Gaze Variable Selection (GVS) Toolbox

Samuel Rivera
Department of Psychology and Department of Electrical and Computer Engineering, The Ohio State University, Columbus OH, USA

Hyungwook Yim
Department of Psychology, The Ohio State University, Columbus OH, USA

Catherine A. Best
Department of Psychology, Kutztown University, Kutztown, PA, USA

Dirk B. Walther
Department of Psychology, University of Toronto, Ontario, Canada

Vladimir M. Sloutsky
Department of Psychology, The Ohio State University, Columbus OH, USA

Aleix M. Martinez
Department of Electrical and Computer Engineering, The Ohio State University, Columbus OH, USA

Abstract

Eye tracking has become a standard tool in experimental psychology because of the tight link between gaze and attention, the high temporal resolution (60 - 2000Hz), and its applicability in various populations (e.g., infants, animals). An open problem, however, is identifying the measures of gaze, or dependent variables, that give insight into cognition. Previous identification methods required expertise in the experiment paradigm, which were informal and had the possibility of omitting relevant gaze information. To resolve this problem, we developed a formal method for variable selection that identifies those variables that systematically differ between two experimental groups. The procedure, which has been implemented as a public MATLAB toolbox, applies standard machine learning algorithms to an exhaustive set of variables that are automatically identified from the raw eye gaze data (x, y coordinates). Here, we present the Gaze Variable Selection Toolbox in detail, then demonstrate the method with a visual categorization case study. The toolbox automatically identified that changes in fixation proportions and early fixation targets result from supervision.

Keywords: eye-tracking, variable selection, machine learning

Introduction

Eye tracking is an attractive method to study human cognition since eye movements provide an approximate measure for attention allocation (Hoffman & Subramaniam, 1995; Corbetta et al., 1998) with high temporal and spatial resolution. Therefore, studies utilizing eye tracking made interesting contribution across different disciplines such as in scene perception (Martinez-Conde, Macknik, & Hubel, 2004), category learning (Rehder & Hoffman, 2005a), memory (Hannula et al., 2010), language (Rayner, 1998; Tanenhaus, Spivey-Knowlton, Eberhard, & Sedivy, 1995), reasoning (Hayes, Petrov, & Sederberg, 2011), and research on clinical population (Riby & Hancock, 2008). Moreover, eye tracking paradigms have provided useful insights in disciplines where participants could not provide traditional responses (e.g., reaction time or accuracy) reliably – infants (Feng, 2011; McMurray & Aslin, 2004), and individuals with autism spectrum disorder (Klin, Lin, Gorrindo, Ramsay, & Jones, 2009).

Eye tracking studies, however, require the researcher to select dependent variables (i.e., different features of eye movements) that are appropriate for testing their hypothesis. The selection process is mostly subjective and informal, which depends on the researcher's experience or previous literature that has been verified. However, for a novice researcher or when conducting a new experimental paradigm, selecting appropriate variables is not trivial. Modern eye trackers produce between 60 to 2000 data points a second (i.e., 60Hz - 2000Hz), where each data point includes information such as the position of the eye and pupil diameter. Even though these raw data are converted into measures such as saccade latency, fixation frequency, and dwell time on a particular area of interest (AOI), the question still remains – Which variables will test the hypothesis the best?, and How many variables should one select?

Here we try to solve this problem by proposing a general framework to formalize and automate the selection process. The framework automatically selects variables using machine learning from a large pool of eye tracking variables, that have been used across different disciplines. Among these variables, the selection algorithm calculates the optimal variable(s) that could distinguish different conditions in a given experimental paradigm. The general framework has been implemented as a MATLAB toolbox called the Gaze Variable Selection (GVS) Toolbox which is publicly available at `https://github.com/sriveravi/GVSToolbox`. As we demonstrate with examples below, about 10 lines of MATLAB code is all that is needed to load and filter missing eye-track data from a standard text file, identify the optimal variables for analysis, and validate the variables with cross validation. Alternatively, we provide a graphical user interface (GUI). GVS Toolbox is well documented and also includes algorithms for filtering and extracting basic variables (e.g., fixation, saccade) such that users may freely utilize any algorithms of the toolbox, or extend the toolbox as

---

Correspondence concerning this article should be addressed to Samuel Rivera, Department of Psychology, The Ohio State University, 267 Psychology Building, 1835 Neil Ave., Columbus, OH, 43210, USA. E-mail: rivera.162@osu.edu

This research was partially supported by NIH grant R01 EY-020834 to AM, NSF grant BCS-0720135 and NIH grant R01 HD-056105 to VS, and a Seed Grant by the Center for Cognitive Science (CCS) at OSU to DBW, VS, and AM. SR was partially supported by a fellowship from the CCS.

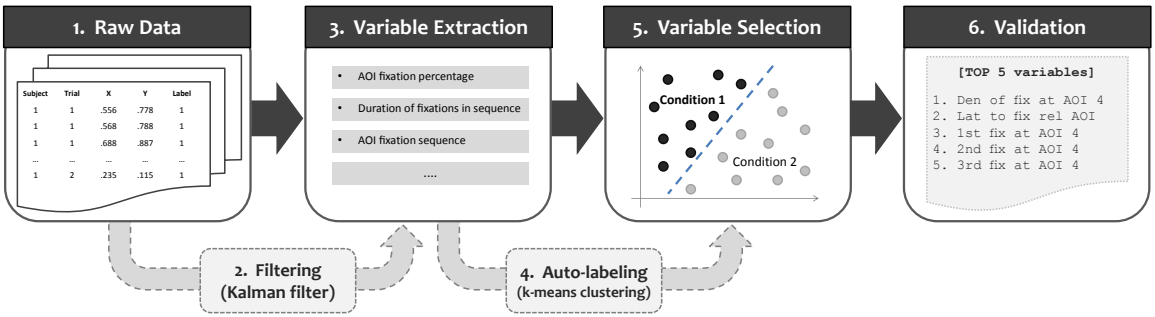

Figure 1. Overview of the GVS Toolbox. Filtering and Auto-labeling steps are optional.

necessary for their needs. In the following sections we will describe GVS Toolbox in detail. Then we demonstrate our method with a visual category learning study.

GVS Toolbox

The overview of GVS Toolbox is illustrated in Figure 1. First, raw gaze coordinates from the eye tracker are preprocessed by filtering out noise and recovering missing data such as from blinks. Then a large set of eye tracking variables are extracted from each trial. The toolbox identifies which variables systematically differ between two experimental conditions. These conditions may refer to a control versus standard experiment, first half versus last half of trials, or any other dichotomy. To select variables without specifying trial groupings, there are functions for unsupervised clustering of trials (i.e., k-means clustering). Finally, the selected variables are validated. The following sections describe each step in more detail. These tools can be accessed with the GUI depicted in Figure 2.

