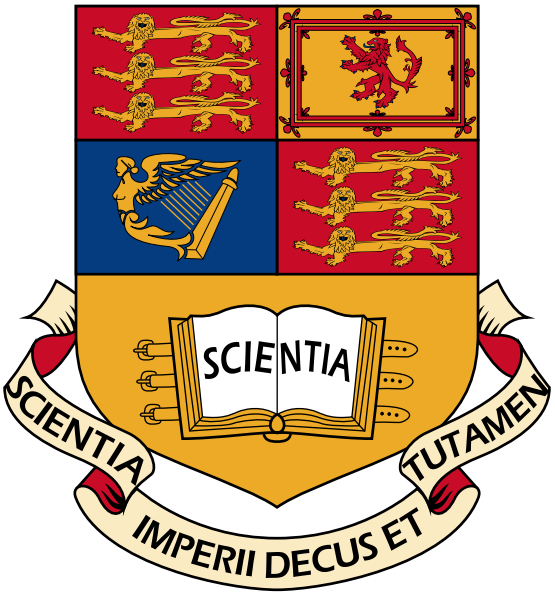
Imperial College London

Déjà Vu: Classification of Memory Using Eye Movements

June 16 2015



Author:
George Nishimura

Supervisor:
Dr Aldo Faisal

Second Marker:
Dr Marc Deisenroth

Abstract

This project aims to classify recognition without awareness using eye movement. We employ popular machine learning techniques to process and organize eye movement to try to solve the binary classification problem "has the participant seen the image before?". We build an application to run trials to collect eye-tracking data and produce a data set of eye movement data with both image and subimage recognition features. Initial analysis of the data set shows there are significant statistical differences between first and second viewing. We use and compare two clustering techniques to automatically generate areas of interests for each category of image, and represent eye movement as strings, Markov chains, and Hidden Markov Models in order to classify the two groups. We achieve a highest result of 68.7% which is lower than the participant average of 72.8% but in general results indicate promise for solving this problem.

Acknowledgements

There are a large number people without whom this report will have never been completed. Firstly, I would like to acknowledge the guidance and mentorship of my supervisor, Dr. Aldo Faisal, who I am grateful for spending the time to introduce me to machine learning, running trials and analysing data. I would also like to thank William Abbott, for taking time away from his PhD to help me navigate the overwhelming amount of literature on the subject of eye movement. I am also grateful for the leadership throughout the year of all the lab's PhD students, most notably Feryal Mehraban Pour Behbahani and Andreas Thomik.

I am grateful to the advice and feedback of Dr. Marc Deisenroth, who helped me understand and communicate the field of machine learning.

I reserve a special mention for my fellow undergraduate and Masters students of the Brain & Behaviour Lab, who consistently kept my spirits high and gave me advice during our group meetings. I appreciate the months of patience as I worked towards a data collection protocol in secret, and the overwhelming support when I needed to find participants for the trials.

Finally I would like to thank my friends, my family and my peers, as their emotional support and patience was critical throughout the year.

Contents

Chapter 1

Introduction

> Memory, my dear Cecily, is the diary that we all carry about with us.
>
> *The Importance of Being Earnest*
> Oscar Wilde

Memory is a critical function of human nature. What we remember shapes how we think, who we are and how we see the world. As Oscar Wilde wrote, it is a a "diary we all carry about with us" whatever we do. But how memory is *written* and how it is *read* are questions that have teased both scientists and philosophers alike. This project is an investigation into an indirect method of accessing memory: the eyes.

Commonly, the interface we associate with memory is explicit: a thought in our consciousness that we can express either orally or through text. However, there are many examples in daily life of functions and activities we can all perform but would find it difficult to express; imagine writing down the steps to tying your shoes for example. This *implicit* memory can exist and be accessed without conscious recognition.

Our eyes are one of the principal ways we understand and communicate with the external world. With its proximity to the brain, it has a direct channel to neurological systems to control where we look and what we look at. At least one of these neurological systems is influenced by our memory, and there is growing research into using eye movement as a means of accessing these systems; in other words, using our eyes as an index into our memory. There would be numerous benefits including: providing another means to investigate memory and its implementation in the brain; giving people who do not have the ability to express their memory explicitly an increased opportunity to understand their past; make our understanding of how we look and what we choose to see, or our model of eye movement, more robust and more accurate.

This project will investigate whether our eye movement can be used as a measure of recognition independent of explicit recognition. We will pursue these objectives by finding the models and features of our eye movement which best identify whether an object we observe is an object we have seen before. In order to accomplish this task, we will need to collect and analyse data for our models.

1.1 Motivation

The pairing of eye movement and the brain is not novel to this investigation; as discussed in our review of existing research on the subject in section 2.3.1, eye movement has been a successful

tool to understand the mind. Especially with regards to memory, eye movement can display implicit biases and involuntary displays of knowledge, sometimes without our awareness. The decision to target memory without awareness, rather than memory, was twofold:

1. to create a more robust model of memory, rather than human behaviour
2. to orient ourselves towards supplementing the brain rather than mimicking it

The first motivation is to make a clear distinction between classification of eye movement behaviour as controlled implicitly by neurological feedback loops and classification of eye movement as a display for human behaviour. We want to access the full domain of memory rather than the range consciousness has access to. This is a subtle distinction but an important one; when we analyse the performance of our classification methods, we do not train nor base our success around comparison with the participant. However, we are interested in the relationship between the two as they are inextricably linked. The second motivation expands upon that link.

If we return anecdotally to the importance of memory in our daily lives, there are often situations when insufficient memory plays a negative role. Forgetting something can come at the cost of a roundtrip back home, a missed date or a fatal mistake. For many sufferers of neurological impairments and damage, an ill-functioning memory is a significant hardship. If eye movement can be a measure of recognition independent of all the neurological systems that need to function in order to *experience* recognition, the people who have suffered damage to those systems will have another means of accessing their memory and piece together something they will have otherwise lost. Unfortunately the scope of the project did not able us to accomplish either goal, but we hope this work builds upon the foundation laid before and guides further work towards those goals.

1.2 Objectives

Memory, as a component critical to human activity but as a subject naturally difficult to investigate, is a field filled with opportunities for novel techniques and cross-fertilization of proven techniques. With our investigation, we approach the problem as a data analysis problem, rather than as a neuropsychology experiment, and aimed to uncover new behaviours of eye movement and memory.

We formally set out and achieve the following three objectives:

- to build a platform to collect eye tracking data
- to collect a novel data set of eye movement for analysis
- to use the data set to better model the relationship between memory and eye movement

In addition to these objectives, and as the project progressed, opportunities to target peripheral objectives were introduced, including:

- to create a versatile tool to collect and display real-time eye tracking data
- to implement and evaluate the latest methodologies of scanpath representation

1.3 Contributions

This report details the following contributions:

- a novel data set of 1200 scanpaths from twenty participants for forty images (twenty repeated)
- an analysis of statistical differences between first and second image viewings (and subimage *seen* and *unseen* features)
- an evaluation and implementation of four representations of scanpaths for classification of eye movement
- results for sixteen different classifiers, with a highest accuracy of 68.7% for classifying *seen* and *unseen* images
- an easy to configure and ready to play application for collecting and displaying eye-tracking data

1.4 Structure of the Report

This report covers the work and findings of our investigation as we followed its principal and peripheral objectives. In order to properly communicate the work, it will first be necessary to outline the existing research and relevant practices in machine learning (section 2.1), eye-tracking (section 2.2) and memory (section 2.3). Many of the techniques we will use will refer to topics in this section.

The implementation of the project is divided into two sections: Chapter 3 Data Collection and Chapter 4 Classification. Data Collection refers to the application and outcome of the aggregation of eye movement data for our novel data set. We will cover both an overview of the eye-tracking application we engineered in section 3.1 and the procedure, stimulus and results of our trials in section 3.2. Classification will cover the algorithms and techniques we employ to extract recognition from eye movement and our efforts to understand the results. This includes an overview of how we pre-process the data in section 4.2 and four sets of classification techniques: a discriminative approach in section 4.3, a spatial approach in section 4.4, a string-edit approach in section 4.5 and using the Markov property in section 4.6.

Finally we conclude with an evaluation of our investigation in Chapter 5 and final remarks on our contribution and our hopes for the future in Chapter 6.

Chapter 2

Background

2.1 Machine Learning

Machine learning is the field of study centred around algorithms and techniques for identifying patterns and relationships in data [1][2]. With increasing availability of large amounts of data and computational processing, automatic methods, particularly those derived from statistics, have become more popular as a tool to sort, predict and understand data.

Machine learning is a broad umbrella split commonly into three subsections: supervised, unsupervised and reinforcement learning. In this investigation, we will not be using reinforcement learning, but will be applying techniques from both supervised and unsupervised learning.

The distinction between supervised and unsupervised learning is labelling; namely, supervised learning is concerned with mapping inputs to outputs, e.g. linear regression, finding the line of best fit for a set of points. Unsupervised learning is concerned with finding unforeseen patterns in data, e.g. clustering, grouping points located near each other. As this investigation is primarily a classification problem, which is in the domain of supervised learning, most of the techniques used will be from this subsection of machine learning. However, a few unsupervised techniques are useful for dealing with and organizing large amounts of data.

It is useful while discussing certain techniques to break down input, or data, into a set of *features*. A *feature* is an individual quantity or characteristic of the input; for example, if the input was a set of two-dimensional points, each point has the features x-position and y-position.

2.1.1 Probability Models

One of the principal explorations within machine learning is how to build accurate models from data. The data we have is the *observed data* and we are trying to model the systems which produced the data. However, these models are not exact or true recreations of the original system and therefore include a measure of uncertainty. The common measure is probability.

Using a probability model provides a measure of confidence of the model in representing the data, but requires finding the appropriate functions and parameters for the model.

2.1.1.1 The Normal Distribution

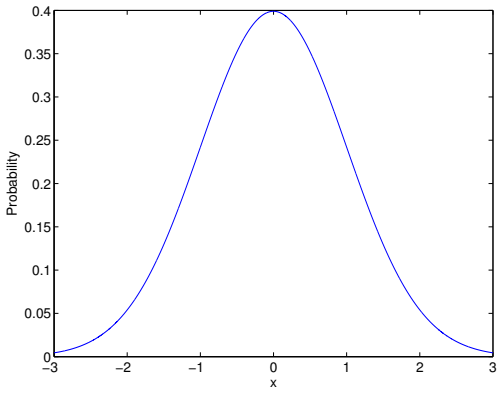
The normal, or Gaussian, distribution is a popular and useful probability distribution, with a probability density function $\mathcal{N}(x|\mu, \sigma^2)$ (pdf):

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{2.1}$$

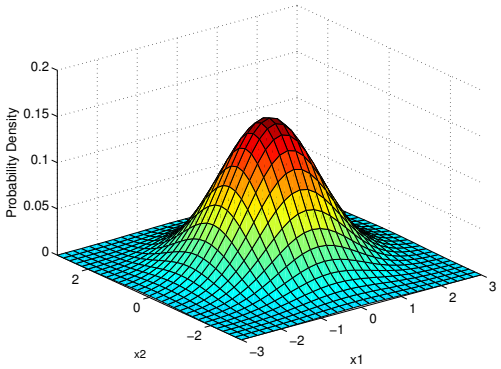
where x is the random variable, and μ and σ^2 are the parameters of the distribution, representing the mean and the covariance of the data respectively. The multivariate counterpart is:

$$\mathcal{N}_n(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{2.2}$$

where $\boldsymbol{x}$ and $\boldsymbol{\mu}$ are n-dimensional vectors and $\boldsymbol{\Sigma}$ is a n-by-n covariance matrix.



(a) Univariate normal ($\mu = 0$ and $\sigma^2 = 1$)



(b) Bivariate normal (zero mean, unit covariance)

Figure 2.1: Probability density functions for the (a) univariate and (b) bivariate normal distributions

Figure 2.1a and figure 2.1b show the pdfs for the univariate and bivariate normal distributions with zero mean and unit covariance.

2.1.1.2 Maximum Likelihood

Maximum likelihood estimation (MLE) is a mathematical technique to estimate the parameters that maximize the likelihood of a model. For example, for a model using a probability distribution, the parameters θ of the distribution can be estimated from the probability $p(x|\theta)$. We can take a look at an example using the univariate normal distribution in equation (2.1).

The parameter θ for the normal distribution is μ and σ^2. Using equation (2.1) for the univariate normal pdf, we can derive the likelihood of a set of $n$variables x as:

$$\mathcal{L}(x|\mu,\sigma^2) = \prod_{i=1}^{n} \mathcal{N}(x_i|\mu,\sigma^2)$$

The likelihood $\mathcal{L}$ represents the probability that each variable x_i was generated from the model with parameters μ and σ^2. If we assume to be individually and independently distributed (i.i.d), the probability of observing the entire set x is the product of the probabilities of each individual x_is.

Using the properties of exponents, and noticing that σ^2 is constant with respect to n, we can turn the product into a sum:

$$\mathcal{L}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma)^{\frac{n}{2}}} \exp\left(-\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{2\sigma^2}\right)$$

When working with probabilities, especially those from the exponential family, it is often easier to work with the log-likelihood $\log \mathcal{L}$ rather than $\mathcal{L}$ for two reasons: one, it prevents arithmetic underflow, as multiplication of very small fractions are replaced with addition of negative numbers; two, it gives us the opportunity to use properties of logarithms (which we use to turn the product into a sum).

Taking the log-likelihood l:

$$\begin{aligned}
l(x|\mu,\sigma^2) &= \log \mathcal{L}(x|\mu,\sigma^2) \\
l(x|\mu,\sigma^2) &= \log \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} \exp\left(-\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{2\sigma^2}\right) \\
l(x|\mu,\sigma^2) &= \frac{-n}{2}\log 2\pi + \frac{-n}{2}\log\sigma^2 + -\frac{1}{2}\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{\sigma^2}
\end{aligned}$$

For MLE, we find the optimal parameters by differentiating the function and setting the gradient equal to zero. Because the logarithm function is monotonically increasing, it will not change the maximums of the function. Therefore the parameters that maximize the log-likelihood will also maximize the likelihood function. For the normal distribution, we differentiate the log-likelihood with respect to each parameter μ and σ^2 and set it equal to zero to get:

$$\begin{aligned}
\hat{\mu} &= \frac{\sum_{i=1}^{n}(x_i-\mu)}{\sigma^2} \\
\hat{\sigma^2} &= \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n}
\end{aligned}$$

which is the empirical mean and empirical covariance respectively.

2.1.1.3 Maximum A Posteriori (MAP)

Maximum A Posteriori (MAP) is a parameter estimation technique that utilizes Bayes Theorem. It estimates the parameters θ from the data x by maximizing the posterior probability of θ given x. Bayes Theorem defines the posterior probability as:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

In a Bayesian framework, probability is a measure of belief, and the posterior represents our belief after observing x. Our belief about θ is proportional to what we believed before observing x (the prior $p(\theta)$) and the probability of θ producing x (the likelihood $p(x|\theta)$).

The optimal parameters are then calculated by finding the posterior probability for θ_M:

$$\begin{aligned}
\theta_M &= \arg\max_{\theta} p(\theta|x) \\
\theta_M &= \arg\max_{\theta} \frac{p(x|\theta)p(\theta)}{p(x)}
\end{aligned}$$

$p(x)$ will be constant for different values of θ so the equation can be simplified to:

$$\theta_M = \arg\max_{\theta} p(x|\theta)p(\theta)$$

The addition of the prior $p(\theta)$ is what separates the MAP estimate from the MLE estimate.

2.1.1.4 Expectation Maximization

Expectation Maximization (EM) is an iterative process to find the MAP or MLE estimate [3]. It is commonly used when computing the ML and MAP estimates directly is difficult, such as if there are latent, or hidden, variables.

If we define X to be our observed data set, Z to be our latent variables, and θ to be our parameters, the goal of EM is to maximize the joint log-likelihood:

$$l(\theta) = \sum_{i=1}^{N} \log p(X, Z|\theta) \tag{2.3}$$

EM iteration is broken down in to two steps: the E step and M step. At iteration step t, the E step aims to calculate the expectations of the parameters with respect to the observed data and the previous parameter values:

$$Q(\theta, \theta^{t-1}) = \mathbb{E}[l_c(\theta)|X, \theta^{t-1}] \tag{2.4}$$

Q is an auxiliary function. The M step finds the next set of parameters θ^t by maximizing the Q function with respect to θ:

$$\theta^t = \arg\max_{\theta} Q(\theta, \theta^{t-1}) + logp(\theta) \tag{2.5}$$

The algorithm iterates until θ_t converges to the optimal value θ_M. Convergence to a local optimum is guaranteed, as the maximum likelihood is strictly non-decreasing at each iteration, and increasing if the expectation of the auxiliary function increases.

2.1.2 Classification

Classification is a part of supervised learning whose goal is to assign inputs their respective discrete classes (labels). A simple example would be the function *is_even*, which given the domain of natural numbers as an input will *classify* each number as either even or odd. In other words, it applies a class label of *even* or *odd*.

is_even is an example of a binary classifier (as it has only two classes) and can be modelled by a well-known relationship: if the number is divisible by two. However, for this investigation, the relationship, or relationships, are not so easily known.

In order to model these ambiguous relationships, probability will be used. With probability, classification can be broken down in to a two-step problem:

1. infer the probability of a class
2. decide whether the label will be assigned

2.1.2.1 Naive Bayesian Classifier (NBC)

Naive Bayesian classifiers (NBC) is a set of simple classifiers built on the foundations of Bayesian probability. Bayesian probability is often regarded as a measure of belief; new data is interpreted as evidence, which affects a prior belief, to create a new posterior belief.

Naive refers to the treatment of the input data features as conditionally independent given the class label. In other words, the assumption that the different features of the data all contribute independently to the probability of the class label. This way, the probability of a class label given the data can be obtained from the posterior:

$$p(c|x) = \frac{p(x|c)p(c)}{p(x)}$$

It is unlikely that features are conditionally independent in real-life systems, however NBCs have been shown to be an effective estimation regardless [4]. To use NBCs, we combine the posterior probability with a decision rule such as "pick the class which maximizes the posterior":

$$\hat{c} = \arg\max_{c \subseteq C} p(c|x) = \frac{p(x|c)p(c)}{p(x)}$$

in other words, the MAP estimate of the class label.

Training the classifier, or model, is the process of using data to find the optimal parameters or distributions for a classifier. In the case of the NBC, it is finding the right likelihood and prior functions. For example, we could use the normal distribution as our likelihood function, which has parameters μ and Σ, if it fits our data well. This an example of a *parametric* and *generative* classifier; parametric as it uses fixed *parameters* of the training data rather than the entire training data itself, and generative as once we find the parameters, we can also *generate* likely observations. There are several techniques to finding the optimal parameters: maximum likelihood, maximum a posteriori and expectation maximization, which we discussed in section 2.1.1.2.

2.1.2.2 K-Nearest-Neighbour Classifier (KNN)

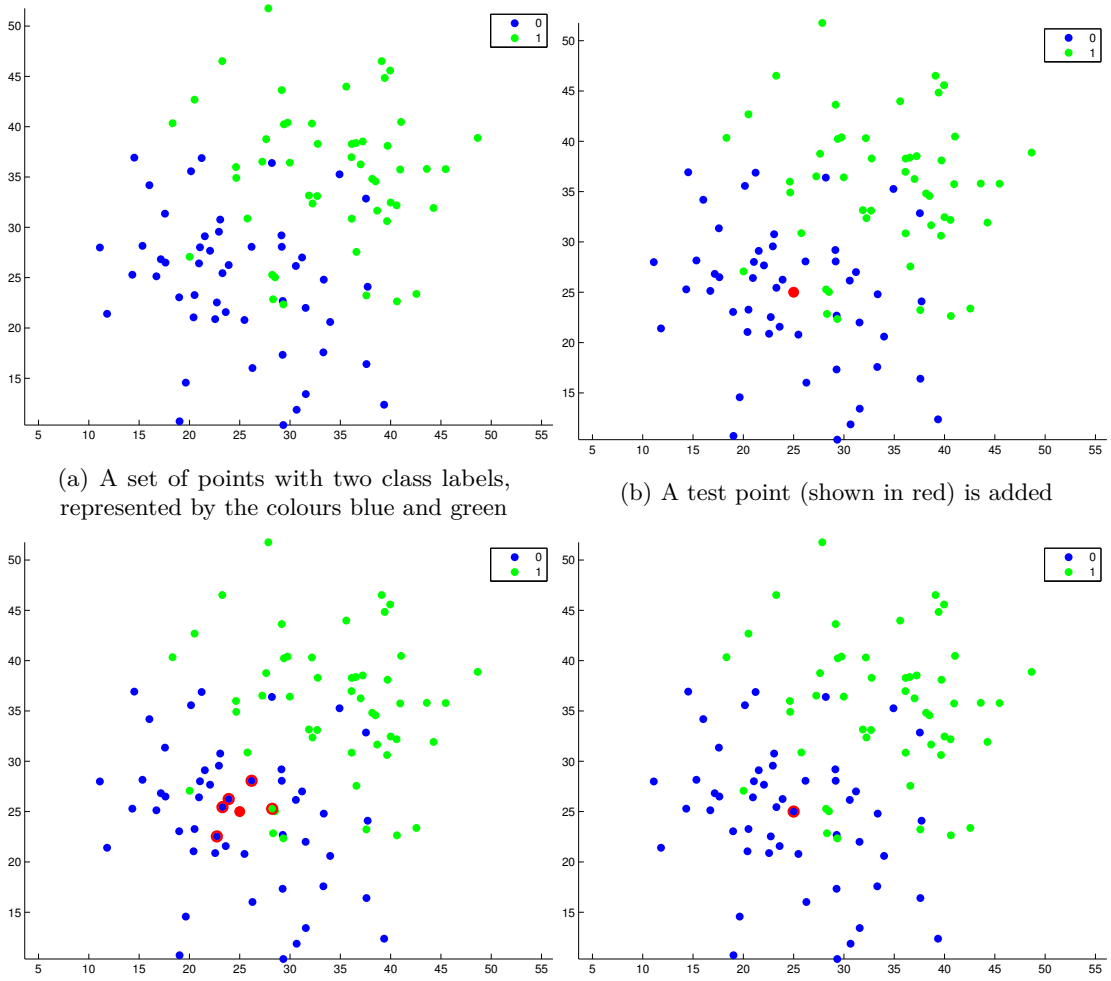
An example of a non-parametric classifier is the K-Nearest-Neighbour (KNN) classifier. KNN uses the entire training data set and some measure of distance such as Euclidean distance to label points based on their neighbours.

Figure 2.2a shows an example of a training dataset with two classes and K set to 5. To classify a new test point with KNN, such as the red point in figure 2.2b, the closest 5 neighbours from the training set are found using the distance metric. Figure 2.2c highlights the neighbours. The test point is assigned by a majority vote: the best represented class among its neighbours. A probability score can be calculated by dividing the number of neighbours with the majority class by K. In our simple example, the point is classified as blue in figure 2.2d, as four of its five neighbours (a probability of 80%) are blue.

As a binary classifier, an odd value for K is sufficient for avoiding any ties. However, to deal with outliers, a weighting can be added as an additional cost to classification. If the weighting was inversely related to distance, closer neighbours will have greater weight than further away neighbours.

