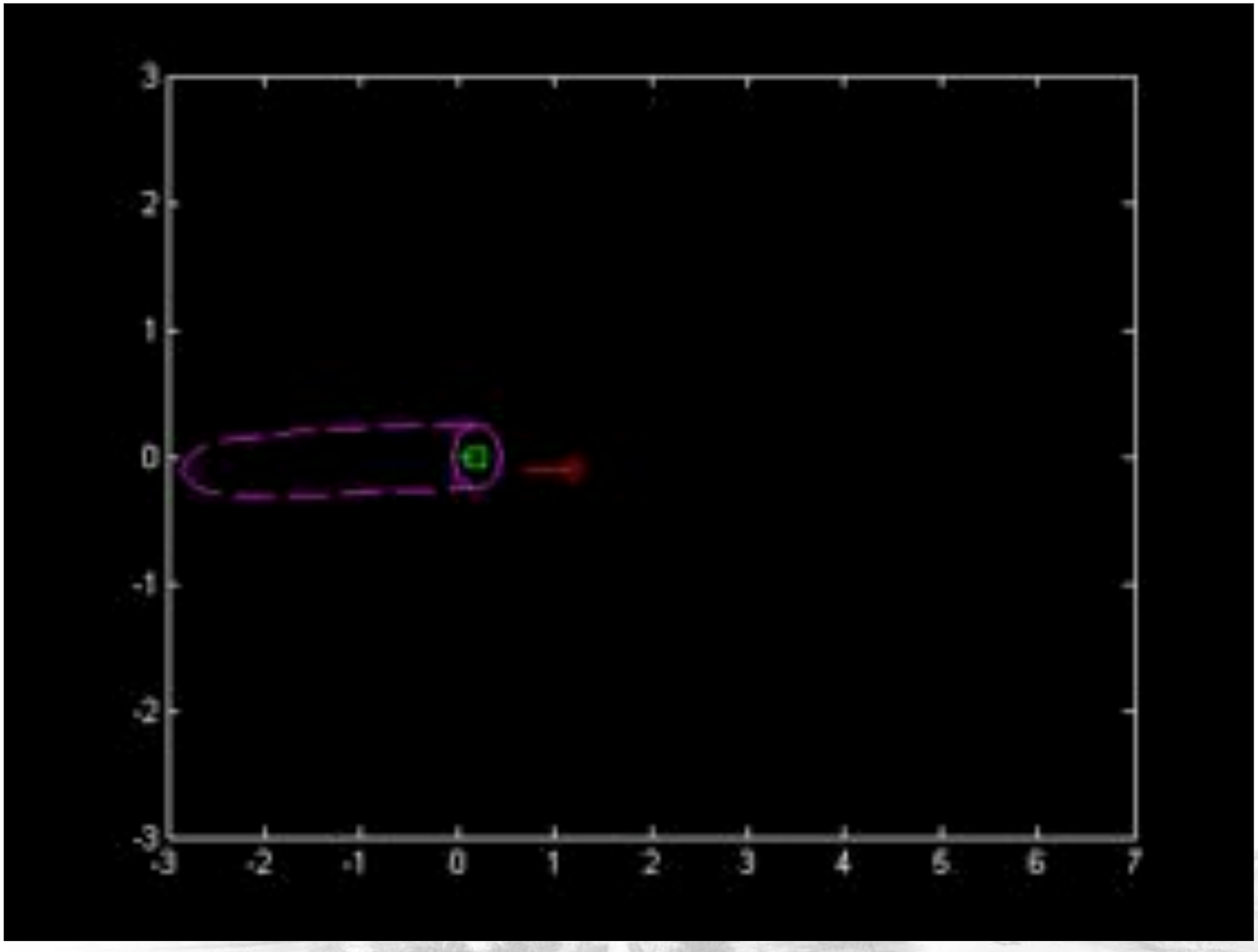


Unsafe Set for Capture
Time Horizon: 5s

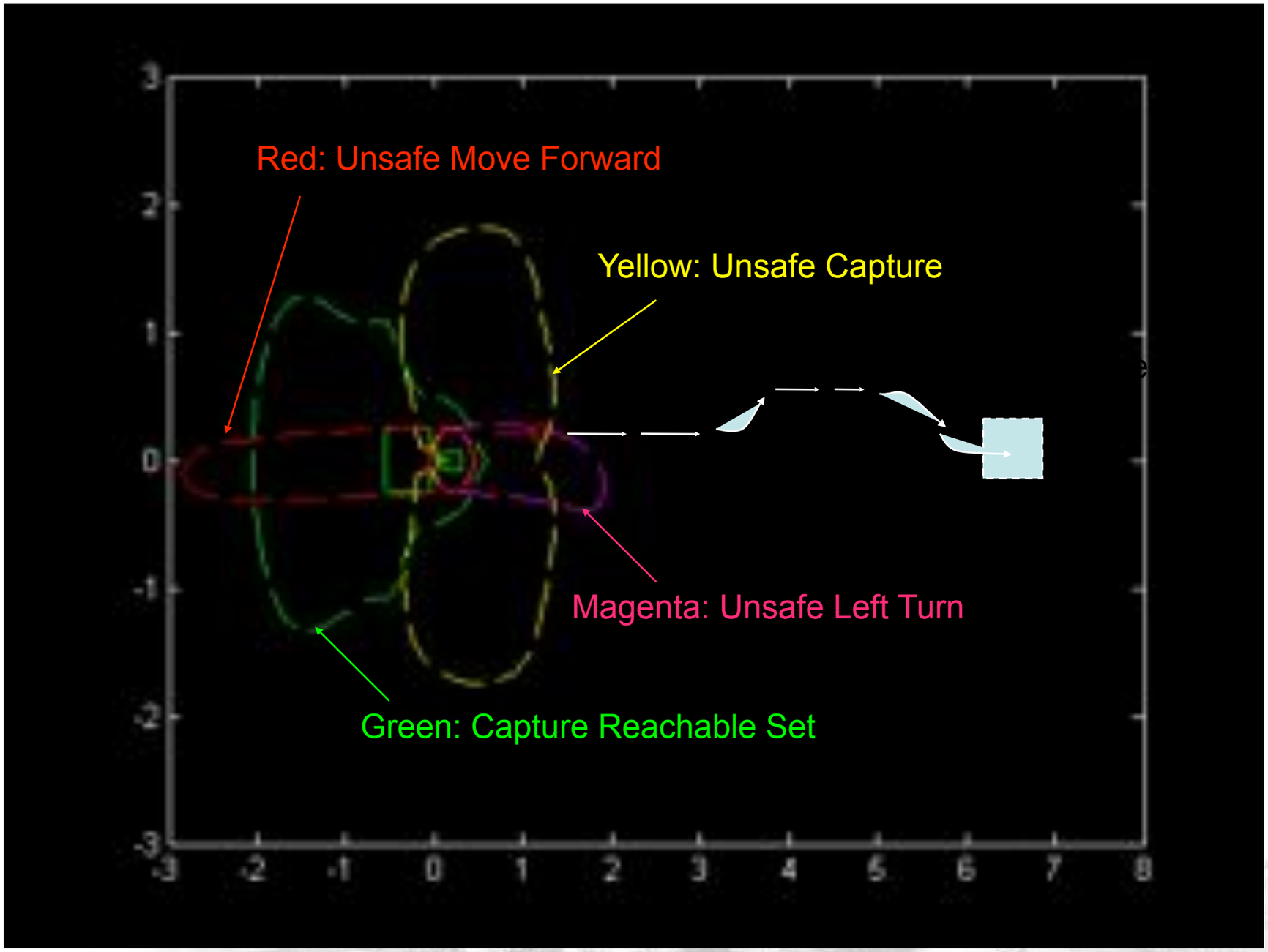
- Simulation of unsafe escape maneuver
- UAV cannot turn fast enough to avoid collision with tanker



