“Operator-defined SUAV Classification Tree Algorithms for Waypoint Prediction and Increased Use of Autonomy”

Phil Root

16 May 2012

Abstract

The growing divide between the potential of fully automated teams of Unmanned Aerial Vehicles (UAVs) and the very limited autonomy seen in operational use deserves attention. This paper proposes a limited step to realize increased autonomy use in operational settings by applying decision tree learning algorithms to UAV operator's flight patterns. These algorithms learn and then propose UAV trajectories based on the individual preferences and patterns of each operator. This approach is distinct from previous methods in that it seeks to tailor autonomy suitable to each operator rather than seek a panacea for all. We show that C4.5 classification trees have numerous advantages over competing algorithms, and limited experimental results have provided great promise for future research.

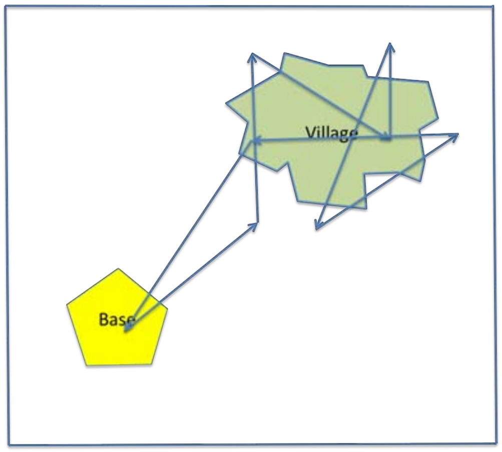
1 Introduction

Small Unmanned Aerial Vehicles (SUAV) hold great promise for those operators in austere environments. Already used ubiquitously throughout the US military, SUAVs are increasing used by domestic law enforcement agencies. Soldiers and Marines rely on the Raven SUAV pictures above, for example, to perform a variety of missions to include surveillance of bases, reconnaissance of convoy routes, and aerial patrolling in a variety of combat locations. SUAVs differ from their larger brethren in that they can be assembled and launched without any tools or runways. Soldiers have launched these aircraft from rooftops, remote bases, and hillsides. Their use is limited only by their batteries and the operator's imagination.

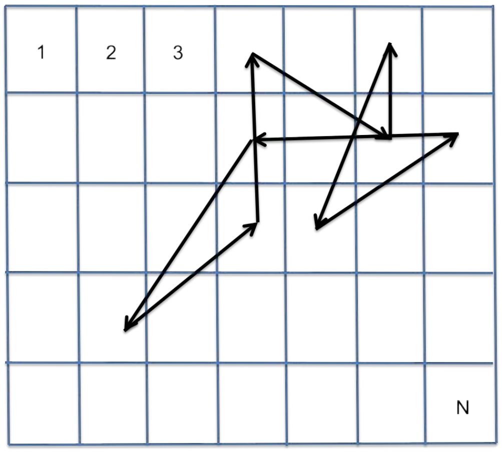
1.1 Current Use of Small UAVs

SUAVs are commonly used by small groups of Soldiers and Marines to conduct reconnaissance in advance of some military operation. Figure 1a depicts the notional reconnaissance of a village by a SUAV launched from a nearby base. The SUAV operator chooses this flight trajectory for some set of reasons based on his/her knowledge of the terrain and the target. The challenge for engineers and researchers is how to infer this rationale from a flight trajectory depicted in figure 1b that is devoid of this informaiton.

There exists a spate of research that discusses optimal search patterns for such SUAV intent on patrolling an area such as [3] or servicing targets that arrive via a given distribution such as [4]. Their work greatly



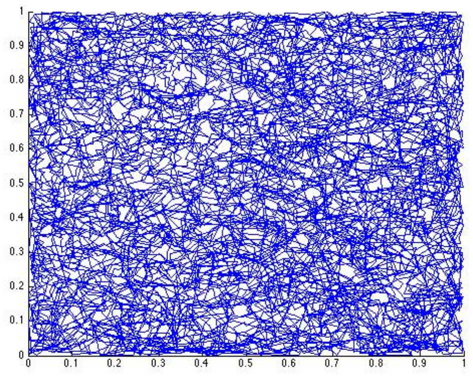
(a) Notional reconnaissance of a village