2.1.2.3 Logistic Regression

A popular approach to binary classification is logistic regression, which unlike NBCs, is a *discriminative* parametric classifier. Discriminative means that logistic regression aims to classify



(a) A set of points with two class labels, represented by the colours blue and green

(b) A test point (shown in red) is added

(c) The five closest neighbours to the test point are found (outlined in red)

(d) The test point is coloured blue as majority of its five closest neighbours ar eblue

Figure 2.2: Example of K-NN binary classification of a test point where K is 5

by computing the conditional probability $p(c|x)$ directly, instead of marginalizing over the joint probability. The disadvantage is the inability to generate likely pairs from a joint probability.

Logistic regression aims to model the relationship between the inputs and outputs as a function. Specifically, it aims to model the relationship as a function of the form:

$$p(y|x, w) = Ber(y|sigm(wx))$$

where y is the label, x is the input, w are the parameter weights, Ber is the Bernouilli distribution and $sigm$ is the sigmoid or logistic function. The Bernouilli distribution $Ber(x|\theta)$ is a distribution such that:

$$Ber(x|\theta) = \begin{cases} \theta & if x = 1 \\ 1 - \theta & if x = 0 \end{cases}$$

The sigmoid, or logistic, function is a continuous function which returns a value between 0 and 1, and is characterized by its S-shape graph. The definition of the sigmoid function is

$$S(x) = \frac{1}{1 + \exp(-t)}$$

In combination with the Bernouilli distribution, the conditional probability $p(c|x, \boldsymbol{w})$ with weights $\boldsymbol{w}$ is

$$p(c|x, \boldsymbol{w}) = \frac{1}{1 + \exp(-w_0 + \hat{w}x)}$$

As this is a binary classification problem, if we combine this with a decision rule similar to what we suggested with NBCs (the MAP estimate of the class label), we will pick the label which has $p(c|x, \boldsymbol{w}) < 0.5$. This is true if

$$-w_0 + \hat{w}x > 0$$

and the decision boundary is located at

$$-w_0 + \hat{w}x = 0$$

2.1.3 Clustering

Clustering is a technique in unsupervised learning to group unlabelled points together based on a set of metrics, usually a form of distance.

For this section, we will show two strategies for clustering the points in figure 2.3a, which was created using three multivariate normal distributions with different means.

2.1.3.1 Soft and Hard Clustering

Clustering is concerned with assignment points to their appropriate clusters.

For hard clustering, the assignment is one-to-one; each point will be assigned to exactly one cluster. Soft clustering is stochastic; the responsibility for the point can be shared among several clusters. A soft-assignment can be turned in to a hard-assignment by finding the cluster with the largest responsibility for the point.

2.1.3.2 Mixture of Gaussians (MOGs)

A popular clustering technique is to use a mixture model of normal distributions referred to as a Mixture of Gaussians (MoG). Mixture models use clusters defined by probability distributions, and use soft-assignment based on the probability of the point belonging to each distribution.

A point is therefore not strictly assigned to a cluster, but assigned to several clusters, who share the responsibility for that point. Reponsibility is a latent, or hidden, variable of the model. This probability assignment is modelled as a posterior probability:

$$p(c_i = k|x_i, \boldsymbol{\theta}) = \frac{p(x_i|c_i = k, \boldsymbol{\theta})p(c_i = k|\boldsymbol{\theta})}{\sum_{j=1}^{K} p(x_i|c_i = j, \boldsymbol{\theta})p(c_i = j|\boldsymbol{\theta})}$$

where K is the number of clusters.

For a Mixture (model) of Gaussians, the likelihood function is the normal distribution, and the prior probability is the cumulative responsibility that distribution has across all the points. To find the optimal parameters, including the latent variables, EM (see section 2.1.1.4) is used.

Bayesian Information Criterion The downside to the classic MoG approach is that the number of clusters K needs to be defined *a priori*, which in some cases can be cumbersome. If we leave the number of clusters unbounded, we can *overfit* the data; i.e. make the model too specific to the training set that it does not generalize well to new data.

One technique to find the number of clusters is to use a measure of *model quality* to rank between models with different K values.

Bayesian Information Criterion is a measure to assess the quality of a model with respect to the dataset [5]. Formally defined:

$$BIC = -2\ln(l) + \alpha\log(n) \tag{2.6}$$

where k is the maximized likelihood $p(x|M, \theta_M)$ of model M and with parameters θ_M, x is the observed data, n is the number of observations and α is the number of free parameters.

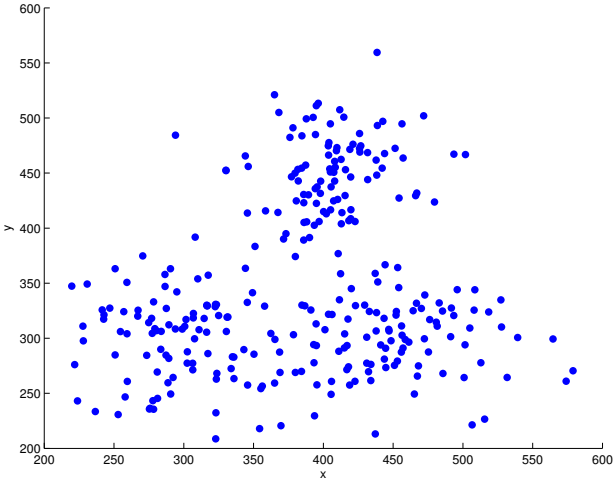
A lower BIC value indicates a better quality of fit. BIC dissuades overfitting by penalizing the number of free parameters used in the model (see α in equation (2.6)). A similar criterion to BIC is Akaike Information Criterion (AIC); however BIC discourages complex models more than AIC.

2.1.3.3 DeCarlo-Santella Algorithm (DCS)

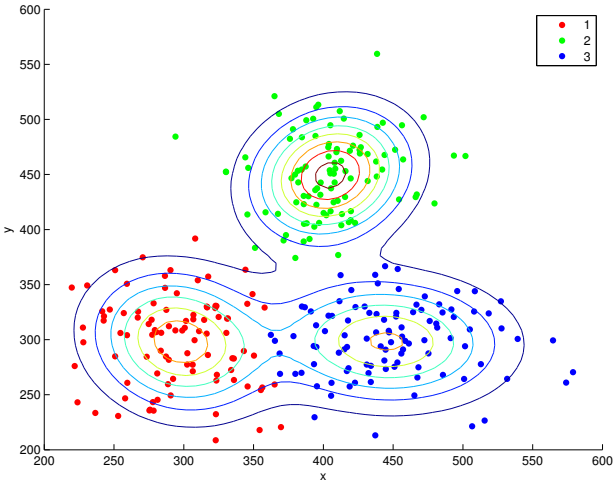
An alternative clustering algorithm is one suggested as an alternative to k-means (another clustering algorithm that we will not cover) [6]. The algorithm, which uses the mean-shift procedure, was brought to the eye movement community by DeCarlo and Santella. We will refer to this algorithm as the DeCarlo-Santella (DCS) algorithm for convenience, although the mean-shift procedure and its use as part of a clustering algorithm was first proposed by Fukunuga and Hostetler [7]. The main advantages of DCS is that it produces consistent results and the number of clusters does not have to be calculated in advance, which are weaknesses of other clustering techniques including MoGs.

The idea behind DCS is a two-step process:

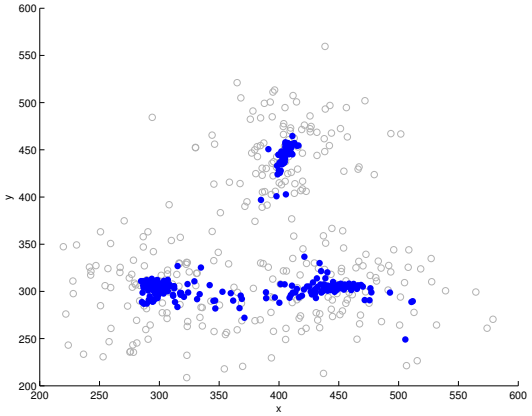
1. shift the points into a denser, easily separable configuration
2. cluster using a simple distance algorithm



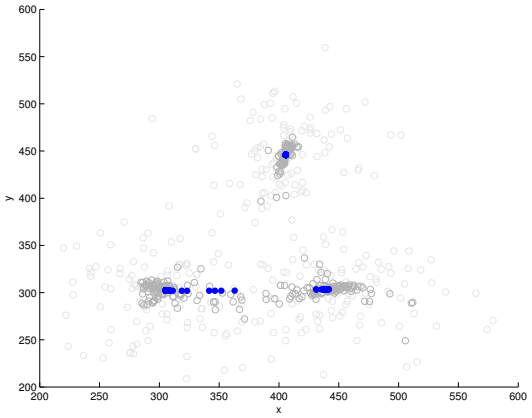
(a) Set of random points generated with three bivariate normal pdfs



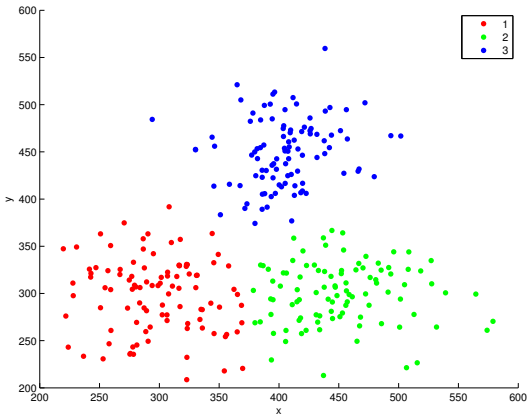
(b) Assignment after clustering using a Mixture of Gaussians where K = 3



(a) The points after four iterations of the mean-shift procedure. The original locations are shown in grey



(b) The points after six iterations of the mean-shift procedure. The original locations and the locations after two iterations are shown in light grey and grey respectively



(c) Assignment after clustering with the DeCarlo-Santella algorithm

In order to perform step one, the mean shift procedure is used. The procedure iteratively moves each point to the weighted mean of all points, until all points converge (to save computational performance the number of iterations is often capped) . The weight is the use of an interchangeable kernel function to avoid sensitivity to extreme outliers, typically a multivariate Gaussian kernel.

Once the points have converged to their local means, a distance based clustering algorithm can be applied. This creates clusters for every group of points separated by a certain threshold of distance.

2.1.4 Markov Processes

The Markov property states that given an ordered sequence of states, the probability of the current state can be determined sufficiently by the previous states.

$$p(s_t|\theta, s_{1:t-1}) = p(s_t|s_{1:t-1})$$

First and second order Markov properties enforce the stronger condition that the current state can be determined sufficiently by the previous and the previous two states respectively.

$$p(s_t|\theta, s_{1:t-1}) = p(s_t|s_{t-1})$$
$$p(s_t|\theta, s_{1:t-1}) = p(s_t|s_{t-1}, s_{t-2})$$

It is often more practical to also include that the probability is invariant to time; namely the probability of state s occuring at time t_1 is the same as the probability of it occuring it at any other time t_2, given the same previous one or two states (for first and second order Markov properties).

$$p(s_{t_1}|s_{t_1-1}) = p(s_{t_2}|s_{t_2-1})$$

A Markov process is a process that obeys the Markov property. The probability of a sequence of states S that obeys a first order Markov property can be decomposed in to a product:

$$p(s_{1:T}) = p(s_1)\prod_{t=2}^{T} p(s_t|s_{t-1})$$

In order to model a first order Markov process, we need to calculate the probability to transition from state s_{t-1} to s_t. This is typically captured in stochastic n-by-n transition matrix A, where n is the number of possible states, and A_{ij} is the probability of transitioning from state i to state j. The rows of A sum to 1.

A can be calculated using empirical data by counting the number of occurrences where state i transitions to state j, and normalizing across the rows. However, if the resulting matrix is sparse (i.e. with a large number of zeros), the transition probability may be zero. This may or may not capture appropriate behaviour. To avoid this problem, we can use add-one smoothing, or pre-populate each row with at least one occurrence (add one smoothing) [1].

2.1.5 Hidden Markov Models

A Markov chain is a useful model for sequences of states which we can observe, but is not suitable for situations where the state space is hidden, or *latent*. To model hidden state spaces, we want to use the hidden Markov model (HMM).

We can build the definition of HMMs using two previously discussed concepts: Mixture of Gaussians (MOG) and the Markov chain. In Mixture of Gaussians, we referred to the *responsibility* of a cluster, how much of each point is assigned to the cluster, as a latent variable. This was because the responsibility is not something we observe, but is implicitly modelled by our MoG. The observations are the actual data points, and the clusters' responsibility is the hidden state space.

Let us define a MoG with two clusters k_1 and k_2 and observe one data point x_1. The likelihood that x_1 belongs to cluster k_1 is the probability $p(x_1|k = k_1)$, which is just the normal distribution with parameters μ_1 and σ_1^2. In other words, we can think of it as the probability that k_i generated, or *emitted*, x_1. If we observe another data point, x_2, we can formulate the same thing: the probability x_2 was emitted by k_1 is $\mathcal{N}(x_2|\mu_1, \sigma_1^2)$. For a MoG, the probability of observing both x_1 and x_2 is the product of the individual posterior probabilities, where the individual posterior probability for k_1 is:

$$p(k = k_1|x_1) \propto p(x_1|k = k_1)p(k_1)$$

where the prior probability $p(k_1)$ is the proportion of points cluster k_1 is responsible for. In the MoG model, the probability of x_2 belonging to k_1 is independent of the probability x_1 belongs to k_1. The prior probability $p(k_1)$ does not take into account any of the history of the previous observations (after training). However, if there was dependency between states, we would have to update the prior in our equation:

$$p(k_t = k_1|x_1) \propto p(x_1|k_t = k_1)p(k_t = k_1|k_{1..t-1})$$

With the first order Markov property, the prior probability becomes a product of one-step transition probabilities, or:

$$p(k_t = k_1|x_1) \propto p(x_1|k_t = k_1)p(k_1)\prod_{i=2}^{t} p(k_i|k_{i-1})$$

This is the posterior probability of the HMM, which combines the emission probability of MoG with the transition probabilities of the Markov chain. A graphical illustration of a HMM is shown in figure 2.5

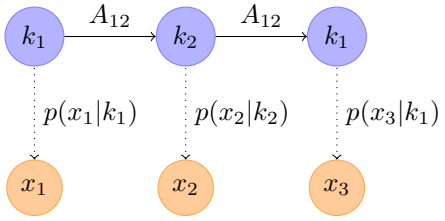


Figure 2.5: A model of a HMM with a sequence of hidden states space $\{k_1, k_2, k_1\}$ emitting the observations $\{x_1, x_2, x_3\}$

Hidden Markov Models have discrete latent states, $\bar{Z} = z_1, ..., z_t$ and either continuous or discrete observations $\bar{X} = x_1, ..x_t$. As a generative model they can be used to generate samples and compute the likelihood of a sample (using log-likelihood), as well as make inferences or predictions. To find the parameters of the model, the EM algorithm (for HMMs also known as the Baum-Welch algorithm) can be used.

2.1.6 Evaluation

An important part of any investigation is how to measure performance. For probability models this is especially critical, as they are only approximations of real-world systems. For binary classification, an obvious measure is accuracy, where

$$Accuracy = \frac{number\ of\ correct\ predictions}{number\ of\ predictions}$$

While accuracy is a useful metric, it does not portray the full picture; for example, let us say we built a model to predict coin tosses with the following rule:

$$for\ all\ inputs,\ predict\ heads$$

If we tested the model by tossing the coin 10 times, and observe heads 8 times, we can calculate an accuracy of 80%. This would seem to imply a successful model (as we would expect a random model to achieve around 50% accuracy). Such naive evaluations can distort the validity of a model.

In order to provide a greater breakdown of performance, a confusion matrix can be used.

A confusion matrix is a four-celled table representing the four values of binary classification:

- **true positives (TP)** - the number of correct positive predictions
- **false positives (FP)** - the number of incorrect positive predictions
- **true negatives (TN)** - the number of correct negative predictions
- **false negatives (FN)** - the number of incorrect negative predictions

The distinction between a *positive* and *negative* prediction is arbitrary for many binary classification problems, including ours. A confusion matrix then has the following form:

TP	FP
FN	TN

From these values, we can calculate four derived performance metrics, including the aforementioned *accuracy*:

Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	the proportion of correct predictions
Sensitivity	$\frac{TP}{TP+FN}$	also known as the true positive rate, the proportion of positive conditions that were correctly predicted
Specificity	$\frac{TN}{TN+FP}$	also known as the true negative rate, the proportion of negative conditions that were correctly predicted
Precision	$\frac{TP}{TP+FP}$	the proportion of positive predictions that were correct

We can model the relationship between the true positive and false positive rate (or 1 - the true negative rate) as a curve called the Receiver Operating Characteristic (ROC) curve. The ROC curve shows us the performance of the classifier as the threshold for prediction is increased. The diagonal represents an ideal random classifier (50%), with anything above the line showing better

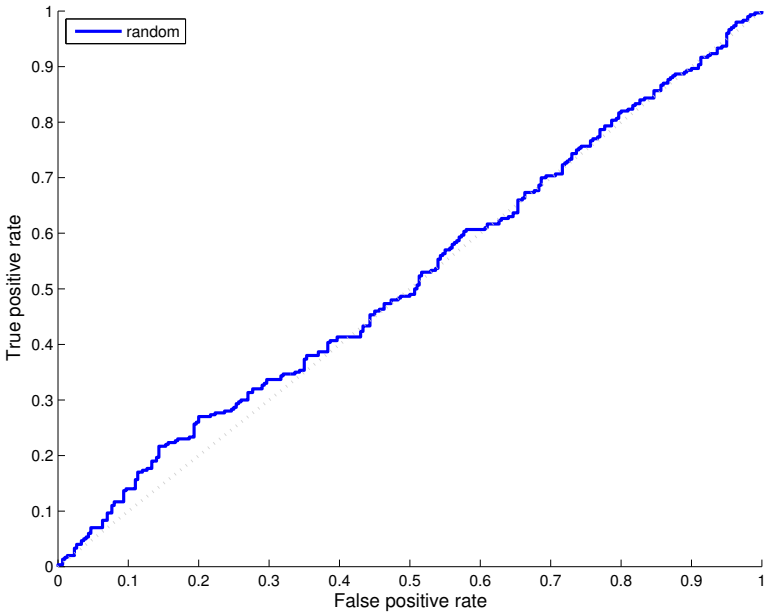


Figure 2.6: A ROC curve for the random classifier

than random performance. Figure 2.6 displays the ROC curve of an actual random classifier. A perfect classifier would be a single point in the top left corner, as the true positive rate would be 100% and the false positive rate 0%.

The area under the ROC curve (AUC) is a a measure of how probable a positive sample will be classified as positive. Thus an AUC score of 1 represents perfect positive classification.

2.1.6.1 Cross Validation

In order to get values for the confusion matrix and ROC curves, we need to test our classifiers with real-world inputs, and compare them with the correct labels. However, in order to build an accurate model, we need to train it, usually by feeding it real-world data and its correct labels. Training and testing data need to be separated; otherwise it calls into question the validity of the evaluation (the success of parametric classifiers like KNN would artificially skyrocket).

To maximize the utility of our data, we can use cross-validation to recycle our data as both testing and training data. For example, 2-fold cross validation would involve splitting the data into two sets D_1 and D_2 and perform two evaluations with the following arrangement:

1. Train the model with D_1, test against D_2
2. Train the model with D_2, test against D_1

The values can be aggregated and averaged to provide an overall score. A common implementation of cross-validation is Leave-One-Out Cross Validation (LOOC). With ten data sets, LOOC involves running 10 validations, where one data set per validation is reserved for testing, and the other nine used for training. The advantage of LOOC, besides simplicity, is that it is often advantageous to use as much data as possible for training the model (for robustness and accuracy).

2.1.7 Tools

Our analysis of the data was done using Matlab version 2014a/b [1]. Many of the implementations of machine learning techniques can be found either in the Statistics and Machine Learning Toolbox [2] or Kevin Murphy's HMM toolbox [3].

2.2 Eyes

Our investigation is concerned with questions of *why* our eyes move; however we must first address the question of *how* our eyes move and explore the basic mechanics of the eyes.

2.2.1 Eye Physiology

Eyes are principally concerned with visual perception, or the process to convert light energy into nerve signals [8]. In order to facilitate this process, the cornea, at the front of the eye, focuses incoming light. The light then passes through the pupil, whose diameter is controlled by the iris, on towards and through a crystalline lens. Ultimately, the light energy is picked up at at the back of the eye by the retina.

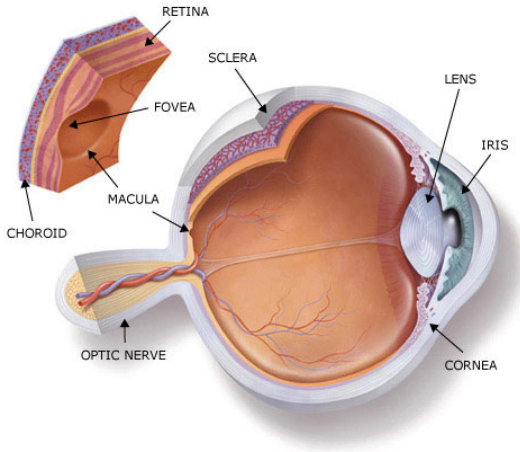


Figure 2.7: Anatomy of the eye. Taken from `http://www.aapos.org/terms/conditions/22`

The retina houses "sensors" which convert the incoming light energy in to electrical energy. There are two types: rods and cones. The physical differences between rods and cones leads to different functions: rods are sensitive to low-level, achromatic light (such as seeing in the dark) and cones are sensitive to high-level chromatic light (such as normal daytime vision) [9]. The retina contains both rods and cones; however the total distribution is not equal (there are roughly 120 millions cones to only 7 million rods) nor is the spatial distribution equal across the retina. A particularly important subsection of the retina is called the fovea, where there is a highly dense collection of cones. The fovea therefore is the part of the eye which has the highest focus as it maximises the ability to sense high levels of chromatic light energy.

The retina sends the electric nerve signals to the optic nerve, which transports it to the visual cortex in the brain.

[1] http://uk.mathworks.com/products/matlab/
[2] http://uk.mathworks.com/products/statistics/
[3] http://www.cs.ubc.ca/ murphyk/Software/HMM/hmm.html

2.2.2 Visual Field

Because of the unequal distribution of rods and cones in the retina, our visual field is not uniform. The fovea, with its dense population of cones, represents our best eyesight: the portion of the visual field most in focus. The fovea is about 1-5 degrees of the visual angle and only 6-8% of the entire visual field [10]. Outside the fovea, the para-foveal and peripheral area are dominated by the low-resolution rods, which leads to a dramatic drop-off in terms of resolution [11].

Because of the non-uniform distribution, only a small percentage of our vision can be at focus, so physical movement of the eye is signficant: eye movement is the mechanism for which our eyes keep or put objects in our visual field in focus. Therefore the study of eye movement is an investigation into how or why objects are put into that focus.

2.2.3 The How of Eye Movement

There are six muscles which can move the eye and six corresponding degrees of freedom: sideways movement is performed by the medial and lacteral recti, vertical movement is performed by the superior and inferior recti, and twists are performed by the superior and inferior obliques [9].