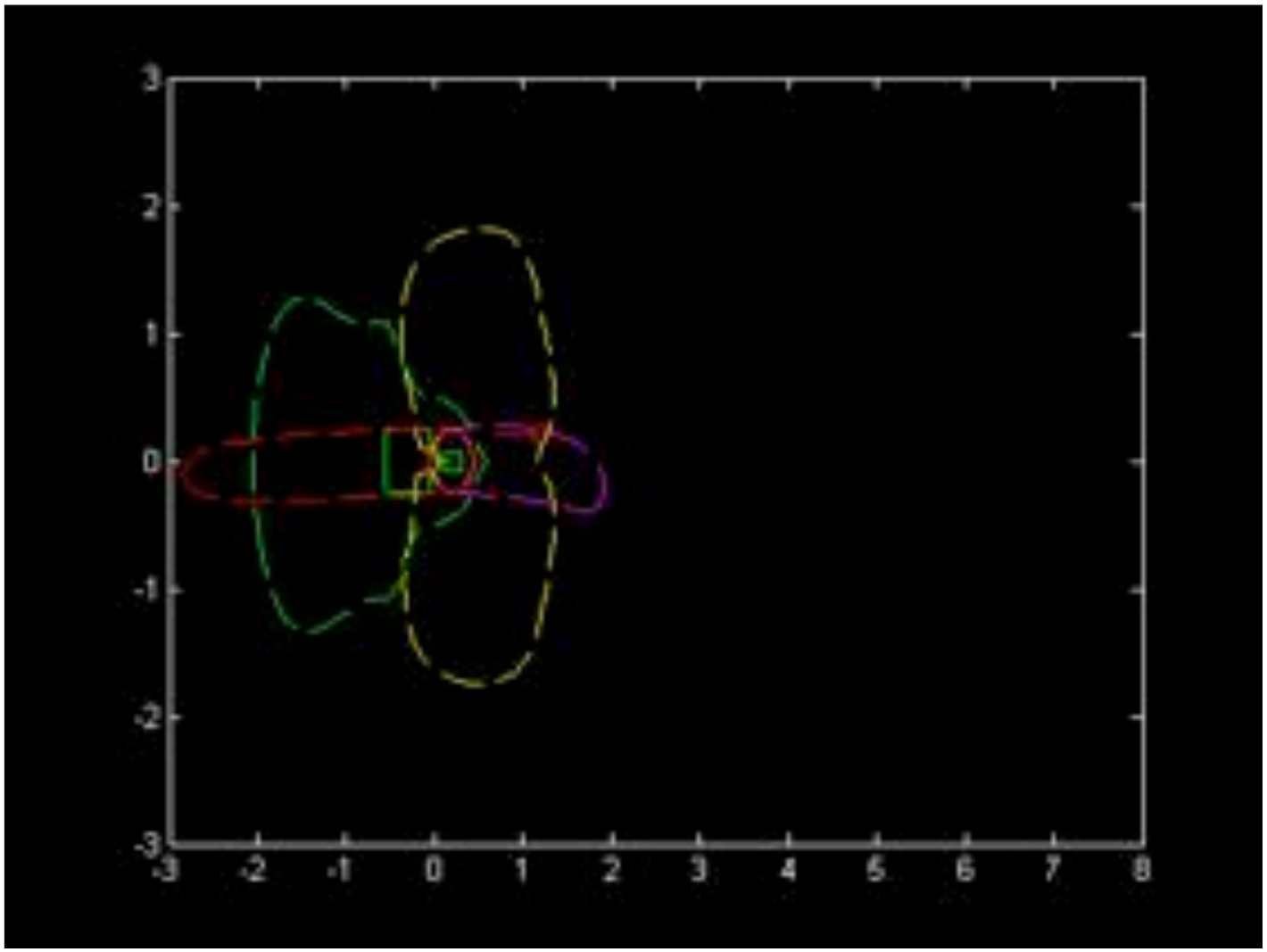
- Simulation of safe escape maneuver
- UAV first speeds up to give itself room for a turn



- UAV starts in unsafe zone for capture
- Want to reach capture zone without any collisions

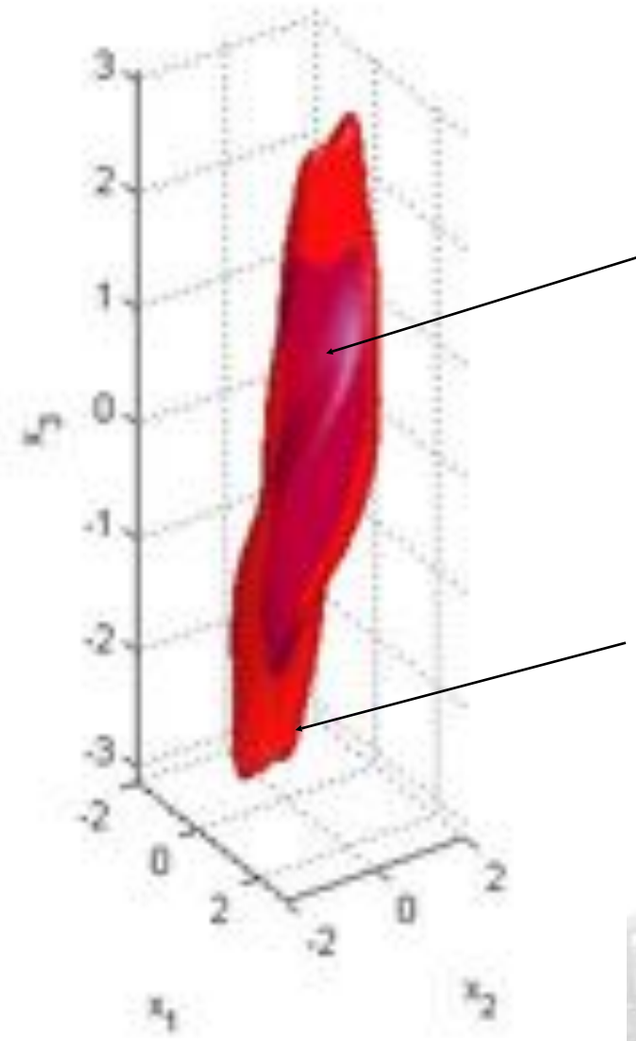


- Visualization of unsafe sets together with capture sets allows for construction of a sequence of safe maneuvers to enter capture zone