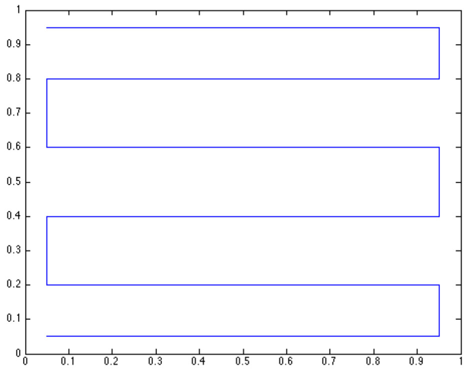
(b) UAV trajectory as seen by flight planner

Figure 1: Current operational SUAV use

advances the technical capability of collaborative SUAVs to achieve a common mission, but in practice we see virtually no such use of autonomy. It is perhaps most compelling to demonstrate this dichotomy graphically. Figure 2a depicts the state of the art persistent patrolling technique from [3] that iteratively links points via a Traveling Salesman Problem (TSP) solver where the points are selected according to some distribution. Figure 2b, on the other hand, depicts a typical "lawnmower" search pattern as used in practice. Moreover, we typically see the best SUAV operators capable of controlling a single SUAV aircraft at a time whereas the more advanced search and patrolling algorithms assume that a single operator can control multiple aircraft.



(a) Persistent patrolling algorithm for multiple SUAVs



(b) Typical patrolling technique used in practice

Figure 2: Comparison of theoretical versus actual patrolling algorithms used in practice

This difference between theoretical and actual SUAV operation is well-known and studied as in [2]. What has generally not been acknowledged is that every operator is different with varying degrees of experience, education, risk tolerance, and willingness to trust autonomy. Any attempt to improve the autonomy for all users could benefit from the realization that not all methods will be universally adopted. Hence this work proposes a limited first step toward increased autonomy by first learning a particular operator's preferences and patterns, and then proposing sequential waypoints based on the learned model.

We summarize the intent of this research as to answer the following central question.

Central problem: What is the best method to learn an operator's patterns for SUAV reconnaissance so as to propose future waypoints with a high probability of user acceptance?

1.2 Rationale for Classification Tree Implementation

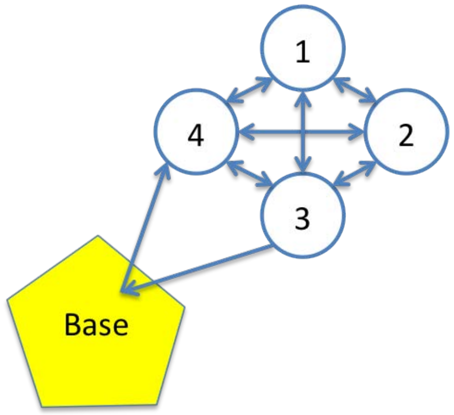
There are several proposed advantages to a classification tree learning algorithm compared to other alternatives for this particular application. Any algorithm that addresses the central question for this research must have two key qualities: first it must be a sufficiently powerful algorithm so as to describe a wide variety of operator patterns, and secondly, the algorithm must lead to increased rates of user adoption. We submit that C4.5 classification trees are well-suited to address both of these requirements as we will discuss.

Clearly classification schemes are sufficiently powerful to address the problem as posed. Most operators will only provide on the order of hundreds of training data examples, and these examples will have dimensions on the order of tens and perhaps hundreds. C4.5 classification trees are more than sufficiently powerful for this class of problem.

