I-AM: Intelligent Air Mobility

The I-AM system is an Agent Based Modeling and Simulation framework for studying Unmanned Aircraft Systems (UAS) and the related UAS Traffic Management (UTM) system. A simulation includes a variety of agent types:

* ATOC: Air Traffic Operations Center
	+ Monitors USS and UAS activity
	+ Provides visualization of simulation
	+ Provides a Matlab movie of the simulation
* GRS: Grid Reservation System
	+ Given a flight path, provides the grid element indexes which are flown over
	+ Registers which USS are operating over which grid elements
	+ Given a list of grid elements, provides a list of USS operating in each grid element
* USS: UAS Service Supplier
	+ Deconflicts flights
	+ Gets authorization for flight
	+ Provides UAS with flight info
	+ Informs USS of contingencies in system
* UAS: aircraft agent
	+ Flies mission
	+ Provides telemetry data throughout the flight
* SIM: although the simulation is not strictly an agent, it communicates with agents as necessary by passing messages

# Functions and Associated Data Structures

## CS6380\_load\_ABMS\_data: provides common data to all agents

% load\_ABMS\_data - script provides common variable definitions to agents

%

% communication info

BROADCAST = '\*';

TELEMETRY = 'TELEMETRY';

ANNOUNCE\_SELF = 'ANNOUNCE\_SELF';

SIM\_TYPE = 'SIM';

USS\_TYPE = 'USS';

UAS\_TYPE = 'UAS';

ATOC\_TYPE = 'ATOC';

GRS\_TYPE = 'GRS';

REQUEST\_GRID = 'REQUEST\_GRID';

GRID = 'GRID';

IN\_GE = 'IN\_GE';

NOT\_GE = 'NOT\_GE';

USS\_GE = 'USS\_GE';

REQUEST\_FLIGHTS = 'REQUEST\_FLIGHTS';

GE\_FLIGHTS = 'GE\_FLIGHTS';

CFB = 'CFB';

BID = 'BID';

ACCEPT\_BID = 'ACCEPT\_BID';

REJECT\_BID = 'REJECT\_BID';

ACCEPT\_CON = 'ACCEPT\_CON';

REJECT\_CON = 'REJECT\_CON';

AWARD\_CON = 'AWARD\_CON';

% agent constraints

Z\_MIN = 10;

Z\_MAX = 20;

MAX\_SPEED = 10;

DIST\_THRESH = 0.1;

H\_T = 1;

% characters

UNDERSCORE = '\_';

## CS6380\_drive\_A3\_ABMS: sets up simulation and calls simulation function

function [res,M] = CS6380\_drive\_A3\_ABMS(max\_t,draw,film,dump)

% CS6380\_drive\_A3\_ABMS - overall driver for A3 ABMS

% On input:

% max\_t (float): max simulation time

% draw (Boolean): draw during simulation

% film (Boolean): make a movie (not implemented)

% On output:

% res (struct vector): results

% Call:

% r1 = CS6380\_drive\_A3\_ABMS(100,0,0);

Sets up the lane structure (5x5 grid with teleports at each grid crossing)

This function uses a global variable (g\_fig) to close the agent table figure after the simulation is done.

## CS6380\_A3\_ABMS: name of I-AS simulator (!!)

function [res,M] = CS6380\_A3\_ABMS(fnames,ports,max\_t,del\_t,draw,film,dump)

% CS6380\_A3\_ABMS - A3 ABMS simulator

% On input:

% fnames (struct vector): names of agent function (filenames)

% ports (nx2 array): launch/land ports (x1 y1)

% max\_t (float): max time to simulate

% del\_t (float): time step increment

% draw (Boolean): display each simulation step

% film (Boolean): make a movie from data

% On output:

% res (struct vector): agent info at each step

% .agents (px9 array): info for each agent

% col 1: agent type (1: USS; 2: UAS: 3: ATOC)

% col 2: x coord

% col 3: y coord

% col 4: z coord

% col 5: dx heading in x

% col 6: dy heading in y

% col 7: dz heading in z

% col 8: ground speed

% col 9: last time called

% Call:

% r1 = CS6380\_A3\_ABMS(fnames,ports,max\_t,del\_t,0,0);

This function uses the name: SIM\_tom\_1 in the From field for messages. It sets up a table of all the agents. Outputs messages to a file named “popo” if the “dump” variable is 1. Draws a step by step visualization if the variable “draw” is set to 1; and makes Matlab movie if the variable “film” is set to 1.