- Given some latency in communication, UAV would remain in current maneuver for t_{lat} seconds, until command for next maneuver arrives

- Reachable set for next maneuver can be propagated back in time by t_{lat} seconds using control law for current maneuver to account for latency



Inner Blue set: Reachable set of Capture maneuver over 3s

Outer Red set: Reachable set of Capture maneuver propagated back by t_{lat} = 2s using Precapture control law

- Under uncertainty in tanker and UAV inputs, wants to know whether capture and collision avoidance is possible under worst case tanker inputs and optimum UAV inputs

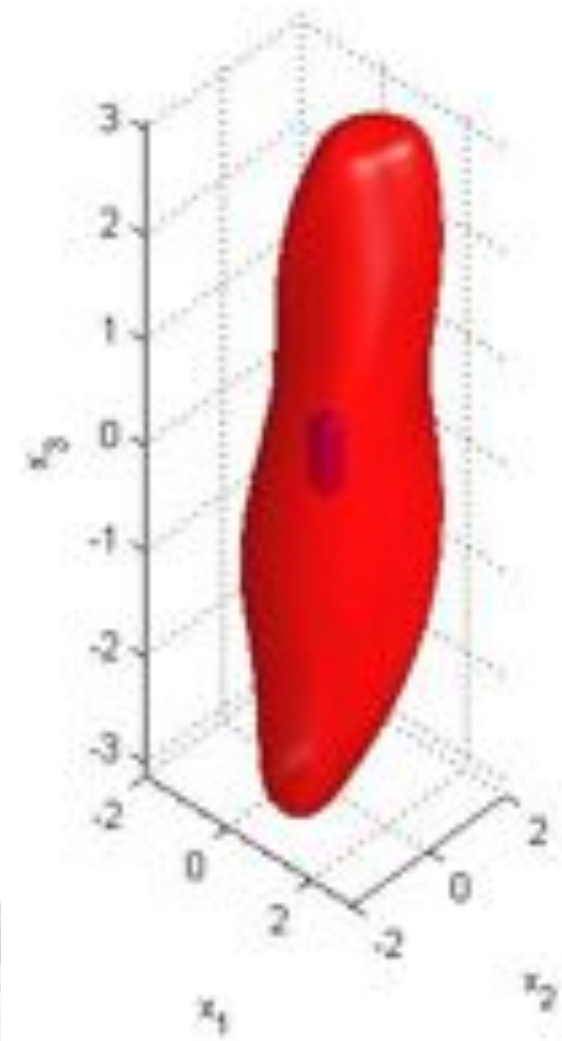
- Hamiltonians for optimum UAV input and worst case tanker input

Capture sets:

$$H(x,\ p) = \max_{d \in D} \min_{u \in U} p^T f(x,u,d)$$

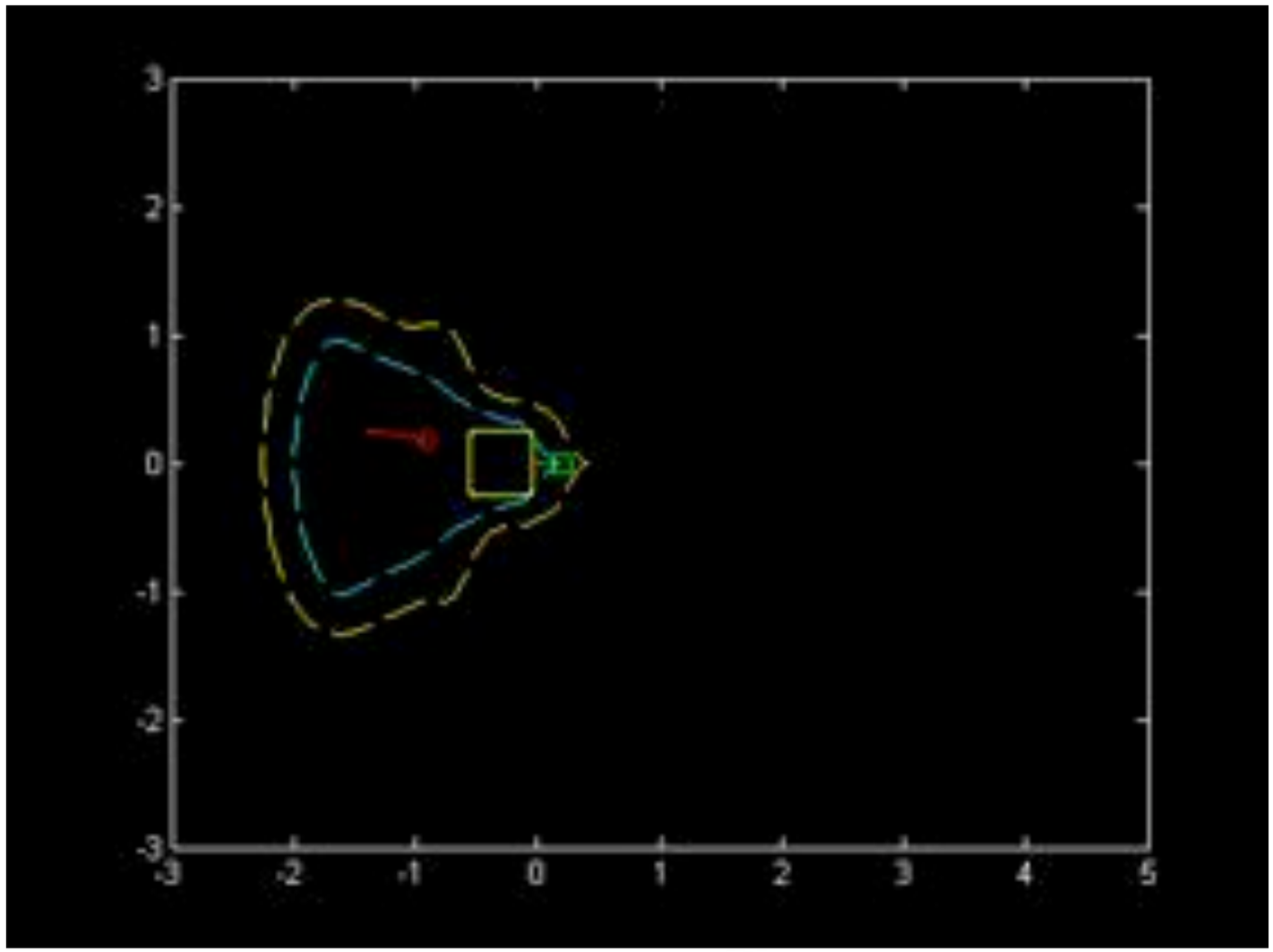
Unsafe sets:

$$H(x,\ p) = \max_{u \in U} \min_{d \in D} p^T f(x,u,d)$$



Reachable set for capture maneuver under optimum UAV input
Time Horizon: 5s

- Starting inside the reachable set generated under uncertainty, UAV can use optimum input to achieve capture, even under tanker disturbance



- The decision authority model describes the member of the distributed system who has the authority to command the UAV at a given time
- The UAV generally responds *only* to controller input, but there are some very interesting issues that should be understood:
  - Transition between controllers at the boundary of the safe sets
  - Late receipt of messages
  - Conflicting message receipt

- What is really interesting is the tradeoff between *explicit* and *implicit* decision authority
  - should the UAV's decision authority be constructive, or defined by "omission"?
  - should there exist certain cases where the UAV *must* obey the operator no matter what?