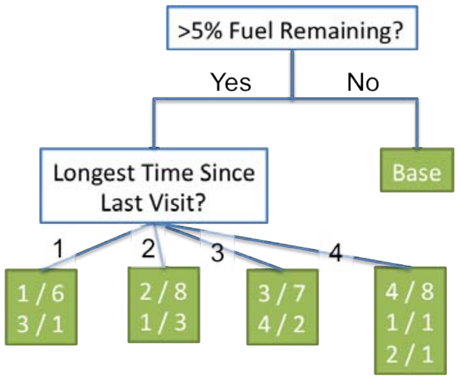
One key advantage of the family of decision tree algorithms is that their resulting model is much more *transparent* and *interpretable* that other machine learning algorithms. The graphical decision tree is simple to understand even without a background in the material, and the decision tree provides rationale as to why the algorithm proposed a particular waypoint. Breiman et al outlined the utility for Classification And Regression Trees (CART) in [1]. Although C4.5 classification trees are implemented differently than CART, they enjoy the same benefits particularly when applied to this problem. Breiman begins by providing a medical example to motivate his work where a binary classification tree applied to nineteen variables predicted high risk patients with only three splits. The resulting classification tree is hence very easy to understand and implement particularly because is represents the logic that most medical professionals follow when making decisions. Similarly, in section 3 we will show that applying the C4.5 algorithm to a "lawnmower" search trajectory with a forty-two variables requires only four nodes or splits to accurately predict all future points, and the graphical representation of this classification tree is very simple to grasp. This paper will also explore the utility of other learning algorithms such as Support Vector Machine (SVM) algorithms. SVM also provides graphical interpretations for their solutions as they seek to graphically maximize the margin between categorical data. But a plot of the SVM separation oracle is really only interpretable in two dimensions, and the problem at hand will have a much higher dimensionality where SVM graphical methods are insufficient.

Perhaps one of the greatest advantages of our proposed modifications to CART and C4.5 is that our algorithm does not propose the unique best solution, but rather selects the answer probabilistically based on the distribution of labels in the optimal leaf, as we will show. Figure 3a depicts a notional reconnaissance mission where the SUAV launches from a base to a nearby set of waypoints. Perhaps the operator directs the SUAV to visit the point that has been visited least recently; stated differently visit the point with the largest amount of time since the last visit. But the operator doesn't follow this pattern with probability 1. Consider the far right leaf where point 4 has had the longest time since being visited last. In this case, the operator has visited point 4 eight times, but he/she has also visited point 1 once, and point 2 once. This information about points 1 and 2 is valuable as it reflects the some real information about the operator's mental decision model. While it is feasible to alter many learning algorithms to take advantage of this variance of the outcome, it is very simple for decision trees to both depict this information and take advantage of it.

We claim that the final advantage of decision trees over alternative learning algorithms is the *malleability* of the model. Returning to figure 3, suppose that the operator flies thirty waypoints as shown in figure 3b and then decides that fuel is too low and commands the SUAV to return to base. This one data point to command the SUAV to base reflects only one point of the thirty previous training examples. Hence, this decision to return to base is likely to be misclassified as some other leaf in the tree. It is far simpler to append a new heuristic to the existing decision that reflects this low density decision ("Return to base whenever fuel $< 5\%$") than it would be to alter the output of a SVM or logistical regression model. Thus, users can more easily modify or tailor their learned algorithm to further personalize the resulting recommendations. All this would yield a higher user acceptance rate given the ease of modification and personalization.



(a) Notional reconnaissance mission



(b) Notional decision tree

Figure 3: Notional reconnaissance mission and decision tree

2 Implementation

Our implementation method consisted of creating a Graphical User Interface (GUI) to allow data collection, applying learning algorithms to the collected data, and seeking feedback from users as to the viability of this approach.

