Jane Blore, Stephan Boettcher, Andrew Keith, Mark Soreco
5-2-2014

CSE 6730

Final Report

Final Project – Drone Surveillance

Project Description, Problem Statement, and Approach

Drones are widely used for intelligence gathering efforts. These assets operate in urban and rural areas, acquiring targets and maintaining visual contact until sufficient information is gathered. There is a great deal of possible variation in this simple scenario. The targets vary widely by priority, location, and movement characteristics. Differing terrain in urban and rural areas can also add complexity to the drone mission.

While these machines are automated, they cannot fly forever. Periodic maintenance and refueling are still required. The drone may also have to return to base early as the result of malfunction.

Drones have a finite capacity to track targets. Far more desirable targets exist than it would be reasonable to observe. Levels of satisfaction as a percentage of targets successfully tracked can be established as a measure of performance. These levels may vary for the type of target, target priority or location.

There are a few questions we would like to explore with this simulation. How many drones are required to meet a required intelligence gathering satisfaction level? Does this number vary with the location or other attributes of the area? Is there a saturation level at which point additional drones do not provide a reasonable increase in the satisfaction level for the additional cost? Many more questions and possible areas of investigation exist, but we will focus on the above.

This project will attempt to determine the number of drones required and how that number varies with the scenario changes described above. Specifically we will examine the effect of target choice heuristic on the fleet size required for a desired intelligence gathering satisfaction level.

In order to answer these questions, a simplified urban environment will be simulated with variable difficulty of target acquisition. Varying drone fleet sizes and types will be compared for overall satisfaction of the requested target surveillance. The heuristic for selecting targets from the requested list will also be varied in order to investigate its impact on drone fleet size required to reach a certain surveillance satisfaction level. These controlled factors will be investigated subject to normally uncontrollable environmental factors, listed as nuisance factors. Nuisance factors represent factors that may affect the performance of the drone fleet, but cannot be directly changed by the owner of the drone fleet. For instance, although the fleet owner cannot control the clutter in a city, it will directly affect how easy it is to track targets. The full list of factors and response variables is shown in Table 1.

Response and Factors				
	Name	**Metric**	**Type**	**Range**
Response	Intelligence Gathering Satisfaction	Percent targets tracked successfully (%)	Continuous	0, 100
Control Factor	Fleet Size	Number of drones (#)	Continuous	1, 10
	Endurance	Loiter time (hours)	Continuous	0, 24
	Tracking Method	Tracking heuristic	Categorical	1,2,3
Nuisance Factor	City Size	Number of map nodes (#)	Continuous	0, 100
	City Clutter	Average probability of acquisition (%)	Continuous	0, 100
	Target	Type of Target	Categorical	Pedestrian, Vehicle

Table 1: Response and Factors

Related Work

Brief Description of key lit review items

Major Assumptions

As with any simulation, a number of assumptions were made in this simulation.

Conceptual Model and Simulation Description

This will be a discrete event simulation with parallel computation. The core construct is a multi-server priority queue process. The model can be conceptualized at various levels of detail. At the highest level, we have a queuing model with servers and customers depicted in Figure 1. Customers represent target assignments, while servers represent the various intelligence processes. All intelligence processes except the drone process will be modeled by single server standard queues with service times drawn from random distributions. The drone process is modeled by a lower level simulation of target acquisition and tracking.

Following the path of a target assignment through the model in Figure 1, target assignments will be generated by tips from Human Intelligence (HUMINT) process. The Combined Air Operations Center (CAOC) process will prioritize these targets. The drone process, which is the core of the simulation, will have a multi-server priority queue. Instead of being drawn from a distribution, the drone process service time will be determined by a separate sub-simulation described later. Following the drone process, target assignments are sent to Imagery Intelligence (IMINT) process for analysis to determine if the drone process was successful. If it was, the target assignments are sent to the Pentagon process for closing and disposal. If not, the target assignment must go back to the CAOC process for re-prioritization.

During drone process sub-simulation, drones will choose a target from the list and proceed to attempt target location and tracking. The paths the targets can follow will be based on a network of the roads in

the surrounding area. Drones will follow the target they are currently tracking, therefore also traversing the city along that network. Multiple drones will complete these activities simultaneously, providing parallel slack for parallel processing. The next sections give further details on this sub-process.