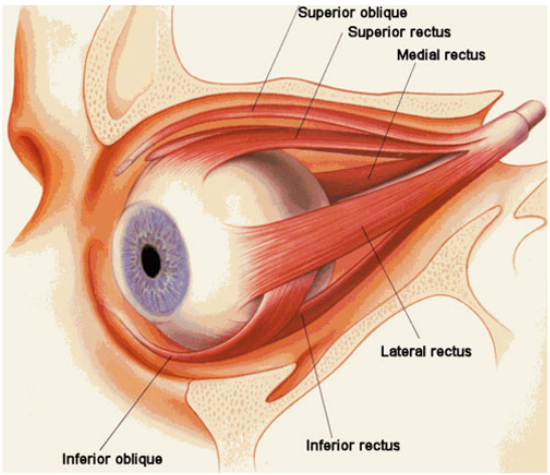


Figure 2.8: Muscles which move the eye. Taken from `http://www.aapos.org/terms/conditions/22`

The muscles are controlled in a neurological feedback loop controlled by three major areas of the brain: the occipital cortex, the superior colliculus and the semicircular canals. The areas map with the functional roles of eye movements: voluntary, involuntary and reflexive.

2.2.4 The Why of Eye Movement

There are four categories of positional eye-movement: saccades, smooth-pursuit, vergence and vestibulo-occular movements [12]. Other changes in the eye, such as pupil dilation, are not investigated in this project.

This project is principally concerned with saccades, but for thoroughness: smooth-pursuit movements are movements of the eye tracking a moving object across the visual field (e.g. focusing on a bird flying in the sky); vergence movements are the movements each eyes make in opposite directions to account for the spacing between each eye (i.e. to focus on something in the near distance); vestibulo-occular movements are eye movements to correct for external movements (e.g. head movements).

2.2.4.1 Saccades and Fixations

Saccades are the rapid pre-programmed movement of the eye intended to move the fovea across the visual field [9]. The periods between saccades, when the eyes are relatively still and the fovea fixed, is called a fixation. In reality, eyes are seldom still; there are also movements such as microsaccades which are so small that they do not disrupt fixations. Fixations take up approximately ninety percent of total viewing time and can last in the range of 100-600 milliseconds [13]. Studies have shown that the next desired location of a saccade can incur processing up to 200ms [9]. In contrast to the fixation, because of the speed of movement, the eyes are effectively blind during a saccade. Therefore fixations are the best representation of visual acuity for still images, and form the basis for the data collected in eye-tracking research.

2.2.5 Eye Tracking

Eye tracking is the practice of recording eye movements, specifically the location of the gaze over time. While there are are many different approaches to eye tracking - mounted or remote, Electro-Oculography (EOG) or Pupil Centre Corneal Reflection (PCCR) - our investigation will use a remote PCCR eye tracker, therefore we will only cover the background relevant to those systems.

2.2.5.1 Pupil Centre Corneal Reflection

Pupil Centre Corneal Reflection is a non-intrusive technique which detects the direction of gaze through the reflection of light in the cornea and pupil [10]. An image sensor is placed at a known fixed position relative to the screen with a corresponding light source. The light is directed at the eyes, and the movement of the head can be tracked, as well as the movement of the pupil within the eye socket, by tracking the position of the glint (or the light reflected in the corneal reflection).

In order to track in 3D space, two cameras with partnering light sources need to be placed a fixed width apart. With the use of 3D geometry, the direction of the gaze can be measured as well as the movement of the head within a three-dimensional trackbox.

In order to relate the position of the glint with the direction of the screen, PCCR eye trackers are typically bundled with calibration software. By looking at nine points on the screen (the center and the extremes), the user's eye movements can then be tracked at any other point on the screen.

2.2.5.2 Tobii

Tobii are a Swedish technology company specialising in eye-tracking products [14]. Tobii have produced many iterations and versions of remote and mounted eye trackers, commonly used for either consumer gaming or academic research.

In this project, the Imperial Bioengineering department provided the Tobii EyeX Controller hardware, which is packaged with software and software development kit (SDK).

Hardware The Tobii EyeX Controller is a remote eye-tracker compatible with a computer running Windows 7 or above through USB 3.0. The system is available as a Development Kit for a consumer-realistic price.

The EyeX is a PCCR eye tracker with two image sensors and proprietary software to detect eye movement [10]. The tracker's specification state that it supports monitors up to a size of 27in. The sampling rate of the tracker is around 55-60Hz. One of the main advantages of the

Tobii remote tracker is a generous trackbox, or the space in which the controller can maintain eye-tracking. A user can move their head within a plane of 48cm x 39cm, and at a distance of 45cm - 80cm.

Software The Tobii EyeX comes packaged with a SDK that makes it easy to set-up, calibrate and communicate with. The SDK is available as a library in a select few languages such as C, C++ and the .NET stack.

Calibration must be performed per user to achieve reliable eye movement data. A profile is created, with information about whether the user wears contact lenses, glasses or neither, and then eye tracking measurements are taken at eight points on the edges of the screen and a point at the center. User profiles and calibration is set up using Tobii software, independent of any developer-engineered application using the SDK.

The .NET Tobii API provides several way to interface with the eye-tracker and provides many examples. With the inclusion of pre-compiled library files (.dll), applications have access to the API and its methods [15]. The API supports the different ways a developer can create .NET applications, including Windows Presentation Foundation (WPF).

The API is more comprehensive than is necessary to explain, including easy ways of defining trackable zones. For this investigation, the lower-level API is suitable, which allows reading the lightly-processed data from the eye-tracker in a similar vein as other input buffers. There are three available data streams:

- gaze points, in x and y positions relative to the top-left corner of the screen
- right and left eye position in three-dimensional space relative to the screen
- fixations, calculated from the gaze points using two fixation algorithms, including Tobii's proprietary and blackbox algorithm.

For this investigation we will rely on our own algorithms to detect fixations.

2.2.5.3 Fixation Detection

Eye trackers such as the Tobii provide gaze point data, a measurement of the location of the gaze at a certain time. Therefore the data is agnostic as whether the point belongs to a saccade or a fixation. A number of algorithms have been devised to classify raw gaze data (x,y,t), with 2D position x and y and timestamp t, into fixations and saccades. We will be using Dispersion Threshold Identificaton (I-DT), as according to experiments conducted by Salvucci and Goldberg, shows both high accuracy and robustness [16]. Other algorithms, such as Velocity Threshold Identification, are either equivalent or have tradeoffs in terms of implementation ease, number of parameters and speed.

Dispersion Threshold Identification An approach to distinguishing between fixations and saccades is to use a typical characteristic of fixations; due to their lower velocity, fixation gaze points tend to cluster together (typically over areas of interest) [16]. Dispersion Threshold Identification (I-DT) uses the dispersion D of points, defined as

$$D = y_{max} - y_{min} + x_{max} - x_{min}$$

for the primary metric for clustering gaze points.

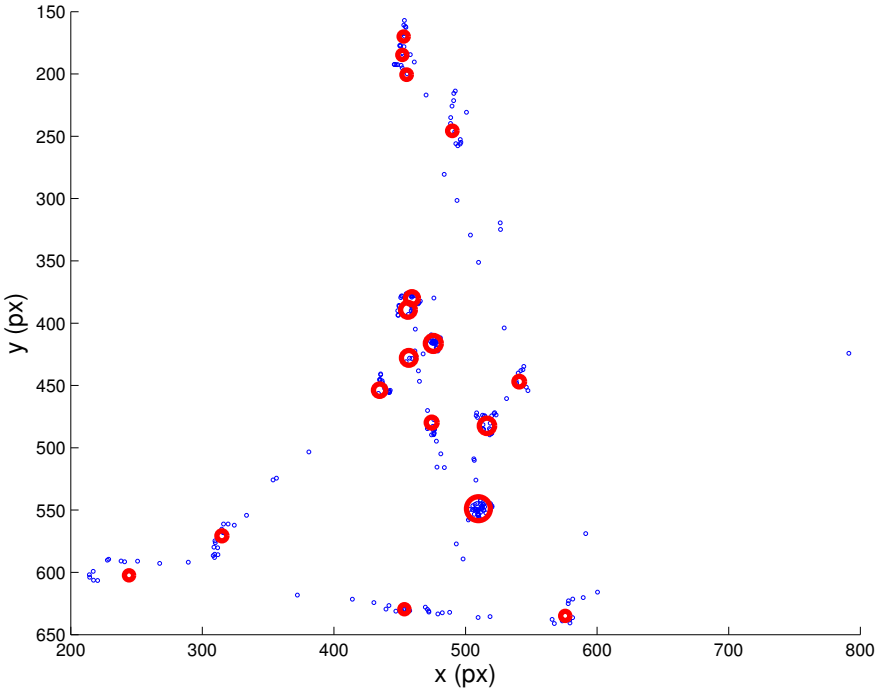


Figure 2.9: A set of gaze points (blue) with corresponding fixations (red) calculated using I-DT. The radius of the circles correlate to the duration of the fixation

Listing 2.1: Our version of the Dispersion Threshold Identification algorithm in pseudocode

```
fixations = []

while points.isNotEmpty:
  window = [points(1)]
  while window.duration < 100ms & points.hasNext:
    add points(next) to window
  end
  if window.dispersion <= threshold & window.duration > 100ms
    while(window.dispersion <= threshold & points.hasNext)
      add points(next) to window
    end

    add centroid(window) to fixations
    remove window from points
  else
    remove points(1) from points
  end
end

return fixations
```

It does this by considering *windows* of points; groups of gaze points that span the minimum duration of a fixation (typically 100ms). It adds consecutive gaze points that maintain a dispersion value under a certain threshold, using known or experimentally-retrieved values. Once the property is violated, the window is closed and a new one, starting from the violating gaze point, is created. Points which do not belong to a window are marked as part of a saccade, and thus ignored. The pseudocode for the algorithm is shown in figure 2.1.

Figure 2.9 shows the original raw gaze data points (in blue) and its corresponding fixations (in red) found using I-DT. The number of fixations is significantly less than the number of original gaze points.

2.2.5.4 Metrics of Eye Movement

In order to quantify and compare eye movement data, it needs to be represented in certain a structure or measure.

Raw 2D eye movement data is typically a triple of (x,y,t), or the location of the gaze in the x and y-dimensions and the timestamp at measurement. Converting the raw gaze data into fixations can be performed by specialized algorithms such as I-DT (see section 2.2.5.3).

Take for example the four fixations calculated while a user viewed the "image" in figure 2.10a. The sequence of four fixations (labelled in their respective order) compose the scanpath. The radius of the circle shows the *dwell time*, or duration of the fixation, such that a larger circle indicates a longer dwell time. With only the fixations, a number of metrics can be derived, including:

- fixation duration
- number of fixations in a scanpath
- total fixation time
- spatial distribution of fixations

We can then calculate the saccades as the vectors connecting fixations. In figure 2.10b, these are the three purple vectors labelled a, b, and c. Corresponding metrics can be found such as:

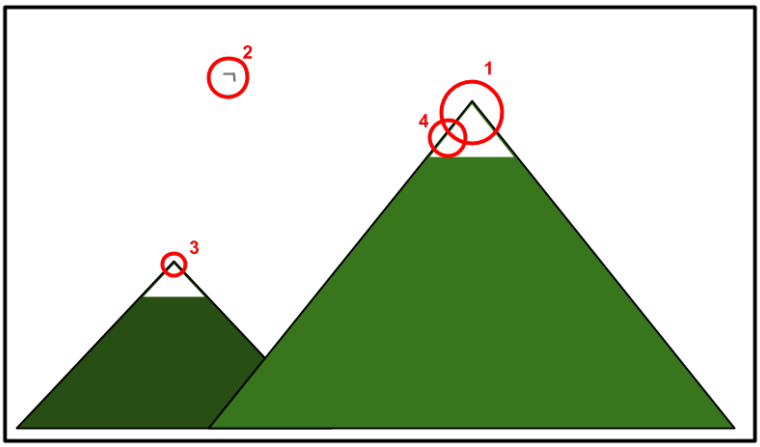
- saccade length (magnitude of the vector)
- saccade amplitude (radians/sec), or saccadic length normalized against the distance to the screen
- number of saccades
- saccade directions (the angle between two saccades)
- saccade duration
- ratio between time in fixations and time in saccades

Finally, a popular organization is to mark sections of the image or screen as Areas-of-Interest (AOI) or Regions-of-Interest (ROI, although we will be using the AOI nomenclature). In figure 2.10c, we defined three AOIs: A, B and C. Once AOIs have been identified, either by humans or algorithms, the following measures can be calculated:

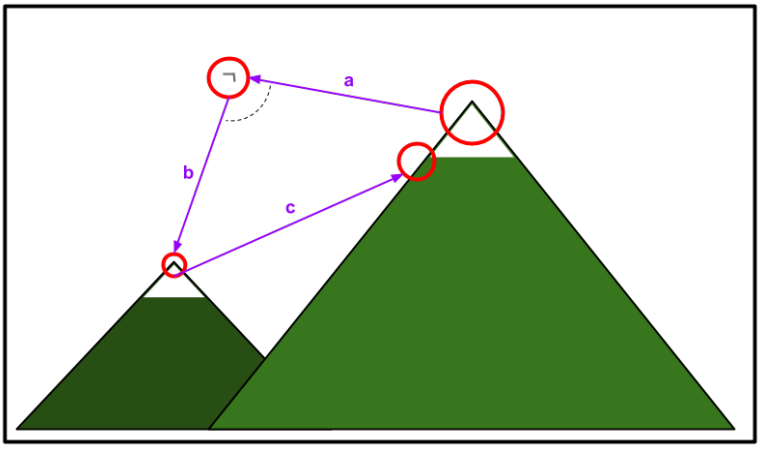
- number of fixations per AOI
- time spent per AOI
- first/second order entropy between AOIs
- number of revisits (left and returned) or repeats (consecutive fixations) per AOI

2.2.5.5 Scanpaths

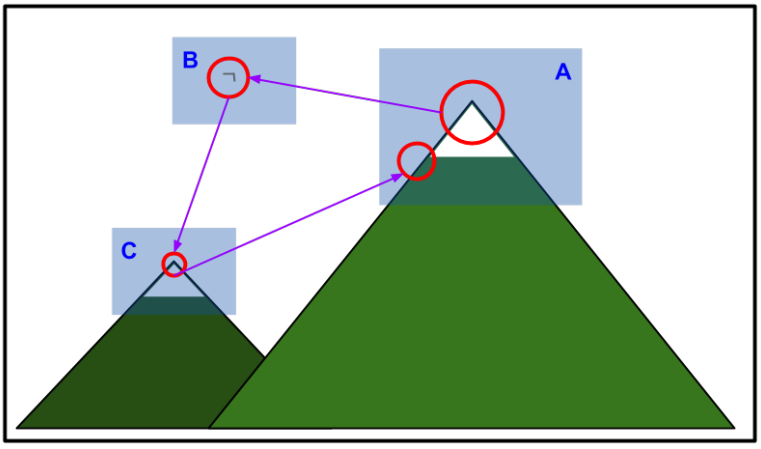
Scanpaths are the term given to a person's series of fixations over time. In essence, they are composed of the saccade vectors connecting fixations, such as the scanpath in figure 2.11. They may also be represented as a series of AOIs, if fixations are replaced with the AOI region in which they located.



(a) An image with locations of four fixations 1,2,3, and 4 displayed as red circles with radius proportional to duration



(b) The saccades a, b and c which connect the four fixations



(c) The three AOIs A, B, and C in the image

Figure 2.10: Examples of eye movement features

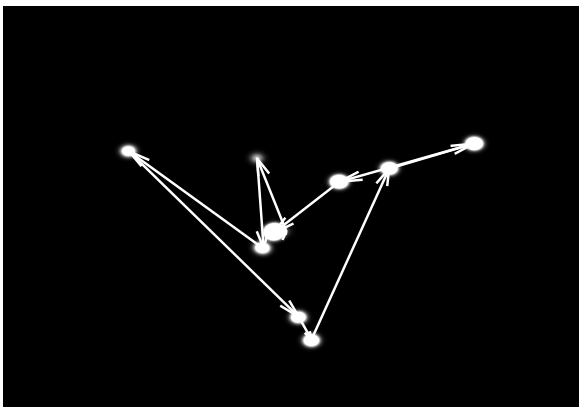


Figure 2.11: An example of a scanpath represented its saccadic vectors

For many eye movement procedures, scanpaths is the natural data structure used to analyse eye movements. Therefore, there is growing research into how to compare and analyse pairs and groups of scanpaths [17].

A scanpath such as the one in figure 2.11 is a list of three-dimensional fixation objects (x,y,t), with x and y position and timestamp t. However, oftentimes we are not interested in the exact x and y position, but it's relationship to a semantic region of the observed image, i.e. an Area of Interest. Therefore, we can reduce a scanpath as a series of AOIs. Let us refer back to figure 2.10 used in the previous section, with the drawing of the mountaintop. In that example, there are three AOIs defined: A, B and C. In the scanpath, there are four fixations (marked 1, 2, 3 and 4). If we represent the fixations by the AOI they are located in, the scanpath can be represented as:

$$A \rightarrow B \rightarrow C \rightarrow A$$

This representation of the scanpath reduces it to a finite sequence of fixed states, which open up the opportunity to model it as a string or a Markov Process.

String Editing With scanpaths represented as a string, we can use a set of distance metrics known as *string-edit* distance. String editing measures the similarity between two strings of characters by counting the cost of editing one string into another. An example of a string-edit metric is the Levenshtein distance.

The Levenshtein distance between two strings is the minimum number of steps required to transform one string into another using one of three operations: addition, deletion, and subtitution of a character [18]. For example, the Levenshtein distance between `imperial` and `emperor` is 4:

```
imperial
emperial # substitute 'i' => 'e'
emperoal # substitute 'i' => 'o'
emperorl # substitute 'a' => 'r'
emperor # delete 'l'
```

Levenshtein distance can be calculated efficiently using a bottom-up algorithm to populate a m-by-n matrix with the minimum cost to transform a string of length m into a string of length n. Each cell of the matrix (i,j) contains the minimum cost to transform the ith letter of the first string into the jth letter of the second string.

As longer strings are penalized by the number of transformations, the measure is normalized against the length of the longer string. The maximum number of steps using Levenshtein

operations is the length of the longer string (substitute every character in the shorter string, and then add the missing characters). To qualify as a similarity measure, Levenshtein distance is defined as $f(s_1, s_2) = 1 - L_c$, where L_c is the normalized cost of transforming s_1 into s_2. Identical strings produce a value of 1, and the function is symmetric: the value of $f(s_1, s_2)$ is the same as $f(s_2, s_1)$.

Levenshtein distance has been used to compare scanpaths successfully in a wide variety of applications. While the original measured proposed by Levenshtein had a cost of 1 for all three operations, other researches have experimented with increasing the cost of the operations [19]. For example, in an investigation on the eye movements of TV viewers, Josephson and Holmes weighted the cost of substituting aoi_1 and aoi_2 by the number of other AOIs physically between aoi_1 and aoi_2 [20]. Similarly, Takeuchi and Habuchi concluded the use of Euclidean and City-Block distance metrics to weight substitution costs improved similarity performance over the original uniform cost [19].

String editing is a useful measure for pairwise comparison of scanpaths, but there have also been efforts to use it as a measure of groupwise similarity. Galgani et al and Duchowski et al. used Levenshtein distance as a metric as the basis of a non-parametric binary classifier [21] [17]. By comparing the test scanpath against each of the scanpaths in the group, a groupwise similarity metric was calculated as the average of the pairwise similarities. For Galgani et al., classification was determined by the higher groupwise similarity. Duchowski trained a classifier for each group, determining the thresholds for similarity that resulted in good performance by using ROC curves (see section 2.1.6). In the situation where neither threshold was reached, the higher average, or groupwise similarity, was used.

Markov Process The second representation of a scanpath is a Markov chain, or a state of discrete states with transition probabilities from one state to another. This representation is particularly appropriate as it models the hypothesized behaviour of the human visual system (HVS); as discussed in section 2.2.4, a saccade is movement with ballistic properties. In other words, the HVS processes and selects the next target fixation and saccades are calculated accordingly. Therefore this dependence between two fixation locations and selection is a transition that can be modelled using a first-order Markov property (see section 2.1.4 for details on the Markov property and Markov Processes).

In relation to eye movement, the sequence of states can map directly to the sequence of fixations (represented by the AOI in which they are located). The corresponding first-order Markov property states that the location of the next fixation can be determined by the location of the previous fixation.

This model of the scanpath produces a n-by-n transition probability matrix, where n is the number of distinct AOIs. We can then use this structure to measure each scanpath, and compare two scanpaths.

The utility of this representation and potential has been of interest to the eye movement community for decades. Stark and Ellis investigated the use of the Markov property at different orders to identify structured processes within eye movements [22]. Hacisalihzade et al. continued the work to use a discordance matrix as a dissimiliarity metric, which we will refer to as *Markov Error* (ME) [23]. The Markov Error is the normalized error, or element-wise difference, of each element between two n-by-n transition matrices A and B such that:

$$D = \frac{1}{n^2} \sum_{i,j=1}^{n} |A_{ij} - B_{ij}| \tag{2.7}$$

This is a dissimilarity metric as identical transition matrices will result in a score of zero.

A similar metric is Markov-Path-Distance (MPD), which measures the aggregated discordance of two transition matrices after a certain number of steps [24]. If $A_{i,j}$ is the probability of

transitioning from state i to state j in one step, the probability of transitioning from state i to state j in k steps is A_{ij}^k. The MPD between two n-by-n transition matrices A and B is thus:

$$MPD = \sum_{i,j=1}^{n} \left| A_{ij}^k - B_{ij}^k \right| \tag{2.8}$$

$$1 < k < \inf \tag{2.9}$$

The value of k can be adjusted to retain the most information and target non-uniform transition probabilities. A k of one is equivalent to the unnormalized ME. A sufficiently large k will lead to the stationary distribution, therefore a k between those values is suggested.

A final measure of a Markov transition matrix is to use Shannon's definition of information entropy. As shown by Ciuperica and Girardin [25], entropy as applied to Markov chains can be calculated as:

$$H_t = -\sum_{i \subseteq N} \pi_i \sum_{j \subseteq N} A_{ij} \log A_{ij} \tag{2.10}$$

where N is the set of states, A is the transition probability matrix, and π is the stationary distribution.

This measure, unlike MPD and ME, is not a comparison or similarity metric. It is a measure of randomness; thus a matrix with high entropy (such as a uniform matrix) will have a value closer to 1, whereas more predictable transitions have a lower entropy. In the context of eye movement, this can be used to measure either the predictability of eye movement or, as Krejtz et al. did, a measure of complexity. In their study, entropy was used to quantify a participant's switching patterns while observing pieces of art to compare against attractiveness and curiosity [26].

From the Markov chain, naturally some research has led into the use of HMMs. Hidden Markov Models (HMM) intuitively seem like an appropriate model of a scanpath because as a Markov process they represent both the temporal and spatial dimensions of eye movement, but there are often latent states not directly observed that are useful to model.

Several studies have used HMM as a means of evaluating scanpath strategies. Pieters et al. used HMMs to model the eye movement of participants viewing advertising [27]. In their study, the hidden states represented whether the user's attention was global or local, relating it to their search strategy. Simola et al. mapped three cognitive processes (scanning, reading and deciding) to three hidden states in evaluation of search strategies while online search [28].

Chuk, Chian and Hsao used a HMM to model each user's eye movements in a face recognition task [29]. The hidden state spaces were the AOIs of the image and observations the spatial dimensions of fixations. They also clustered the HMMs to identify two types of recognition strategies: holistic and analytic.