2.1 Experimental Setup

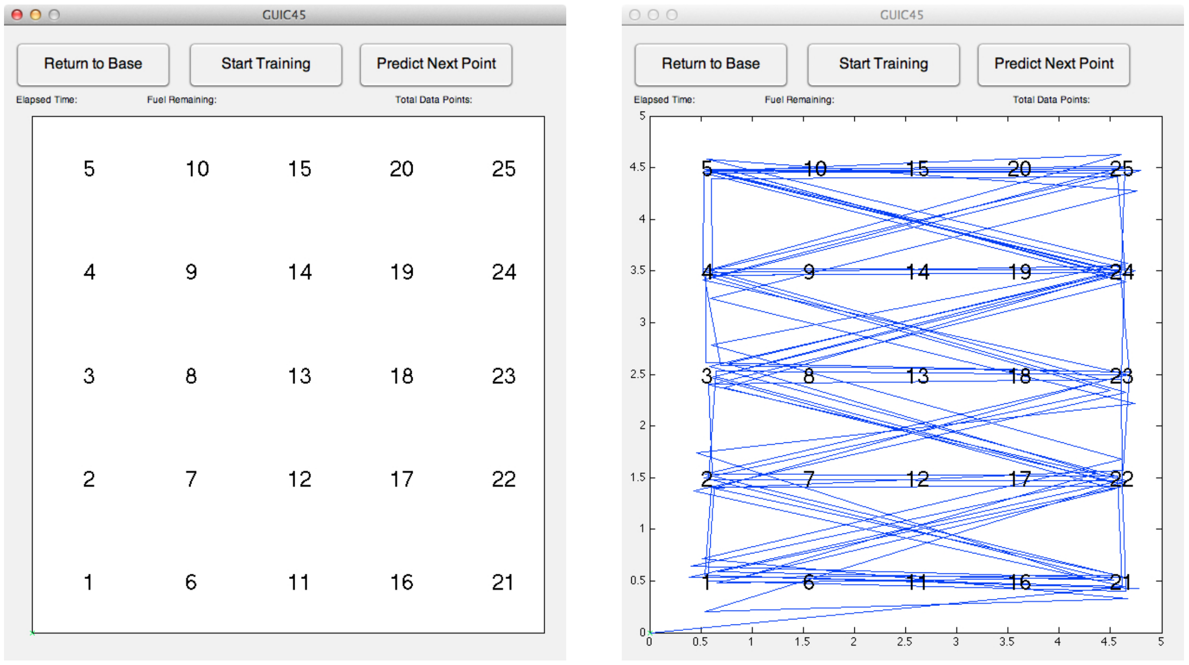
We created a MATLAB GUI that allowed us to gather data quickly and reliably over a wide range of different mission profiles. We discretized a square environment into N points. We chose $N = 25$ arbitrarily, and the GUI allows for increased or decreased discretization. Figure 4a depicts the MATLAB GUI upon initialization. We always "launch" the simulated SUAV from the origin in the lower left corner.

Depressing the "Start Training" button causes the GUI to display a set of crosshairs so that the operator can select points on the map representing SUAV waypoints. Each additional waypoint adds another training example for the learning algorithms to be described below. Each example captures the state vector at the time the operator chose that point. We designed the state vector in an attempt to capture data that would be useful given an unknown mission profile. Stated differently, the learning algorithm has no knowledge *a priori* as to the type of reconnaissance or mission so the state vector must contain sufficient data to amply describe a wide range of operator types. We chose to collect forty-two attributes for each training example as follows.

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 - x_7 \\ x_8 - x_{12} \\ x_{13} - x_{17} \\ x_{18} - x_{42} \end{bmatrix} = \begin{bmatrix} \text{time elapsed} \\ \text{fuel remaining} \\ \text{last } M = 5 \text{ points visited} \\ \text{latitude of last } M = 5 \text{ points} \\ \text{longitude of last } M = 5 \text{ points} \\ \Delta T \text{ for } N = 25 \text{ points} \end{bmatrix}$$

Variable x_1 and x_2 are the time elapsed and fuel remaining, respectively. Variables x_3 to x_7 are the last $M = 5$ discretized waypoints closest to the selected coordinate. Similarly, we collected the latitude and longitude of the last M coordinates visited. Finally, variables x_{18} to x_{42} are the length of time since that point was last visited, ΔT. As stated previously, we chose $N = 25$ as a balance between prediction fidelity and problem complexity, and we felt that twenty-five points were sufficient for our experimental work. We chose to track the previous $M = 5$ points for similar reasons although we can adjust both N and M should we ever require higher fidelity. In this training mode, the GUI updates the elapsed time, fuel remaining, and number of waypoints after each selection.

We took great care to ensure we accurately determined the time that a simulated SUAV passed by each of the N points. Figure 4b depicts a set of training data, and the path often passes from point 5 to point 25.