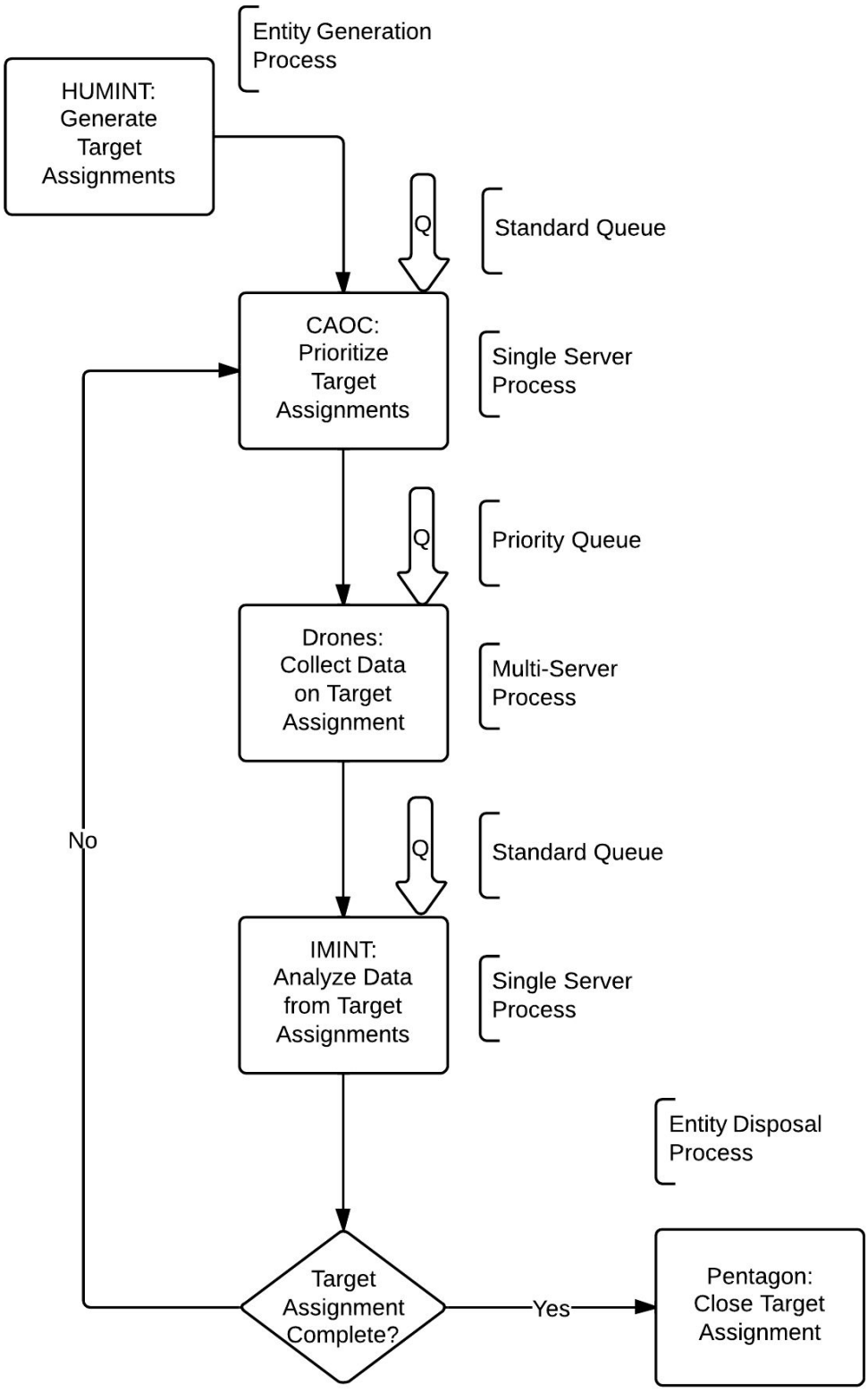

Figure 1: Process Overview

Maps

The mapping from the physical world to the modeling world will be accomplished by using an undirected connected graph, where nodes can represent streets or intersections. Figure 2 shows a notional implementation of the map network for a section of Atlanta. Each node will have a set of properties that encapsulate the distance to the next node, a set of drone detection probabilities, and which nodes are connected to the current node. An undirected graph was chosen over a cellular automata style matrix representation to speed up computation as well as maintain a discrete event style simulation. The nodes will be as equally spaced as possible along each road. A maximum node distance will be defined in the code to ensure a minimum simulation fidelity will be maintained.

The street or intersection properties at each node dictate the probability that the drone sensor will be able to locate the target in that stretch of road or intersection. These probabilities will simulate things such tree coverage on the road, crowded streets, awnings or other obstructions to the drone's sensor. The weather will not be modeled in this simulation as we are assuming the drone will be below any cloud banks. Intersections will have a higher probability of detection as they generally provide a less obstructed view from the air. These probabilities will be assigned randomly to each node at the start of the simulation. These probabilities may change slightly over time to reflect changes with time of the day. However, the change will not be that large as EO/IR sensors function well at night.