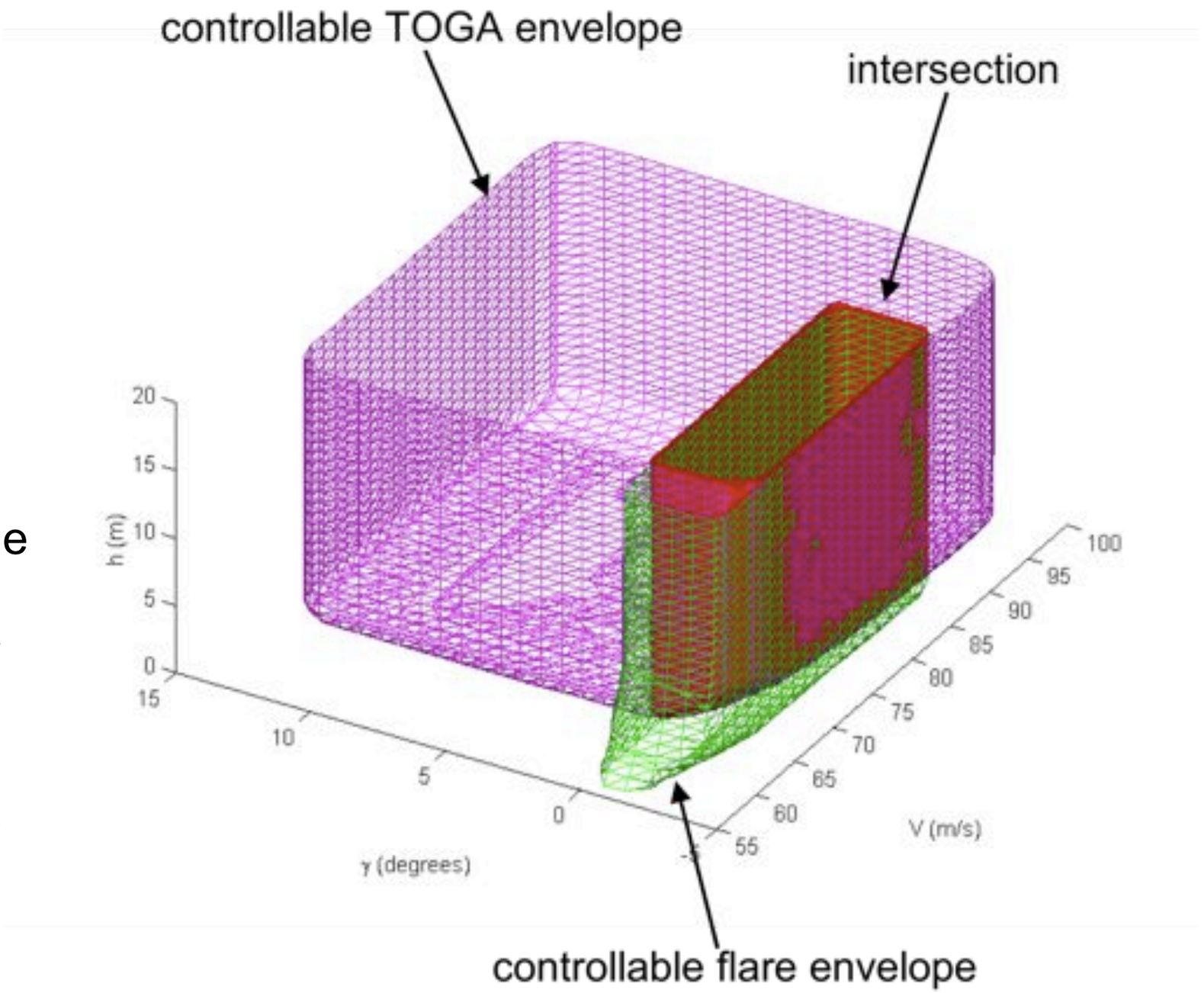
- Our goal is to provide a mechanism that puts these decisions in the hands of a system modeler, since once solution/policy will not suit all domains.

- Each state has its own intrinsic reachability analysis
- However, controllability in one state does not guarantee controllability in another!

- In cases such as these, the system can actually infer from the control laws specified whether transition *with current conditions* is “safe”

- First, however, we need to know the reachability of each state!

Tomlin, et al., 2006



$$\mathcal{P} = \{\mathcal{P}_{\text{SM}0}, \mathcal{P}_{\text{SM}1}, \ldots, \mathcal{P}_{\text{SM}N}\}$$

where each $\mathcal{P}_{\text{SM}i}$ is a concurrently executing protocol state machine with its own definition as follows:

$$\mathcal{P}_{\text{SM}} = \langle Q, X, V, \Sigma, \gamma, \delta, Y, f, \mathcal{G}_\tau \rangle$$

where Q represents the discrete states of the system, X is the set of continuous state variables, V is the set of input (control) variables, Y is the set of messages which can be sent, Σ is the input alphabet (i.e., set of messages which may be received), γ is the set of guard conditions which prevent transition, δ is the set of edges between Q nodes, f is the set of ordinary differential equations specifying the dynamics of the system, $f : X \times V \rightarrow X$, and $\mathcal{G}_\tau$is the set of continuous reachability of f, defined as the (safe) backwards reachable from the stable (final) conditions of each f_j corresponding to $q_j \in Q$ over time interval τ_j. It is possible that certain $\mathcal{P}_{\text{SM}}$ definitions may have for certain portions of its definition the null set; these $\mathcal{P}_{\text{SM}}$ models are not autonomous, and model the embedded human's interaction.