2.3 Memory

The study of memory, as a topic of neuropsychology, is still a field with lots of mystery. Functions of memory such as how it is stored and how it is retrieved, as well as which parts of the brain those functions take place, are hypothesised under memory models. Experimental data collected over time helps refine, support and eliminate these models of memory. For example, many of the research studies discussed below follow on from data collected from patient H.M. in 1962, whose damage to the hippocampus helped identify the role and importance of the hippocampus in memory [30].

With increased technology, the ability to map the functions of memory to parts of the brain have eliminated and uncovered new models. While it is not necessary for the purposes of our study to delve in to the full research, it is important to appreciate the foundations for some of the research we will discuss in later sections 2.3.1. More critically, we need to ensure that the feasibility of the project matches with the expectations of current theories on memory.

A model that is referred to often in the eye-tracking research is the principal component model with dual stage processing [31][32]. This model puts the hippocampus, a small region of the brain, as an index to memory traces, analogous to how a program stack maintains pointers. Anything that is consciously experienced creates a memory trace and the hippocampus stores the index to the trace. If we think of memory as an object on the heap, which itself can be fragmented across different parts of the heap, then the index is the original pointer to the object. Where the memory is "stored" and what regions of the brain is necessary to encode and retrieve are dependent on the type of memory. The key to the retrieval of the memory trace is a cue, or stimulus, which was also used during the encoding process. In layman's terms, this is referred to as a trigger; for example, hearing your wedding song may remind you of the experience of dancing at the reception. A point to note is that the hippocampus does not discriminate; it cannot assess the importance of a memory, and indexes anything that is consciously experienced. This is why Moscovitch refers to the hippocampus as a "stupid" module [31].

The expansion to the model includes two stages for recollection: a quick retrieval of studied content possibly without awareness, as awareness includes the use of other parts of the brain, and a second slower process which brings about the awareness of the recollection. Theoretically, this implies that if the regions of the brain which the memory are actually stored are disrupted, by lesions or other effects, there would still be one form of recognition from the initial process.

While our investigation is primarily concerned with the interface, or input and output, of these processes, rather than its implementation in neurological systems, this model especially does create a strong foundation for success for our analysis. It is important to note, however, that there are other models of memory, with varying levels of evidence and popularity. While memory research has improved dramatically with the advent of technologies such as functional Magnetic Reasoning Imaging (or fMRI), it is still difficult to test memory in controlled conditions, and the validity of past experiments have been called into question[33] [32]. Therefore, instead of navigating all the evidence, our study will stay relatively independent from any particular model. The principal component model is a convenient vehicle in which to explain the current state of memory research. We make no attempt to prove or disprove any of the models.

As well as models of the overall system, different types of memory are categorised under specific terms. One of the most relevant to our investigation is the difference between *explicit*, of which we are consciously aware, and *implicit*, of which we are not aware, memory. Other words for explicit include *declarative*, and implicit is interchangeable with *non-declarative* or *procedural*. Common examples for implicit memory are everyday activities such as knowing how to tie your shoelaces. The study of implicit memory is a particularly difficult subject of experimentation, and a major reason for the increased use of eye-tracking as a method of investigation.

There is also a distinction between *recollection* and *familiarity*; recollection being tied to remembering, or recalling the memory with its context, and familiarity referring to recognising or knowing something but not realising from what context.

There are also types of memory dependent on the object being encoded. Memories of conscious experiences are *episodic*. Remembering the bindings between two things, for example a pairing of a face and a scene, is an example of *relational* memory. Finally, *semantic* memory is remembering the meaning or context of an object, for example seeing a picture of Tom Cruise and remembering that he is an actor.

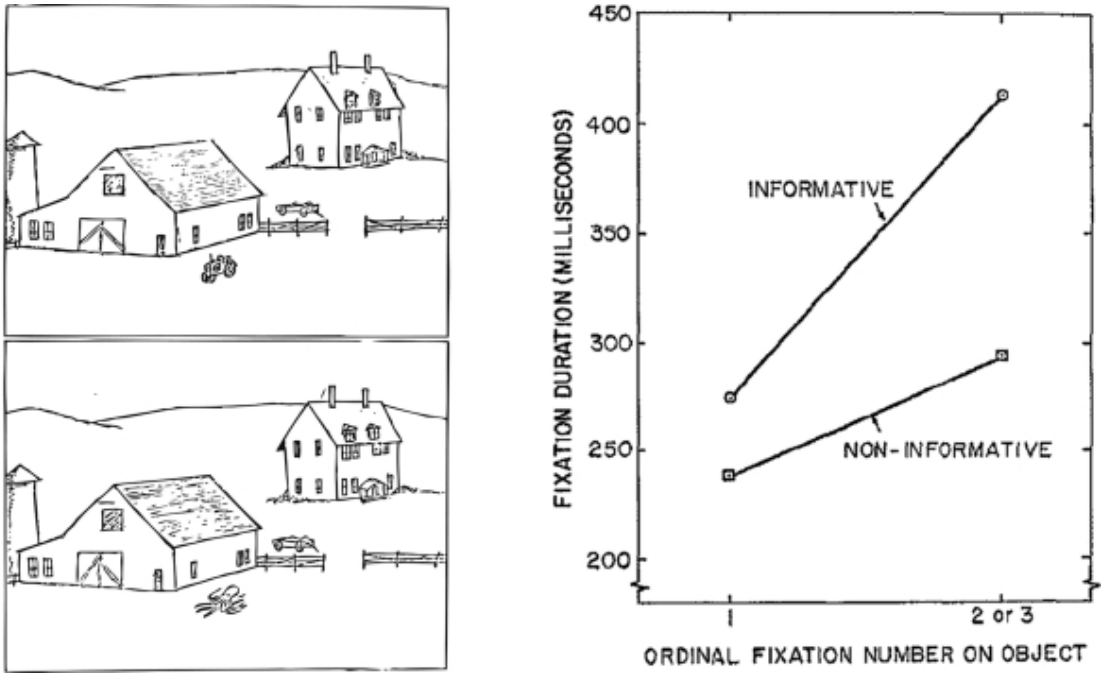


Figure 2.12: Stimulus used by an experiment by Loftus and Mackworth (taken from paper) [37]

2.3.1 Current Research

Eye tracking is a useful tool for investigating certain memory-related properties as eye movement is typically influenced by two factors: physical properties, such as luminance and hue, and brain-related properties, such as episodic or semantic memory [34]. We are interested in the influence of memory, but will have to account for the influence of physical properties too. In the following section, we will be looking at support and results for this field of research.

2.3.1.1 Eyes, the Brain and Memory

Experiments investigating the relationship between eye movement and the brain have occurred for decades, especially those concerned with memory and its retrieval.

In 1967, Yarbus formulated an experiment to track participants' eyes as they viewed Repin's *An Unexpected Visitor* [35]. Yarbus showed that the scanpaths differed depending on which task they were given prior to looking at the painting, e.g. identifying the wealth of the family. We will refer to this as the *task effect*.

An experiment at Stanford University showed the reverse relationship; the influence of eye movement on the brain. In this study, the goal was to measure the relationship between eye movements and memory performance: namely if the number of fixations or the total fixation time during the studying of an image correlated with correct recollection of the given image in the test phase [36]. The experiment discovered a significant positive correlation between the number of fixations and memory performance. Simply translated, the more we look at the image, the better we remember it.

Loftus and Mackworth then showed the effect of semantic memory on eye movement [37]. They found, by showing participants images of a barnyard seen such as the one in figure 2.12, that fixations were earlier, longer and more frequent for objects that were unexpected (like an octopus) rather than a tractor.

Another direction of research looked in to the effectiveness of eye-movement desensitization-reprocessing therapy (abbreviated as EMD-R) to treat patients with post traumatic stress disorder.

EMD-R is a technique to reduce the stress of recollecting traumatic memories by making the patient move their eyes at the moment of recollection. Running four experiments with University of Sydney students, Andrade, Kavanagh and Baddeley discovered that vividness (measured by a surveyed response) was reduced while making saccadic eye-movements [38]. A follow-up experiment testing only autobiographical memory found a similar correlation; eye movements were more effective as a distraction than tapping or other repetitive tasks. Interestingly the vividness of even further future recollections of those memories was affected [39].

At the University of Massachusetts Amherst, eye movement was used to study the familiarity effect; namely, if the familiarity of the targets affected the performance of the search. Eye movement was measured to test whether searching for a familiar target embedded within unfamiliar distractors takes longer than an unfamiliar target within familiar distractors [40]. In addition to manual reaction time, number of fixation and duration of fixations were measured to help understand the possible cognitive processes at play. Number of fixations correlated with the familiarity effect, but there was no significant relationship with individual or average fixation duration.

2.3.1.2 Awareness Experiments

A much closer area of research in the past decade has been the investigation of using eye-movements as a metric for recognition independent of conscious awareness.

An experiment investigating the relationship between eye movements and the hippocampus and surrounding media temporal lobes (MTL) was conducted in 2009 [33]. In the first experiment, background scenes and superimposed face pairs were studied, and then participants were asked to identify, based on the scene cue, which of the three faces were the correct pairing. Examples of the stimulus can be seen in figure 2.13 Correctly identified faces were viewed longer cumulatively than just selected faces. Researchers analysed the set of instances where a disproportionate (more than 10% of viewing time) amount of time was spent on a matching face (DPM), the set of instances where a disproportionate amount of time was spent on a non-matching face (DPNM), and the activity of the MTL using functional magnetic resonance imaging (fMRI). Results showed there was a greater strength of the blood-oxygen-level dependent signal in certain MTL lobes for DPM than for DPNM, regardless of explicit memory retrieval. The prefrontal cortex (PFC) was also analysed, and the activity of the connection between the hippocampus and PFC correlated with whether the participant correctly or incorrectly identified the target. These results do not contradict the hypothesis that eye movements can be an expression of hippocampal recognition, but without activity from other regions of the brain (such as the PFC), then explicit awareness might not be reached.

Figure 2.13: Study and test stimulus used in a relational memory experiment (taken from paper) [33]

Experiments at the University of Illinois and University of California, Davis, investigated the use of memory as a veridical index of memory disassociated from conscious recognition [41]. The experiments asked participants to study a set of faces during a study phase, and then select a face from a three-face display if they studied that face, or to select randomly if the face was not present. Eye movements were compared to verbal responses, and a second experiment was conducted to increase the number of verbal false positives (incorrectly recognised as studied). To increase the difficulty of the challenge, the three face display used morphed images; a morphed face is the target face "morphed", or mixed algorithmically, with the non-target face. The procedure also

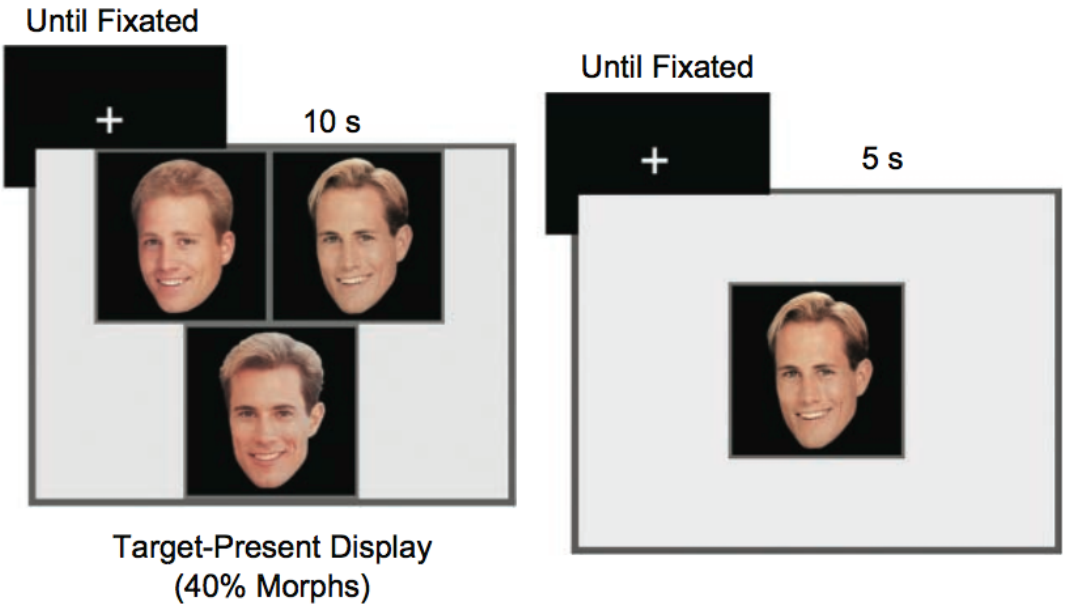


Figure 2.14: Stimulus for study and test phases to test eye movement as an index to memory (taken from paper) [41]

included a morphing coefficient, either 20% or 40%, to test the relationship between accuracy and difficulty (measured by the morphing coefficient).

Results suggested that eye movements were a better indicator of studied faces than behaviour - namely that eye movement indicators of recognition occured earlier and were more accurate than behavioural responses. Eye movements also seemed to be insensitive to the morphing coefficient, and in the experiments occurred within the first two seconds of seeing the images.

Schwedes and Ventura investigated the use of eye movements as a means of uncovering revealed information in an experiment with students from Saarland University [42]. The experiment asked participant to select one out of six presented faces under three conditions: in the concealed display, the participant knew one face but was instructed to select one of the other five; in the revealed display, one face was known and the participant was instructed to select it; in the neutral display, no faces were known and the participant was instructed to select a random face.

The results indicated a pattern that separated the recognition effect, whether the participant recognised a face, and the response intention effect, whether the participant selected a face, in eye movements. The recognition effect manifested itself by fixation duration over the first three fixations - faces that were recognized, regardless of whether the intent was to conceal or reveal them, had higher fixation durations during the first three fixations (analysis actually showed only the first two were needed). The neutral display did not follow this pattern. The response intention effect could not be identified within the first three fixations, and only under the analysis of total fixation duration did the selected face have a significant difference in the neutral display condition.

A study at the University of Barcelona tested the oculomotor reaction to auditory cues in retrieving long-term memory traces of picture-location pairings [43]. The set-up included a screen depicting four possible picture locations,arranged in a two-by-two grid, and eye-tracking sensors. Participants in the experiment went through an encoding phase in which a unique sound was played, followed by a single picture in one of the four locations. The encoding phase was purposely large in order to overwhelm memory capacity and create the opportunity to test both conscious and unconscious memory retrieval. During the testing phase, participants were asked to focus on the centre of the screen, and one of the encoded auditory cues was played. Participants were asked to search for the correct location the picture should appear in. Three separate responses

were captured per trial: one, the eye movement in relation to one of the four regions of interest; two, whether the participant could verbally recollect which of the four locations the associated picture appeared in; three, whether the participant could correctly identify the picture. A second experiment also asked the participant for their confidence judgment on being correct.

The metrics used for eye-movement involved the relative number of fixations and the proportion of time gaze was fixated on the correct square during the search period. Results in both experiments found that, even for instances where the participant reported an inability to remember the correct location, eye movements tended to show a disproportionate number of fixations (and to a lesser degree, dwell time) on the correct location. This is further evidence that suggests eye movement could implicitly indicate memory retrieval.

Chapter 3

Data Collection

3.1 Application

One of the principal objectives of this investigation was generating a novel dataset for purposes of analysis. In addition, the lab did not have an established means of collecting eye movement data for a customized purpose. Therefore, the development of an eye movement collection application was necessary. This section describes the implementation of this application. The actual data collected through trials will be discussed in section 3.2.

3.1.1 Background

WPF Windows Presentation Foundation (WPF) is a .NET application framework that makes it easy to create rich user experiences. Using XML as the view markup, developers write code in any .NET language (such as C# or Visual Basic) to write the *code-behind*, including adding behaviours such as event handlers. Properties and interactions can be programmed in either the code-behind or the markup.

3.1.2 Application Objectives

The data collection application was designed to collect eye tracking data from a single user sitting in front of a monitor. The parameters of the application were primarily defined by the use of the Tobii EyeX Controller, described in section 2.2.5.2. Compatibility with Tobii influenced other set-up criteria, including the use of the Microsoft .NET framework, Windows Presentation Foundation.

The implementation of the application began before the trial protocol was fully defined, therefore it was designed to be robust against any changes. It was also designed with the secondary objectives of reusability and multi-purpose; a possible framework to be used by others in the lab, or for situations outside of this investigation. We will detail such a use-case in section 3.1.4.2.

3.1.3 Application Architecture

The application is an executable designed to "play" a user-given experiment and write out data files. An experiment is defined as a series of *phases* by the user in a text file called `phases.txt`. The user also supplies any images in the *images* folder, optionally subdivided into folders. Both the *images* folder and `phases.txt` are expected to be in the root directory of the executable.

Playing an experiment involves sequentially traversing the list of phases in serial order. Each phase is responsible for alerting the application when it has completed so that the application can either move to the next phase, or close.

When a phase is active, it has access to the filesystem, keyboard, eye tracker, and window. Phases are pre-defined, but are abstracted such that it is easy to add custom phases to the application if required. All phases are instantiated at application start-up. A phase is activated by calling its `start()` method and is given a callback to signal when the phase has completed. It also exposes a `close()` method, where it is expected to clean up any of its responsible memory (such as an open write buffer).

The application is displayed in full-screen but with a maximum-sized rectangle of screen space defined as a *working area*. The working area is used to prevent the application from relying on the edges of the screen, where the calibration is weaker, for extremely large screens.

3.1.3.1 Phases

The application lifespan is subdivided into discrete sections called phases.

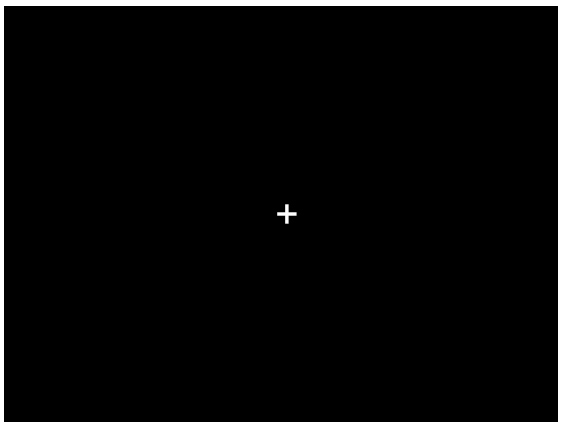


Figure 3.1: The centering image used to divert users' attention to the center of the screen

The following phases are currently supported, with brief overviews provided:

- `Study Phase`: show a series of images and write eye tracking data to file. Each image is preceded by a *centering* image, shown in figure 3.1.
- `Score Phase`: equivalent to the study phase, except asks the user to press one of two buttons (red or green) to proceed to the next image, and the keys are printed to file.
- `Calibration`: Play a sequence where a dot moves around the extremes of the screen and print the eye movement data to file.
- `Instruction/Key Trigger`: Print text on screen. Ended either by a time limit (instruction) or by pressing a button (Key Trigger)
- `Eye Trigger`: User must look at a cross in the centre of the screen in order to move on.
- `Demo Phase`: Display an image and show eye tracking data on screen. Supports dual screen.

Both the study and score phases are a time-triggered series of images. The series is defined by a `TestScript`, and the images are loaded using an `ImageLoader`. The script's file path (relative to the application root directory) is passed in as the first parameter of the phase in `phases.txt`. Other phases also support parameters. An example of a `phases.txt` can be found in section B.1 of the Appendix.

3.1.3.2 ImageLoader and ImagePool

On instantiation, an `ImageLoader` loads all the images in the *images* folder and converts it in to a list of `ImagePool`s. A pool is created for each sub-folder in the *images* folder, with the first `ImagePool` consisting of any JPEG and PNG images in the *images* folder itself. The order of folders and images is deterministic and lexicographical.

Each image is captured as a `DisplayImage`, which contains its source path and some application-relevant meta data.

In addition to storing the `DisplayImage`s, an `ImagePool` provides the access to retrieving the images. This includes supporting four getter methods:

- `next()`: get the first free image, or reset if no image is free
- `prev()`: get the last used image
- `randomNew()`: get any random image
- `randomOld()`: get any previously used image

`ImagePool`s are implemented using two lists, one for used images and one for free (unused) images.

3.1.3.3 TestScript

A `TestScript` is a series of triples composed of an `Action` (different from the C# Action), pool index, and duration. `Action` is an enum with values `Next`, `Previous`, `RandomNew` and `RandomOld`, mapping directly to each of the `ImagePool` access methods. Pool index and `Action` define which image will be shown and duration is the number of milliseconds the image will be shown. `Action` is the only necessary parameter as both pool index and duration, if not defined, will default to 0 and 5000 respectively.

`TestScript`s are expected to be UTF-8 encoded text files, whose location are passed in as a parameter to the phase. An example of a `TestScript` can be found in section B.1.1 of the Appendix.

Each image shown in the script is preceded by a neutralising image with a cross in the centre (as seen in figure 3.1). The neutral image is defined as the image in the *images* folder titled `neutral.jpg`. Each `ImagePool` is responsible for alternating between a script image and the neutral image, and the neutral image is shown for three seconds. We plan to add a configuration file so that these settings are easier to customize than changing source code.

3.1.3.4 Data Input

There are two methods of input for the application: keyboard and eye tracker. All forms of input fit the event handler pattern used by WPF.

The only key reserved for use by the global application is the escape key, which pressed during any phase can be used to close the application. The active phase is commanded to exit using its `close()` method, and is therefore responsible for ensuring the data is properly written to in the case of unexpected termination.

To access the eye tracker, the application uses the available libraries provided by Tobii. As discussed in section 2.2.5.2, Tobii provides a low-level API to access data streams.

The application accesses two global data streams: the raw gaze position in relation to the display and the position of the users eye in 3D space.

Access to the eye tracker is given to each phase through a `Tracker` singleton instance, which is responsible for creating and disposing of the streams.

In order for the eye tracker to read input correctly, it needs to be calibrated per person. This is independent of the application, which assumes the eye tracker is configured correctly.

3.1.3.5 Data Output

The application can output `DataFile`s, append only files (with extension `.dat`) filled with chronologically-sorted, row-based, tab-separated data. In order to work with other programs[1], strings are ordered at the end of each row. `DataFile`s insert an application defined POSIX timestamp.

The name of the `DataFile` is defined by the phase which creates and manages the `DataFile`. For the pre-defined phases, such as the `Study Phase`, we have adopted a minute-granular naming convention. For example, if the application was started at 4pm on June 16th, the file would be named:

```
06-16-16-00-1-image-gaze.dat
```

where the fifth number, 1, is the unique phase ID given to the phase during instantion by the application.

3.1.4 Use

3.1.4.1 Data Collection Trials

The engineering of the application coincided with the development of the data collection protocol discussed in section 3.2. As designed, the application was used successfully in a number of trials. The configuration is discussed in section 3.2.4, and the exact `phases.txt` can be found in the Appendix in figure B.2.

3.1.4.2 Imperial Festival

Independent of this investigation, the application was used in another environment: the Imperial Festival[2]. The Festival took place over the weekend of 9th and 10th May 2015 at Imperial College London, and is a free and open to the public. The event is designed to recognise great research by the institution as well as provide activities, talks and music events for all ages. As part of the Bioengineering exhibit, there was a stall dedicated to research into eye movement and eye-tracking.