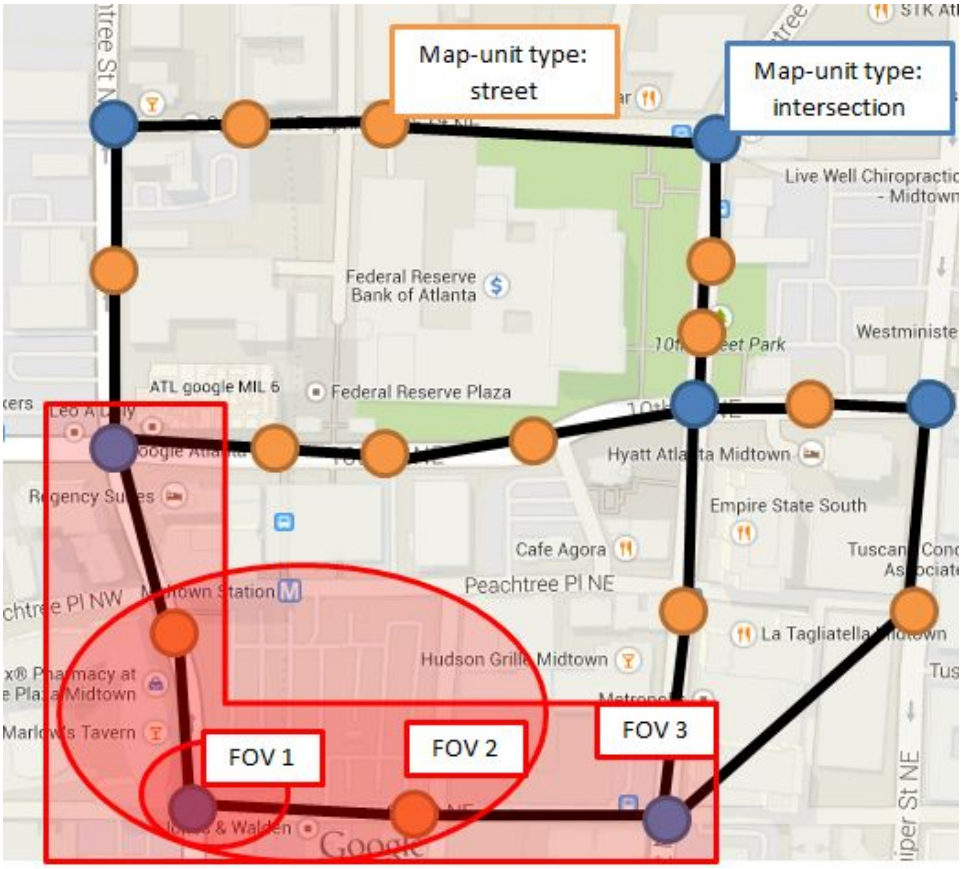

Figure 2: Map Network

Modeling the acquisition process on a network, rather than a grid, supports the project goals in several ways. Our focus is to determine the relative performance of different drone fleets, not the tracking performance of individual sensors. To compare the effectiveness of different size fleets with different sensor and dwell capabilities, we need to evaluate overall target tracking satisfaction under many different fleet scenarios. Rather than implementing a sensor algorithm to geometrically scan the ground, we abstract this process by assigning streets and intersections probabilities of tracking success. This allows us to use a discrete event model that more effectively takes advantage of look-ahead and parallel processing. Unfortunately, we do lose some fidelity: the fleet performance in the more abstract tracking model may not represent reality as well as the fleet performance with a spatially defined scanning algorithm. However, since we are assuming that every drone in any fleet has the same tracking algorithm, the loss in fidelity is expected to have a minimal impact on the final ranking of fleet options. Furthermore, this approach allows us to evaluate the heuristic for target-drone assignments.

Drone Behavior

Drones will be modeled only as sensor field of views (FOV) projected on the map. This corresponds to a fixed sensor mounted on to the bottom of a small UAV. The FOV may vary in size for different drones, which will be modeled by allowing drones to search a different number of nodes simultaneously. For instance, if a drone has FOV 1 then it will only be able to search one node at a time. However, if it has FOV 2, it will be able to search a node and all of its neighbors. FOV 3 corresponds to neighbors of neighbors and so on, as seen in Figure 2. As the drones move across the map, the sensor FOV will be focused on different nodes. When the drone is not actively tracking a target, it is not constrained to the graph. Each drone will maintain an array that determines the distance and flight time to any other node on the map. Flying to a node will be modeled by a hold function, during which the sensor will be in a sleep mode and will not look for targets until it arrives on station at the new node.

Drones will have a finite "on-station" dwell time which is determined by their fuel levels and time until their next maintenance action. Each drone will have two timers, a "joker" and a "bingo". The joker time will be the last time at which a drone will be able to accept a new target. The bingo timer will be the time until the drone has to return to base for either refueling or maintenance.

Target Behavior

Targets are not aware that they are being tracked. As a result, they maintain normal behavior patterns and do not attempt to avoid detection. Target assignments will be initiated when an intel tip is created. After the target arrives on the graph, it will loiter in the vicinity of the tip for a predetermined amount of time before moving in a random direction. Targets will be broken into groups by type: vehicle or person on foot. Each group will have a different movement speed distribution. These distributions will be normal about the average speed of a walking person and average vehicle speed limit respectively. An individual target will maintain a constant speed throughout their movements. Upon entering an intersection node, a target will randomly choose a direction in which to proceed according their type's speed. Targets have an option to exit the simulation at any point in time and at any node, road or intersection. This models targets entering buildings at any point along the road.

