CS 6780 : Final Project Report
Learning Written Gesture Sequences for Robot Control

Danelle Shah (dcs45), Joe Schneider (jrs465)

***Abstract*—In this work, we introduce an algorithm for learning written multi-stroke gesture sequences, with motivation towards natural robot control. We briefly survey the current research in the field of gesture recognition and describe the overall approach we have taken. We describe the feature sets used and discuss several supervised learning techniques for the classification problem. We ultimately use a probabilistic multiclass classifier to label multi-stroke gestures, single strokes, and stroke transitions, which achieved average accuracies of 97.5%, 83.4% and 75.4%, respectively. Using combinations of these probabilistic classifiers, we introduce a heuristic tree search algorithm for reconstructing multi-stroke gesture sequences. For sequences of up to ten strokes, our algorithm achieved greater than 88% average accuracy, which is comparable to existing literature.**

I. Introduction and Motivation

Our goal in this project is to implement a robust, intuitive mechanism for the control of a fleet of ground robots. Cornell is competing in the MAGIC 2010 competition in Australia in November 2010, and we expect that this interface will play a part in that implementation. We use machine learning techniques to identify written gestures, as well as heuristic tree search algorithm in order to identify those gestures in a sequence. Much research has been done on both character and word recognition, and we leveraged that knowledge base to implement a series of gestures used in concert. We have built a generic framework that can recognize complex commands (but not yet map them to actions) in series, and in future work we expect to implement the ultimate goal of a feature rich, intuitive, and robust interface for robotic control.

II. Overview of Approach

Many modes of communication have been devised to communicate human commands to machines, specifically robots. Chief among them is the keyboard and mouse due to their ability to handle the breadth of commands. The limitation of both the keyboard and mouse is their artificial nature. Humans are more skilled at communication with other humans using other methods, such as diagrams and gestures. The difficulty with communicating with machines via diagrams and gestures is the inability to accurately distinguish thier meanings, especially when humans never perform them *exactly* the same twice. Another difficulty is encountered when gestures are inherently vastly different from one instance to the next, as in the case of path specification.

While research has been done on commanding robots using written gesture commands, the focus has been exclusively on navigation commands such as paths and goal points [1], [2]. Additionally, [2] use at most three gestures for the entirety of their interaction with their robots. We wish to achieve a more robust and capable system that can communicate high level commands intuitively to semi-autonomous robots, as well as low-level explicit commands when appropriate. We plan on achieving comparable results to [2] initially, but with a more robust framework. One crutch both [1], [2] used is hard coding gestures, such as X's and paths. They could not handle more than single strokes, with the one exception of an X, which was defined as two single stroke gestures that intersect. [1] were successfully able to dictate formation commands to a small group of robots using this interface on a PDA. They used a library of 4 gesture commands (pre-recorded route shapes), with one additional command to specify the formation configuration. [1] had no capability to utilize combinations of characters, and depended entirely on single character recognition.

A secondary goal is to design a system capable of recognizing new gestures with a minimal number of examples without sacrificing quality. Yang and Xu [3] used Hidden Markov Models (HMM) to achieve 99.78% accuracy with 100 samples per gesture, and 91.65% accuracy with *just 10 samples per gesture!* The gestures they used were the written form of the numbers 1–9.

Several early commercial character recognizers including Grafitti (palm) [13] and ANPR (Apple) [14] that came out in the 90's were quite successful, but limited, and have since been discontinued. CalliGrapher (by PhatWare) that remains somewhat successful (their iPhone App WritePad has 2.5 out of 5 stars with over 3000 reviews) [15]. There are many more recent commercially available software, but text recognition remains largely excluded from everyday interfaces between machines and humans. The primary algorithms used for gesture recognition in the literature are HMMs [1], [3], [4], [6], [8], [9], [11], [12], neural networks [5], [10], and sometimes both [7]. As we will discuss in detail later on, these techniques rely on some simplifications of the problem of gesture and word recognition that we wish to overcome. While the problems successfully addressed using HMMs and neural networks are in many ways similar to the problem we address here, the added complication of multi-stroke gestures and ambiguous gesture starts and ends prompted us to choose a different approach. The algorithm we present here is a combination of Sparse Multinomial Logistic Regression (SMLR, pronounced "smaller") and heuristic tree searching.

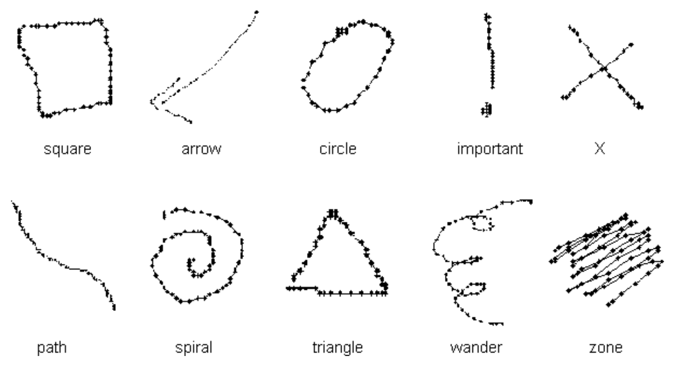

Fig. 1. Example gestures collected. A total of 80 data points per gesture were used for training and 80 for testing.

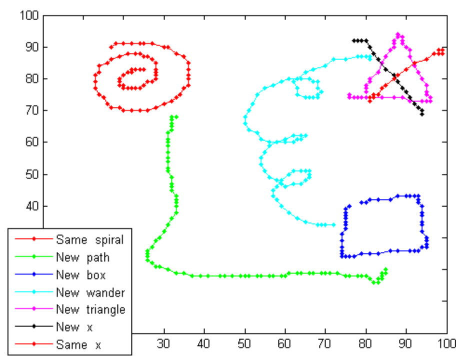

Fig. 2. An example gesture sequence with seven strokes. Classifying gestures first requires that individual strokes are combined appropriately.

III. Data Set and Features

In our work we used ten written gestures from two users (shown in Figure 1), where each is assumed to represent a unique command, with no relation to the other gestures in the sentence (future work will address inter-gesture correlations). Most of these gestures are composed of a variable number of pen strokes, the number and style of which are not rigidly enforced.