Importing Raw Data

The first step to use the toolbox is to import raw eye tracking data. This could be done by using the following MATLAB command:

```
>> ds = loadFromTable( 'exampleTable.txt' );
```

The data should be prepared in a tab delimited text file, with an example in Table 1. The file should contain columns *subject, trial, gazeX, gazeY,* and *label*. Each row corresponds to a single recorded sample from the eye tracker. The subject and trial columns index the subjects and trial numbers, respectively. The (gazeX, gazeY) columns contain the (x, y) gaze coordinates that are scale in the $[0, 1]$ range. The top left display corner corresponds to the values $(0, 0)$, while the bottom right corner is $(1, 1)$. Missing values are denoted by -1. The final label column contains zeros or ones corresponding to the different experimental conditions. In the case that it is unclear which group a trial belongs in, a label of 2 in the source file denotes an *indeterminate* trial. Indeterminate trials are automatically excluded from the variable selection and validation analyses. The indeterminate labels are useful for experiment error, or excluding trials that do not meet experimental criterion.

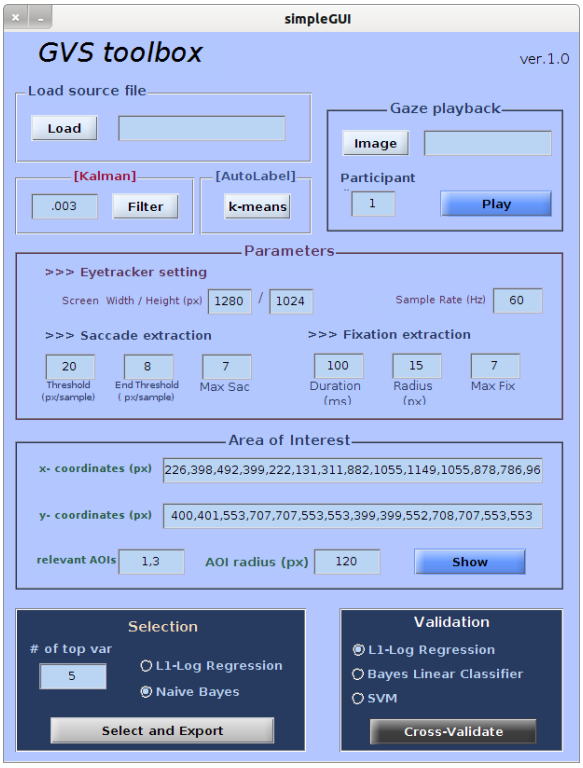

Figure 2. The Graphical User Interface. Run the 'simpleGui' to start the GUI.

Filtering

Eye tracking data usually contains missing values from blinks as well as measurement noise which complicates identification of fixation and saccades among other variables. The toolbox provides a filtering tool that interpolates missing data and smooths the effect of noise using a Kalman filter (Murphy, 2004). The Kalman filter is a recursive algorithm that estimates the state of a discrete-time stochastic process, given a description of that process as a linear stochastic difference equation, and previous observations. It has been widely used for filtering and tracking motion trajectories that are well described by the physical laws of motion (Auger et al., 2013; Kalman, 1960), making it applicable for tracking eye movements. This step is optional, and may be replaced by alternative methods (e.g., Saez de Urabain, Johnson, & Smith), as shown in the Appendix.

To filter missing data and noise, the *smoothTracks* function is used as follows:

```
>> ds = smoothTracks( ds );
>> ds = smoothTracks( ds, .0003 );
```

The optional second argument relates to process variance and controls for noise level. The default value of 0.0003 works well for gaze scaled in the $[0, 1]$ range. However, for noisier data the value could be reduced (e.g., 0.0001). Using a value too small will overly smooth the gaze, resulting in interpolated gaze that slowly follows behind the true gaze trajectory.

subject	trial	gazeX	gazeY	label
1	1	-1	-1	0
1	1	-1	-1	0
1	1	0.7463199	0.3817881	0
1	1	-1	-1	0
1	1	0.7563701	0.391761	0
1	1	-1	-1	0
		⋮		
1	16	0.7540916	0.3970774	0
1	16	0.7646027	0.363432	0
1	16	0.7640033	0.3662336	0
1	16	0.7629339	0.3637718	0
		⋮		
10	16	0.3934239	0.5095221	1
10	16	0.3964407	0.5062761	1
10	16	0.3893011	0.5015657	1
10	16	0.3845963	0.4944581	1
10	16	0.3877983	0.4998075	1
10	16	-1	-1	1
10	16	-1	-1	1

Table 1
Example data tab delimited text file for 10 *participants conducting* 16 *experiment trials. The columns headings (gazeX, gazeY) denote the (x, y) gaze coordinates that are scaled in the range [0,1], with missing values denoted by* -1*. The label column contains zeros or ones to denote the experiment group each trial belongs to (i.e., control versus experiment condition). Trials with label value of* 2 *are excluded from analysis.*

Variable Identification

Fixations and Saccades. After filtering, fixations and saccades are extracted. Fixations are detected using a dispertion threshold algorithm (Salvucci & Goldberg, 2000) when eye gaze is maintained within a 15 pixel radius for at least 100ms. Saccades are detected using a velocity-threshold identification algorithm (Stampe, 1993) when an eye movement exceed smooth pursuit velocity of 30° per second or 0.5° per sample at 60Hz. The default values were optimized for human adults using a 1024 by 1280 pixel display of 27.0 by 33.8 cm and an eye tracker at 60Hz. Adjustments should be made for infants or special populations, and different sampling rates and display sizes. The following commands enable users to change these values. However, these commands can be skipped when accepting the default values.

```
>> varParams.imageSize = [1024; 1280]; % display [height; width]
>> varParams.fixMinNumSamples = 6; % (6 samples)/(60 Hz)= 100ms
>> varParams.fixMaxCircleRadius = 15; % circle with radius 15px
>> varParams.velThreshold = 20; % 20px shift per sample to start saccade
>> varParams.stopThreshold = 8; % 8px shift during saccade to signal end
```

Variable Extraction. The variables are a set of dependent variables that have been traditionally used across disciplines (Holmqvist et al., 2011), and are typically determined with respect to fixations and saccades at specific areas of interest (AOI) within a display. AOIs are (sub)regions of the presented stimulus that can be grouped in a meaningful way, such as color uniformity or the structural nature of the object. The AOIs can further be labeled in various ways based on their role in the experiment (e.g., category relevant feature). The toolbox assumes circular AOIs defined by their center coordinates and radius. In the case that circles overlap, AOIs are restricted to the closest AOI center.