Each target will have properties such as priority, intelligence gathering value, ease of tracking, length of visual contact required, possible location, speed and direction of movement. The goal length of visual contact for sufficient intelligence collection is an important property. The closer to the goal time a target is tracked, the more likely that the data analysis process will determine the assignment complete. If the analysis process determines that there is insufficient intelligence, the assignment will have to re-enter the prioritization queue.

Target detection

Target detection is determined by the node probability and the sensor FOV location. The drone will only have one chance per node to detect the target. However, if the drone fails to acquire the target, it can proceed in the target's predicted direction to the next node and attempt to acquire the target there. Once a target is acquired, there is also a probability that the drone loses the target, based on the street or intersection loss probability and the target-specific loss multiplier.

Targeting Heuristics

There are three targeting heuristics that will be compared.

Naive Heuristic: In the most basic heuristic, drones will take target assignments strictly in priority order and will track a target until the required tracking time has been accomplished. There are no limits to how long a target will search for a target or how many times it will re-attempt the same target.

Local Heuristic: In this approach, the drone will search the local area for the assigned target. If it doesn't find the target, it sends the target assignment to the analysis process with zero tracking time, ensuring that the target will require "rework". This approach is intended to maximize the time the drone spends tracking versus searching. However, it will require a lot of reassignments and does not take target priority into account.

Impatient Heuristic: Each drone will have an "egg timer" which determines how much simulation time the drone will have to acquire and track the target. The length of this timer is dependent on the priority of the target. If the drone is unable to finish tracking the target before the egg timer ends, the target is sent to the analysis queue without reaching its goal tracking time. As a result, it has a lower probability of assignment completion. This increases the probability of "rework" but also limits how long a drone will spend searching for or tracking a target with a high cost-benefit ratio.

Parallelism

Due to the modular nature of this simulation, we are able to implement a parallel Discrete Event Simulation. A notional block diagram of the program flow is in Figure 3. Parallelism is achieved by having each drone as its own logical process, as well as having a controller unit and a intel processing unit. The computational load of each event, whether it is processing the target, searching for a target, or moving around on the map, is minimal. Rather than splitting up each task to be computed in parallel, the simulation will have multiple workers spread out the event workload to for the greatest performance increase.