During training it is possible to isolate exactly which strokes belong to which gesture, but in the application of a gesture command interface the aggregation of strokes into gestures becomes part of the problem itself. We cannot directly observe the start and end of each gesture; we can only infer separation between gestures from the shape of their component pen strokes.

As an analogy of the problem, imagine that each pen stroke is a letter, each gesture is a word, and a series of gestures makes a sentence. To illustrate the combinatorial nature of aggregating pen strokes into valid gestures, consider a sentence without spaces, such as “ISHOTSOUP”. Just as “IS HOT SOUP” and “I SHOT SO UP” are both valid segmentations of the sentence, the written gestures we chose can have similar pen strokes among neighboring gestures in sequence. To address this we learn three different classifiers associated with gestures, strokes, and stroke transitions, which are described in the next sections. An example of a sequence can be seen in Figure 2. Each color represents a single stroke, with the first stroke always designated with a “same” indicator.

A. *Gesture Features*

As explained in Section II, handwritten gesture recognition has been extensively studied and many feature sets have been developed for similar problems. For our gestures, we have adopted a portion of the g-48 set presented in [17] and [18]. The g-48 set contains 48 features computed from a global gesture trajectory, and are generally well-suited for multiple-stroke gestures (only 42 features from this set were used). Please refer to [17] for more details on this feature set.

In addition to the g-48 feature set, we employed another set of features using histogram of tangents, or “HoT features” (please refer to the midterm report for a detailed description of HoT features). Nine additional features corresponding to initial orientation angles, the amount of time spent drawing each gesture, the total number of strokes, and cw/ccw orientation indicators complete the feature set. In all, 99 features were used for learning gesture type.

B. *Stroke Features*

To identify stroke type (i.e. which gesture type a single stroke belongs to), we used the same features as described in Section III-A (with the exception of number of strokes). Note that some strokes may “obviously” belong to a particular gesture type, especially those that are themselves a full gesture, such as `spirals`. Other strokes, especially straight lines, may easily belong to several gesture types and we should not expect them to be correctly classified reliably. In the results presented in Section V-B, we will see that this is in fact the case.

C. *Stroke Transition Features*

As explained in Section III, learning gesture sequences is a hard problem when the gestures transitions are not known. This requires us to learn *simultaneously* where gesture transitions are and what the gesture types are, resulting in a (worst case) combinatorial search. To help “rule out” unlikely gesture transitions we have defined nine stroke transition features, calculated from consecutive strokes (such as time elapsed between drawing strokes, relative position of strokes, etc.). If both strokes belonged to the same gesture the stroke transition was labeled as that gesture class, and if the strokes belonged to different gestures the stroke transition was labeled as `new gesture`.

Like strokes, we would expect some stroke transitions to be more easily classified than others. For example, an `important` gesture consists of a vertical line and a dot directly below the line. We would expect the consistency of this interstroke relationship to help classify two consecutive `important`-strokes as belonging to the same `important` gesture.

IV. Classification

Once features have been extracted from our data, we are ready to train our classifier. Of course there are many different classifiers available, and they can broadly be separated into two categories: binary classifiers and multiclass classifiers. Binary classifiers include Gaussian Discriminant Analysis [23], Support Vector Machines [24], Relevance Vector Machines [25], and many others. These classifiers tend to be simpler and require less training data, but are best applied to binary classification problems. The problem we are trying to solve would benefit more from a *multiclass* classifier. The most common approach to multiclass classification extends binary classifiers to the multiclass problem such as the "one versus all" (OvA) approach, which learns N binary classifiers (where N is the number of classes). Another option is the "one versus one" (OvO) approach, in which a binary classifier is learned for each pairwise combination of classes (resulting in $N(N-1)/2$ classifiers) [26], [27], [28], [29], [30]. Generalizing binary classifiers to the multiclass problem has been studied extensively [31], [32], [33], [34]. Some other multiclass classification techniques include Softmax [35], Generalized LDA [36], multiple logistic regression [37], nearest neighbor [38], and decision trees [39], [40].

In the next sections, we briefly discuss three classification techniques that were evaluated for the midterm report: Support Vector Machines, Relevance Vector Machines, and K-Nearest Neighbors. In Section IV-D, we describe the final multiclass classification method implemented, called Sparse Multinomial Logistic Regression (SMLR), which was used to obtain all following results. (Please note that the SVM, RVM and KNN classifiers were evaluated using a different data set than SMLR, so classification results can not be directly compared. They are included here only to demonstrate that these methods were examined.)

A. Support Vector Machines

Support Vector Machines (SVMs) [41] have been widely used for classification in many fields, especially for machine learning tasks such as text categorization [42] and pattern recognition [43]. SVMs is a supervised learning methods that constructs a hyperplane in high (or even infinite) dimensional space to be used for classification or regression. It attempts to maximize the "functional margin", i.e. the distance from the hyperplane to the nearest training datapoints. A "soft margin" allows for some misclassification, in instances where classes are not seperable. While SVM is generally a *linear* classifier, it can be extended to find *nonlinear* boundaries using non-linear kernel functions. For the midterm report, we trained a linear pairwise SVM classifier on the data (200 data points from each class were used to train, and 50 to test). The one-versus-one (pairwise) classifier performed the best, with about 99.67% accuracy on average.

B. Relevance Vector Machines

Relevance Vector Machines (RVMs) have identical functional form to SVMs, but use Bayesian inference to provide *probabilistic* classification (which will be useful later when we implement our HMM) [25], [44]. RVMs can derive accurate prediction models with fewer basis functions than a comparable SVM. They also output probabilistic predictions, automatically estimate parameters, and allow the use of arbitrary basis functions. For the midterm report, we trained a Gaussian pairwise RVM classifier on the data (200 data points from each class were used to train, and 50 to test), which had comparable performance to SVM for the pairwise case (average accuracy of 99.7%).

C. K-Nearest Neighbors

Both SVM and RVM are binary classifiers (although they can be extended to multiclass classification as discussed in Section IV), so we have implemented K-Nearest Neighbors (KNN) as a simple multiclass classifier. KNN is a useful algorithm when there is no known underlying structure to the data, but is prone to overfitting and requires an appropriate distance measure be defined [47]. It also requires that a new data point be compared to the entire training set (which may be prohibitively expensive if the data set is large). Because KNN is very general and can be applied to many different problems, much work has been done to analyze the appropriateness of the algorithm [45], [46]. For the midterm report, we ran KNN on two cases: one nearest neighbor and 5 nearest neighbors. KNN gave results comparable to SVM and RVM (average accuracy of 99.5%), while truly being a multiclass classifier.