Communicates with the ATOC agent by sending a “DRAW” message type if vis required, and a “FILM” message type if a movie is to be made. Before exiting it sends a “SEND\_FILM” message to ATOC if movie required (this is done through the percept message variable and the movie is returned as the Data variable).

Simulator sets up percept for each agent, calls it and gets action; accumulates all actions (including messages), and then calls each agent in turn to update the state of the world based on its actions.

## CS6380\_ATOC\_tom\_1: Air Traffic Operations Center

Gets the grid info from GRS agent so it can draw the visualization. Handles simulator requests for visualization and movie. Keeps track of all USS and UAS and prints a table of them. Reads telemetry data.

## CS6380\_USS\_tom\_3: UAS Service Supplier (example)

Will generate flights with a fixed probability. Handles GRS messages: gets grid, provides grid element usage info; gets USS grid element operations data (when deconflicting a flight); provides flights to any USS requesting them for specific grid elements. Negotiates with UAS to schedule a flight; gets UAS telemetry data [current bug: does not check that the UAS is flying its flight!).

# Data Structures

## percept fields:

* x: agent x coord
* y: agent y coord
* z: agent z coord [x;y;z] is location vector
* dx: agent x heading
* dy: agent y heading
* dz: agent z heading [dx;dy;dz] is direction vector
* speed: agent speed in direction vector
* time: current time
* messages (mx1 struct vector): each message has fields:
	+ To: name of agent for whom message is intended (\* for BROADCAST)
	+ From: agent sending message
	+ Type: indicates type of message
		- REQUEST\_GRID = 'REQUEST\_GRID';
		- GRID = 'GRID';
		- IN\_GE = 'IN\_GE';
		- NOT\_GE = 'NOT\_GE';
		- USS\_GE = 'USS\_GE';
		- REQUEST\_FLIGHTS = 'REQUEST\_FLIGHTS';
		- GE\_FLIGHTS = 'GE\_FLIGHTS';
		- CFB = 'CFB';
		- BID = 'BID';
		- ACCEPT\_BID = 'ACCEPT\_BID';
		- REJECT\_BID = 'REJECT\_BID';
		- ACCEPT\_CON = 'ACCEPT\_CON';
		- REJECT\_CON = 'REJECT\_CON';
		- AWARD\_CON = 'AWARD\_CON';
		- DRAW = 'DRAW';
		- FILM = 'FILM';
		- SEND\_FILM = 'SEND\_FILM';

## Action fields:

* dx: x heading
* dy: y heading
* dz: z heading
* speed: magnitude of velocity
* messages (as above)

## flights: keep info about all flights; fields:

* id: index in flights array
* traj: flight trajectory
* speed: flight speed
* start\_time\_or: originally requested start time
* start\_time: actual start time
* stop\_time: scheduled arrival time
* UAS: name of UAS making flight
* telemetry: data for flight
* ge\_list: grid element list for flight (flies over these grid elements)
* ge\_USS\_list (bx1 struct vector) with fields:
	+ ge\_index: linear index of grid element
	+ USS: list of USS operating in the corresponding grid element
* USS:
* USS\_reponse: (dx2 table): grid element, USS pairs; removed as flight data received
* s\_flights: scheduled flights
* bidders: list of UAS indexes (in UAS table) bidding on flight
* status: status of flight: 1 initial; 2 waiting for bids; 3 all bids received; 4 accepted contract; 5 awarded contract