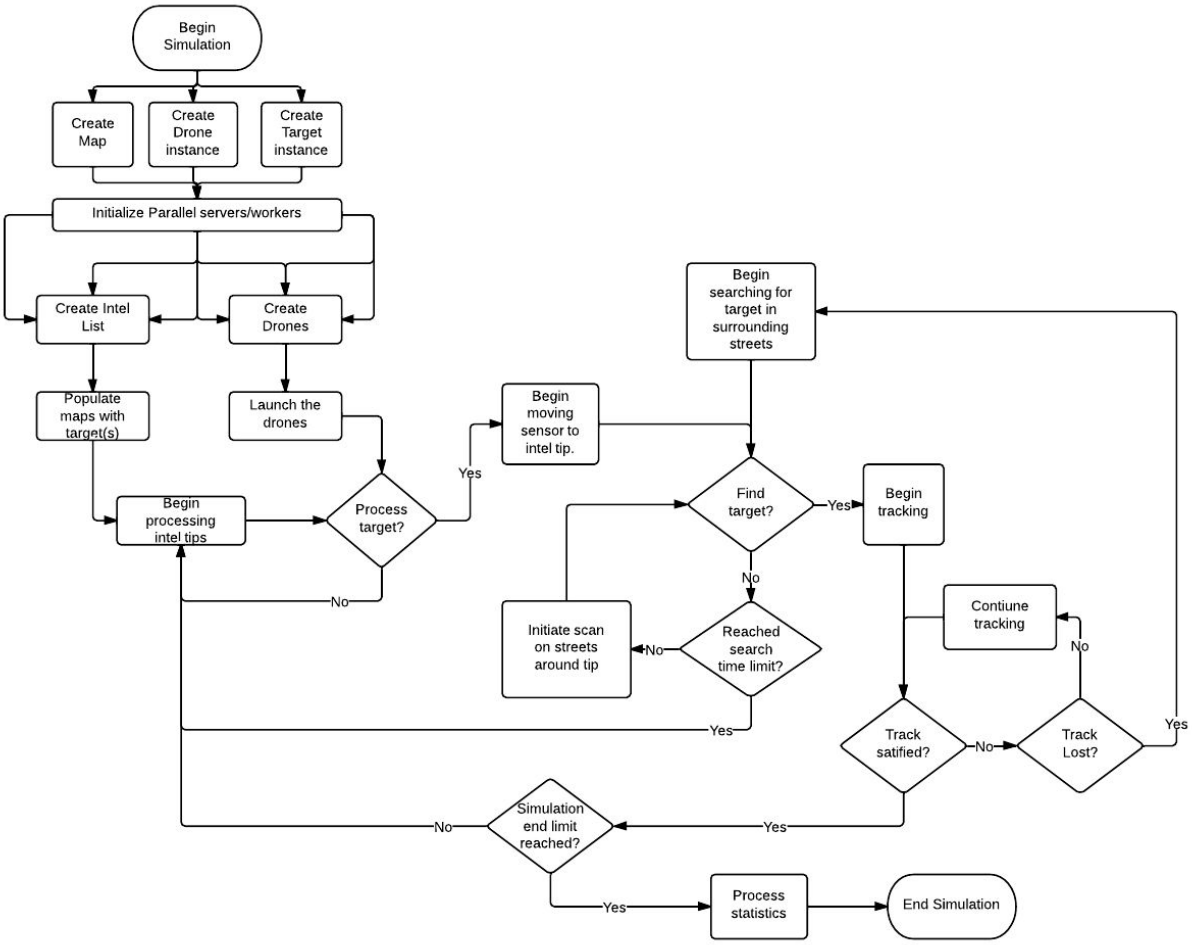

Figure 3: Program Logic Flow

In Figure 3, the simulation begins by creating the target and drone classes as well as the simulation map. Then, the parallel workers and servers are setup. The breakdown of the workers and servers can be seen in Figure 1. As can be seen in Figure 4, each drone-target combination is a logical process, as is the controller, the intel generator and the postprocessor. All of these processes allow for a highly parallel simulation to function.

Related Work – Move to Appendix

Calendar Queues

Brown, Randy. "Calendar Queues: A Fast O(1) Priority Queue Implementation for the Simulation Event Set Problem." *Communications of the ACM* 31.10 (1988): 1220-227. Web.

A calendar queue is a O(1) data structure to hold pending event list for a discrete event simulation. Previous O(1) data structures struggled with overflow lists making them only useful for very large queue sizes. The calendar queue can be thought of as having years and days, where each day has a sorted linked list of events. The difficulty in implementation is choosing the right length of day and year as the number of events changes. A rule of thumb is to double the number of buckets (days in

the year) when the number of events in the queue doubles. The bucket width (length of a day) should be about the same as the average separation of adjacent queue elements. Pseudo code for implementation included. Performance is much better than other approaches for large and very large queue sizes but worse for small queues (fewer than 10).

Design of UAV Systems

www.southampton.ac.uk/~jps7/Aircraft/Design/Resources/UAV/Resources/ASE261.11.Payload.ppt. N.p., n.d. Web. 24 Mar. 2014.

This work covered the multitude of trade spaces inherent to drone design. Tradeoffs such as sensor resolution vs weight, power vs size, and weight vs size were covered. The sensors covered were primarily Electro-Optical and Infrared (EO/IR), but Synthetic Aperture Radar (SAR) mapping was also discussed. EO/IR sensors perform well in both day and night conditions, but typically do not function well in weather. The basic EO/IR equations for pixel pitch, max sensor range and field of view were defined.

Empirical Comparison of Priority-queue and Event-set Implementations

Jones, Douglas W. "An Empirical Comparison of Priority-queue and Event-set Implementations." *Communications of the ACM* 29.4 (1986): 300-11.

The hold model is used to compare queue structures to facilitate extrapolation to different size queues and different applications. The hold model includes enqueue and dequeue operations derived from several different probability distributions. Many structures are compared, including heaps, lists (including Henriksen's), binomial queues, pagodas, and splay trees. The results show that linked lists are optimal for very small lists (fewer than 10). Splay trees are stable and among the fastest structures at O(log n). A full comparison with average and worst case performance is included for all structures considered.

Self-adjusting Binary Search Trees

Sleator, Daniel Dominic, and Robert Endre Tarjan. "Self-adjusting Binary Search Trees."*Journal of the ACM* 32.3 (1985): 652-86.

The splay tree is a self-adjusting form of the binary search tree that achieves amortized O(log n) performance which is the same as a balanced tree. Splaying operation allows the tree to adjust to actual usage, avoids overhead, and is conceptually simple. Drawbacks include more local adjustments during access and more expensive individual operations. The basic restructuring operations are zig (terminating single rotation), zig-zig (two single rotations), and zig-zag (double rotation). These restructurings result in more commonly accessed sub-trees being brought closer to the root. It is also possible to have a semi-adjusting tree where the restructuring is more limited.

Ladder Queue

Tang, Wai Teng, Rick Siow Mong Goh, and Ian Li-Jin Thng. "Ladder Queue." *ACM Transactions on Modeling and Computer Simulation* 15.3 (2005): 175-204. Web.
Calendar queue struggles with certain distributions, especially highly skewed, and when the parameters change rapidly; this is likely due to the heuristic used to adjust bucket width and quantity. Ladder Queue achieves more consistent O(1) performance by (a) sorting as late as possible (b) limiting resizes to a single structure (c) resizing bucket width dynamically for each bucket and (d) dynamically adjusting operating parameters without sampling. The basic ladder queue structure is a top, several rungs, and a sorted bottom. New rungs are spawned when a bucket in a given rung gets too crowded. Dequeues pull from the bottom only and enqueues are put into unsorted buckets. Theoretical and empirical justification is provided to support claims of improved performance. This seems to be the gold standard for pending event lists. Detailed pseudo-code is also provided.