D. Sparse Multinomial Logistic Regression

Sparse Multinomial Logistic Regression (SMLR) is a supervised learning algorithm for probabilistic multiclass classification[52]. There are several advantages to using SMLR as opposed to other classifiers like SVM and RVM. Unlike extensions to SVM and RVM, SMLR has a true multiclass formulation which learns a weight vector w such that the likelihood of label i for data point x is given by:

$$P\left(y^{(i)}=1|x,w\right)=\frac{\exp\left(w^{(i)T}x\right)}{\sum_{j=1}^{m}\exp\left(w^{(j)T}x\right)}$$

As the name implies, SMLR produces a *sparse* weight vector by using a Laplacian prior on the weights, and although calculating the maximum a posteriori (MAP) multinomial regression with a Laplacian prior scales unfavorably with the number of bases (which may be very large), the component-wise update equation has a monotonically increasing closed form solution. Solving component-wise SMLR results in computation cost and storage requirement linear in the number of bases. Like RVMs, it uses Bayesian inference to provide probabilistic classification and can incorporate arbitrary bases, including non-Mercer kernels, but SMLR converges to a unique maximum, and is not at risk of local minima as RVMs are. For these reasons, we have chosen to use Sparse Multinomial Logistic Regression as our final classification algorithm, results of which can be found in the next section.

V. Classification Results

A. *Gesture Classification*

The SMLR gesture classifier was learned on 88 examples from each class and tested on 53 examples from each class. Figure 3 shows the confusion matrix for the learned classifier, which gave an average accuracy of 97.5%. `X`'s were the most difficult to detect, with an accuracy of only 91.2%. While these results are not as good as some gesture recognition methods in previous literature (discussed in Section VII), recall that many of our gestures are "flexible," i.e. don't have an underlying template (notably `path`, `zone` and `wander`). To correctly identify these gesture types, the classifier can't necessarily consider only what a gesture *looks* like, but also *how* the gesture was drawn. This is a much harder problem than classifying gestures such as numbers and letters, which not only have "template" structures, but are also written with known orientation. Considering the added complexity of our problem, we find our results very promising. Note that baseline (i.e. random) classification would yield 10% accuracy.

The confusion matrix, however, does not tell the whole story. We are not only interested in the number of correct/incorrect classifications, we are also concerned with how *confidently* the classifier chooses the correct label. Recall that SMLR gives probabilistic classifications, so not only do we know the most likely label, but we have a measure of *how likely* the label is. Consider the binary classification problem in which the algorithm outputs highest likelihood for the correct label 100% of the test data, but only with probability = 0.51. In other words, the likelihood of the incorrect label is very close to that of the correct label. It is not only important that the classifier correctly label the test data, but also that it does so with high confidence.

Figure 4 shows the average label likelihood calculated by the gesture classifier. Notice that while it looks very similar to the confusion matrix, it is not identical. Here, we can see that the average likelihood calculated for the correct label is 96.9%. The difference between these two matrices will be more apparent in the next sections.

B. *Stroke Classification*

The SMLR stroke classifier was learned on single strokes using 89 examples from each class and tested on 82 examples from each class. Figure 5 shows the confusion matrix for the learned classifier, which gave an average accuracy of 83.4%. `X`-strokes were the most difficult to detect, with an accuracy of only 56.1%. While these results may seem poor, recall that this classifier is trying to label the gesture type of a single stroke. In the case of a straight line, the stroke could belong to any number of gesture types. In fact, this is why `X`-strokes are so difficult to correctly classify. Note that baseline (i.e. random) classification would yield 10% accuracy.

Again, we can refer to the average likelihoods in Figure 6. Notice here that `X`-strokes have highly non-zero likelihoods associated with `triangles` and `arrows`. This makes sense, as `triangles` and `arrows` are often drawn with a collection of angled straight-line strokes. Similarly, `important`-strokes had non-zero likelihoods as `boxes`, which are also often drawn using a collection of vertical straight-line strokes. Despite these expected confusions, the average likelihood calculated for the correct label is still 81.2%.

It is worth noting that, although this classifier is not always able to correctly classify the stroke, it is still useful in ruling out what the stroke *isn't*. For example, it may not be possible to determine whether a straight-line stroke belongs to an `X`, `triangle` or `arrow`, but the classifier would correctly yield near-zero likelihoods for `circles` and `spirals`. In Section VI, we will see that this will help the sequence search algorithm by effectively pruning parts of the search tree with near-zero probabilities.

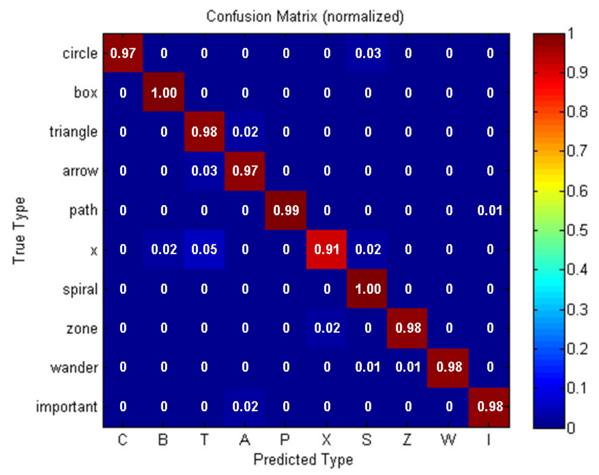

Fig. 3. Confusion matrix for gesture classifier, where diagonal elements represent correctly labeled test data.

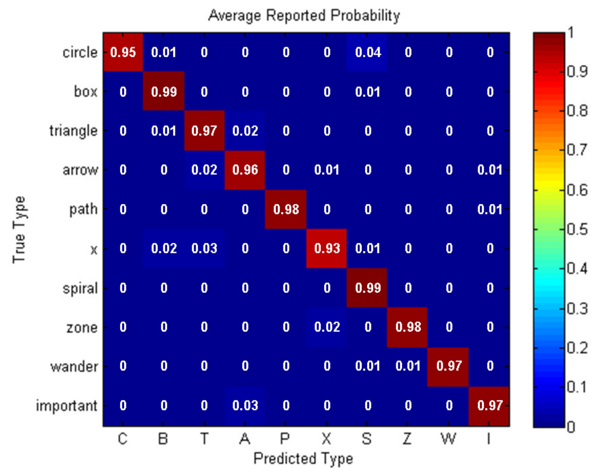

Fig. 4. Average calculated gesture class likelihoods, where diagonal elements represent correct labels.