The following commands are an example that shows how to define three AOIs, and extract gaze variables.

```
>> aoiPositions = [20,40,80;40,80,160]; % [x1,x2,x3; y1,y2,y3]
>> aoiRadius = 20; % 20 pixel radius of AOI center
>> relevantAOIs = [1;3];  % AOIs 1 and 3 are relevant
>> [ features, varParams ] = extractVarsDist( ds.trackCell,...
     aoiPositions, aoiRadius, relevantAOIs, varParams );
```

The first two lines specify the position and size of three AOIs. The third line specifies which of the AOIs are relevant (first and third in this case). For example, in a category learning task, relevant AOIs could be those that contain category relevant features. The final line extracts all gaze variables as listed below. Although the toolbox contains many of the popular eye tracking variables in the literature, some are not listed or have not yet been discovered. The toolbox has a straightforward method for expanding the variable list to include any additional gaze measures (see Appendix).

1. *AOI fixation percentage* describes the percentage of time fixated at the different AOIs during a trial. All non-AOI fixations are discarded in the calculation. For a stimulus with q AOIs, this variable is encoded as a q-dimensional feature vector with a value for each AOI. The fixation percentages are normalized so that they sum to 1 whereas if there are no fixations at AOIs all percentages are set to 0.

2. *Relevant AOI fixation density* is represented as a scalar value between zero and one which describes the percentage of the total time fixated at the relevant AOI(s).

3. *AOI fixation sequence* describes the sequence of AOI fixations during one trial. We limit this sequence to seven fixations, starting with trial onset. We encode a fixation sequence of f fixations over q AOIs as a $q \times f$ binary matrix, where each column of the matrix has a 1 in the position corresponding to the fixated AOI, and zero otherwise. If there are fewer than f fixations, the last columns are set to 0. This binary encoding of the fixation sequence allows us to describe any sequence of fixations without imposing an ordering of the AOIs. In addition, the fixation sequence is represented as a sequence of relevant and non-relevant AOI fixations. This representation yields a $2 \times f$ binary matrix, in which each column has a 1 in the first row if a relevant AOI is fixated or a 1 in the second row if a non-relevant AOI is fixated. If there are less than f fixations, the last columns are set to 0. Note that it is necessary to use a pair of binary variables to encode each fixation of the latter representation to account for three cases: fixations at a relevant AOI, non-relevant AOI, and fewer than f fixations. Therefore there will be alternating ones in each row, and zeros in the last columns for both rows if less then f

fixations were made. The number of fixations to consider as well as the start position were determined with validation (CV) of our case study data. In cross validation, the training data are separated into k partitions, and for each partition, samples are classified using a classifier that is trained with the remaining $k-1$ partitions. The percentage of correctly classified samples over all partitions is the CV accuracy. We varied the parameters (start fixation and number of fixations) and calculated the CV accuracy when using only the fixation sequence to classify samples. Using the first few fixations gave the best results, with no improvement as later fixations were included. These paramters are easily changed, as shown in the Appendix.

4. *Duration of fixations in sequence* describes the duration of each fixation in the sequence described by the *AOI fixation sequence*. This variable is encoded by an f-dimensional vector.

5. *Total distance traveled by eye* is a scalar value describing the total distance traveled by the eye gaze during a trial. The distance is found by adding the Euclidean distance between all adjacent (x, y) gaze position samples from the eye tracker.

6. *Number of unique AOIs visited* is a scalar describing the total number of unique AOIs fixated during a trial.

7. *Saccade sequence* is similar to *AOI fixation sequence* but describes the sequence of AOI saccades during one trial. All saccades that do not terminate at AOIs are discarded. The sequence is limited to the first seven saccades. The number of saccades to consider as well as the start saccade were determined using CV, but these parameters can be changed as shown in the Appendix. We encode a saccade sequence of s saccades over q AOIs as a $q \times s$ binary matrix. Each column of the matrix has a 1 in the position corresponding to the target AOI, and zero otherwise. If there are fewer than s saccades, the last column(s) are set to 0. In addition, the saccade sequence is represented as a sequence of saccades to relevant and non-relevant AOIs. This representation yields a $2 \times s$ binary matrix, with each column containing a 1 in the first row if the saccade target is a relevant AOI or a 1 in the second row if the saccade target is a non-relevant AOI. If there are fewer than s saccades, the last column(s) are set to 0. Therefore there will be alternating ones in each row, and zeros in the last columns for both rows if less then s saccades were made.

8. *Relative number of saccades to an AOI* is the saccade analogue of *AOI fixation percentage* and describes the relative number of saccades terminated to the AOIs during one trial. An image with q AOIs yields a q-dimensional feature vector with each entry counting the number of saccade targets to the corresponding AOI. The vector is normalized by the sum of all entries such that the entries sum to 1 unless there are no saccades. In that case, all entries are set to 0.

9. *Fixation latency to relevant AOI* describes the delay before fixating at a relevant AOI. It is defined as the duration from the trial start to the beginning of the first fixation at a relevant AOI.

10. *Saccade latency to relevant AOI* describes the delay before a saccade to a relevant AOI. It is defined as the duration from the trial start to the end of the first saccade to a relevant AOI.

The toolbox by default considers the sequence of AOIs visited during each trial, but there are many other sophisticated measures of long term temporal dynamics that may be appropriate in certain applications (Hayes et al., 2011). Therefore, our toolbox allows interested users to add any other variables of interest (see Appendix). After the variables are calculated and transformed to a numeric representation, they are aggregated into a single *feature vector* $\mathbf{x} = (x_1, x_2, \ldots, x_d)^T$ whose d entries correspond to the variables described. For clarity, *features* denote the entries of the feature vector which encode the eye tracking variables, while *variables* correspond to the measures of the eye-track enumerated above. Therefore, d is much larger than 10, because encoding certain variables requires multiple feature values. Furthermore, each trial results in its own feature vector.

Optimal Variable Selection

The goal of the toolbox is to determine which aspects of gaze systematically differ between two groups, or experimental conditions defined by the researcher. The key idea is to use binary classifiers to determine which gaze features best separate the two groups. Those measures of the gaze will typically become dependent variables for further analysis. The toolbox identifies the subset of features from the set defined above using the machine learning methods of Naive Bayes Ranking or L1 Logistic Regression. Each method is described in detail below.

Naive Bayes Ranking (NBR). NBR assumes that if the labeled feature vectors can be accurately classified given a single feature, x_i, then that feature separates the two classes well. In essence, the classification accuracy is a surrogate for the class separability achieved by the particular feature. Therefore, the features are ranked from best to worst according to decreasing classification accuracy.