On the Accuracy of National Intelligence Estimates

Smith, Abbot E. "On the Accuracy of National Intelligence Estimates." *Central Intelligence Agency*. Central Intelligence Agency, 04 Aug. 2011. Web. 31 Mar. 2014. <https://www.cia.gov/library/center-for-the-study-of-intelligence/kent-csi/vol13no4/html/v13i4a04p_0001.htm>.

This is a declassified CIA resources which details intelligence gathering practices and the accuracy of that intelligence. It talks about what percentages are associated with general probability terms. For example, the CIA considers intelligence it labels “probable” or “likely” to have at least a 60% chance of being accurate. There is also a lot of discussion regarding the difficulty in assessing accuracy of intelligence due to the difficulties in verifying tips. Other actions may have distributed enemy operations or movements from occurring that otherwise would have proved intelligence accurate. The bottom line from this article is that intelligence accuracy is nearly impossible to assess.

Unmanned Aircraft Systems Sensors

Weatherington, Dyke D. *Unmanned Aircraft Systems Sensors*. Tech. Washington, DC: OSD UAV Planning Task Force OUS(AT&L) Defense Systems Air Warfare, 2005. Print.<www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA472380>.

This OUSD(AT&L)/Defense Systems Air Warfare paper outlines the different types of UAVs used in IRAQI FREEDOM as well as the different kinds of sensors that were used on the UAVs. UAVs were broken into four categories: Small, Tactical, Operational, and Theater. These categories were driven by how large of a payload the UAV could carry. Within each category, the UAVs were ordered by their endurance. Small UAVs typically carry 1-6 lbs of payload and are able to stay on station anywhere from 40 minutes to 15 hours. Tactical UAVs typically carry 50-200 lbs and can stay on station from 5 hours to 11+ hours. The paper also covered the typical sensor package for each type of UAV.

Using On-line Simulation in UAV Path Planning

Karmrani, Farzad. *Using On-line Simulation in UAV Path Planning*. Thesis. Kungliga Tekniska H Stockholm, 2007. Stockholm Sweden: KTH Computer Science and Communication, 2007.

This thesis is actually a collection of three papers on UAVs. A common theme in the papers is how UAVs may use a priori information, or intelligence, to improve search algorithms and target location. The papers also discusses how UAV paths follow along road networks. It notes that the paths cannot be predetermined; those paths must be able to adjust as the mission develops. This mission may be “military surveillance, search & rescue missions,fire detection and disaster operations,” all of them require dynamic flexibility.

This paper gives a history of UAV usage, noting that the reconnaissance missions like the ones in this project did not become an application for the UAV until the 1950s. The earliest unmanned flying craft appeared in 1849 as pilotless balloons.

UAV path planning is the primary focus of this paper, meaning planning the path exactly before the mission begins. This is done with a simulation based method that varies the predictability of the target and the field of view. The third paper in the set describes the implementation of the road network in detail and provides some pseudo code for reference.

A Trends Analysis of Image Processing in Unmanned Aerial Vehicle

Lee, Jae-Neung, and Keun-Chang Kwak. "A Trends Analysis of Image Processing in Unmanned Aerial Vehicle." *International Journal of Computer, Information Science and Engineering* 8.2 (2014): n. pag.

This paper presents an overview of several image processing trends with respect to techniques and capabilities. Improvements in computing power, unmanned vehicle size, and image processing algorithms have resulted in the capability for unmanned vehicles to automatically identify targets. This image processing can be computed onboard the vehicle itself as well as on desktop class machines. These advancements and low hardware requirements have facilitated image processing in real time in some cases. This suggests that surveillance imagery analysis has matured to a large degree and image analysis workload is mainly limited by the human factor, rather than computational time limits of hardware or algorithms.

Analyzing Probabilistically Constrained Optimism

Lees, M., B. Logan, and G. Theodoropoulos. "Analysing Probabilistically Constrained Optimism." *Concurrency And Computation: Practice And Experience* 21.11 (2009): 1467-1482. *Inspec*.

The Decision-Theoretic Read Delay (DTRD) algorithm is a hybrid alternative to strictly conservative or optimistic synchronization algorithms. The DTRD algorithm determines an optimal delay for events to maximize the rate of time progression in local processes. This paper compares the performance of DTRD algorithms and Time Warp algorithms against varying assumptions of input processes. In general, DTRD algorithms can outperform Time Warp algorithms. However, the analysis finds that as the variance of inter-arrival times increases, the DTRD performance approaches Time Warp performance. DTRD achieves performance gains by exploiting predictability of arrival processes, which becomes more difficult as the variance increases. In the general case, the overhead of the DTRD algorithm is offset by the performance gains.

Intelligence Gathering Post-9/11

Loftus, Elizabeth F. "Intelligence Gathering Post-9/11." *American Psychologist* 66.6 (2011): 532-41.

The author outlines the issues with intelligence based on interviews. Mainly the article is focused on the fact that although a great deal of intelligence collection comes from interviews or interrogations, this information can be faulty for many reasons. Memory distortion, false confessions, and intentional deception are all reasons that intelligence proves to be false. A few specific anecdotes are related in the article to reinforce the author's thesis. No real measures of intelligence inaccuracy are provided, just generalities about the prevalence of the issue. This real-world phenomenon could be replicated by including a large number of intel tips that prove to be inaccurate or completely false.