C. *Stroke Transition Classification*

The SMLR stroke transition classifier was learned on seven classes: `boxes`, `triangles`, `arrows`, `X`, `zone`, `important` and `new gestures`. The nine-dimensional feature vector was calculated from consecutive strokes; if both strokes belonged to the same gesture the stroke transition was labeled to that gesture class, and if the strokes belonged to different gestures the stroke transition was labeled as `new gesture`. Four gesture types (`circle`, `path`, `wander` and `spiral`) were never drawn with more than one stroke, so these stroke transition likelihoods were set to zero.

The classifier was learned on stroke transitions using 46 examples from each class and tested on 80 examples from each class. Figure 7 shows the confusion matrix for the learned classifier, which gave an average accuracy of 75.4%. Notice now, `X`-stroke transitions are the easiest to classify with an accuracy of 87.7%, as users tend to draw these gestures very consistently. Note that baseline (random) classification would yield 14.3% accuracy.

Here, we can clearly see the difference between Figures 7 and 8. Although stroke transitions were correctly classified 75.4% of the time, the average likelihood of correct classification was only 65.1%. This indicates that there are many cases where the classifier outputs the correct label, but the classification likelihood is significantly less than 1. This may indicate that different features are needed to better distinguish between labels.

VI. Learning Sequences of Gestures

As alluded to in Section III, even the best gesture classifier can only be effectively applied once the transitions between each gesture have been identified. This is in itself very difficult, because there is no indicator for the beginning and end of gestures, only pen strokes. To simultaneously define gesture transitions and classify gestures, we apply a heuristic tree search method to find the maximum likelihood gesture sequence by examining successive strokes, and searching only the most likely paths.

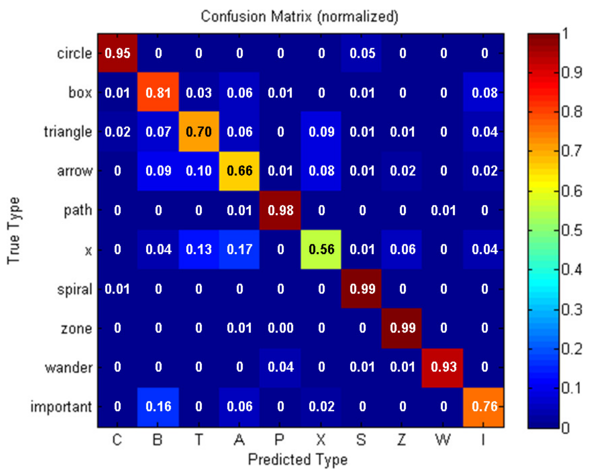

Fig. 5. Confusion matrix for stroke classifier, where diagonal elements represent correctly labeled test data.

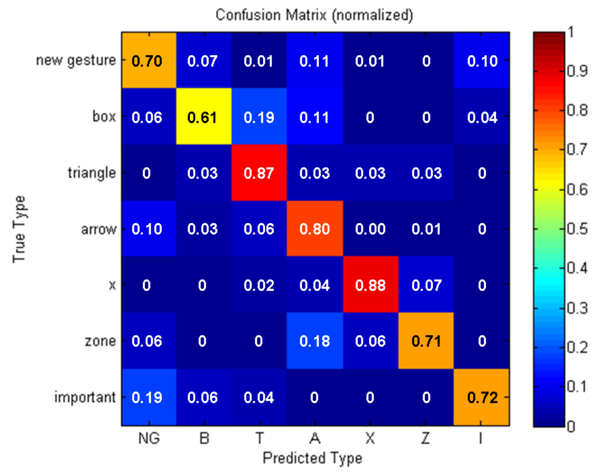

Fig. 7. Confusion matrix for stroke transition classifier, where diagonal elements represent correctly labeled test data.

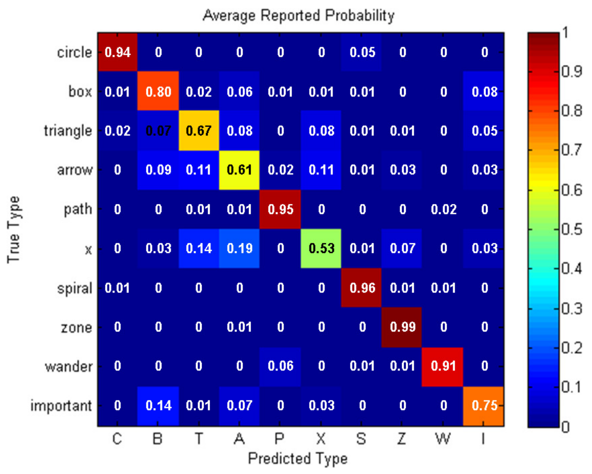

Fig. 6. Average calculated stroke class likelihoods, where diagonal elements represent correct labels.

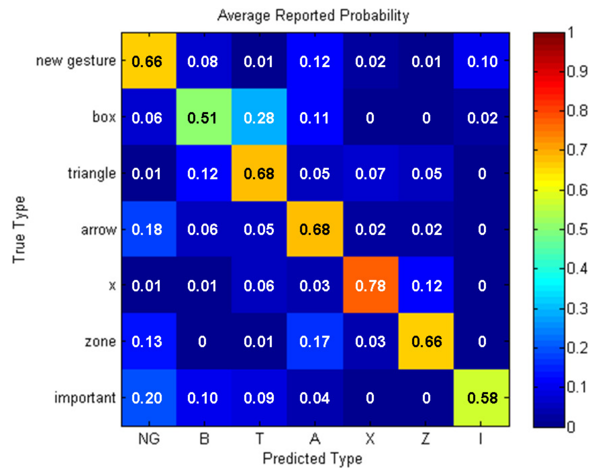

Fig. 8. Average calculated stroke transition class likelihoods, where diagonal elements represent correct labels.