The application was used to give members of the public an opportunity to experience eye-tracking in a novel and entertaining way. For the event, the application was configured to display primarily in `DemoPhase`s. This allowed for a series of images that can be shown both on the monitor with the eye tracker, and a larger second screen available to the audience. The most popular images were a collection of 'Where's Wally?[3]' pictures, which attracted a lot of interest from children. The second, larger screen would show a moving dot, representing the real-time eye movement of the child in the chair as they searched the picture for Wally. Members of the crowd could help guide the user if they wished. `DemoPhase`, triggered on input from the keyboard, can also can show the entire gaze data point history, which a few participants used to self-evaluate their search strategieis. Over the weekend over a hundred people used the application, and several hundred more observed its output.

[1] Matlab
[2] https://www.imperial.ac.uk/be-inspired/festival/
[3] http://www.whereswally.co.uk/

3.2 Data Collection Trials

The data collection application was used as a part of a series of trials to collect novel eye movement data. In this section, the set-up, procedure and initial analysis of the results from the trials will be described.

3.2.1 Objectives

The objectives of the data collection trials was to collect eye-tracking data to test and train our classifiers. The goal of the classifier was to be able to discriminate between eye movement that indicates recognition independent of awareness. We targeted two levels of recognition, which we will refer to as *image* and *subimage* recognition. *Image* refers to macroscopic recognition of the image, whereas *subimage* refers to specific features of the image. The principal objective of this investigation is to classify recognition in general, not solely of an entire image. The behaviour of eye movement on feature-level recognition may be a critical component of recognition. Furthermore, later users of the data may find this additional feature of the data set useful.

While it is straightforward to determine whether a participant has seen an image before, one of the objectives was to detect recognition *independent* of awareness. In order to study this characteristic, it was necessary to collect results directly from the participant about explicit recognition of the images. This data also enabled another means to evaluate our classifier in section 4.1.

As part of our preliminary analysis, we investigated the following hypotheses:

- eye movement differed between first and second viewings of the same image (*image*)
- eye movement differed for previously viewed features (*subimage*)

We tested several metrics, and for the purposes of showing the value of the data, will use the following six metrics (which showed good results) for *image* recognition:

- μ_{sl}, mean saccade length (pixels)
- μ_{st}, mean saccade duration (ms)
- σ_{at}, standard deviation of total fixation time in each AOI (ms)
- σ_{aft}, standard deviation of fixation duration in AOIs, ie grouping consecutive fixations which are in the same AOI (ms)
- σ_{ft}, standard deviation of fixation durations (ms)
- Σ_{tft}, total fixation time (ms)

We will concentrate on one metric for *subimage*, or feature, recognition:

- μ_{ft}, average fixation duration (ms)

AOIs were defined a priori using human analysis. Saccade length is was not normalized against eye position (ie it is not saccadic amplitude), although the data is available.

3.2.2 Stimulus

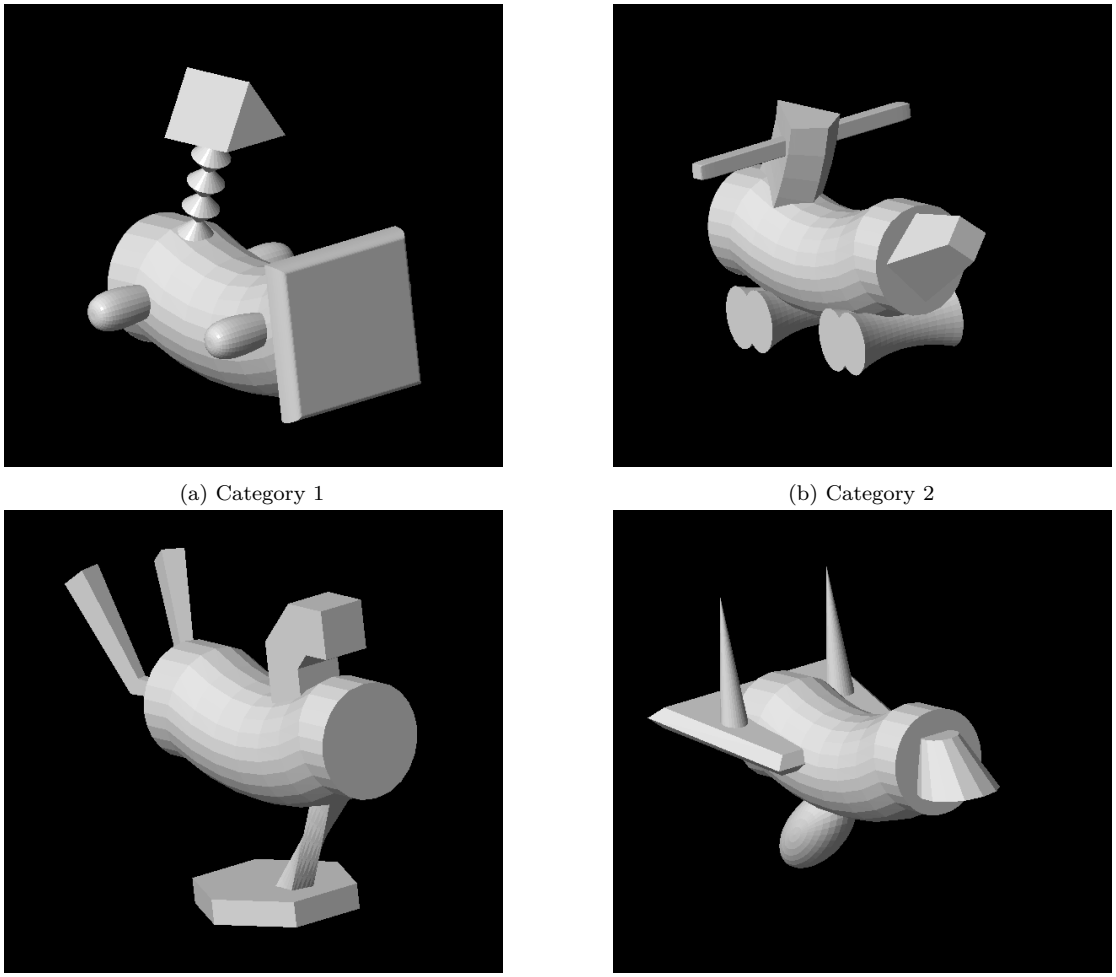
To support the objectives of the experiment, an appropriate stimulus needed to be found, with the following characteristics:

- *novel to the subjects*: to ensure participants had not seen the stimulus before

- *various*: a number of different images was needed to collect as much data as possible
- *modular*: in order to study both *image* and *subimage* recognition, features needed to be shared across images
- *visually neutral*: hue, contrast, and visual saliency have all been shown to affect eye movement [34]. In order to test for memory-related movement, images would preferably minimize the effect of these properties.

The stimulus used achieves all the objectives. The images were generated for use as experimental stimulus and are referred to as *fribbles*. The original set of models were generated by Mike Tarr from the Department of Psychology at Carnegie Mellon University [4], however our source comes directly from its use as a data set for an experimental study in haptic research by Yildrim & Jacobs [44]. The adapted data set provide the additional benefits of being monochromatic and increasing slightly stronger inter-relation features.

Fribbles are made-up objects that categorise into species. Each species is visually distinct, but *fribbles* within a species are only a small Hamming distance, or substitutions, away from each other [45]. In other words, each object is made up of subset of components shared across the species. Documentation for the differences can be found in section C.



(a) Category 1

(b) Category 2

(c) Category 3

(d) Category 4

Figure 3.2: Examples from each species (or category) of fribble

[4]Stimulus images courtesy of Michael J. Tarr, Center for the Neural Basis of Cognition and Department of Psychology, Carnegie Mellon University, http://www.tarrlab.org

Our experiment utilized four species of fribbles, an example of each is shown in figure 3.2. The collection of images showing the same species of fribble will be referred to as a *category*. Every fribble is distinct inter-species and intra-species in their structure and composition. All images were shown in the same size, although the proportion of the fribbles to the image differed across fribbles. The orientation of the fribble within a species was consistent. An image shown multiple times was shown in exactly the same configuration.

3.2.3 Preparation

The stimulus was shown using the application described in the previous section. The corresponding test script can be found in section B.1.1 of the Appendix. The application was displayed on a 27in monitor Viewsonic Monitor connected to an Intel i7 PC running Windows 8.1. To collect eye-tracking data, the Tobii EyeX was placed on the bottom edge of the monitor (see the red lights in figure 3.3). Each participant was asked to calibrate with the Tobii tracker prior to the start of the experiment. There were no notable difficulties with calibration, regardless of whether the participant wore glasses or contact lenses. An image of the physical set-up is shown in figure 3.3.

Figure 3.3: A reconstruction of the actual physical setup of the data collection trial

All experiments were conducted in the Brain & Behaviour Lab in the Department of Bioengineering at Imperial College London. All participants were students at Imperial College London studying in the Department of Bioengineering or the Department of Computing. A privacy curtain was used to separate the participant from the surrounding environment, and as an additional measure against audio disruption participants were encouraged to wear noise-reducing ear protection as seen worn in figure 3.3. No notable occurences took place and observer notes will be made available with the data set. Experiments lasted approximately twelve minutes. In total, data for twenty participants was collected and analyzed.

3.2.4 Procedure

The experiment consisted of two phases: a study phase and a score phase. The participant was not told the number of phases beforehand, although they were given a rough estimate of

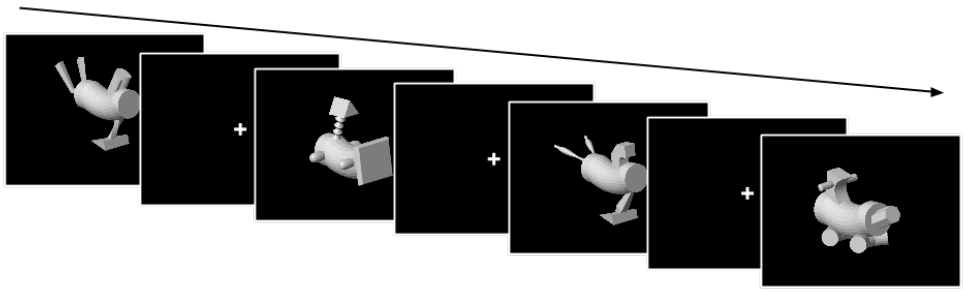


Figure 3.4: Subsection of series of images show during the study phase. Fribbles are from category 3, category 1, category 3 and category 2 in order from left to right.

the length of the experiment. Each phase was preceeded by instructions in white-on-black text presented on the screen. Instructions appeared on the screen until the participant pressed the green button to continue. The only verbal instruction provided to the participant was the location of the green and red keys necessary to conduct the experiment. The keys can be seen in figure 3.3, and were the space (green) and insert/numpad0 (red) keys. Additionally, participants were able to ask any questions to the observer sitting behind them, although such occurences were rare (and related to clarification of an on-screen instruction during the interval between phases). The observer's primary purpose was to note down any occurrences which may lead to invalid results, such as major distractions or unexpected program behaviour. Again, such occurrences were rare.

The experiment began and ended with a calibration test. The participant was asked to follow a dot which appeared in the eight cardinal positions of the screen. The dot appeared for five seconds before moving to the next counter-clockwise position. Eye movement data and the position of the dot were recorded in the case of experimental results being affected by calibration.

3.2.4.1 Study Phase

Before the study phase, participants were given two instructions:

- "please look at the cross until it disappears"
- "otherwise, please look freely at the images"

No indication of the purpose of the experiment was provided. During the study phase, a total of twenty images were shown from across three categories (in other words, twenty fribbles were shown from three species). Each image was preceeded by an image with a centering cross, to provide a consistent starting point for eye movement for each image across participants, that displayed for three seconds.

The distribution of images was: five images from category 1, five images from category 2, and ten images from category 3. The images from category 3 were interpolated between the other two categories such that no category was shown twice in a row. Each image was displayed for exactly five seconds before automatically switching to the next centering cross (or text to signal the end of the phase). No image was repeated.

3.2.4.2 Score Phase

Preparation for the score phase began immediately after the conclusion of the study phase. The instructions for the score phase were more complicated than the study phase and included:

- "Phase 2 is a game"

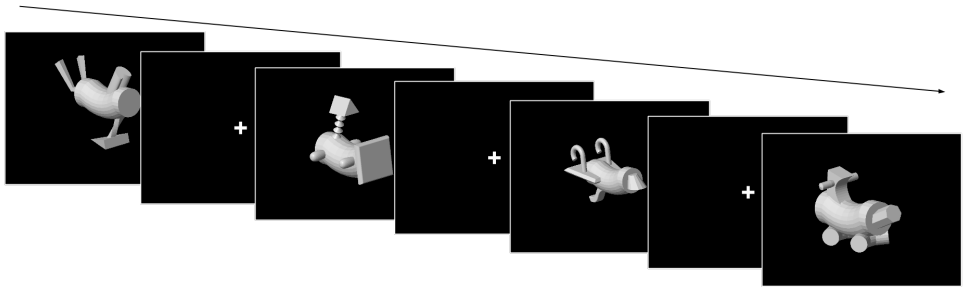


Figure 3.5: Subsection of series of images show during the score phase. Fribbles are from category 3, category 1, category 4 and category 2 in order from left to right.

- "The goal of the game is to maximize the number of points"
- "If You Saw The Image In Phase 1, Press Green"
- "If You Did NOT See the Image In Phase 1, Press Red"
- "You Get +1 Points For A Correct Answer And -1 For An Incorrect Answer"

In essence, participants were asked to press one of two buttons per image: green if the image was shown during the study phase, and red if they image was not shown during the study phase. Participants had to press a button per image, and were scored 1 point for a correct response, and -1 point for an incorrect response.

During the score phase, a total of 40 images were shown from across all four categories. Each image was preceeded by an image with a centering cross, to provide a consistent starting point for eye movement for each image across participants, that displayed for three seconds. Each image was displayed for five seconds. If the participant had not pressed the red or green button by five seconds, the screen turned black until they input a response, which then triggered the transition to the next centering cross.

The distribution of the images was equal, with ten images from each category. If we divide the order into groups of four images, each category was represented exactly once in each group. The positions of the category alternated, and was the same for all participants[5]. The inclusion of the fourth category balanced the number of seen versus unseen images (for collecting participant input), and to make the number of images seen larger, thereby making the task more difficult.

Due to the increased number of images and the additional time for participant input, the score phase was 2-3 times longer than the study phase.

3.2.5 Results

For ease of nomenclature, a displayed image will be referred to as a *Viewing Experience* (VE). Therefore each participant produced sixty VEs (20 in the study phase and 40 in the score phase). Each VE is preceeded by the three second centering image.

3.2.5.1 Participant Performance

Performance of the participants ranked high, with a mean score of 18.2 points (an average accuracy of 72.8%). All scores are shown in figure 3.6a. The lowest score was 10, or 25 correct and 15 incorrect responses. The highest score was 26, or 33 correct responses and 7 incorrect responses. Figure 3.6c shows the breakdown of each score into its four constituent parts: correctly identified

[5]Due to an application bug, the first five participants saw a different fribble from within a species than the last fifteen. The exact ordering shown to each participant is available with the data set.

as seen (green correct), correctly identified as not seen (red correct), incorrectly identified as seen (green incorrect) and incorrectly identified as unseen. Figure 3.6b shows how performance varied across categories, or species, of the images. Unsurprisingly, category 4, which was novel to the score phase, resulted in the best accuracy. Finally, figure 3.6e showed the breakdown of responses per each VE. Each index contained the same category and same level of novelty, although the exact image could vary. Correct responses for the first twelve images (mean points = 11.33) and final twelve images (mean points = 9.50) showed slight decreased performance on average over time.

Timestamps were also collected for each keypress, to show the amount of time per VE before the participant responded. Figure 3.6d shows the difference between mean time spent on correct and incorrect responses. In general, participants spent more time on incorrect responses. Figure 3.6f shows the mean time per VE index.

3.2.5.2 Eye Data

Collection for all sixty images shown to each participant was successful. Data indicated that for the 500ms prior to all 1200 VEs, gaze was directed (number of points greater than 150px less than 50%) near the center of the screen for all but 47 VEs[6].

Figure 3.7 displays examples of the raw gaze data collected for two images from the same participant.

3.2.5.3 Fixation Detection and Saccade Extraction

Processing of gaze points into fixations used the I-DT algorithm as described in section 2.2.5.3, with a spatial threshold of 35 pixels. We discuss the implication of this threshold in section 4.2.1. Saccades are calculated as the vectors connecting successive fixations, without further processing.

Figure 3.7 shows two examples of scanpaths from the same participant with the corresponding image shown in the background. Figure 3.7a is a category 1 image and figure 3.7b is a category 2 image. The arrows represent saccades and the circles represent fixations, with the intensity (size) of the circle proportional to the dwell time (or fixation duration).

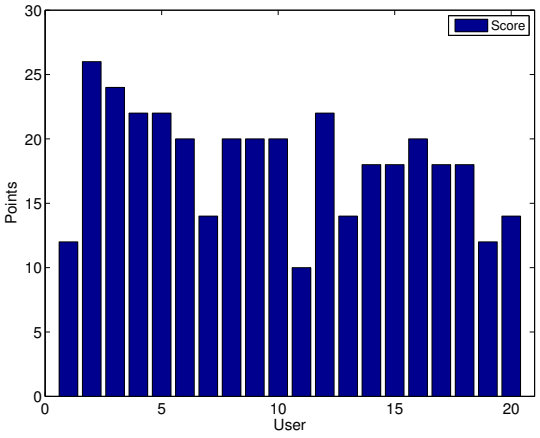
3.2.5.4 First Versus Second Viewing

	μ_{sl}	μ_{st}	σ_{at}	σ_{aft}	σ_{ft}	Σ_{tft}
First	126.418	42.623	758.528	450.869	103.936	4266.571
Second	157.842	79.758	702.328	350.446	94.479	3868.940
p-value	$p < 0.001$	$p < 0.001$	$p < 0.05$	$p < 0.001$	$p < 0.05$	$p < 0.001$
DOF	399	399	399	399	399	399
t-value	-13.020	-6.362	2.617	4.912	2.513	8.291

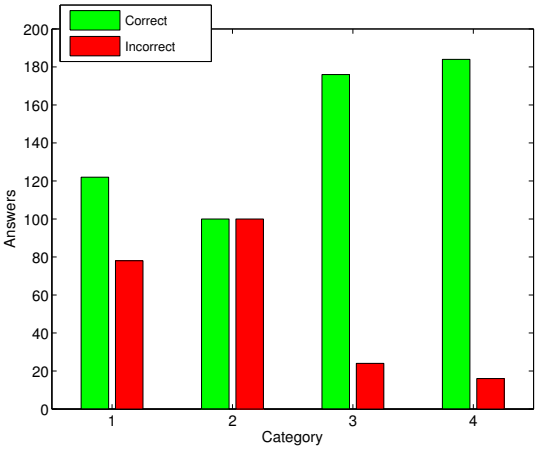
Table 3.1: Results table for the paired t-test for the recognition effect

We will now show the utility of the data by performing some initial analysis to show the differences between first and second image viewing. We chose six metrics, listed in table 3.1, which showed statistical differences between first and second image viewing. However, in order to show that these effects are not mediated by other differences, we will also show the differences between two other groups: first and second *category* viewing and first and second *phase* viewing. These two

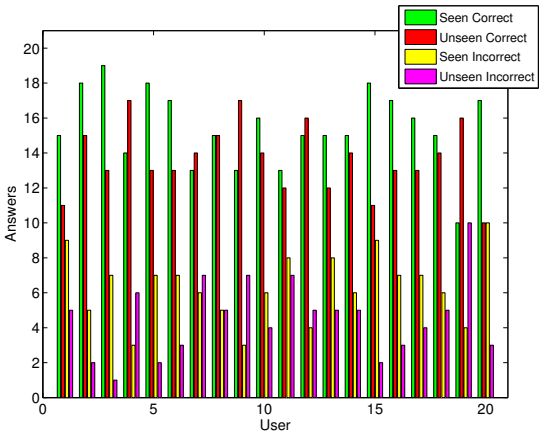
[6]For purposes of classification, and after visual inspection of the VEs, we decided to use all 1200 VEs. Future uses may want to be more selective, and the centering data is all available with the data set



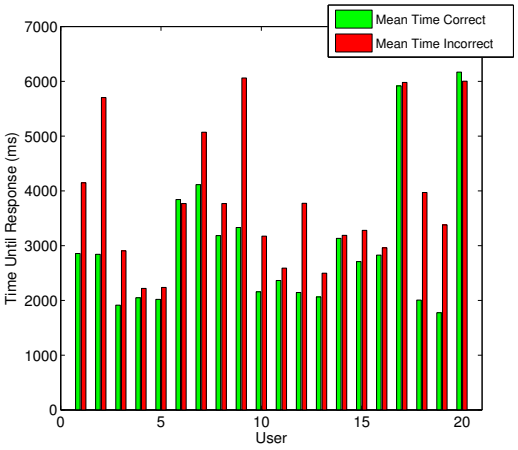
(a) Total number of points per participant



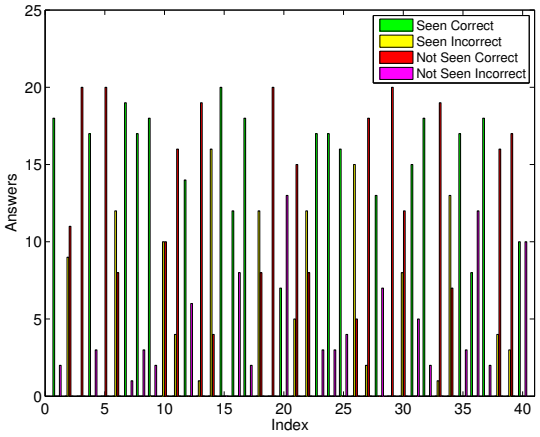
(b) Number of correct and incorrect responses per category



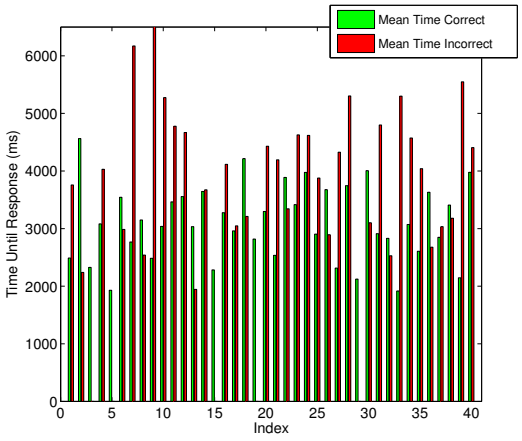
(c) Breakdown of responses from each participant



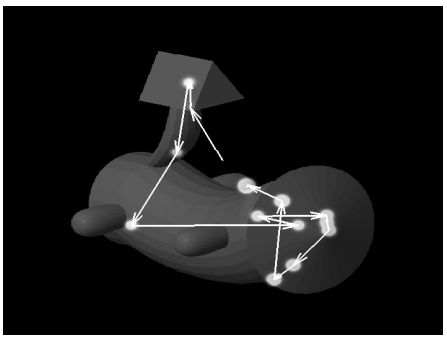
(d) Mean time spent on responses per participant



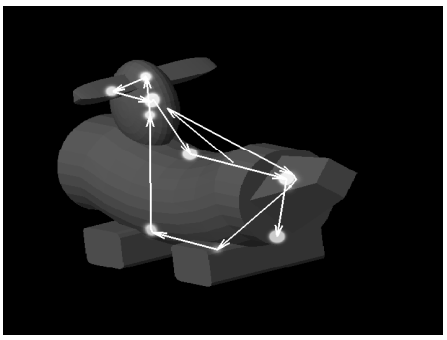
(e) Responses for each VE index



(f) Mean time spent on responses per VE index



(a) Eye movement data for a category 1 image



(b) Eye movement data for a category 2 image

Figure 3.7: Example of two scanpaths from the same user for two different categories

	μ_{sl}	μ_{st}	σ_{at}	σ_{aft}	σ_{ft}	Σ_{tft}
First	137.759	69.354	715.315	335.180	95.957	3944.957
Similar	123.832	37.838	776.645	452.387	107.004	4331.949
p-value	$p < 0.001$	$p < 0.001$	*$p > 0.05$*	$p < 0.001$	*$p > 0.05$*	$p < 0.001$
DOF	398.000	398.000	398.000	398.000	398.000	398.000
t-value	4.190	4.114	-1.883	-3.818	-1.878	-4.280

Table 3.2: Results table for the two-sample t-test for the similarity effect

map to two different possible other effects, besides recognition, which could cause differences in eye movement.