Architecture

The overall queue structure will be separated into logical processes to facilitate parallelization, as shown in Figure 4. In LP1, HUMINT and CAOC are combined into one LP because they are sequential processes that would not benefit much, if at all, from parallelization. The same is true for IMINT and Pentagon in LP2. On the other extreme, the drone processes get maximal benefit from parallelization since they are all concurrent. Although the drones and targets all work in the same map, they can be treated as separate LPs because targets and drone FOVs can be collocated on a single node. Each drone LP includes a drone, the target it is tracking, and an instance of the map. Note that the three drones shown in Figure 4 are notional and we can have arbitrarily many subject to optimizing the load matching. Lastly, we have the LP0 the controller process which also holds the future event list.

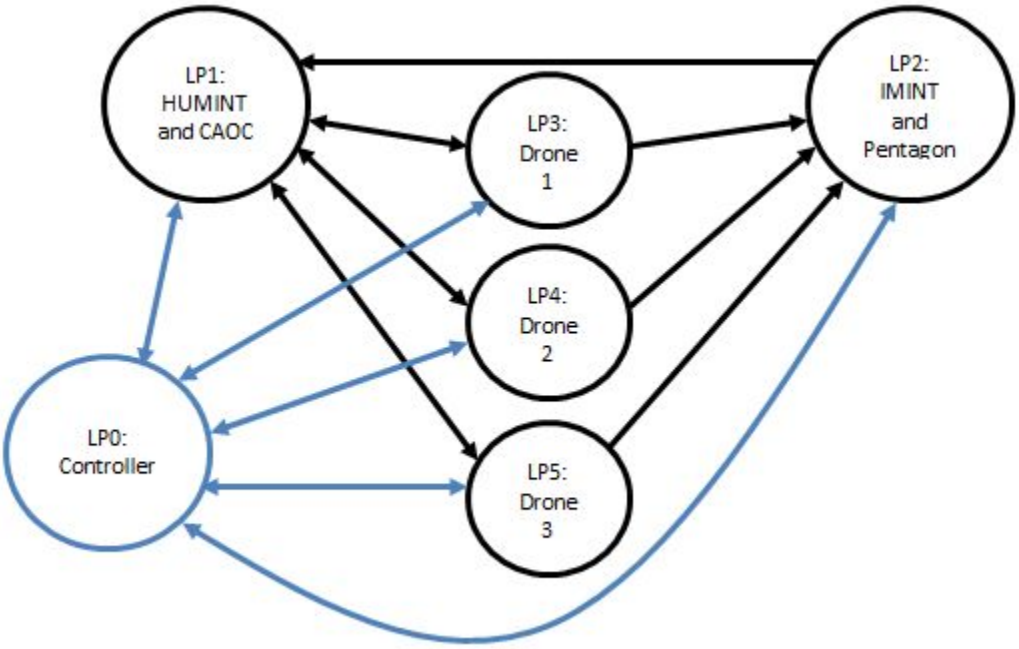

Figure 4: Logical Process Architecture

Major Logical Processes

LP0

Controller
Future Event List: Ladder Queue
Cuts: Mattern Algorithm
Rollback: Anti-messages

LP1

HUMINT: Generate Target Assignments, collect statistics
CAOC: Assign Priority based on Intelligence Value for Target Assignments

LP3-LP*n*

Map: Create map instance
Target Movement: Manage target movement, map exits/entries
Drone Movement: Manage FOV movement, refueling, maintenance
Target Acquisition: Determine when the target has been acquired
Target Tracking: Determine length of time target has been tracked
Drone Process Completion: Determine when drone has processed Target Assignment

LP2

IMINT: Determine if Target Assignment is complete based on % goal time tracked
Pentagon: Collect statistics from completed Target Assignments

Parallelization Approach

The simulation will use a Time Warp algorithm to manage simulation time. Specifically, global virtual time will be calculated asynchronously using the Mattern algorithm. The structure of the logical processes and the drone LPs in particular, are designed to maximize look-ahead time. A conservative algorithm, such as deadlock detection and recovery, is an alternative method if Time Warp is found to be a less desirable approach for this simulation.

Future Event List

The future event list will be structured as a ladder queue. Individual queues within each Logical Process (LP) will be constructed as sorted linked list. The distributions for the individual LPs are small enough that the linked list performance will be similar to or better than other queuing structures. However, the future event list is expected to become quite long and requires a structure like the ladder queue that can efficiently manage large queues.

Data Collection

There are inherent difficulties with collecting data for this simulation. As such, we will rely on historical and published data to inform our simulation.

Assignment, Prioritization, and Analysis

To estimate the rate at which target assignments are generated, the time it takes to prioritize an assignment, and the time it takes to analyze targeting video, we derived theoretical distributions from the limited publicly available, unclassified data. Additionally, we leveraged professional knowledge to determine feasible parameters for these distributions. These estimates are used as input data for the HUMINT and CAOC processes in Figure 3.