Starting with the first stroke, there M possible classes the stroke may belong to, represented as nodes in a tree. Using the stroke classifier described in Section V-B, calculate the likelihood L that the stroke belongs to a particular gesture type GT and assign to the respective node. Find the leaf node i with the highest probability, and "expand" that node by looking at the next stroke in the sequence. That stroke may either belong to the same gesture as i or to a new gesture, creating $M+1$ child nodes from i. For each child node k calculate $L(k, A_k)$, the likelihood of the whole stroke sequence up to node k (A_k are the ancestors of node k), using the appropriate gesture, stroke, and stroke transition classifiers. Repeat this process by expanding the highest likelihood leaf node (i.e. nodes that have not already been expanded) until the highest likelihood leaf node corrseponds to the last stroke. This algorithm is known as a best first search and is guaranteed to find the optimal sequence (where "optimal" is defined as the highest likelihood). Please refer to the pseudo-code presented in Algorithm 1 for details. Here, G_i is the sequential gesture number of node i.

The likelihood of a sequence up to node i, L_i, is the product of stroke, gesture, and stroke transition probabilities as defined by the stroke labels. It is intuitive that as the sequence gets longer, its aggregate likelihood will decrease (as we are multiplying numbers ≤ 1 – or adding numbers ≤ 0 in the case of log-likelihoods). This has the undesirable effect of encouraging the algorithm to perform breadth-first-search (this is worst-case for trees with constant depth)! Therefore, we implement a heuristic to "penalize" nodes closer to the top of the tree:

$$L'_i = L_i \cdot \left(\frac{N_i}{N}\right)^\alpha$$

where N_i is the stroke number of node i, N is the total length of the sequence, and α is a number between 0 and 1. (Note: L here is a log-likelihood.) This heuristic encourages a more depth-first-search behavior, but sacrifices the guarantee of optimality. The effect of this heuristic is evaluated in Section VI-B.

A. *Performance and Error Metrics*

Two performance metrics are used to evaluate the efficiency and effectiveness of the gesture sequence learning algorithm: (1) the number of misclassified strokes, and (2) the number of nodes explored in the tree search. A stroke is considered misclassified once if it's gesture *type* is incorrect and again if it is considered part of a new gesture when it is actually a continuation of the previous gesture (or vise-versa). These errors measure the accuracy of the algorithm as a whole.

The number of nodes explored in the tree search measures the computational efficiency of the algorithm. While the worst-case search will yield a search of $M(M+1)^{N-1}$ nodes (where M is the number of gesture types and N is the number of strokes), this can become an infeasible search for long sequences. Even if the search is feasible, it may not be possible to perform in real time. However, if the individual

Algorithm 1 Finding the optimal stroke sequence

```
$k = 0$
leaf nodes = $\emptyset$
for all $j \in GT$ do
    create new node $k = k + 1$
    set $A_j = \emptyset$, $N_k = 1$, $G_k = 1$, $GT_k = GT(j)$
    evaluate $L'_k = L(k) \cdot \left(\frac{1}{N}\right)^\alpha$
    add $k$ to leaf nodes
end for
find $i$ such that $L'_i = max(L'_n)$ where $n \in$ leaf nodes
while $N_i < N$ do
    for all $j \in GT$ do
        create new node $k = k + 1$
        set $A_k = [A_i, i]$
        set $N_k = N_i + 1$, $G_k = G_i + 1$, $GT_k = GT(j)$
        evaluate $L'_k = L(k, A_k) \cdot \left(\frac{N_k}{N}\right)^\alpha$
        add $k$ to leaf nodes
    end for
    create new node $k = k + 1$
    set $A_k = [A_i, i]$
    set $N_k = N_i + 1$, $G_k = G_i$, $GT_k = GT_i$
    evaluate $L'_k = L(k, A_k) \cdot \left(\frac{N_j}{N}\right)^\alpha$
    add $k$ to leaf nodes
    remove $i$ from leaf nodes
    find $i$ such that $L'_i = max(L'_n)$ where $n \in$ leaf nodes
end while
return $A_i$
```

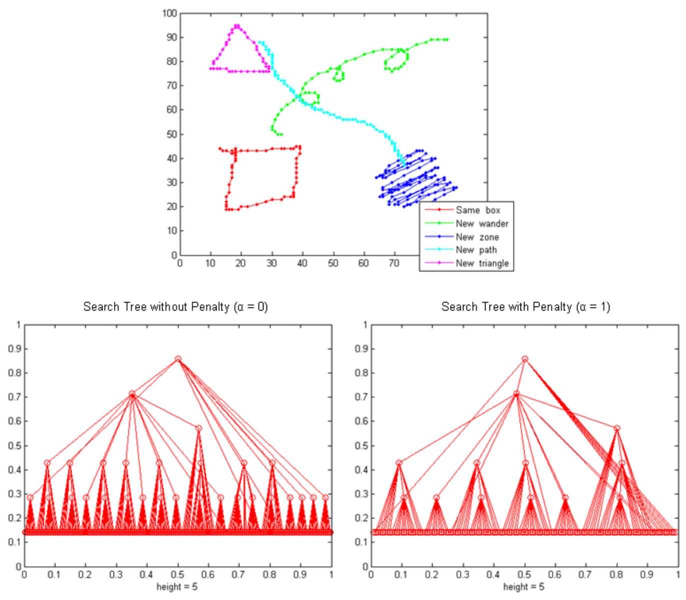

Fig. 9. An example 5-stroke sequence (top), reconstructed correctly by both the optimal search algorithm in 300 nodes (left), and the penalized search algorithm in 130 nodes (right).

classifiers (described in the previous sections) are "good enough," they will have a similar effect as tree pruning, where branches with low likelihoods are never explored. The hope is that the tree can be searched fast enough to allow real-time evaluation of stroke sequences, since we aim to use them to control robots.

Figure 9 illustrates an example of a 5-stroke sequence. The worst-case search for this sequence would require a search of 146,410 nodes. However, even without a penalty term the algorithm is able to find the maximum likelihood sequence by searching under 300 nodes. Using an aggressive penalty term of $\alpha = 1$ the algorithm is able to find the same sequence by searching only 130 nodes!

B. *Sequence Reconstruction Results*

77 different gesture sequences, ranging from 3–10 strokes, were reconstructed using different penalty values ($\alpha \in \{0, 0.1, 0.5, 1.0\}$). Recall that a higher penalty promotes depth searching when node likelihoods at different depths in the tree are similar. Average accuracy and percent of tree searched are plotted in Figures 10 and 11, respectively.