The Bayes classifier assigns a sample, $\mathbf{x}$, to the class having the highest posterior distribution. More formally, assume that the class-conditional density functions of a feature, given its class, $p(x_i|y)$, are modeled as normally distributed with mean and variance, μ_{iy} and σ_{iy}, respectively. Then by applying the Bayes formula, the posterior probability of class y is $P(y|x_i) = \frac{p(x_i|y)P(y)}{p(x_i)}$, where $P(y)$ is the prior of class y, and $p(x_i)$ is a scale factor which ensures that the probabilities sum to 1. The toolbox assumes $P(y = 1) = P(y = 0) = 0.5$, corresponding to the assumption that *a priori* a sample is equally likely to come from either class. The scale factor is the same for both classes, so it can be omitted in the classification rule. Finally, the predicted class label, $\hat{y}$ is given by:

$$\hat{y} = \arg \max_{j \in \{0,1\}} p(x_i|y = j)P(y = j). \quad (1)$$

The classification accuracy used here and throughout is the leave-one-subject-out cross-validation (LOSO-CV) accuracy. In LOSO-CV, the samples belonging to one participant are sequestered, and the remaining samples are used to train the classifier. The sequestered samples are then classified with the learned classifier, and the procedure is repeated for every participant in the database. The total number of correctly classified samples divided by the total number of samples is the LOSO-CV accuracy.

L1 Logistic Regression (L1-LR). L1-LR is a linear classifier model, which returns a probability that a sample belongs to a particular class. It accomplishes this by modeling the natural logarithm of the ratio, or odds, of two probabilities as a linear function of $\mathbf{x}$. More formally,

$$ln\left(\frac{p(y=1|\mathbf{x})}{1-p(y=1|\mathbf{x})}\right) = \mathbf{w}^T\mathbf{x} - b, \tag{2}$$

where ln denotes the natural logarithm. The two class probabilities are then given by

$$p(y=1|\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x}+b)} \tag{3}$$

$$p(y=0|\mathbf{x}) = \frac{\exp(-\mathbf{w}^T\mathbf{x}+b)}{1+\exp(-\mathbf{w}^T\mathbf{x}+b)}. \tag{4}$$

The parameters defining the linear function, $\mathbf{w}$ and b, are estimated via Maximum Likelihood (ML) estimation. A regularization term λ is introduced to penalize large elements in $\mathbf{w}$, the vector of linear function coefficients. Using an L1-norm regularizer yields a sparse model. More formally, the regularized ML objective is,

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w},b} \sum_{i=1}^{N} logP(y_i|\mathbf{x}_i) - \lambda\|\mathbf{w}\|_1, \tag{5}$$

where $(\mathbf{x}_i, y_i), i = 1, \ldots, N$ are the full feature vectors and their associated labels, λ is a user determined real valued positive regularization parameter, and $\|\cdot\|_1$ denotes the L1-norm. Increasing the value of λ will result in more elements of $\mathbf{w}$ being shrunk to zero, i.e., a sparse weight vector $\mathbf{w}$. Variable selection is performed by increasing the value of λ until a desired number of $\mathbf{w}$ elements are non-zero. The elements of $\mathbf{x}$ corresponding to the non-zero elements of $\mathbf{w}$ are the top ranked variables. These top ranked variables can then be sorted from best to worst by sorting the corresponding entries of $\mathbf{w}$ in order of descending absolute magnitude. We use the L1-LR implementation of (Schmidt, 2011).

Normalizing and Labeling Trials. As is standard practice when classifying multivariate data, each feature x_i is first normalized to zero mean and unit variance over the entire data set. In addition, each trial feature vector $\mathbf{x}$ is associated with a class label, $y \in \{0, 1\}$, that specifies which group the trial belongs to. It is possible to automatically determine group labels for the trials using MATLAB's k-means algorithm. The k-means algorithm is an iterative algorithm that attempts to partition a set of vectors into clusters such that the within cluster variance is minimized. Therefore, k-means will generally label the trials such that the feature vectors having the same label have the smallest Euclidean distance from each other. This can be used as an exploratory method to automatically find groups within data.

Note that whatever method of labeling is used, binary classifiers ignore the ordering of trials and considers all trials within conditions at once. Therefore, it is possible that qualitative gaze changes that occur within a condition but from earlier to later trials will be ignored. To check for these types of changes across trials, the earlier and later trials should be labeled as 0 and 1, respectively. To ignore trials of a certain condition, recall that

trials labeled as 2 are ignored in the analysis. This allows a convenient method for checking qualitative differences between earlier and later trials within an experiment condition.

The following example commands show how to use the k-means clustering, and select the top 5 variables that classifies the two conditions best.

```
>> numKeep = 5;  % number of variables to select
>> ds.allLabels = autoLabel( features ); % optional k-means clustering
>> topIdxNBR = sortVariablesNB( features, ds.allLabels, ds.sampsPerSubj, numKeep );
>> topIdxLR = sortVariablesLR( features, ds.allLabels, numKeep );
```

In the first line, the *numKeep* variable take the number variable to select. The second line executes the optional k-means clustering procedure, and the last two lines select the top 5 variables using Naive Bayes Ranking and L1 Logistic Regression.

To show the top variable the *describeVariables* is used. The following example shows results taken from the L1 Logistic Regression.

```
>> describeVariables( varParams, topIdxLR );
```

Variable Validation

Once the variables that distinguish the experimental conditions are identified, we use them to classify each trial's group membership. This requires that we train a classifier to distinguish between two classes of data. Recall that each trial results in a feature vector, or *sample* $\mathbf{x}$. A classifier defines a decision rule for predicting whether a sample is from class 0 or 1. We use linear classifiers because of their ease of interpretation (Martinez & Zhu, 2005) – the absolute model weights give the relative importance of the eye tracking variables. We illustrate in Fig. 3 with a 2-dimensional linear classifier model specified by $\mathbf{w}$ and b. $\mathbf{w}$ is the normal vector of the hyperplane which separates the feature space into two decision regions, and b is the distance from the origin to the hyperplane (i.e., the offset).

All samples $\mathbf{x}$ above the hyperplane are assigned to class 1 while the samples below are assigned to class 0. Data samples $\mathbf{x}$ existing on the boundary satisfy $\mathbf{w}^T\mathbf{x}-b=0$. Therefore, samples are classified according to the sign of $\mathbf{w}^T\mathbf{x}-b$. In this example $\mathbf{w}=(-.55,.83)^T$, so the second dimension, x_2, is more informative for classification. Note that in our case the feature space has not two but up to 264 dimensions, depending on the cut-off for variable selection.