Target Movement

To estimate the distribution of pedestrian target speeds, we will reuse our data collection from the evacuation project. Posted speed limits in urban areas will be used to estimate the average and standard deviation of vehicle target speeds.

Drone Characteristics

Technical specifications of most drone platforms are surprisingly available on the internet. These numbers will be used to determine average time in between maintenance activities, time in the air, sensor capabilities, and other pertinent drone characteristics.

City Characteristics

We are compiling a simulated urban environment based on common characteristics of high profile cities. Example cities studied include Kabul, Los Angeles, Atlanta, and Tehran. Most cities exhibit large regions of grid like organization with some areas of complex structure.

Simulator Overview

All info on the inner workings of the simulation should be here

Major Classes

Target Assignment

Intelligence Value: Real number between 0 and 100 representing subjective value of target
Priority: Real number between -100 and 100 representing priority queue order

Target

Generated when corresponding Target Assignment is at head of priority queue and begins service by drone process
Type: Vehicle or Pedestrian
Stealth: Multiplier to adjust base probability of acquisition and loss for any node
Speed: Multiplier to adjust time it takes to complete any node

Goal Tracking Time: Goal time length of visual contact between drone and target

Drone

Field-of-view (FOV) Size: Nodes that a drone can search/track at once (defined by neighbors)
Fidelity: Multiplier to adjust base probability of acquisition and loss for any node and target
Speed: Multiplier to adjust time it takes to complete any node
Max Dwell Time: Time available before drone must refuel

Map Node

Type: Intersection or Street
Base Probability of Acquisition: Base probability drone acquires target
Base Probability of Loss: Base probability drone loses target
Length: Time it takes the fastest mover to traverse

List of Python Code Files to Date

CAOC.py
Drone.py
DroneSim1.py
Initialization of objects and parallelization
DroneSimController.py
DroneType1.py
HMINT.py
IMINT.py
LadderQueue.py
Structure for Enqueue, Dequeue, and Rung Creation functions built
LogicalProcess.py
Map.py
Generates the map nodes
Message.py
Target.py
TargetPriorityQueue.py

Verification and Validation

What did we verify and how

Testing Methodology

This simulation

Output Analysis and Results

Significant versus insignificant factors

Conclusions

Nothing yet

Areas for Further Study

Other areas we could have investigated

Source Code

The source code developed for this project is attached in the files listed below, including main model files and analysis and testing files:

Bibliography

Brown, Randy. "Calendar Queues: A Fast O(1) Priority Queue Implementation for the Simulation Event Set Problem." *Communications of the ACM* 31.10 (1988): 1220-227.

"Design of UAV Systems." www.southampton.ac.uk/~jps7/Aircraft/Design/Resources/UAV/Resources/ASE261.11.Payload.ppt. N.p., n.d. 24 Mar. 2014.

Jones, Douglas W. "An Empirical Comparison of Priority-queue and Event-set Implementations." *Communications of the ACM* 29.4 (1986): 300-11.

Karmrani, Farzad. *Using On-line Simulation in UAV Path Planning*. Thesis. Kungliga Tekniska H Stockholm, 2007. Stockholm Sweden: KTH Computer Science and Communication, 2007.

Lee, Jae-Neung, and Keun-Chang Kwak. "A Trends Analysis of Image Processing in Unmanned Aerial Vehicle." *International Journal of Computer, Information Science and Engineering* 8.2 (2014): n. pag.

Lees, M., B. Logan, and G. Theodoropoulos. "Analysing Probabilistically Constrained Optimism." *Concurrency And Computation: Practice And Experience* 21.11 (2009): 1467-1482. *Inspec*. Web. 3 Apr. 2014.

Loftus, Elizabeth F. "Intelligence Gathering Post-9/11." *American Psychologist* 66.6 (2011): 532-41.

Sleator, Daniel Dominic, and Robert Endre Tarjan. "Self-adjusting Binary Search Trees."*Journal of the ACM* 32.3 (1985): 652-86.

Smith, Abbot E. "On the Accuracy of National Intelligence Estimates." *Central Intelligence Agency*. Central Intelligence Agency, 04 Aug. 2011. Web. 31 Mar. 2014. <https://www.cia.gov/library/center-for-the-study-of-intelligence/kent-csi/vol13no4/html/v13i4a04p_0001.htm>.

Tang, Wai Teng, Rick Siow Mong Goh, and Ian Li-Jin Thng. "Ladder Queue." *ACM Transactions on Modeling and Computer Simulation* 15.3 (2005): 175-204.

Weatherington, Dyke D. *Unmanned Aircraft Systems Sensors*. Tech. Washington, DC: OSD UAV Planning Task Force OUS(AT&L) Defense Systems Air Warfare, 2005. <www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA472380>.

Appendices

Appendix A: Complete Literature Survey

Appendix B: Data Collection on Pedestrian Walking Speeds

Appendix C: Full Results

Appendix d: Interface Descriptions