With no penalty ($\alpha = 0$), the sequence reconstruction algorithm yielded an average accuracy of 93.2%. For short sequences (3 strokes), an average accuracy of 97.6% was obtained. Even with an aggressive penalty ($\alpha = 1$), the algorithm yielded an average accuracy of 92.6% accuracy. For short sequences (3 strokes), an average accuracy of 97.6% was obtained. Note that baseline (random) sentence reconstruction of a 3-stroke sequence will yield an accuracy of 0.8%. With no penalty, an average of 0.55% of the total search tree was evaluated for the 77 test cases (maximum of 11.4%). For $\alpha = 1$, an average of 0.37% of the total search tree was evaluated for these test cases (maximum of 3.7%).

Figures 10 and 11 suggest that while adding a penalty term decreases the sequence reconstruction accuracy slightly (this is expected, as we are no longer guaranteed to find the maximum likelihood sequence), it decreases the computational cost significantly. It may be reasonable to use a small penalty term ($\alpha = 0.1$) to help avoid breadth-first-search behavior without sacrificing much accuracy.

It should be noted that our search strategy is prone to cascading errors. In other words, if a stroke is misclassified, there is a greater chance that the subsequent stroke(s) may also be misclassified. Imagine that the first of two `X`-strokes is incorrectly classified as an `arrow`. The next stroke (the second `X`-stroke) will either be classified as a `new X` (one error), `same arrow` (one error), or `new ***` where "`***`" is not an `X` (two errors). Because the first stroke was misclassified, the second stroke will necessarily be (at least partially) misclassified.

The computational complexity of our algorithm is also sensitive to stroke sequencing. Since we are employing a tree search (with a large branching factor), a difficult-to-classify node towards the top of the tree may cause much more of the tree to be explored than if the same node were toward the bottom of the tree. This may be partially alleviated by choosing a different heuristic or by performing a bidirectional search.

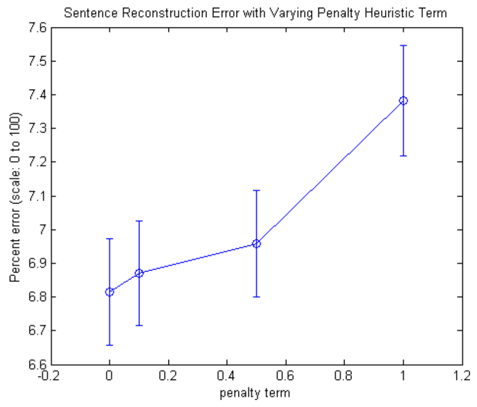

Fig. 10. Percent stroke error of sequence reconstruction using heuristic tree search algorithm with varying penalty terms.

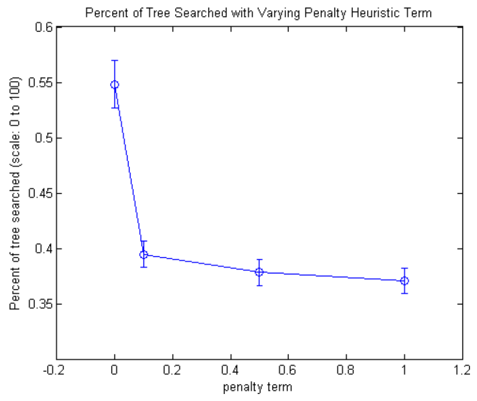

Fig. 11. Percent of total tree searched using heuristic tree search algorithm with varying penalty terms.

VII. Comparison to Existing Literature

As discussed in Section II, the field of gesture recognition is hardly new. Here, we briefly compare our results to some closely-related existing literature.

In [48], the authors examined a similar problem as ours within the context of recognizing hand-drawn circuit diagrams. Addressing the issues of segmentation and stroke recognition using Dynamic Bayesian Networks, their algorithm yielded an average accuracy of 92%, which is comparable to our sequence reconstruction results. While [48] addressed the problem of interspersed drawings (making the combinatorial problem discussed in Section VI significantly more difficult than in our case), they are able to rely more heavily on spatial relationships between gestures. The circuit gestures (resistors, transistors, etc.) in [48] are also not "flexible", although their gesture recognition did have to handle variability in size and orientation.

The multi-stroke icon recognition problem presented in [17] is most similar to our gesture recognition problem, and in fact we used many of the features devised by the authors. Their average accuracy was 95% with various icon datasets, some of which which were arguably more complicated than the 10 gestures we studied. We achieved an accuracy of

97.5% using significantly fewer training examples, however, and the gestures in [17] had fixed orientations, whereas ours were not constrained to a particular orientation (except the `important` gesture).

With respect to application, the authors of [2] also present robot control as motivation for written gesture recognition. While the experiments in [2] are very similar to the end goal that we hope to achieve, we believe that our framework allows for a superior interface to emerge. [2] used a very limited set of rigidly-defined gestures, and the focus was almost exclusively on the integration between gesture and robot (i.e. after the gesture has been recognized, how to control the robot), rather than the gesture recognition itself. We emphasize the need to be able to handle a multitude of command gestures (and even learn new ones easily) in order to create an expressive gesture environment.

VIII. Discussion and Conclusions

While the results presented in Sections V-VII are promising, there are still several things that can be done to improve accuracy and reduce computational complexity. First, we can add an `other` gesture, composed of a colelction of strokes that does not make any gesture. This will be helpful when determining whether a stroke transition is *inter*-gesture or *intra*-gesture. For example, imagine a group of strokes is being run through the gesture classifier. The collection of strokes may not resemble any gesture at all, but it will still be classified as *something* with a minimum probability of $1/M$. By adding an `other` gesture, it will allow a sequence of strokes to have low probability for *all* gesture types. We predict that this will greatly improve sentence reconstruction.

Another way to improve classification accuracy would be to change the features used. Specifically, we would like to examine how a kernel basis function affects accuracy. This may greatly increase computation (if the number of samples is large), but increased accuracy may be worth the added complexity. Using Gaussian kernels centered around data points may also provide a more robust writing style.