(a) GUI upon initialization (b) GUI with training data

Figure 4: GUI upon initialization and after collecting some training data

We modified the Bresenham algorithm to rapidly determine which of the N points we intersected passing from point 5 to point 25, and updated their resulting ΔT. This step was nontrivial but vital to the accuracy of the remainder of the work.

The "Return to Base" button commands the simulated aircraft to return to the origin but does not stop the training session. The operator stops submitting training examples at some point, and depresses the "Predict Next Point" button. This button submits the previously collected training data and the current state to various learning algorithms to predict the next point using the following decision tree algorithms.

2.2 Decision Tree Algorithms

We first implemented a binary classification tree in MATLAB due to the simplicity to do so as an initial step. The following code excerpt implemented the actual algorithm.

```
1 x=[h.t,h.fuel,h.IntLast,h.LatLast,h.LongLast,h.deltaT];
2 y=[h.label(1:end-1)]
3 DT = classregtree(x(1:end-1,:), y, 'method','classification', ...
4     'categorical', [3:2+h.M], 'prune','on','names',h.names);
5 view(DT)
6 h.prednext=eval(DT,x(end,:));
```

Line 1 concatenates the state variables to form the example data, and line 2 contains the training labels. It is of note that the last row of `x` contains the state vector used for predicting the next point, so when we call the `classregtree` command in line 3, we reference all but the last line of `x`. We define the last M points as "categorical" data as compared with "numeric" data as part of the `classregtree` command call. Line 5 allows us to see the resulting model, and line 6 evaluates the resulting `DT` model given the current state in the last row of `x` to predict the next point.

We then implemented a C4.5 classification tree algorithm using Weka APIs available to MATLAB as shown in the following code.

```
1 train = matlab2weka('route-train',h.names,train,classindex);
```

```
2 model = trainWekaClassifier(train, 'trees.J48')
3 test = matlab2weka('route-train',h.names,test,classindex);
4 [predicted, class] = wekaClassify(test,model);
5 pikr = rand(1); count=1;
6 while pikr > sum(class(end,1:count))
7     count = count +1;
8 end
9 next = char(test.classAttribute.value(count-1));
```

The `matlab2weka` function uses the Weka APIs to create a JAVA object for the training data in line 1, and line 2 forms a C4.5 classification model using this training data. The `test` variable contains the current state vector for use in predicting the next point in line 3. We then tailored the `wekaClassify` function to output the probability distribution of the labels within the selected leaf rather than simply the classification label with the maximum probability in line 4. The code in lines 5 to 8 then picks a random variable, `pikr`, and we select the label associated with this value within the probability distribution. Line 9 determines which label to assign based on this stochastic process.

2.3 Support Vector Machine Algorithm

In addition to the binary and C4.5 classification algorithms outlines above, we implemented a Support Vector Machine (SVM) algorithm for purposes of comparison. More specifically, we implemented a "one-vs-all" classification scheme using the code highlights below.

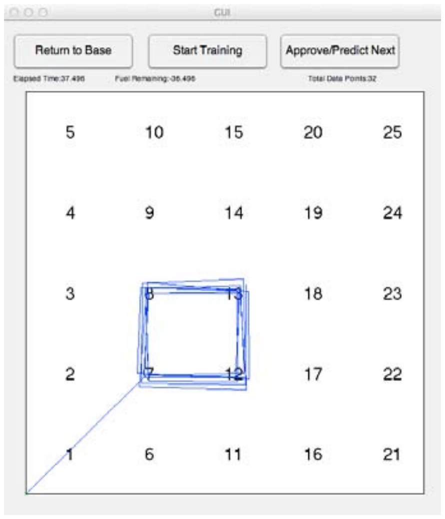
```
1  klabels = unique(data.label(1:end-1));
2  for i = 1:length(klabels)
3      k=klabels(i);
4      tmplabel = y;
5      tmplabel(tmplabel==k)=1;
6      tmplabel(tmplabel~=1)=0;
7      svmStruct = svmtrain(train,tmplabel);
8      yout = [yout svmeval(x(end,:),svmStruct)];
9  end
10 out = klabels(find(yout==max(yout)));
```

For each unique label in the training data, the SVM algorithm alternatively converts the training labels to binary data, and determines the optimal support vectors and weights in lines 1 to 7. The `svmeval` command in line 8 evaluates the classifier as $c = \sum_i \lambda_i k(s_i, x) + b_i$ where the SVM model contains the optimal weights λ_i, bias b_i, and support vectors s_i for the selected kernel $k(\cdot,\cdot)$ and training data x. Line 10 then chooses the classifier with the highest real value.