Several varieties of linear classifiers exist. The toolbox uses the Bayes classifier with equal covariances, L1-LR, or the Support Vector Machine (SVM) algorithm.

Bayes with equal covariances (Bayes). When both classes are assumed to be multivariate normally distributed with the same covariance Σ, means μ_1 and μ_0, and equal priors, the Bayes classifier decision boundary is a hyperplane given by $\mathbf{w}=\Sigma^{-1}(\mu_1-\mu_0)$ and $b=\frac{1}{2}\mathbf{w}^T(\mu_1-\mu_0)$ (Duda, Hart, & Stork, 2001).

L1 Logistic regression. Recall that L1-LR yields a probability that a sample belongs to a particular class. It uses the model of Equation (2), where $\mathbf{w}$ defines the normal of the hyperplane, and the sign of $\mathbf{w}^T\mathbf{x}-b$ determines the class label.

Support Vector Machine. SVM is a linear classifier which maximizes the margin between two classes of data (Burges, 1998). If the training samples are perfectly separable by a hyperplane, we can find $\mathbf{w}$ and b such that the data satisfy the following constraints,

$$\mathbf{x}_i^T\mathbf{w}-b\geq 1 \text{ for } y_i=1 \tag{6}$$

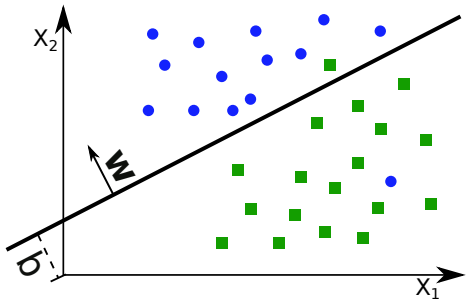

Figure 3. Illustration of a linear classifier. $\mathbf{w}$ is the normal vector of the hyperplane which separates the feature space into two decision regions, and b is the distance from the origin to the hyperplane. The blue circles represent samples from class 1, while the green squares represent samples from class 0. All but one of the blue circles exists on the positive side of hyperplane, and are classified correctly.

$$\mathbf{x}_i^T \mathbf{w} - b \leq -1 \text{ for } y_i = 0 \tag{7}$$

Essentially, these constraints specify that the samples from the different classes reside on opposite sides of the decision boundary. The margin between the classes, defined by $\frac{2}{\|\mathbf{w}\|_2}$ where $\|\cdot\|_2$ defines the L2-norm, is then maximized subject to the above constraints. The dual formulation of the constrained optimization problem results in a quadratic program for $\mathbf{w}$ and b. In the case that samples from each class are not linearly separable, a penalty is introduced to penalize the amount that a sample is on the wrong side of the hyperplane. Again, the dual formulation results in a quadratic program for $\mathbf{w}$ and b (Chang & Lin, 2001).

To calculate the LOSO-CV accuracy and wight vector for a particular linear classifier, use any of the following MATLAB commands. The commands corresponds to the three linear classifier (i.e., Bayes, L1-LR, SVM) in order.

```
>> [aLDA,wLDA] = runLDALeave1Out(features(topIdxLR,:),ds.allLabels,ds.sampsPerSubj);
>> [aLR,wLR] = runLRLeave1Out(features(topIdxLR,:),ds.allLabels,ds.sampsPerSubj);
>> [aSVM,wSVM] = runSVMLeave1Out(features(topIdxLR,:),ds.allLabels,ds.sampsPerSubj);
```

Recall that the weight vectors give the relative importance of the variables in determining which group trials belong to. In practice, however, more rigorous statistical tests are warranted. Thus, the toolbox provides a function that exports top variables into a tab delimited text file for further analysis with standard statistical packages, as shown in Table 2. To export the top variables selected using L1-LR to the file 'resultFile.txt', run the MATLAB command,

```
>> exportToTable( ds, features(topIdxLR,:), 'resultFile.txt'␣)
```

Case Study: Adult category learning

To demonstrate the performance of the GVS Toolbox we used an adult category learning experiment from Yim, Best, and Sloutsky (2011). Adults were ask to learn a single category while their eye-gaze was recorded using a Tobii T60 eye-tracker (Falls Church, VA) at the sampling rate of 60Hz. In one condition adults were instructed that there was a single

subject	trial	label	var_1	var_2
1	1	0	965.02	0
1	2	0	1282.2	0
		⋮		
1	15	0	2668.5	0
1	16	0	8674.8	1
		⋮		
10	16	0	4630.9	1

Table 2
Example of an exported tab delimited text file of top 2 variables of 10 *participants conducting* 16 *experiment trials. The headings 'var_1' and 'var_2' correspond to the top two variables identified by the algorithm.*

feature defining a category (i.e., supervised) whereas in another condition no instructions were given (i.e., unsupervised). Category members were flower-like objects with six petals as in Fig. 4. Category members had a single defining petal that had the same color and shape, whereas other petals changed their color and shape in a binary fashion. Category A and B had the defining petal at the same location (i.e., bottom right) but with different color/shape, whereas category C and D had the defining petal at the same location (i.e., left) but with different color/shape.

The experiment had 8 blocks, with 8 learning trials followed by 4 testing trials in each block. In a learning trial, a category member (e.g., category A) was displayed in the center of the screen one at a time for 1.5 seconds each. In a testing trial, a novel category member of the to-be-learned category and a novel member of a contrasting category (e.g., category B) were presented side by side. Test stimuli were displayed on the screen until the participant made a decision via key press about which stimulus was a member of the learned category. The to-be-learned category remained the same for the first 4 blocks. A second to-be-learned category was introduced (e.g., category C), where the defining petal had a different color/shape/location, in the final 4 blocks without notice to the participant.

Results

Trials were grouped as supervised (class 1) versus unsupervised (class 0). This resulted in 768 supervised and 1472 unsupervised class samples for the learning phase, and 384 supervised and 736 unsupervised class samples for the testing phase in both the category A or B and C or D conditions. Each trial resulted in a 152-dimensional feature vector for the learning phase, and a 264-dimensional feature vector for the testing phase.