The best-first-search approach we presented in Section VI is certainly not the only way to do sequence reconstruction, especially in terms of computational complexity. Recalling that the worst-case search will require examination of $M\left(M+1\right)^{N-1}$ nodes, this technique may not even be feasible for long sequences of strokes. One way to improve computational efficiency is to use a different search heuristic. For example, an A* search uses an optimistic search heuristic that still guarantees optimality.

One thing we did not discuss in this paper (although it was presented in the midterm report) is dimensionality reduction. For the midterm report, we used PCA to reduce the feature dimensionality from 98 to 44 while maintaining 90% of the data variance. Due to time constraints, we did not do an analysis of dimensionality reduction using SMLR for this paper. One should note, however, that the sparsity promoting priors of SMLR lend itself naturally to dimensionality reduction. After training the classifier, bases that have zero weights (for all classes) are not used by the classifier and can be ignored when collecting new data. If these bases correspond to actual features, this may give some intuition about what features are useful for distinguishing between classes. If the bases correspond to kernels centered around training data, it may indicate a set of "template" gestures.

Lastly, incorporating environmental information (by adding to the feature set) could provide invaluable information. If we use this algorithm to send controls to robot teams (as we plan to do), information about the environment, such as where a gesture is drawn with respect to a robot, can be extremely helpful for segmenting stroke sequences into gestures that make sense. Additionally, we can incorporate priors on gesture transitions that are application-specific. For example, if a `circle` translates to "select this robot," and an `arrow` means "travel to this location," there may be a high probability of an `arrow` immediately following a `circle`. This is information we ignored for this project, but plan to incorporate in future work.

IX. Future Work

We plan to build on the framework we have developed to implement a comprehensive interface for reliable and natural robotic control in the coming months. We will focus on improving the recognition rates of full sentences by refining the feature set. We also plan on enhancing the tree search algorithm's speed (without sacrificing optimality) by improving our heuristic function. When connecting the interface to a robot collective, each gesture will represent some (possibly abstract) command from a human user to the robot team, as well as the level of autonomy expected. For example an `arrow` may mean "Go to this location by any path you'd like," whereas a `path` may mean "Go to this location by taking this specific route." This requires that the gestures be translated into some useable form and will take a significant amount of development, but we hope to achieve this by next summer. To translate the gestures, we will make use of Dr. Kress-Gazit's work in Linear Temporal Logic (LTL) [49], [50], [51]. LTL has the ability to capture typical control specifications such as reachability and invariance, as well as complex specifications like sequencing and obstacle avoidance. It is our hope that the generation of these controllers using a handwritten gesture interface will be functional for the MAGIC 2010 competition next November.

The "Holy Grail" of this research is to incorporate probabilistic interpretations of human input into the LTL framework, which is rigid by definition. For example, the following questions are posed: What should the robot do if it is only 80% confident in the translation? What if the user commands something that is impossible ("breaks" another LTL rule)? What if the user's command is ambiguous? These questions are hard (but very interesting!) problems that we hope to address in the future.

REFERENCES

[1] D. Anderson, C. Bailey and M. Skubic, "Hidden Markov Model symbol recognition for sketch-based interfaces," AAAI Fall Symposium. Menlo Park, CA: AAAI Press, pp. 15–21, 2004.

[2] M. Skubic, D. Anderson, S. Blisard, D. Perzanowski and A. Schultz, "Using a qualitative sketch to control a team of robots," Proceedings 2006 IEEE International Conference on Robotics and Automation (ICRA2006), pp. 3595–3601, 2006.

[3] J. Yang and Y. Xu, "Hidden Markov Model for Gesture Recognition," Tech. Report CMU-RI-TR-94-10, Robotics Institute, Carnegie Mellon University, 1994.

[4] S. Iba, J. Vande Weghe, E. Weghe, C. Paredis and P. Khosla, "An Architecture for Gesture-Based Control of Mobile Robots," Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 851–857, 1999.

[5] S. Waldherr, R. Romero and S. Thrun, "A Gesture Based Interface for Human-Robot Interaction," Autonomous Robots, vol. 9, pp. 151–173, 2000.

[6] S. Calinon and A. Billard, "Stochastic Gesture Production and Recognition Model for a Humanoid Robot, Proceedings. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2004), vol. 3, pp. 2769–2774, 2004.

[7] A. Corradini, "Real-Time Gesture Recognition by Means of Hybrid Recognizers," Lecture Notes In Computer Science, Revised Papers from the International Gesture Workshop on Gesture and Sign Languages in Human-Computer Interaction, vol. 2298, pp. 34–46, 2001

[8] C. Lee and Y. Xu, "Online, interactive learning of gestures for human/robot interfaces," IEEE International Conference on Robotics and Automation (ICRA1996), pp. 2982–2987, 1996.

[9] T. Starner and A. Pentland, "Visual recognition of american sign language using hidden markov models," International Workshop on Automatic Face and Gesture Recognition, pp. 189–194, 1995.

[10] B. Verma, M. Blumenstein and S. Kulkarni, "Recent Achievements in Off-line Handwriting Recognition Systems," In Proceedings of the International Conference on Computational Intelligence and Multimedia Applications, pp. 27–33, 1998.

[11] A. Vinciarelli and J. Luettin, "Off-line cursive script recognition based on continuous density HMM," Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition, pp. 493–498, 2000.

[12] A. Vinciarelli, "A survey on offline Cursive Word Recognition," Pattern Recognition, vol 35, no. 7, pp. 1433-1446, 2002.

[13] I. Mackenzie and S. Zhang, "The Immediate Usability of Graffiti," Graphics Interface, pp. 129–137, 1997.

[14] L. Yaeger, B. Webb and R. Lyon, "Combining Neural Networks and Context-Driven Search for On-Line, Printed Handwriting Recognition in the Newton," Lecture Notes In Computer Science, Neural Networks: Tricks of the Trade, vol. 1524, pp. 275–298, 1998.

[15] CalliGrapher, Phatware Corp., http://www.phatware.com/index.php?q=product/details/calligrapher

[16] J. Shlens, "A Tutorial on Principal Component Analysis," La Jolla, CA 92037: Salk Institute for Biological Studies, 2005.

[17] D. Willems, R. Niels, M. van Gerven, L. Vuurpijl, "Iconic and multi-stroke gesture recognition," Pattern Recognition, vol. 42, num. 12, pp. 3303–3312, 2009.