3 Results and Feedback

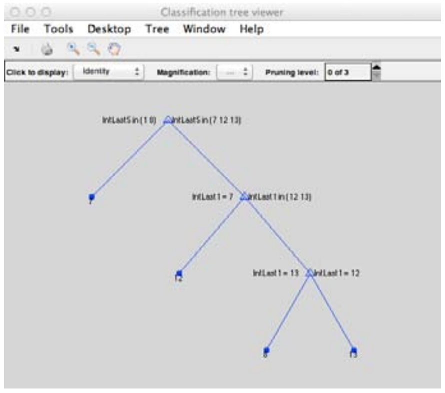
We found the MATLAB GUI a highly effective tool to gather data and evaluate all the classification schemes listed above. We used the GUI to input several common patrolling techniques such as orbiting around a point and a "lawnmower" pattern. The orbit pattern depicted in figure 5a just follows a sequence of four points. The resulting binary classification tree depicted in figure 5b required thirty-two training examples before it accurately predicted the next point in the pattern. The C4.5 classification scheme required only eleven training examples before it converged to the correct pattern. This is not an accurate comparison because the binary classification tree was deterministic in that it always chose the label that constitutes the majority of the leaf. We altered the C4.5 algorithm to randomly select the label from the probability distribution of the leaf, so the C4.5 algorithm only "converges" given this repeated orbit pattern after all the leaves are pure.

The lawnmower pattern depicted in figure 6a is more complex. It follows the pattern {5, 25, 24, 4, 3, 23, 22, 2, 1, 21, 22, 2, 3, 23, 24, 4, 5}. The binary classification tree required ninety training examples to converge to the model shown in figure 6b. Given a smaller training set, the binary classification algorithm



Elapsed Time:37.496 Fuel Remaining:-36.496 Total Data Points:32

(a) Orbit patrol trajectory

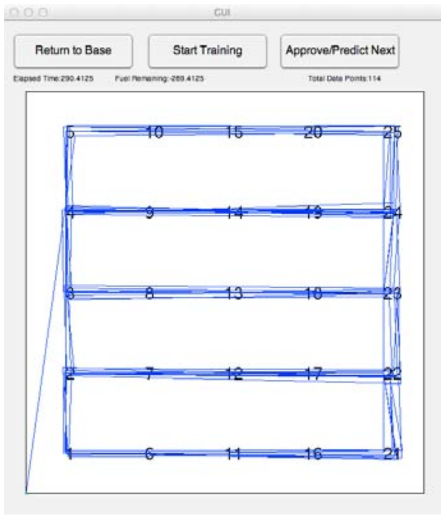


Click to display: Identity Magnification: — Pruning level: 0 of 3

(b) Orbit classification tree

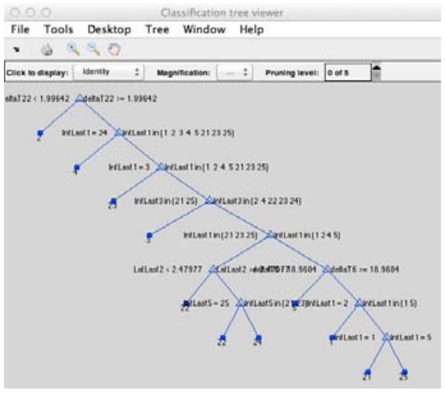
Figure 5: Orbit trajectory and classification tree

could correctly predict several elements of the sequence, but not the entire sequence. The modified C4.5 algorithm required only fifty-eight training examples to converge to a static and predictive model.