We identified the most important variables for separating the experiment conditions, and validated those variables using the three linear classifiers. The LOSO-CV error is reported as a function of the number of top features used for classification in Fig. 5. Recall that the features encode the eye tracking variables. The results show that a very small numbers of features yields a high classification rate, and including more features does not improve the accuracy. Using the top five L1-LR features and the Bayes classifier, we achieved LOSO-CV accuracies of 74% and 73% in the AB and CD learning conditions,

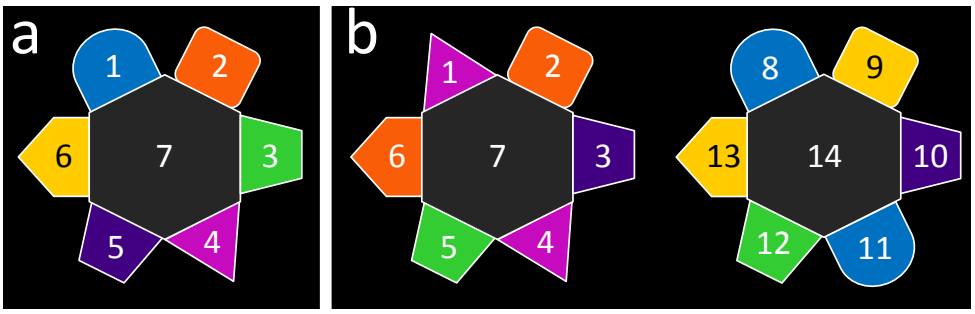

Figure 4. Example of category exemplars used during (a) learning trials (i.e., category A), and (b) test trials (i.e., category A on the left, and category B on the right). Numbers on the petals represents AOIs, which were not displayed to the participants. In the experiment category relevant features were – category A: a pink triangle at position 4(11); category B: a blue semi-circle at position 4(11); category C: an orange square at position 6(13); and category D: a yellow pentagon at position 6(13).

respectively. Similarly, we achieved 79% and 76% accuracy in the testing conditions.

The fact that the classifier shows stable performance just beyond a few features suggests that a small number of variables are sufficient for discriminating the two experimental conditions. The top five variables for NBR and L1-LR are listed in Table 3. We boldfaced the variables that were consistently ranked in the top five across both the category A or B and category C or D conditions and all feature selection algorithms. Note that AOI 4 for the category A or B condition is equivalent to AOI 6 in the category C or D condition. The consistently top ranked variables in the learning condition were *latency to a fixation at the relevant AOI* and *fixation density at the relevant AOI*. The top variables in the testing condition were *first and second fixations* as well as *fixation density at the relevant AOI(s)*.

We checked the linear classifier weight vectors to further elucidate the relative merit of these variables and the relationship between the variable values and the experiment conditions. Here we focus on the results for the Bayes classifier with the top 5 features of L1-LR, but similar trends were found for the other classifiers. For the A or B learning condition, the top variable *fixation density at the relevant AOI* had the largest positive weight of 0.68, indicating that a majority of fixations at the relevant AOI is positively associated with supervision. The second ranked variable, *latency to a fixation at the relevant AOI* had a negative weight of -0.57, indicating that *shorter* latencies are associated with supervision. The final three variables had weights under 0.1 in magnitude, indicating that they were much less predictive of the experiment condition.

For the C or D learning condition, the relative magnitudes of the top two variables were consistent with the A or B condition. However, the third variable 1^{st} *fixation at AOI 5* had a weight of 0.27, indicating that early fixations at AOI 5 were associated with supervision. This may be because AOI 5 neighbors the relevant AOI for categories C and D. The last two variables, *fixation density at AOI 7* and *fixation density at AOI 1* had weights of -0.36 and -0.31, respectively. This indicates that *shorter* fixation times at these irrelevant AOIs are associated with supervision, a result consistent with the overall findings.

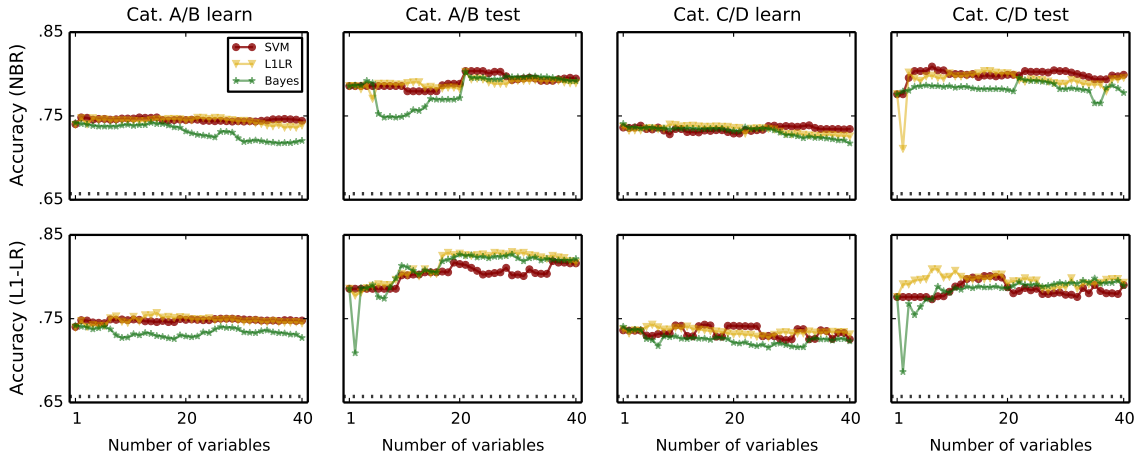

Figure 5. Leave-one-subject-out cross-validation accuracy for each classifier (i.e., SVM, L1-LR, and Bayes) as a function of the number of top ranked variables used for classification. The top row shows accuracy for the variables that were selected through NBR, and the bottom row for L1-LR. The most left column shows accuracy of the classifier when category A and B were presented in the learning trials, followed by their test trials, learning trials with category C and D, and their test trials. In almost all cases, the classification accuracy was near the maximum after including very few features and did not change much when including more. Dotted lines represent chance accuracy for the classifier (i.e., .657).

For the A or B testing condition, the variables 1^{st} *and* 2^{nd} *fixations at a non-relevant AOI* had weights -0.63 and -0.30, respectively, indicating that while the first fixations were more predictive, looking at a non-relevant AOI on the first two fixations was associated with a lack of supervision. The variable *fixation density at the relevant AOI* had a weight of 0.63, indicating that a larger proportion of fixations at the relevant AOI(s) is associated with supervision. For the C or D testing condition, 1^{st} *and* 2^{nd} *fixations at a non-relevant AOI* were again associated with a lack of supervision, with the first fixation being more informative. The weights were -0.75 and -0.58, respectively. *Fixation density at the relevant AOI* had a positive weight of 0.59, consistent with the A or B result.

In sum, the supervised and unsupervised conditions were best distinguished by how quickly the participants looked at the relevant AOI, and how long they maintained their gaze on it. The result relates to previous literature where successful category learning accompanies attention optimization to the relevant features (e.g., Rehder & Hoffman, 2005b; Kruschke, 1992; Nosofsky, 1986; Blair, Watson, & Meier, 2009), and implies that participants in the supervised condition were more successful in learning the category than in the unsupervised condition. Behavioral accuracy measures in Yim et al. (2011) also supports this fact where only a couple of participants were able to learn the category in the unsupervised condition.