[18] D. Willems and R. Niels, "Definitions for features used in online pen gesture recognition," Tech. rep., NICI, Radboud University Nijmegen (2008). URL http://unipen.nici.ru.nl/NicIcon/

[19] V. De Silva and J. Tenenbaum, "Global Versus Local Methods in Nonlinear Dimensionality Reduction," Advances in Neural Information Processing Systems, vol. 15, pp. 705–712, 2003.

[20] J. Tenenbaum, V. de Silva, J. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," Science, vol. 290, pp. 2319–2323.

[21] S. Roweis and L. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," Science, vol. 290, pp. 2323–2326, 2000.

[22] M. Belkin and P. Niyogi, "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation," Neural Computation, vol. 25, pp. 1373–1396, 2002.

[23] T. Hastie and R. Tibshirani, "Discriminant analysis by Gaussian mixtures," Journal of the Royal Statistical Society, Series B, vol 58, pp. 155–176, 1996.

[24] N. Cristianini and J. Shawe-Taylor. An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press, 2000.

[25] M. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine," The Journal of Machine Learning Research, vol. 1, pp. 211–244, 2001.

[26] T. Hamamura, H. Mizutani and B. Irie, "A Multiclass Classification Method Based on Multiple Pairwise Classifiers," Proceedings of the Seventh International Conference on Document Analysis and Recognition, vol. 2, pp. 809, 2003.

[27] T. Wu, C. Lin and R. Weng, "Probability Estimates for Multi-class Classification by Pairwise Coupling," Journal of Machine Learning Research, vol. 5, pp. 975–1005, 2003.

[28] Aldebaro Klautau and Nikola Jevtic and Alon Orlitsky, "Combined Binary Classifiers with Applications to Speech Recognition," Nearest-Neighbor Ecoc With Application to All-Pairs Multiclass SVM, pp. 2469–2472, 2002.

[29] H. Zhao and B. Lu, "An Efficient Selection of Binary Classifiers for Min-Max Modular Classifier," 2005 IEEE International Joint Conference on Neural Networks, vol. 5, pp. 3186–3191, 2005.

[30] N. Yukinawa, S. Oba, K. Kato and S. Ishii, "Optimal Aggregation of Binary Classifiers for Multiclass Cancer Diagnosis Using Gene Expression Profiles," IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), vol. 6, no. 2, pp. 333–343, 2009.

[31] C. Hsu and C. Lin, "A Comparison of Methods for Multiclass Support Vector Machines," IEEE Transactions on Neural Networks, vol. 13, no. 2, pp. 415–425, 2002.

[32] B. Sun and D. Huang, "Support Vector Clustering for Multiclass Classification Problems," Evolutionary Computation, vol. 2, pp. 1480–1485, 2003.

[33] E. Mayoraz and E. Alpaydin, "Support Vector Machines for Multi-class Classification," Proc. 5th IWANN, Spain, vol. 34, pp. 833–842, 1999.

[34] J. Rennie and R. Rifkin, "Improving Multiclass Text Classification with the Support Vector Machine," Tech. Report AIM-2001-026, 2007.

[35] K. Duan, S. Keerthi, W. Chu, S. Shevade and A. Poo, "Multi-category Classification by Soft-Max Combination of Binary Classifiers," 4th International Workshop on Multiple Classifier Systems, 2003.

[36] C. Park, H. Park, "A Comparison of Generalized Linear Discriminant Analysis Algorithms," Pattern Recognition, vol. 41, no. 3, pp. 1083–1097,2008.

[37] M. Campbell, "Multiple logistic regression modelswhat are they?" Midwifery, vol. 20, no. 3, pp. 236–239, 2004.

[38] Fix, E., Hodges, J.L. Discriminatory analysis, nonparametric discrimination: Consistency properties. Technical Report 4, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[39] B. Kijsirikul, N. Ussivakul and S. Meknavin, "Adaptive Directed Acyclic Graphs for Multiclass Classification," Lecture Notes In Computer Science, 7th Pacific Rim International Conference on Artificial Intelligence: Trends in Artificial Intelligence, vol. 2417, pp. 158–168, 2002.

[40] J. Lee and I. Oh, "Binary Classification Trees for Multi-class Classification Problems," Seventh International Conference on Document Analysis and Recognition, pp. 770–774, 2003.

[41] V. Vapnik, "Statistical Learning Theory," Wiley, New York, 1998.

[42] T, Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," Universität Dortmund, Fachbereich Informatik, LS-8 Report 23, 1997.

[43] C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," Data Mining and Knowledge Discovery, vol. 2, no. 2, pp. 121–167, 1998.

[44] C. Bishop and M. Tipping, "Variational Relevance Vector Machines," Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, pp. 46–53, 2000.

[45] K. Beyer, J. Goldstein, R. Ramakrishnan and U. Shaft, "When Is 'Nearest Neighbor' Meaningful?," Proceedings of the 7th International Conference on Database Theory, vol. 1540, pp. 217–235, 1999.

[46] J. Friedman, "Flexible Metric Nearest Neighbor Classification," Technical Report, 1994.

[47] R. Short and K. Fukunaga, "The Optimal Distance Measure for Nearest Neighbor Classification," IEEE Transactions on Information Theory, vol. 27, no. 5, pp. 622–627, 1981.

[48] T. Sezgin and R. David, "Sketch recognition in interspersed drawings using time-based graphical models," Computer Graphics, vol. 32, pp. 500510, 2008.

[49] G. Fainekos, H. Kress-Gazit and G. Pappas, "Temporal Logic Motion Planning for Dynamic Robots," Automatica (Journal of IFAC), vol. 45, no. 2, pp. 343–352, 2009.

[50] H. Kress-Gazit, N. Ayanian, G. Pappas and V. Kumar, "Recycling Controllers," IEEE International Conference on Automation Science and Engineering, pp. 772-777, 2008.

[51] H. Kress-Gazit and G. Pappas, “Automatically Synthesizing a Planning and Control Subsystem for the DARPA Urban Challenge,” IEEE International Conference on Automation Science and Engineering, pp. 766-771, 2008.

[52] B. Krishnapuram, M. Figueiredo, L. Carin, & A. Hartemink, “Sparse Multinomial Logistic Regression: Fast Algorithms and Generalization Bounds.” IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 27, pp. 957968, 2005.