The first is the *similarity* effect; seeing a similar image again. This occurs during the study phase when a participant sees an image from the same category again. Recall that we are seeking displays recognition or recollection, although displays of similarity will also be useful later on. The second is the *task* effect; the difference in movement based on what task the participant is given. Yarbus showed that a task does affect eye movement [35], and in our protocol a participant is given a specific task in phase 2 (the score phase) and an ambiguous task in phase 1 (the study phase).

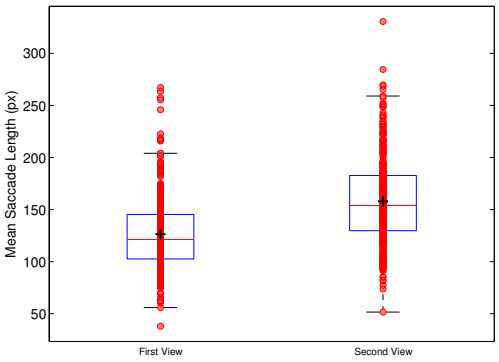
We will show the differences between all six metrics, and justify our claim by showing the differences are not caused by the other two effects. In order to test first and second viewing, which we refer to as the *recognition* effect, we used a paired t-test between the first and second Viewing Experiences (VEs) the participant saw the exact same image. Across twenty participants, this gave us 400 datapoints. For the similarity effect, we used two groups of images for a two-sample t-test: the first image shown of each category and the second image shown from each category. There were 80 datapoints. For the task effect, we compared all the images in phase 1, and all the images in phase 2 not seen in phase 1, using a two-sample t-test. This was a direct comparison of how eye movement changed while viewing an image for the first time in the study phase compared to the score phase. We used category 1 and 2 images for a total of 400 VEs.

The data for tests on the metrics the recognition effect is found in table 3.1, the similarity effect in table 3.2 and the task effect in table 3.3.

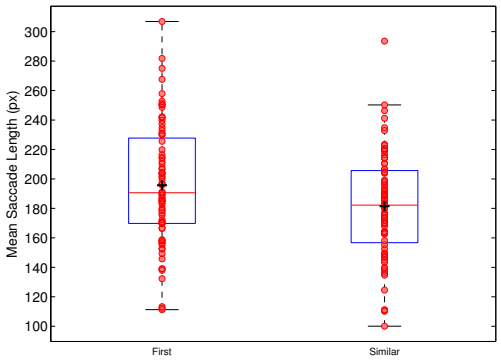
Mean Saccade Length Mean saccade length μ_{sl} is the mean of the magnitude of the saccade vectors. The paired t-test for the recognition effect shows a significant statistical difference ($p < 0.001$) between the two means ($t(399) = -13.02$). μ_{sl} increases on the second viewing. The opposite effect is observed in relation to the similiarity and task effects; statistical differences are shown between the means, but the means decrease. As can be seen through the three box plots

	μ_{sl}	μ_{st}	σ_{at}	σ_{aft}	σ_{ft}	Σ_{tft}
Phase 1	195.717	89.223	661.134	300.680	155.385	3852.118
Phase 2	181.498	82.849	698.236	334.553	164.943	3964.592
p-value	$p < 0.05$	$p > 0.05$	$p > 0.05$	$p > 0.05$	$p > 0.05$	$p < 0.05$
DOF	79.000	79.000	79.000	79.000	79.000	79.000
t-value	2.589	1.684	-1.155	-1.417	-0.996	-2.461

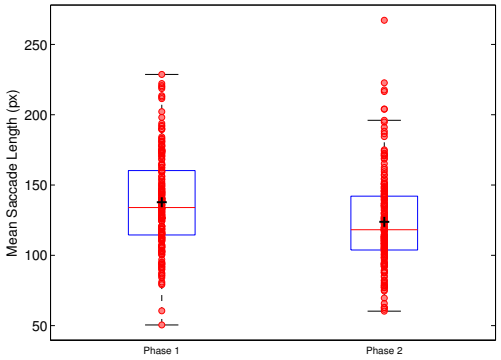
Table 3.3: Results table for the two-sample t-test for the task effect



(a) Recognition Effect for μ_{sl}



(b) Similarity effect for μ_{sl}



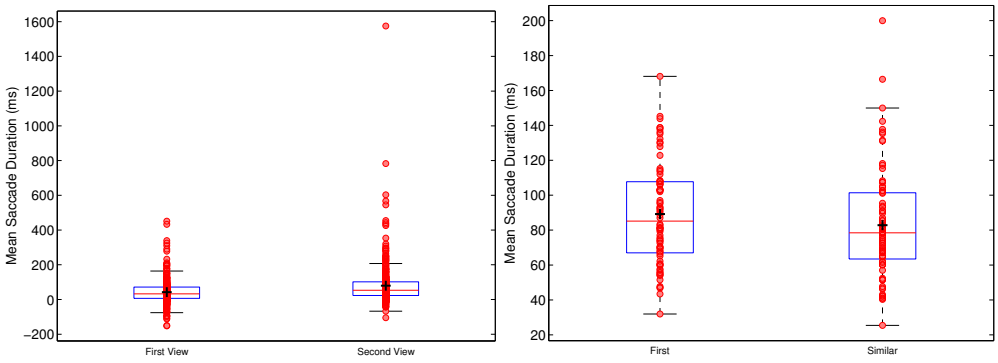
(c) Task Effect for μ_{sl}

Figure 3.8: Boxplots comparing mean saccade length (μ_{sl}) for the three effects

in figure 3.8, the increase of μ_{sl} only incurs for first and second image viewing.

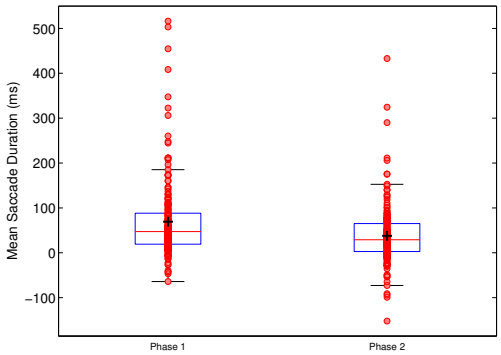
Mean Saccade Duration Mean saccade duration μ_{st} is the mean of the time taken for each saccade. The recognition effect shows a significant ($p < 0.001$) increase ($t(399) = -6.362$) in μ_{st} across first and viewings. There was against a significant but opposite difference for the similarity effect; duration decreased. There was no significant difference for the task effect, although as can be seen in figure 3.9, the group mean decreased.

Standard Deviation of Total Fixation Time in Each AOI Standard deviation of total fixation time in each AOI σ_{at} is the standard deviation between the total amount of dwell time (or fixation duration) the participant spent looking at each AOI. To determine the AOIs, we used a human analysis approach, which approximated the areas of interest for each category of image. The AOIs for each category are the coloured boxed in figure 3.10. There was a significant decrease in σ_{at} for the recognition effect ($p < 0.05$, $t(399) = 2.617$), and no significant differences in the similarity or task effects. The box plots are in figure 3.11.



(a) Recognition Effect for μ_{st}

(b) Similarity effect for μ_{st}



(c) Task Effect for μ_{st}

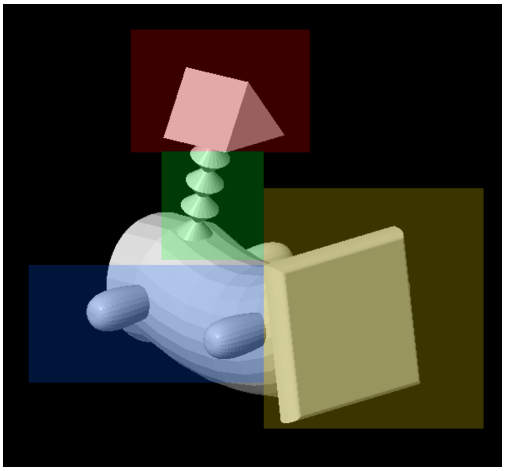
Figure 3.9: Boxplots comparing mean saccade duration (μ_{st}) for the three effects

Standard Deviation of Fixation Duration in AOI Standard deviation of fixation duration in each AOI σ_{aft} is the deviation of fixation durations after grouping consecutive fixations in the same AOI as one fixation. We found a significant decrease with regards to the recognition effect ($p < 0.001$, $t(399) = 4.912$) but a significant increase with regards to the similarity effect ($p < 0.001$, $t(79) = -3.818$). There was no significant difference in the task effect as can be seen in figure 3.12.

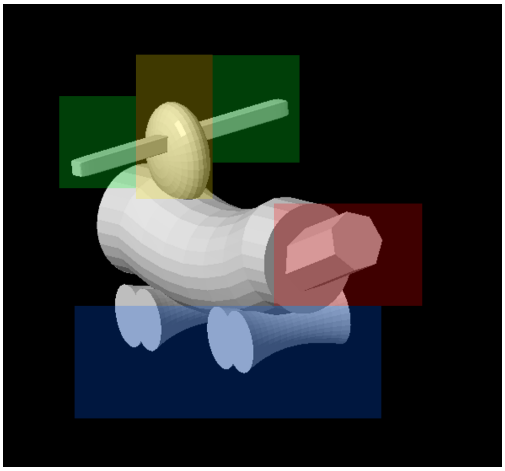
Standard Deviation of Fixation Duration Standard deviation of fixation duration in each AOI σ_{ft} is the deviation of dwell times for each fixation in the scanpath. Only the recognition effect showed a significant difference with a decreased deviation ($p < 0.05$, $t(399) = 2.513$). The two groups and their means alongside the similarity effect and the task effect is drawn in figure 3.13.

Total Fixation Time Total fixation time Σ_{tft}, or the total amount of time the participant spent per VE in fixations rather than in saccades, showed that participants spent less time in fixations for the second viewing of images rather than first. This correlates with our first metric, mean saccade length, as longer saccades would naturally use more time. There was significant differences in all three effects; however, while Σ_{tft} decreased for the recognition effect ($t(399) = 8.291$), it increased for both the similarity ($t(79) = -4.280$) and task ($t(398) = -4.280$) effects.

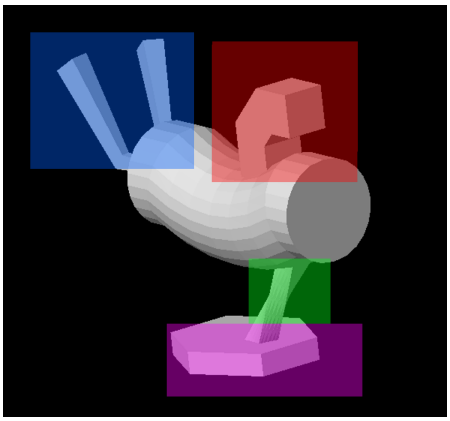
Summary Measuring these six metrics, which were hand-picked as they showed significant differences for the recognition effect, serve two purposes: first, they illustrate the utility of our data and our protocol for finding results for different effects; two, they give us a platform for our classification attempt. However, inspection of the box plots reveals that while significant



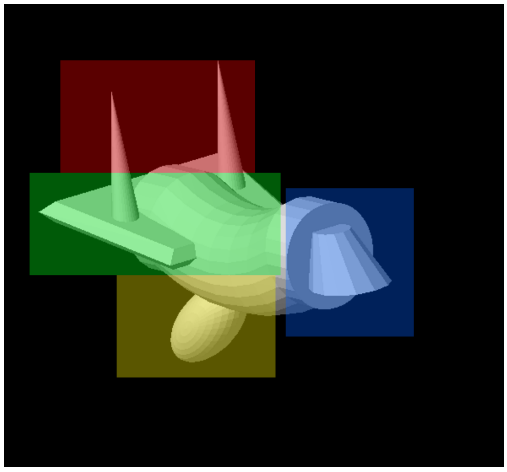
(a) Category 1



(b) Category 2



(c) Category 3



(d) Category 4

Figure 3.10: Human picked area of interest for each category

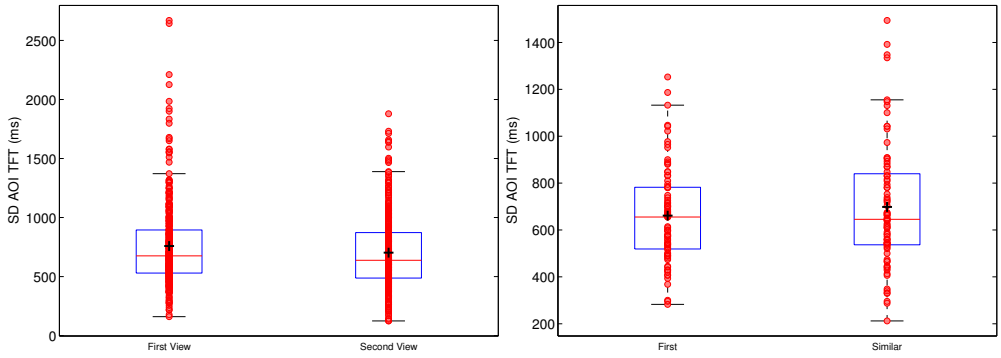
differences may occurs, that implies neither causation nor good discriminative powers; two properties we need in order to build a good classifier.

3.2.5.5 First versus n-th Feature Viewings

With the collected data, we can also compare the results in terms of the individual features of each image (fribble) in VEs. For this analysis, each VE was analysed for which features had appeared before in the experiment. Each fixation was then labelled: targeted towards a seen feature (1); targeted towards an unseen feature (2); neither/unknown (3). categorising all the fixations under their label, and deconstructing the original gaze points which made up the fixation, we show data for one metric:

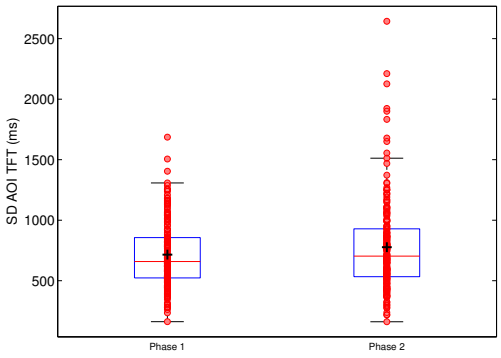
- mean fixation duration $\mu_f t$

For this comparison, we used a two-sample two tests of two groups: mean fixation duration per *seen* feature and mean fixation duration per *unseen* feature. This was calculated by finding the total fixation time on the area of interest of the feature, labelling the fixation as *seen* and *unseen*, and for each scanpath, aggregating the fixation duration for *seen* and *unseen* features.



(a) Recognition Effect for σ_{at}

(b) Similarity effect for σ_{at}



(c) Task Effect for σ_{at}

Figure 3.11: Boxplots comparing standard deviation of AOI total fixation time (σ_{at}) for the three effects

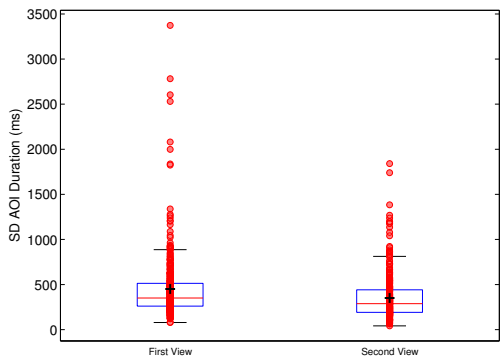
	μ_{sl}
Seen	375.491
Unseen	449.610
p-value	$p < 0.001$
DOF	1493
t-value	-5.6764

Table 3.4: Results table for the two-sample t-test for the subimage recognition effect

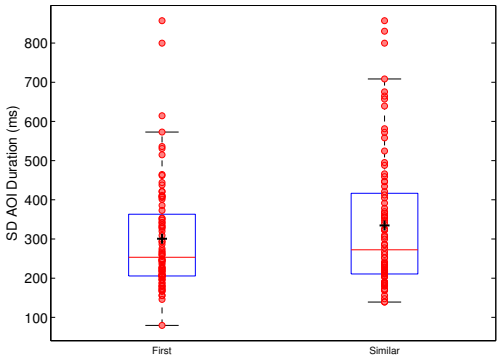
We divided each by the number of *seen* features and *unseen* features they actually viewed (which in most cases were the number of seen and unseen features in the image).

Table 3.4 shows the results of this comparison measuring just mean fixation duration. There is a significant increase in the mean fixation duration ($p < 0.001$, $t(1493) = -5.6764$). This is related to the work mentioned in section 2.3.1 by Greene and Rayner, who studied the effects of familiar distractors in visual search [40]. However, while they found that familiar distractors had fewer fixations, slightly different from our results the duration was comparable to the unfamiliar distractors.

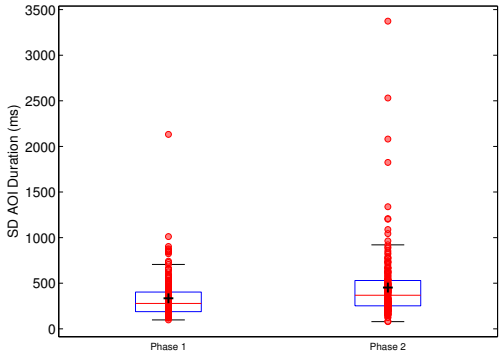
Summary While subimage recognition is a feature we use for classification, we wanted to create a data set with this property of decomposability. Through our initial analysis of just mean fixation duration, we have found a significant difference between *seen* and *unseen*; therefore further investigation may be worthwhile.



(a) Recognition Effect for σ_{aft}

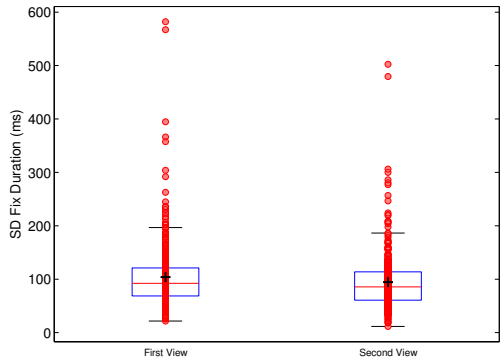


(b) Similarity effect for σ_{aft}

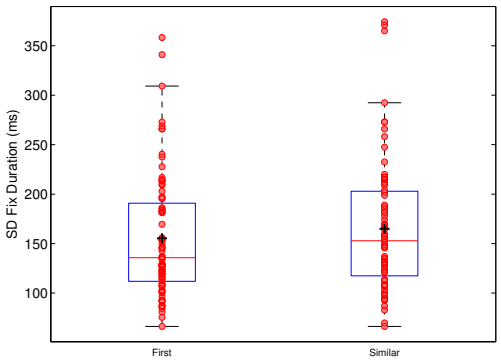


(c) Task Effect for σ_{aft}

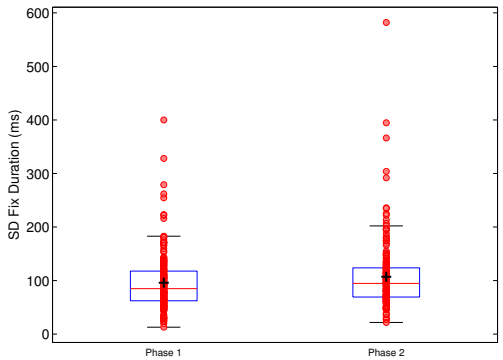
Figure 3.12: Boxplots comparing standard deviation of AOI fixation duration (σ_{aft}) for the three effects



(a) Recognition Effect for σ_{ft}

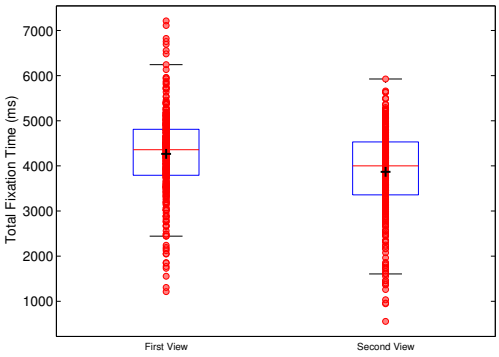


(b) Similarity effect for σ_{ft}

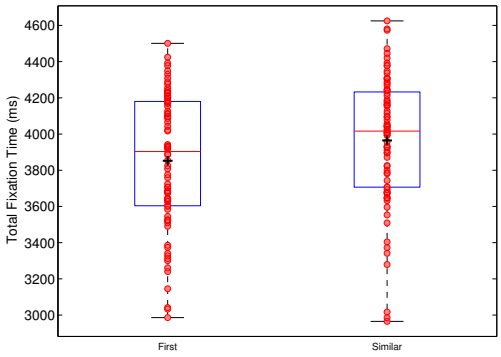


(c) Task Effect for σ_{ft}

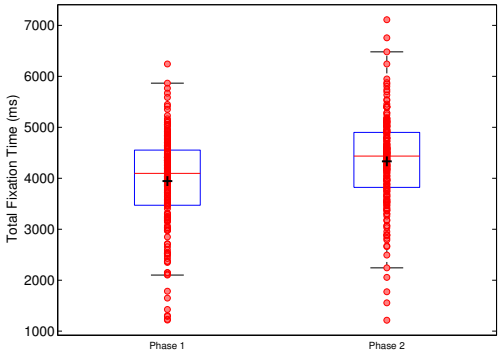
Figure 3.13: Boxplots comparing standard deviation of fixation duration (σ_{at}) for the three effects



(a) Recognition Effect for Σ_{tft}



(b) Similarity Effect for Σ_{tft}



(c) Task Effect for Σ_{tft}

Figure 3.14: Boxplots comparing total fixation time (Σ_{tft}) for the three effects

Chapter 4

Classification

4.1 Method of Evaluation

Classification was tested against the data collected from all twenty participants of the data collection trials described in Section 3.2.

4.1.1 Cross Validation

The data was separated by using the Leave-One-Out-Cross-Validation (LOOC) method (see 2.1.6 for background). As figure 4.1 captures, this created 20 independent tests and 20 corresponding models of each classifier. The models were trained with 19 participants' Viewing Experiences (VEs), with the remaining, i.e. *left out*, user's data used as the test set.