1. There exists a bounded, directional, one-way, worst-case latency for each $(\mathcal{P}_{\mathrm{SM}i},\mathcal{P}_{\mathrm{SM}j})$ ordered pair;

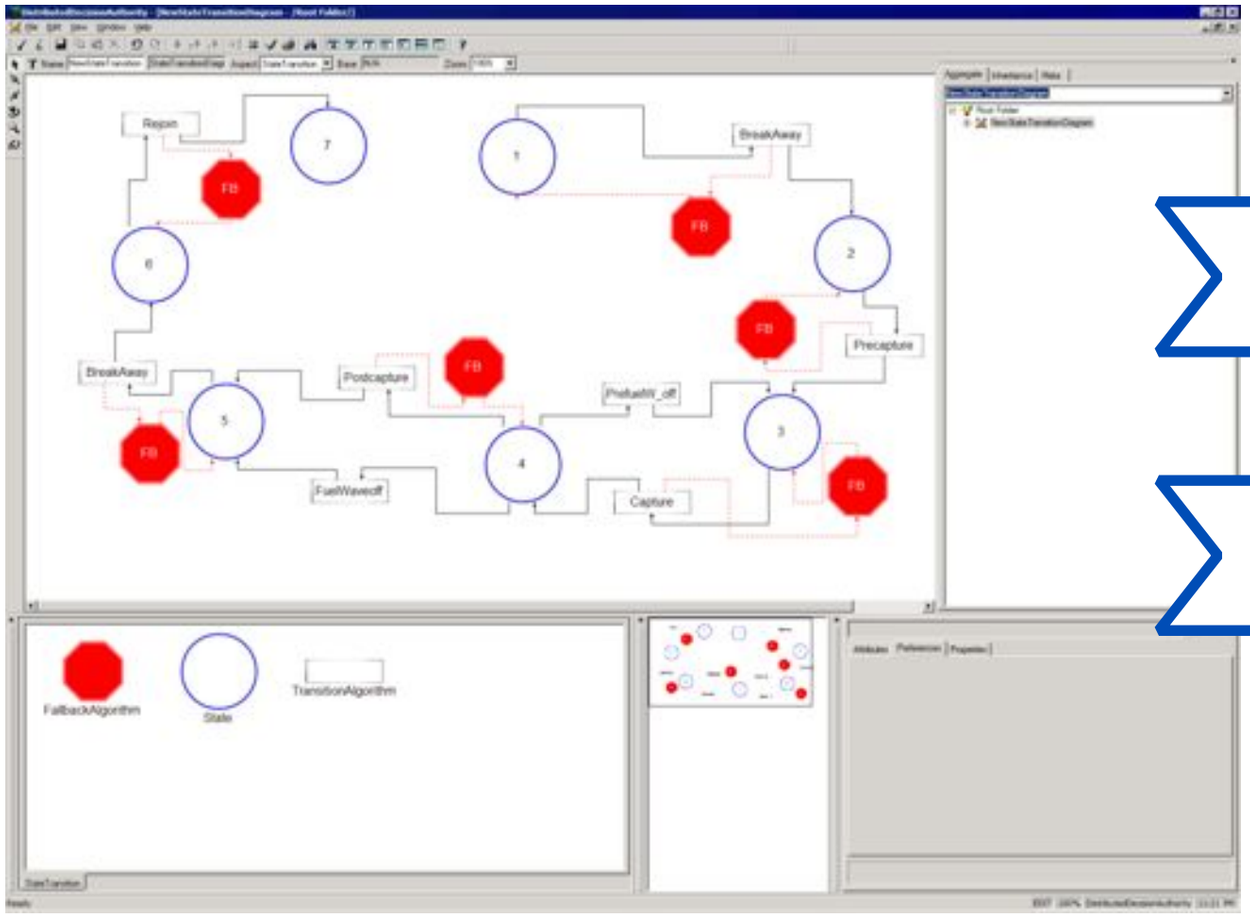
2. The set of messages $\Sigma_j \subseteq \bigcup Y_i$, meaning that the events are members of the outputs of each concurrent $\mathcal{P}_{\mathrm{SM}i}$ (including $\mathcal{P}_{\mathrm{SM}j}$); when Σ_j is specified on a transition, it is accompanied by the sender information;

3. The set of outputs $Y_j \subseteq \bigcup \Sigma_i$, meaning that the generated output messages are members of the events of each concurrent $\mathcal{P}_{\mathrm{SM}i}$ (includeing $\mathcal{P}_{\mathrm{SM}j}$); when Y_j is specified on a transition, it is accompanied by the receiver information, and multiple concurrent receivers may be specified;

4. As specified by the modeler, a $\mathcal{P}_{\mathrm{SM}}$ may generate its own messages, either due to safety or timeout considerations.

5. The guard conditions, γ, may prescribe specific behavioral or timing characteristics which explicitly disregards human-generated events, or explicitly allow autonomous interaction to proceed *without* human-generated events: this specification is the so-called *decision authority model*;

6. $x \in X, \forall \delta_{m,n} \in \delta |\ \exists \gamma_{m,n}.\ x \subseteq (\mathcal{G}_{\tau m} \bigcap \mathcal{G}_{\tau n})$; for any transition between discrete states, the transition is guarded by a condition that the current continuous state be fully contained in the safe backwards reachable set of both the source and destination states.