	Learning condition	
	NBR	L1-LR
A or B	1. **Den of fix at AOI 4**	**Den of fix at rel AOI**
	2. **Lat to fix rel AOI**	**Lat to fix rel AOI**
	3. 1^{st} fix at AOI 4	1^{st} fix at rel AOI
	4. 2^{nd} fix at AOI 4	5^{th} fix at rel AOI
	5. 3^{rd} fix at AOI 4	Den of fix at AOI 1
C or D	1. **Den of fix at AOI 6**	**Den of fix at rel AOI**
	2. **Lat to fix rel AOI**	**Lat to fix rel AOI**
	3. 1^{st} sac to AOI 4	1^{st} fix at AOI 5
	4. 5^{th} fix at AOI 4	Den of fix at AOI 7
	5. 1^{st} fix at AOI 6	Den of fix at AOI 1

	Testing condition	
	NBR	L1-LR
A or B	1. 1^{st} **fix at non-rel AOI**	1^{st} **fix at non-rel AOI**
	2. 1^{st} sac to non-rel AOI	2^{nd} sac to non-rel AOI
	3. 1^{st} fix at rel AOI	2^{nd} **fix at non-rel AOI**
	4. **Den of fix at rel AOI**	**Den of fix at rel AOI**
	5. 2^{nd} **fix at non-rel AOI**	3^{rd} fix at non-rel AOI
C or D	1. 1^{st} **fix at non-rel AOI**	1^{st} **fix at non-rel AOI**
	2. **Den of fix at rel AOI**	2^{nd} **fix at non-rel AOI**
	3. 1^{st} fix at rel AOI	**Den of fix at rel AOI**
	4. 2^{nd} **fix at non-rel AOI**	2^{nd} sac to non-rel AOI
	5. 2^{nd} sac to non-rel AOI	Den of fix at AOI 1

Table 3

The variables above best separated the supervised and unsupervised groups. The bold face entries show variables that were consistently determined most relevant. NBR and L1-LR correspond to the different feature selection algorithms. AOI 4 is relevant in the category A or B condition, and corresponds to AOI 6 in the category C or D condition. We use the following shorthand convention: fixation (fix), saccade (sac), relevant (rel), density (den), and latency (lat).

Conclusion

We have developed a methodology for automatically identifying eye tracking variables that systematically differ between two groups or experimental conditions. Previous research subjectively determined which variables should be analyzed in a particular study. Instead, we used a formal methods that utilized machine learning techniques to find the important variables in an over-complete set of variables. The efficacy of the approach was verified with an adult categorization study.

References

Auger, F., Hilairet, M., Guerrero, J., Monmasson, E., Orlowska-Kowalska, T., & Katsura, S. (2013, Dec). Industrial applications of the kalman filter: A review. *Industrial Electronics, IEEE Transactions on*, *60*(12), 5458-5471. doi: 10.1109/TIE.2012.2236994

Blair, M. R., Watson, M. R., & Meier, K. M. (2009). Errors, efficiency, and the interplay between attention and category learning. *Cognition*, *112*(2), 330–336.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*, 121–167.

Chang, C.-C., & Lin, C.-J. (2001). LIBSVM: a library for support vector machines [Computer software manual]. (Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm)

Corbetta, M., Akbudak, E., Conturo, T., Snyder, A., Ollinger, J., Drury, H., ... Shulman, G. L. (1998, Oct). A common network of functional areas for attention and eye movements. *Neuron*, *21*(4), 761–773.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification (2nd edition)* (2nd ed.). Wiley-Interscience. Hardcover.

Feng, G. (2011, January). Eye Tracking: A Brief Guide for Developmental Researchers. *Journal of Cognition and Development*, *12*(1), 1–11. Retrieved from `http://dx.doi.org/10.1080/15248372.2011.547447` doi: 10.1080/15248372.2011.547447

Hannula, D. E., Ranganath, C., Ramsay, I. S., Solomon, M., Yoon, J., Niendam, T. A., ... Raglandemail, J. D. (2010, October). Use of eye movement monitoring to examine item and relational memory in schizophrenia. *Biological psychiatry*, *68*(7), 610–6. Retrieved from `http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2943005\&tool=pmcentrez\&rendertype=abstract` doi: 10.1016/j.biopsych.2010.06.001

Hayes, T. R., Petrov, A. A., & Sederberg, P. B. (2011, January). A novel method for analyzing sequential eye movements reveals strategic influence on Raven's Advanced Progressive Matrices. *Journal of vision*, *11*(10), 10. Retrieved from `http://www.ncbi.nlm.nih.gov/pubmed/21926182` doi: 10.1167/11.10.10

Hoffman, J. E., & Subramaniam, B. (1995, August). The role of visual attention in saccadic eye movements. *Perception & psychophysics*, *57*(6), 787–95. Retrieved from `http://www.ncbi.nlm.nih.gov/pubmed/7651803`

Holmqvist, K., Nyström, M., Andersson, R., Dewhurst, R., Jarodzka, H., & Van de Weijer, J. (2011). *Eye tracking: A comprehensive guide to methods and measures*. Oxford University Press.

Kalman, R. E. (1960, March). A New Approach to Linear Filtering and Prediction Problems. *Journal of Fluids Engineering*, *82*(1), 35–45. Retrieved from `http://dx.doi.org/10.1115/1.3662552`

Klin, A., Lin, D. J., Gorrindo, P., Ramsay, G., & Jones, W. (2009, May). Two-year-olds with autism orient to non-social contingencies rather than biological motion. *Nature*, *459*(7244), 257–61. Retrieved from `http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2758571\&tool=pmcentrez\&rendertype=abstract` doi: 10.1038/nature07868

Kruschke, J. K. (1992). ALCOVE: An Exemplar-Based Connectionist Model of Category Learning. *Psychological Review*, *99*(1), 22–44.

Martinez, A. M., & Zhu, M. (2005). Where are linear feature extraction methods applicable? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *27*(12), 1934–1944.

Martinez-Conde, S., Macknik, S. L., & Hubel, D. H. (2004, March). The role of fixational eye movements in visual perception. *Nature reviews. Neuroscience*, *5*(3), 229–40. Retrieved from `http://www.ncbi.nlm.nih.gov/pubmed/14976522` doi: 10.1038/nrn1348

McMurray, B., & Aslin, R. N. (2004). Anticipatory eye movements reveal infants' auditory and visual categories. *Infancy*, *6*(2), 203–229.