Not all sixty VEs per participant were used for testing. Instead, to create equal distributions of *seen* and *unseen* images, only forty images were used. These were further divided into their respective image categories, as many of the classification methods relied on category-based segregation. To maintain equal distribution at both category and response-level, the forty images did not include any images from category 4 (whose images were only shown once). Also, five images were excluded for categories 1 and 2. More precisely, the five images that were novel to the score phase were excluded. In total, the forty images broke down in to:

- the 5 category 1 VEs from the study phase and the matching 5 category 1 VEs from the score phase
- the 5 category 2 VEs from the study phase and the matching 5 category 2 VEs from the score phase

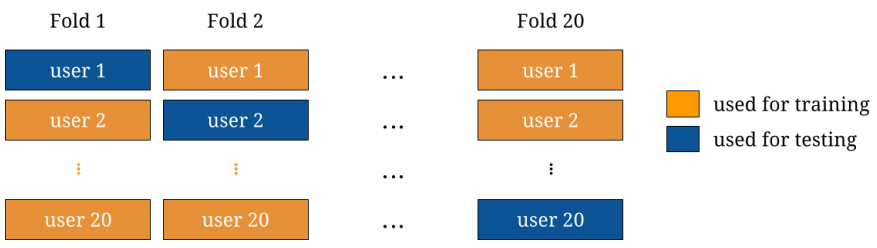


Figure 4.1: Diagram showing how the data from twenty participants was divided in Leave-One-Out-Cross Validation (without the subdivision into categories)

- all 10 category 3 VEs from the study phase and the matching 10 category 3 VEs from the score phase

In total, this created 40 tests for each of the 20 runs, or a total of 800 predictions from each classifier. There were sixty corresponding models, mapping to the twenty participants and three categories.

For the training data, category 4 images were excluded. Training data also mainted equal relationships, but did not discriminate which images were selected. Therefore it randomly sampled 200 category 1 VEs, 200 category 2 VEs and 400 category 3 VEs, such that there was a 50/50 split between first and second viewing. In total, each classifier was trained with 800 VEs for each of the 20 runs.

4.1.2 Performance Metrics

For each classifier, results were aggregated into a confusion matrix, which details the number of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) aggregated for all twenty folds. The positive case for our classifier is defined as *seen*. Accuracy[1], sensitivity, specificity and precision scores were averaged across each fold (and category), and an ROC curve was plotted and graphed for the aggregated results. The ROC curve is given for each classifier, and the performance metrics are compiled into a table, along with the area under the ROC curve (AUC) value.

4.2 Data Pre-Processing

4.2.1 Fixation Detection

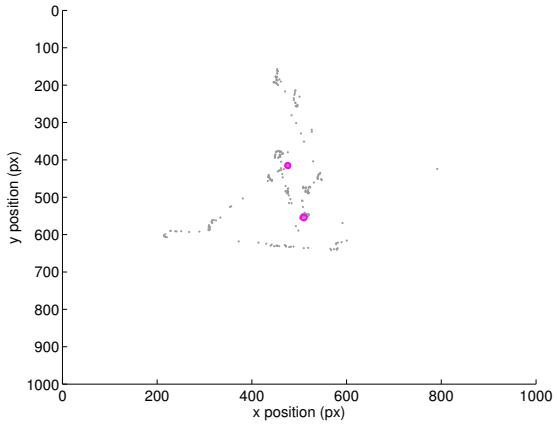
For both training and testing, the input to our classifiers is a scanpath, represented as a series of fixations, with dimensions (x, y, t, d), with x and y position in pixels, t the timestamp of the fixation relative to the beginning of the VE in milliseconds, and d the duration of the fixation in milliseconds. In order to translate the raw gaze data into fixations, we use our implementation of the I-DT algorithm, described in section 2.2.5.3. For the dispersion threshold value, we used a value of 35 pixels. Our initial estimation was based on analysis performed by Blignaut on optimum threshold values for I-DT algorithm, which found a range of 0.7 to 1 degrees to be the optimal threshold [46]. With the average eye position around 45cm, and through empirical analysis, we chose a threshold value of 35 pixels. The effect of the threshold value can be seen in figure 4.2, for three values: 5 pixels (figure 4.2a), our selected 35 pixels (figure 4.2b) and 75 pixels (figure 4.2c).

4.2.2 Automatic AOI generation

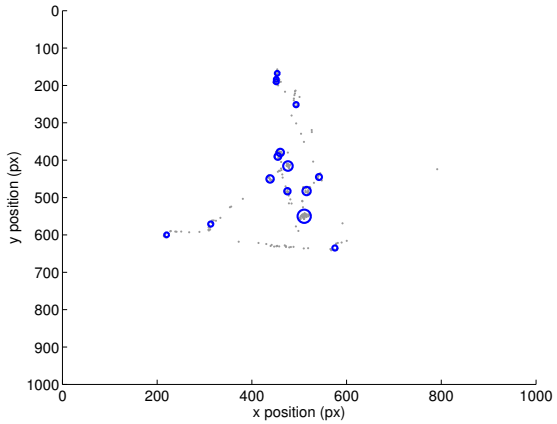
Many of the metrics discussed by the research into the relationship between eye movement and memory use Areas-of-Interest (AOI) demarcation. AOIs are image-dependent areas which have semantic and visual value, such that they tend to attract the attention of users. Roughly speaking, there are three strategies to identify AOIs within an image:

1. manually, i.e. a human marks the areas
2. using characteristics of the image, such as location of edges, distribution of light/shadow
3. using aggregated eye movement data

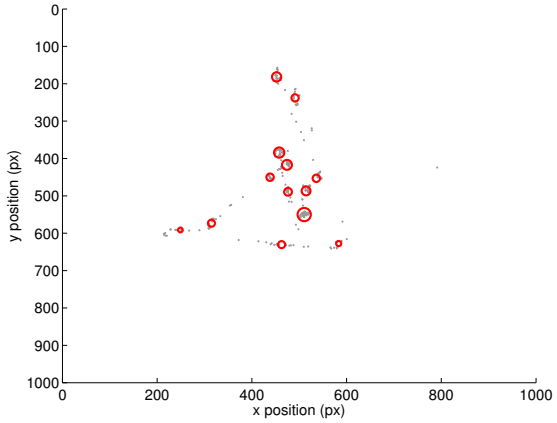
[1] We draw a distinction between average accuracy, which is accuracy averaged over the 60 models, and aggregate accuracy, which is the ratio of the total number of correct predictions over the total number of predictions



(a) I-DT threshold of 5 pixels



(b) I-DT threshold of 35 pixels



(c) I-DT threshold of 75 pixels

Figure 4.2: Calculated fixations using I-DT with a dispersion threshold of (a) 5 pixels, (b) 35 pixels and (c) 75 pixels

While all three methods have appropriate use-cases, this investigation pursues automatic classification based on eye-movement, so unless otherwise stated, the third strategy, based on eye movement, was used as it best fits these objectives. However, for certain tasks, we have relied on human analysis and future work may find more benefit in image analysis.

We use two methods for calculating areas of interest, both based on the spatial distribution of fixations. An area of interest is defined as any area of the image with a high density of fixations, or in other words, *clusters* of fixations. The two methods we will use are the unsupervised clustering algorithms MoG and DCS (see section 2.1.3).

Figure 4.3 shows the difference between the AOI assignment generated by each clustering method. AOIs are represented by the fixations that were used to generate them and each fixation is coloured with their respective AOI assignment.

4.2.2.1 MoG AOI Generation

The first AOI generation technique is to use a Mixture of Gaussians (MoG), where each bivariate normal distribution of the mixture model represents one AOI. MoGs use a soft-assignment where responsibility for a point can be shared across multiple clusters. The assignment is a posterior probability, and a hard assignment can be derived by maximizing the posterior, i.e. using a MAP estimate. The EM algorithm is used as the principal method to calculate the optimal parameters. In order to make the MoG more robust, EM estimation is repeated five times with random starts, and the mixture model with the largest log-likelihood (with respect of the training data) is chosen. The number of iterations is capped to 500 for computational performance.

Traditionally, the disadvantage of a MoG clustering technique is having to know the number of clusters K *apriori*. To overcome this requirement, K is found iteratively by comparing the *goodness of fit* of MoGs with different K values, starting at K is equal. The criterion used to measure *goodness of fit* is the Bayesian Information Criterion (see 2.1.3.2). As an additional measure to overfitting, K values were capped to 10, twice the number of components in each image. Figure 4.3c shows the AOIs generated by the MoG algorithm for a category 1 image.

4.2.2.2 DCS AOI Generation

DCS AOI generation uses the mean shift procedure as described in 2.1.3.3. For the mean shift step, the weightings are calculated using a Gaussian kernel with zero mean and covariance $\sigma_s I$:

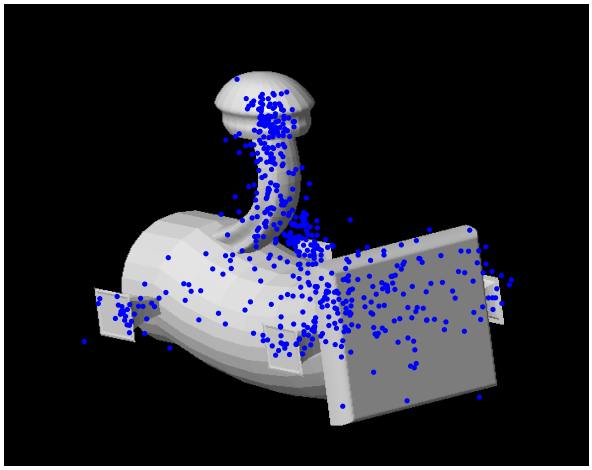
$$k_s(x, y, t) = \exp\left(-\frac{x^2 + y^2}{\sigma_s^2}\right)$$

This kernel, suggested by DeCarlo and Santella, significantly decreases the sensitivity of points σ_s pixels away. σ_s controls the scale in terms of Gaussian distance, and tuning this parameter affects the size and number of AOIs generated, as well as sensitivity to outlier fixations.

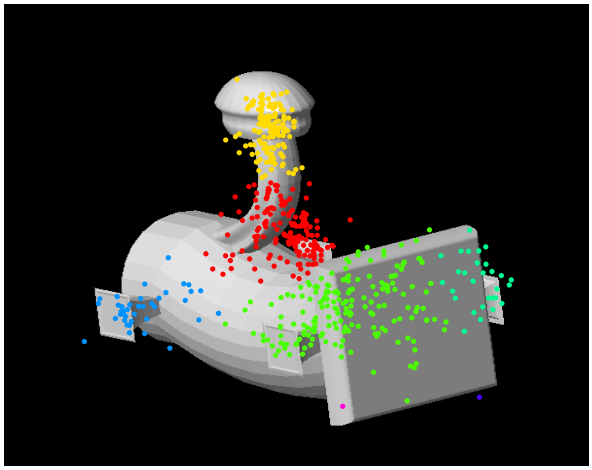
To assign a new fixation to the existing clusters we use a 5-NN classifier (see section 2.1.2.2), where each neighbour is an existing fixation, and the class labels are the AOIs. We use an inverse distance metric, where the weight of a neighbour's vote is inversely related to its distance away from the new fixation. Therefore, closer neighbours have a larger influence. The tie-breaker between equal votes is a random decision.

Clustering is accomplished by finding populations of fixations who are no greater than ϵ pixels away from each other. It is useful to make σ_s and ϵ similar, and for this investigation, both were set to 50 pixels.

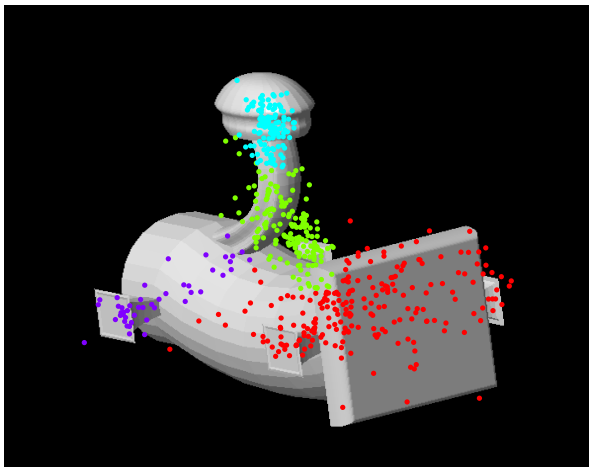
Figure 4.4 shows the effect of changing the value of σ_s and ϵ to the generation of AOIs. Too small a value, such as in figure 4.4a, too many AOIs will be generated. A large value for σ_s may



(a) A set of fixations aggregated over multiple participants while looking at images in category 1

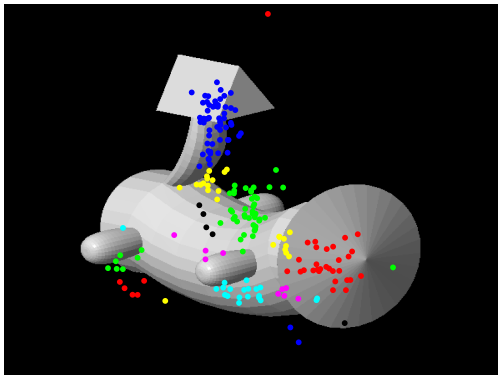


(b) Assignment of fixations to AOIs as calculated using the DeCarlo-Santella algorithm

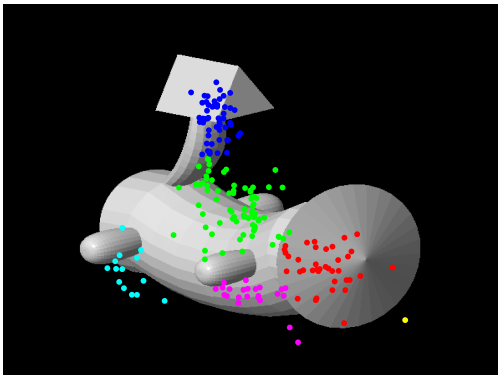


(c) Assignment of fixations to AOIs as calculated using Mixture of Gaussians with 6 clusters

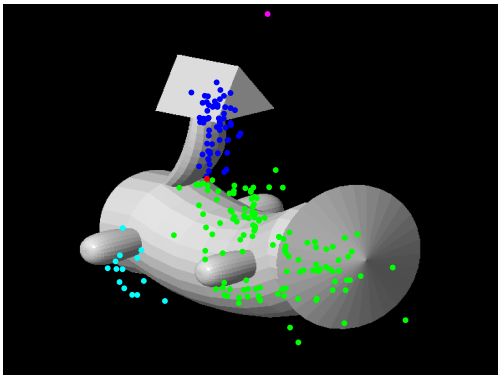
Figure 4.3: AOI assignment of (a) using (b) DCS and (c) MoG



(a) DCS at $\sigma_s = 25$ pixels



(b) DCS at $\sigma_s = 50$ pixels



(c) DCS at $\sigma_s = 100$ pixels

Figure 4.4

result in too few AOIs; notice how in figure 4.4c the image is represented by three large AOIs and two small outlier AOIs.

Figure 4.4b is the assignment when the value of σ_s and ϵ is 50 pixels, or the one used by our classifiers. The DCS algorithm will produce six AOIs but one of the AOIs contains only one point (it is marked in yellow near the bottom right corner of the figure). In figure 4.3b, there are two outlier points in the bottom right corner, marked magenta and purple respectively. To prevent outlier fixations from generating unnecessary AOIs, we pass the assignment (for category-wise AOI generation) through a noise-filtering process. During this cleaning, any AOI which receives less than 1% of total dwell time is removed. The fixations within the removed AOI are either reassigned to the closest AOI (using the same KNN classifier as we used for assignments), or if they are greater than 50 pixels away from the AOI, removed completely.

4.3 Discriminative Classification

4.3.1 Approach

Our first attempt at classification is to use the metrics we identified in the initial data analysis section 3.2.5.4 and compute a probability distribution over it. We will be using logisitic regression (Section 2.1.2.3) as our model due to its speed and effectiveness. As a discriminative classifier, logistic regression computes a function which maps our feature vectors to a value between 0 and 1. We can use a threshold of 0.5 to determine which class label to use.

Training the classifier is fitting the training data to the function; namely finding the different weights w of the function which maximize the accuracy of the classifier.

In total we will be testing six sets of features:

- MSL-CL: mean saccade length
- VEC-CL: mean saccade length and mean saccade duration
- DUR-CL: standard deviation of total AOI duration, standard deviation of AOI duration, standard deviation of fixation duration and total fixation time
- EXP-CL: aggregate of the features in VEC-CL and DUR-CL
- SIG-CL: number of revisits per AOI (two fixations in the same AOI separated by at least one fixation in another AOI), number of repeats per AOI (consecutive fixations within the same AOI), proportion of AOIs visited and ratio of fixations to AOIs
- ALL-CL: all the features above combined

The first four feature sets were explained during the preliminary data analysis in section 3.2.5.4. To reiterate, these were features that showed statistical significance between first and second viewing. The fifth feature set was not previously discussed, but is based on our own personal hypothesis to what may show an AOI as significant (especially with regards to recognition). All of these are metrics of predictability and order; one would expect revisits to correlate with significance, and a larger number of fixations to correlate with greater uncertainty. However, we aim to test this under classification conditions, rather than statistical analysis.

4.3.2 Results

Performance for both classifiers aggregated over the whole cross-validation is represented in figure 4.5, with the performance metrics shown in table 4.5a, the confusion matrices in table 4.8c, and the ROC curves in figure 4.5b.

4.3.2.1 Summary

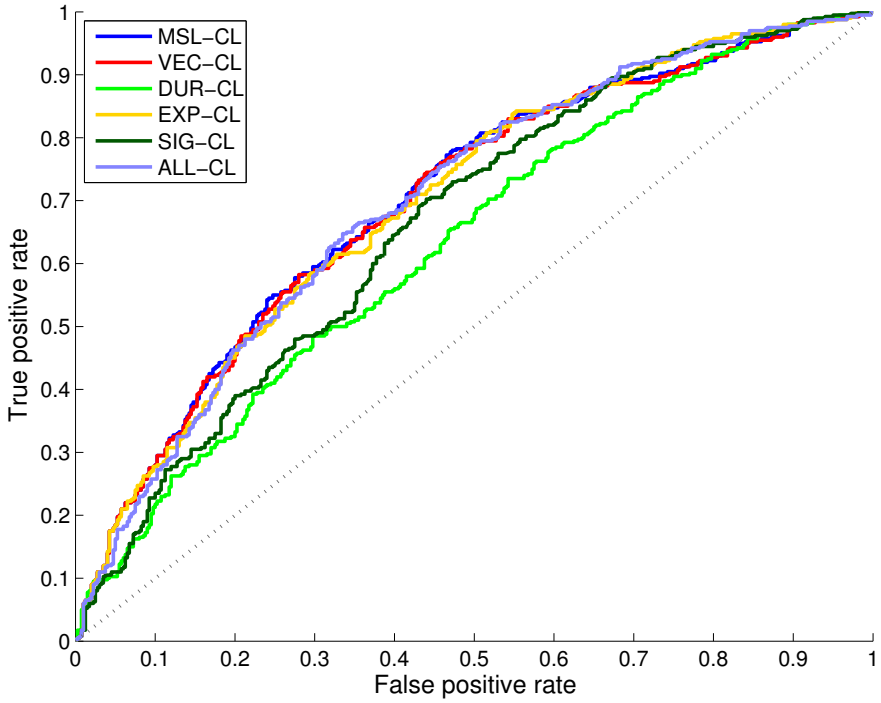
The six classifiers showed disappointing performance, with performance only marginally better than a random classifier. The combined classifier (ALL-CL) and the experimental classifier (EXP-CL) showed the most promise, with average accuracies of 61.3% and 61.0% respectively. The experimental metrics had a slightly lower devation (11.7%) than the aggregate set (13.3%) which may show that the additional features cause greater variability, and therefore, less reliability.

The other four feature sets cover an accuracy range of 56.2 % and 60.4%. The inclusion of mean saccade duration to a feature set comprised of mean saccade length only seems to be marginally lower standard deviation (from 12.9 % down to 12.4%) with no difference in accuracy (60.4%). In fact, out of 800 predictions, only twelve predictions differed between the first two classifiers.

Figure 4.5: Summary of results for MSL-CL, VEC-CL, DUR-CL, EXP-CL, SIG-CL, and ALL-CL classifiers

Classifier	Accuracy		Sensitivity	Specificity	Precision	AUC
MSL-CL	0.604	(0.129)	0.338	0.870	0.593	0.701
VEC-CL	0.604	(0.124)	0.330	0.878	0.591	0.698
DUR-CL	0.562	(0.097)	0.292	0.833	0.578	0.633
EXP-CL	0.610	(0.117)	0.377	0.843	0.602	0.697
SIG-CL	0.563	(0.109)	0.273	0.852	0.580	0.644
ALL-CL	0.613	(0.133)	0.387	0.838	0.609	0.699

(a) Performance metrics for MSL-CL, VEC-CL, DUR-CL, EXP-CL, SIG-CL, and ALL-CL classifiers (parentheses indicate standard deviation)



(b) ROC Curves for MSL-CL, VEC-CL, DUR-CL, EXP-CL, SIG-CL, and ALL-CL classifiers (positive: *seen*)

Classifier	v.s. correct		Classifier	v.s. correct	
	seen	*not seen*		*seen*	*not seen*
MSL-CL			EXP-CL		
seen	333	232	*seen*	319	217
not seen	67	168	*not seen*	81	183
VEC-CL			SIG-CL		
seen	336	235	*seen*	316	243
not seen	64	165	*not seen*	84	157
DUR-CL			ALL-CL		
seen	305	239	*seen*	317	212
not seen	95	161	*not seen*	83	188

(c) Confusion matrices for MSL-CL, VEC-CL, DUR-CL, EXP-CL, SIG-CL, and ALL-CL classifiers - columns are the true conditions, rows are the classifier predictions

The worst two classifiers in terms of accuracy is DUR-CL (56.2%), the latter four metrics used in the preliminary data analysis, and SIG-CL (56.3%), our hypothesized significance classifier.

All six classifiers have a heavy bias towards positive predictions, and thus have a high amount of false positives (and corresponding poor sensitivity) and low amount of false negatives (and corresponding good specificity).

4.3.2.2 Participant and Category Breakdown

The performance of the classifiers across each participant shows two interesting results: the lowest accuracy (46.7%) is fold 14 for three (MSL-CL, VEC-CL and EXP-CL) out of the six classifiers, and fold 13 for three (VEC-CL, EXP-CL and ALL-CL) out of the six classifiers. This shows the influence of the mean saccade length and mean saccade duration features in the combined classifiers for the low and high accuracy cases respectively; fold 13 and fold 14, which correspond to participant 13 and participant 14, may have displayed especially unusual and especially predictable movement behaviours respectively.

Classifier	Participant		Category	
MSL-CL	0.604	(0.066)	0.604	(0.077)
VEC-CL	0.604	(0.064)	0.604	(0.078)
DUR-CL	0.562	(0.045)	0.562	(0.073)
EXP-CL	0.610	(0.074)	0.610	(0.064)
SIG-CL	0.562	(0.050)	0.562	(0.100)
ALL-CL	0.613	(0.074)	0.612	(0.070)

Table 4.1: Category-divided and participant-divided accuracy means and standard deviation (in parentheses) for the logisitic regression classifiers

Unsuprisingly, the standard deviation across the three category is greater than the deviation between participants. For all six metrics, category 1 resulted in the worst accuracy, and category 3 the greatest accuracy. Category 3, as the largest training and test set, is expected to have the most robust and most accurate results without taking into account the classification method. Table 4.1 summarises the average accuracy and standard deviation for participant and category taxonomies.

4.3.2.3 Thresholding

For logistic regression, we used the decision rule:

$$Decision = \begin{cases} seen & if\ p > 0.5 \\ unseen & if\ p <= 0.5 \end{cases}$$

However, given our cross-validation, the highest accuracy may not have been at a 50% decision boundary. Figure 4.6 shows the accuracy of the six models given different decision boundaries. All six classifiers peak earlier than 50%, and improve their accuracy 2-4%.