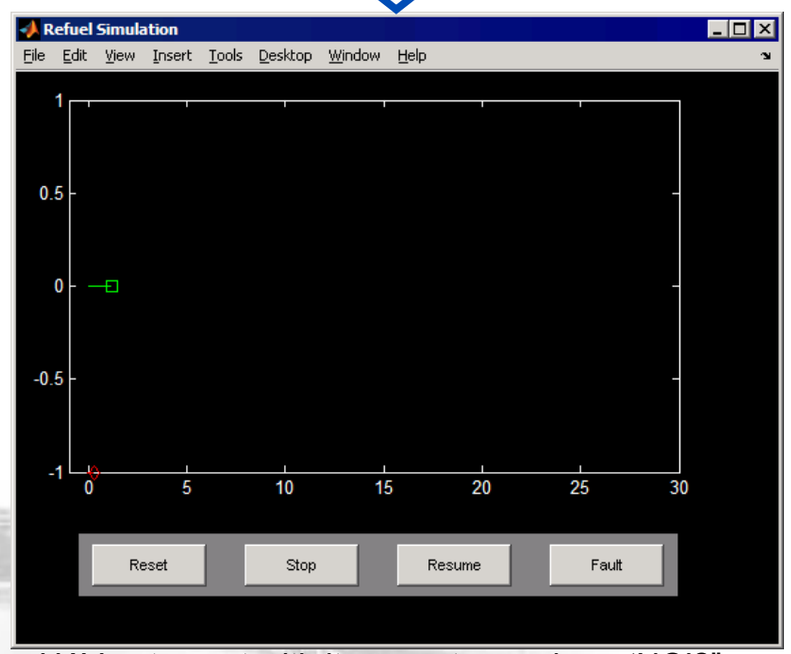
```
function refuelsim(action)
%refuelsim.m
%Author: Jerry Ding
%Date: 02/18/07
%This file creates movie of simulated trajectories of
%Tanker aircraft and UAV with fault injection

if nargin<1,
    action='initialize';
end;

if strcmp(action,'initialize'),
    % The following statements initializes the figure
    oldFigNumber=watchon;
    figNumber=figure( ...
        'Visible','off', ...
        'NumberTitle','off', ...
        'Color','black', ...
        'Name','Refuel Simulation');
    colordef(figNumber,'none')
    axes( ...
        'Units','normalized', ...
        'Position',[0.07 0.3 0.80 0.65]);

    %---------------------------------
    % Information for all buttons
    bottom=0.1;
    left=0.1;
    btnWid=0.15;
    btnHt=0.075;
    btnSpacing=0.05;

    %---------------------------------
    % The CONSOLE frame
    frmBorder=0.02;
    yPos=bottom-frmBorder;
    frmPos=[left-frmBorder yPos 4*btnWid+3*btnSpacing+2*frmBord
    h=uicontrol( ...
```



After attaching semantics to a visual language, we will be able to synthesize the MATLAB scripts, based on generalizations of the prototypes which we've built by hand. Then, "fallback" states can change, based on the model built, not the static code.

This allows protocol designers to graphically design their own protocols for experimentation.

Domain-Specific Modeling Language: used for these specifications

- Distributed Protocol model: no.
  - Still need humans to 'build' the representation, and give authority
- Reachability analysis: no.
  - Downside: Needs TLC from an experienced programmer.
  - Mitigation(s): control laws infrequently change; many states have the same control law (just different initial/final conditions);

- Reachability intersection: yes!
  - Set intersection is fast to perform, and just needs lots of memory
- Simulation interface: yes!
  - Buttons reflect potential events; ghosted buttons reflect un-actions
- Latency traces: yes!
  - Time difference of arrival (between players) is relatively easy to simulate

DSML Generator: used (with domain inferences) to produce these artifacts

- "Interesting" initial conditions for simulation: yes!
  - Finding interesting cases using discrete reachability, based on given latency, can provide a set of simulations which can test (a) the intuitiveness of the system, and (b) potential operators
- Dead end states: (maybe)--there may be some inherent undecidability here
- Mutual exclusion for transitions (events, guards): yes!
- Multi-perspective simulations: yes!
  - Our own simulations are very rudimentary, but are quite rich
- HMI experiments: yes!
  - Human Factors research into UAV interfaces, under latency requirements, now possible for embedded human applications

THE UNIVERSITY OF ARIZONA

We are always looking for good graduate students.

http://ece.arizona.edu/