Murphy, K. (2004). *Kalman filter toolbox for matlab.* Retrieved from `http://www.cs.ubc.ca/~murphyk/Software/Kalman/kalman.html`

Nosofsky, R. M. (1986). Attention, Similarity, and the Identification-Categorization Relationship. *Journal of Experimental Psychology: General*, *115*(1), 39˜57.

Rayner, K. (1998). Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, *124*(3), 372–422.

Rehder, B., & Hoffman, A. B. (2005a). Eyetracking and selective attention in category learning. *Cognitive Psychology*, *51*, 1-41.

Rehder, B., & Hoffman, A. B. (2005b). Thirty-something categorization results explained: Selective attention, eyetracking, and models of category learning. *Journal of Experimental Psychology: Learning, Memory and Cognition*, *31*, 811–829.

Riby, D. M., & Hancock, P. J. B. (2008, September). Viewing it differently: social scene perception in Williams syndrome and autism. *Neuropsychologia*, *46*(11), 2855–60. Retrieved from http://www.sciencedirect.com/science/article/pii/S0028393208001917 doi: 10.1016/j.neuropsychologia.2008.05.003

Saez de Urabain, I., Johnson, M., & Smith, T. (2014). Grafix: A semiautomatic approach for parsing low- and high-quality eye-tracking data. *Behavior Research Methods*, 1-20. Retrieved from http://dx.doi.org/10.3758/s13428-014-0456-0 doi: 10.3758/s13428-014-0456-0

Salvucci, D. D., & Goldberg, J. H. (2000). Identifying fixations and saccades in eye-tracking protocols. In *Etra '00: Proceedings of the 2000 symposium on eye tracking research & applications* (pp. 71–78). New York, NY, USA.

Schmidt, M. (2011). *L1general - matlab code for solving l1-regularization problems.* Retrieved from http://www.di.ens.fr/~mschmidt/Software/L1General.zip

Stampe, D. M. (1993). Heuristic filtering and reliable calibration methods for video-based pupil-tracking systems. *Behavioral Research Methods, Instruments, & Computers*, *25*(2), 137–142.

Tanenhaus, M. K., Spivey-Knowlton, M. J., Eberhard, K. M., & Sedivy, J. C. (1995). Integration of visual and linguistic information in spoken language comprehension. *Science*, *268*(5217), 1632–1634.

Yim, H., Best, C. A., & Sloutsky, V. M. (2011). Cost of Attention as an Indicator of Category Learning. In L. Carlson, C. Hölscher, & T. Shipley (Eds.), *Proceedings of the 33rd annual conference of the cognitive science society* (pp. 1724–1729). Austin, TX.

Appendix

Data structures

The toolbox uses two basic data structures to organize imported data and variable properties. They are 'ds', and 'varParams', as described below. We assume N trials total from p participants, and d variable types.

```
ds % returned by 'loadFromTable' function
    ds.trackCell % (1 by N) cell of gaze coordinate complex column vectors (x+yi)
    ds.allLabels % (1 by N) matrix of group labels in {0,1,2}
    ds.sampsPerSubj % (p by 1) matrix of trial counts per participant

    Access entries
    ds.trackCell{2} % access 2nd trial gaze coordinates
    ds.allLabels(2) % access 2nd trial group label
    ds.sampsPerSubj(2) % access number of trials completed by 2nd participant

varParams % returned by 'extractVarsDist' function
    varParams.dimPerFeat % (d by 1) matrix of the number of features per variable
    varParams.dimNames % (d by 1) cell of descriptive variable names

    Access entries
    varParams.dimPerFeat(2) % access 2nd variable dimension
    varParams.dimNames(2) % access 2nd variable name
```

Changing Parameters

```
Display size
>> varParams.imageSize = [1024; 1280]; % display [height; width]

Fixation parameters
>> varParams.fixMinNumSamples = 6; % (6 samples)/(60 Hz)= 100ms
>> varParams.fixMaxCircleRadius = 15; % circle with radius 15px
>> varParams.numFixationInSeq = 7; % number of fixations to extract

Saccade parameters
>> varParams.velThreshold = 20; % 20px shift per sample to start saccade
>> varParams.stopThreshold = 8; % 8px shift during saccade to signal end
>> varParams.numSaccadesInSeq = 7; % number of saccades to extract
```

General toolbox use

```
Load and filter gaze
>> ds = loadFromTable( 'exampleTable.txt' );
>> dsUnfilt = ds;  % store unfiltered data
>> ds = smoothTracks( ds );

Visualize trial i for a stimulus image saved as 'stimulus.bmp'
>> visualizeTrackDist(  ds.trackCell{i}, dsUnfilt.trackCell{i}, 'stimulus.bmp', varParams )

Extract variables
>> aoiPositions = [20,40,80;40,80,160]; % [x1,x2,x3; y1,y2,y3]
>> aoiRadius = 20; % 20  pixel  radius  of AOI  center
>> relevantAOIs = [1;3];   % AOIs 1 and 3 are  relevant
```

```
>> [ features , varParams ] = extractVarsDist( ds.trackCell,...
    aoiPositions , aoiRadius , relevantAOIs , varParams  )

Optional clustering with k-means
>> ds.allLabels = autoLabel ( features );

Select and validate variables
>> topIdxLR = sortVariablesLR( features , ds.allLabels , 5 );
>> [aSVM ,wSVM] = runSVMLeave1Out(features(topIdxLR ,:),ds.allLabels ,ds.sampsPerSubj);

Describe and export
>> describeVariables( varParams , topIdxLR  );
>> exportToTable( ds, features(topIdxLR,:), 'resultFile.txt' )
```

Alternative filtering

It is possible to load pre-filtered data into MATLAB and skip the filtering step. Alternatively, assuming that a filter function 'customFilter' takes as an argument a complex vector of gaze coordinates and returns the filtered version, the eye-tracks can be filtered as follows:

```
>> for i1 = 1:length( ds.trackCell )
>>    ds.trackCell{i1} = customFilter( ds.trackCell{i1})
>> end
```

Adding new variables

We assume that a new variable function, 'extractVariablesNew' accepts the cell of complex gaze coordinates and returns a (dim, N) matrix of gaze features, where dim and N correspond to the number of new features defining the new variable and N is the total number of trials. Also assume that the original features have been extracted using,

```
>> [ features , varParams ] = extractVarsDist( ds.trackCell,...
    aoiPositions , aoiRadius , relevantAOIs , varParams  )
```

New features can be added to the framework as follows:

```
>> newFeatures = extractVariablesNew( ds.trackCell );
>> features = [features; newFeatures]; % combine old and new features
>> varParams.dimPerFeat(end+1) =  dim; % new dimensions
>> varParams.dimNames{end+1} = 'customVariable'; % variable description
```