Elapsed Time:290.4125 Fuel Remaining:-289.4125 Total Data Points:114

(a) Orbit patrol trajectory



Click to display: Identity Magnification: — Pruning level: 0 of 5

(b) Orbit classification tree

Figure 6: Orbit trajectory and classification tree

In all cases, the decision tree algorithms converged to accurate predictions with a limited number of training examples. Importantly, at no time did the classification schemes suggest an errant point such as a point that had yet to be visited. While this may seem like a trivial achievement, it is key to establish operators' trust.

Interestingly, the SVM algorithm never predicted either of these two test patterns reliably. One possible explanation deals with missing values in the training data. Each training example contains the ΔT for each point, and the orbit pattern never visits some points such as point 5, for example. We initialize all ΔT to -1000, but this initialization could cause numeric issues. Consider the "lawnmower" pattern which visits each point in two “directions”. The pattern visits point 1, for example, both following point 2 and following point 21 depending on the location within the sequence. This is trivial for classification trees to address, but SVM seeks a separation hyperplane that separates all point 1 labels from all others, but point 1 labels have two distinct states when they occur. Trying linear, quadratic, and higher order polynomial kernels similarly failed to achieve satisfactory results. It is possible that these two distinct states are too distant in the 42-dimensional state space to separate from other points.

It is challenging to draw any objective conclusions about the predictive nature of the classification trees in general because we don’t know what the “correct” waypoint would be in all cases. These simple test cases confirm that indeed the classification trees can replicate an operator’s decision process, but we hesitate to overextend this conclusion. Instead we had the opportunity to present this research to the US Army Small UAV Program Office in Huntsville, Alabama. As representatives of the users, they were able to provide us feedback as to the veracity of our claim. They were very supportive of this work as a means to capture operator preferences and patterns, and they validated that this approach was the best they had seen to achieve this end.

4 Conclusions

Our limited experiments make it clear that this approach in general performs well as a means to learn operator preferences and present them with the results so as to increase user acceptance rates.

We find that the modified C4.5 classification tree has several advantages over the binary classification tree. First, it converges more quickly to a stable model in our trials. Secondly, we were able to easily modify the algorithm such that it selects the label stochastically based on the probability distribution of the selected leaf which more closely mirrors the operator’s mental process.

We also find that either classification tree performs better than SVM in predicting the sample patterns. We struggle to believe that SVM has no reliable predictive nature when applied to this problem, although it may indeed be poorly conditioned to address the problem. We imagine that our SVM implementation is faulty in some regard leading to these results. Regardless, one of the key advantages of the classification trees is that they output a model that can be displayed graphically to users for their interpretation and acceptance. An SVM model is much more challenging to present in higher dimensions. Thus we posit that users will more readily adopt a classification tree scheme compared to SVM due simply to this difference.

5 Future Work

We believe that this work has potential across many fields where operators could benefit from increased autonomy but are reticent to do so due to potential safety issues. Consider the Air Traffic Control (ATC) problem where ATC controllers must route aircraft through a busy corridor. This problem has been well-studied as in [5], but in general proposed solutions fail to account for individual differences between ATC controllers in the way that they prefer to route traffic and deal with weather factors. This work could easily be extended to this problem.

References

[1] Leo Breiman, Jerome H. Friedman, Richard Al Olshen, and Charles J. Stone. *Classification and Regression Trees*. Chapman & Hall, 1984.

[2] M.L. Cummings, S. Bruni, S. Mercier, and P.J. Mitchell. Automation architecture for single operator, multiple uav command and control. *The International C2 Journal*, 1(2):1–24, 2007.

[3] Vu Anh Huynh, J.J. Enright, and E. Frazzoli. Persistent patrol with limited-range on-board sensors. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 7661–7668, December 2010.

[4] Marco Pavone, Ketan Savla, and Emilio Frazzoli. Sharing the load. *Robotics & Automation Magazine*, 16(2):52–61, 2009.

[5] Thomas Prevot, Paul Lee, Nancy Smith, and Everett Palmer. Atc technologies for controller-managed and autonomous flight operations. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005.