Another form of thresholding is to only accept predictions with a probability greater than a certain threshold. This would cut out the middle portion of the logistic curve, where the probability of success is much lower, as the confidence of the model is lower. Table 4.7a, table 4.7b, table 4.7c, table 4.7d, table 4.7e and table 4.7f show the change of aggregate accuracy over different threshold values.

If we set a standard of good coverage; namely, the classifier predicts over half the cases, then we we can gain 7-8% for all the classifiers except DUR-CL for predictions with a probability of 65%

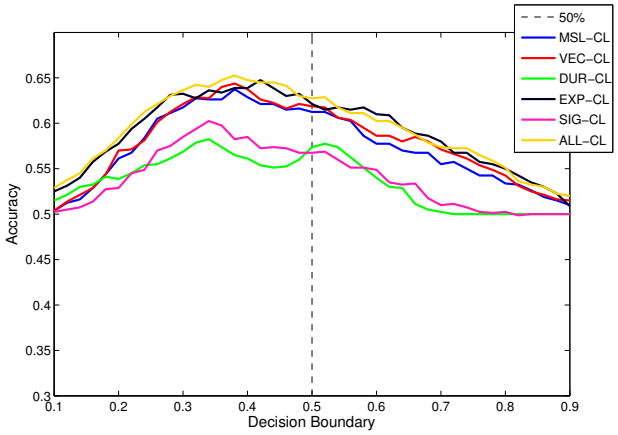


Figure 4.6: The accuracy of the six classifiers at different decision boundary thresholds

Table 4.2: The optimal boundary values for aggregate accuracy values for each discriminative classifier

Classifier	Peak Boundary	Accuracy
MSL-CL	0.38	0.638
VEC-CL	0.38	0.644
DUR-CL	0.34	0.583
EXP-CL	0.42	0.648
SIG-CL	0.38	0.603
ALL-CL	0.38	0.653

confidence or greater. DUR-CL does not cover half the cases until we lower the threshold below 65%, which may explain its poor average accuracy.

Figure 4.7: Columns are number of (P)redictions, number of (C)orrect predictions and total (Acc)urracy

	Category 1		Category 2		Category 3		Total		
Threshold	P	C	P	C	P	C	P	C	Acc
0.85	1	1	8	7	34	32	43	40	0.930
0.75	33	21	56	44	105	87	194	152	0.784
0.65	114	69	133	83	211	157	458	309	0.675
0.55	184	96	185	107	338	239	707	442	0.625

(a) Prediction accuracy of MSL-CL at different thresholds

	Category 1		Category 2		Category 3		Total		
Threshold	P	C	P	C	P	C	P	C	Acc
0.85	1	1	8	7	39	35	48	43	0.896
0.75	38	27	54	42	123	99	215	168	0.781
0.65	116	70	132	82	232	176	480	328	0.683
0.55	185	97	185	107	344	241	714	445	0.623

(b) Prediction accuracy of VEC-CL at different thresholds

	Category 1		Category 2		Category 3		Total		
Threshold	P	C	P	C	P	C	P	C	Acc
0.85	8	8	13	12	12	9	33	29	0.879
0.75	34	22	42	28	38	27	114	77	0.675
0.65	112	61	111	69	106	71	329	201	0.611
0.55	186	94	187	93	296	196	669	383	0.572

(c) Prediction accuracy of DUR-CL at different thresholds

	Category 1		Category 2		Category 3		Total		
Threshold	P	C	P	C	P	C	P	C	Acc
0.85	13	10	19	17	66	56	98	83	0.847
0.75	49	34	61	46	146	119	256	199	0.777
0.65	112	66	128	80	243	188	483	334	0.692
0.55	178	97	179	105	353	252	710	454	0.639

(d) Prediction accuracy of EXP-CL at different thresholds

	Category 1		Category 2		Category 3		Total		
Threshold	P	C	P	C	P	C	P	C	Acc
0.85	0	0	8	7	1	1	9	8	0.889
0.75	28	16	59	47	37	26	124	89	0.718
0.65	119	65	125	77	165	117	409	259	0.633
0.55	191	95	179	93	315	204	685	392	0.572

(e) Prediction accuracy of SIG-CL at different thresholds

	Category 1		Category 2		Category 3		Total		
Threshold	P	C	P	C	P	C	P	C	Acc
0.85	14	10	24	22	68	56	106	88	0.830
0.75	53	36	59	46	162	133	274	215	0.785
0.65	111	65	125	75	266	202	502	342	0.681
0.55	176	95	177	105	355	257	708	457	0.645

(f) Prediction accuracy of ALL-CL at different thresholds

4.4 Spatial Classification

A non-temporal parametric probability model can be found by considering only the spatial dimension of fixations across the image for first and second viewing experiences. In this section we compare the classification based on spatial distribution using two clustering strategies: the DeCarlo-Santella (DCS) algorithm (see 4.2.2.2) and Mixture of Gaussians (MoG) (see 4.2.2.1).

4.4.1 Approach

We built two classifers to compare the two strategies for classification:

- DCS-CL
- MOG-CL

Each classifer was trained using two sets of fixations, one set for each group (first and second viewing).

4.4.1.1 DeCarlo-Santella Clustering

The DCS clustering algorithm is a robust and deterministic algorithm for creating centroids from fixations using the mean-shift procedure (see 2.1.3.3). These centroids represent AOIs, and for the use of the classifier, are represented by the mean μ and the diagonal covariance matrix Σ_c (which was lower-bounded by $\sqrt{50}$ pixels in each dimension) of the fixations.

Each set of fixations resulted in a corresponding set of AOIs. To classify a test scanpath, each fixation f of the scanpath S was compared in turn to each AOI cluster c of the group g. The *gaussian distance* is calculated using the bivariate normal pdf with parameters μ_c and Σ_c; in other words, the likelihood $L(c, f)$ of the fixation belonging to cluster [17].

$$L(c, f) = N(f|\mu_c, \Sigma_C)$$

By taking the maximum log-likelihood for each cluster, and summing the log-likelihoods for each fixation, a total log-likelihood was calculated for the scanpath.

$$\begin{aligned} L(S, g) &= \log p(S|G = g) \\ &= \sum_{f \subseteq S} \arg\max_{c \subseteq g} \log L(c, f) \end{aligned}$$

With the assumption that the prior probability of the group is uniform, classification was determined by the group, or set of clusters, which maximized the log-likelihood.

$$\begin{aligned} \arg_g \max p(G = g|S) &= \arg\max_g p(S|G = g)p(G = g) \\ &= \arg\max_g p(S|G = g)p(G = g) \\ &= \arg\max_g \log p(S|G = g) + Constant \\ &= \arg\max_g L(S, g) \end{aligned}$$

4.4.1.2 Mixture of Gaussian Clustering

For the MoG classifier, we constructed a MoG for each group using only the (x, y) coordinates of its fixations. The parameters of the MoG were estimated using the EM algorithm (see 2.1.1.4). The probability $p(S|M = m)$ of a test scanpath S composed of N fixations $x_1, ..., x_N$ belonging to a MoG m is:

$$\begin{aligned} p(S|M = m) &= \prod_{i=1}^{N} p(x_i|M = m) \\ &= \prod_{i=1} \sum_{k=1}^{m} p(k|x_i) \\ &\propto \prod_{i=1} \sum_{k=1}^{m} p(x_i|k)p(k) \end{aligned}$$

where K_m is the number of clusters in MoG m, and μ_k, π_k and Σ_k are the parameters of cluster k found through EM estimation. Classification was performed by finding the posterior probability of the test scanpath for each MoG and choosing the larger probability. For convenience, and to avoid arithmetic underflow, the log-likelihood was used.

It is important to note that in comparison to the DCS classifer, the MoG classifer uses a soft-assignment for cluster each fixation in a scanpath. Also, as the number of clusters K has to be defined *a priori*, the classifier selected the value K which minimized the BIC value (as described in section 2.1.3.2). The maximum value for K was capped at 10, which is more than the number of human-selected AOIs in each source image.

4.4.2 Results

Performance for both classifiers aggregated over the whole cross-validation is represented in figure 4.8, with the performance metrics shown in table 4.8a, the confusion matrices in table 4.8c, and the ROC curves in figure 4.8b.

The results for both spatial classifiers is noticeably poor. With a prediction accuracy analogous to a random classifer there are very few positive conclusions to be drawn from these classifiers. The ROC displays how similar the quality of the model is to that of the random classifier (the dashed reference line in figure 4.8b).

DCS Classifier (DCS-CL) DCS classifier has the lower accuracy (0.488) and the lower AUC score (0.483) of the two classifiers. The ROC curve shows how similar the performance of DCS-CL is to the random classifer. DCS-CL makes a much larger number of *seen* predictions (469) to *seen* predictions (331). The large standard deviation shows the poor performance of the DCS-CL on certain folds, with a minimum accuracy of 10%.

As table 4.3 shows, DCS-CL does not improve beyond 50.7% in relation to the difference between the log-likelihoods. The slight positive correlation only manages to make the classifier as respectable as random.

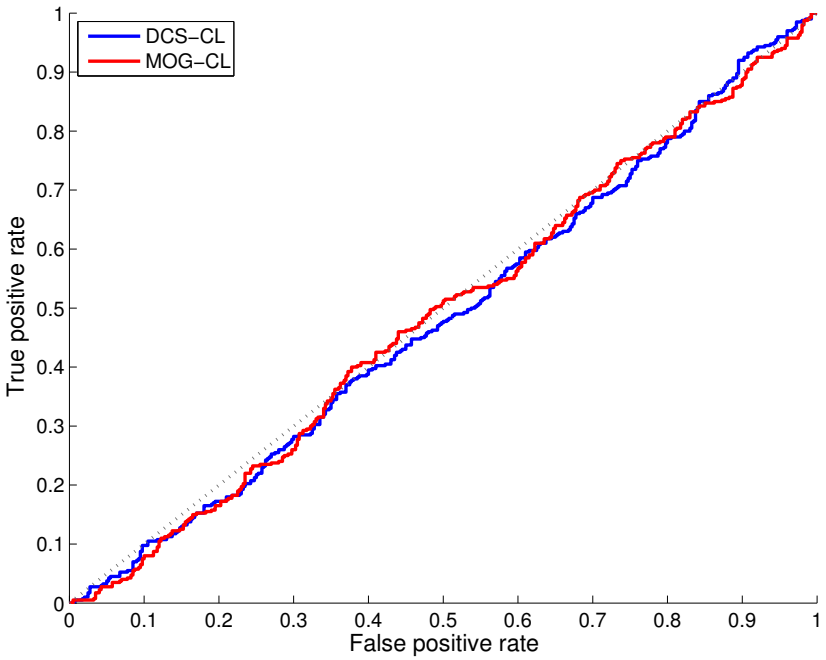
MOG Classifier (MOG-CL) While MOG-CL fared better than DCS-CL in terms of accuracy (0.502) and AUC score (0.487), as neither classifers differentiated themselves from the idealized random classifier (0.50), it's not possible to draw any conclusions from this comparison.

However, in comparison to DCS-CL, there is a negative correlation between the difference between log-likelihoods and the prediction accuracy. In fact, table 4.4 shows, the greater the difference

Figure 4.8: Summary of the results of DCS-CL and MOG-CL classifiers

Classifier	Accuracy		Sensitivity	Specificity	Precision	AUC
DCS-CL	0.488	(0.138)	0.402	0.575	0.500	0.483
MOG-CL	0.502	(0.125)	0.572	0.433	0.449	0.487

(a) Performance metrics for DCS-CL and MOG-CL classifier



(b) ROC Curve for DCS-CL and MOG-CL classifiers (positive: *seen*)

Classifier	v.s. correct		Classifier	v.s. correct	
	seen	*not seen*		*seen*	*not seen*
DCS-CL			MOG-CL		
seen	231	238	*seen*	200	196
notseen	169	162	*notseen*	200	204

(c) Confusion matrices for DCS-CL and MOG-CL classifier - columns are the true conditions, rows are the classifier predictions

	Category 1		Category 2		Category 3		Total		
Threshold	P	C	P	C	P	C	P	C	Acc
7	41	21	36	19	63	31	140	71	0.507
5	70	37	58	29	119	56	247	122	0.494
2	144	72	134	60	276	136	554	268	0.484
1	167	87	170	77	326	158	663	322	0.486
0.1	196	103	196	85	394	198	786	386	0.491

Table 4.3: Prediction accuracy of DCS-CL at different thresholds. Columns are number of (P)redictions, number of (C)orrect predictions and total (Acc)urracy

	Category 1		Category 2		Category 3		Total		
Threshold	P	C	P	C	P	C	P	C	Acc
5	70	30	58	29	119	53	247	112	0.453
2	144	67	134	72	276	137	554	276	0.498
1	167	75	170	93	326	167	663	335	0.505
0.5	184	83	186	100	372	187	742	370	0.499
0.05	198	90	199	106	397	202	794	398	0.501

Table 4.4: Prediction accuracy of MOG-CL at different thresholds. Columns are number of (P)redictions, number of (C)orrect predictions and total (Acc)urracy

between the log-likelihoods, the less accurate the predictions are. This indicates that those low scores are effectively outliers, and have no significance on whether it is a first or second viewing.

4.4.2.1 Participant and Category Breakdown

Classifier	Participant		Category	
DCS-CL	0.488	(0.095)	0.488	(0.044)
MOG-CL	0.502	(0.086)	0.502	(0.038)

Table 4.5: Category-divided and participant-divided accuracy means and standard deviation (in parentheses) for the spatial classifers

The accuracy, as shown in table 4.5 is consistent across categories and participants with a standard deviation of 4.4% and 3.8% for DCS-CL and MOG-CL respectively.

4.5 String-Edit Classification

Scanpaths consist of a sequential property that is ignored in spatial distance metrics such as Euclidean distance. A likely cause of failure for our spatial classifiers may have been this reduction to a set of independent fixations, rather than considering a scanpath as a sequence of fixations. A model that can maintain the relative ordering of fixations preserves potentially crucial information about the similarity between two scanpaths. This goal has lead research into the string representation of scanpaths and the use of string editing distance.

4.5.1 Approach

One can identify the relationship between strings and scanpaths as an analogy; a string is a series of characters like a scanpath is a series of fixations. Therefore, finding the string representation of a scanpath consists of converting fixations, which have a real-valued location and duration, into a discrete set of characters. The transformation we will use is to represent fixations by the area-of-interest (AOI) they are located in. Like the characters of a string, AOIs are a finite set of discrete elements; therefore a one-to-one mapping of character to AOI is trivial.

As discussed in section 4.2.2, what is not trivial is the definition of AOIs, or in its string form, our *alphabet*. As we intend to use a hard-assignment for AOIs, and considering its equivalent performance to MoGs empirically in the spatial classifiers, we will use the DCS method for AOI generation. However, the set of fixations we use to generate AOIs is a control variable we test.

We use the Levenshtein distance as the metric for our string representation. Levenshtein distance measures the similarity of strings as a measure of the *cost* to transform one string into another using one of three operations: deletion, addition and substitution. A variation of the original metric, possibly more representative of real-world systems, is to apply a weighted cost to substitution, such that the substitution of two characters further apart (using another distance metric) is more expensive than one close by. We explore the effectiveness of using a substitution cost.

In total, we modelled four classifiers using the string-editing metric:

- weighted substitution cost and imagewise alphabet (WS-IA)
- uniform cost and imagewise alphabet (UC-IA)
- weighted substitution cost and pairwise alphabet (WS-PA)
- uniform cost and pairwise alphabet (UC-PA)

As stated, all four classifiers used the DCS clustering algorithm to automatically generate AOIs from a set of fixations. For UC-PA and WS-PA, the set was the union of the fixations in the two scanpaths compared. For UC-IA and WS-IA, the set was the union of all the fixations in both groups from the training set. Therefore, while UC-IA and WS-IA generated an alphabet once per model, UC-PA and WS-PA had pair specific AOIs and generated them for each comparison. It may be worth noting that in an online system, AOI generation is a variable cost for the pairwise alphabets and a fixed cost for the imagewise alphabets.

The reasons to use an imagewise alphabet instead of a groupwise alphabet, where each group has its own set of AOIs, is twofold: one, AOIs can be considered properties of the underlying image; two, analysis of the spatial classifiers indicated that the spatial distribution between both groups was insignificant. Applications where neither of these points hold may find value in using a groupwise alphabet.

To assign a new scanpath to the existing AOIs, a K-Nearest-Neighbour classifier was used, with $K = 5$ and a distance-weighted cost (see section 2.1.2.2).

For WS-PA and WS-IA the cost of substitution was weighted using the Euclidean distance between the centers of the AOIs. The cost of addition and deletion remained fixed at 1, but the cost of substitution equalled

$$\alpha \sum_{i=1}^{2} \sqrt{(u_i - v_i)^2}$$

where u and v represent the means of the two AOIs, and α is 0.0025. The value for α should be relative to the units and size of the image, and was found empirically based on performance. This is discussed below in section 4.5.2.4. Substitutions were stored in a n-by-n cost matrix, where n is the length of the alphabet.

4.5.2 Results

The results of the four classifiers against the cross validation set is shown in figure 4.9. The ROC curves and confusion matrix are also shown in figure 4.9b and table 4.9c respectively.

There is clearly a marked improvement over the spatial classifiers. Figure 4.9b shows that the behaviour of the classifier departs the diagonal reference line representing the behaviour of an ideal random classifier.

4.5.2.1 Pairwise v.s. Image wise

The performance between the two definitions of alphabets did not show a significant difference. The accuracy of the imagewise alphabets (UC-IA = 66.6%, WS-IA = 68.7%) is similar to the accuracy of the pairwise alphabets (UC-PA = 65.5%, WS-PA = 68.7%). However, the imagewise alphabets both consistently have higher specificity scores and lower sensitivity scores, a cause of the bias for the imagewise classifiers towards positive *seen* predictions.

4.5.2.2 Weighted v.s. Unweighted

The performance of the weighted classifiers, who use a Euclidean-distance weighted cost for substitution, is greater than that of the uniform costs for both definition of the alphabets, and achieve similar accuracies to each other. WS-IA (68.7%) is higher than UC-IA (66.6%) and WS-Pa (68.7%) is higher than UC-PA (65.5%). They also have lower standard deviations, indicating more consistent results.

4.5.2.3 Participant and Category Breakdown

Classifier	Participant		Category	
UC-IA	0.666	(0.096)	0.666	(0.011)
WS-IA	0.687	(0.090)	0.687	(0.036)
UC-PA	0.655	(0.100)	0.655	(0.010)
WS-PA	0.687	(0.082)	0.687	(0.032)

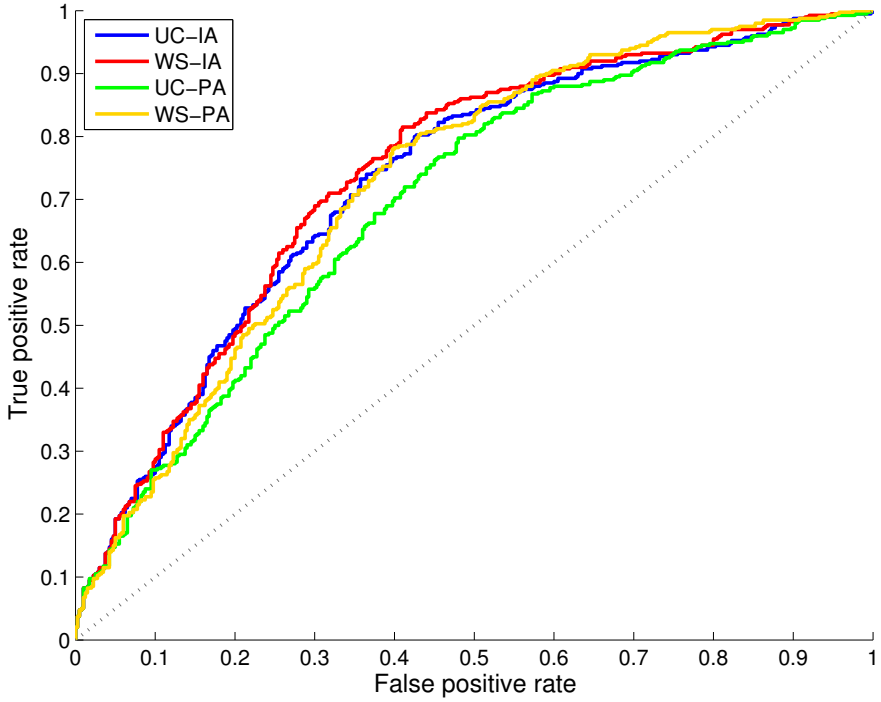
Table 4.6: Category-divided and participant-divided accuracy means and standard deviation (in parentheses) for the string classifers

Accuracy, as shown in table 4.6, is consistent across participants and categories. However, the deviation between categories is much less than across participants, which is unsurprising, as our classifiers are using a similarity metric. Participants whose viewing patterns differ greatly than the group will suffer a much lower accuracy; those with common viewing patterns (similar to the group) will score a higher accuracy.

Figure 4.9: Summary of results for UC-IA, WS-IA, UC-PA and WS-PA classifiers

Classifier	Accuracy		Sensitivity	Specificity	Precision	AUC
UC-IA	0.666	(0.149)	0.642	0.690	0.676	0.729
WS-IA	0.687	(0.124)	0.633	0.740	0.708	0.741
UC-PA	0.655	(0.135)	0.773	0.537	0.739	0.696
WS-PA	0.687	(0.122)	0.742	0.632	0.740	0.725

(a) Performance metrics for UC-IA, WS-IA, UC-PA and WS-PA classifiers



(b) ROC curve for UC-IA, WS-IA, UC-PA and WS-PA classifiers (positive: *seen*)

Classifier	v.s. correct		Classifier	v.s. correct	
	seen	*not seen*		*seen*	*not seen*
UC-IA			UC-PA		
seen	272	137	*seen*	208	82
not seen	128	263	*not seen*	192	318
WS-IA			WS-PA		
seen	284	131	*seen*	241	89
not seen	116	269	*not seen*	159	311

(c) Confusion matrices for UC-IA, WS-IA, UC-PA and WS-PA classifiers - columns are the true conditions, rows are the classifier predictions

4.5.2.4 Search for Alpha

Our results table in figure 4.9 show the performance of two weighted classifiers WS-IA and WS-PA. However, to account for the control variable α, we cross-validated four additional classifiers, using two other α values: 0.001 and 0.005. We chose 0.0025 as it showed the best results, as can be seen in table 4.7.

α	Classifier	Accuracy		Sensitivity	Specificity	Precision
0.001	WS-IA	0.598	(0.117)	0.333	0.862	0.584
0.001	WS-PA	0.601	(0.121)	0.420	0.782	0.590
0.0025	WS-IA	0.687	(0.124)	0.633	0.740	0.708
0.0025	WS-PA	0.687	(0.122)	0.742	0.632	0.740
0.005	WS-IA	0.676	(0.149)	0.723	0.628	0.716
0.005	WS-PA	0.686	(0.130)	0.833	0.538	0.790

Table 4.7: Performance metrics for weighted string classifiers with different α values

4.6 Markov Process Classification

From our promising results with the string editing classifier in section 4.5.2 and poor results in the spatial classifier in section 4.4.2, we have seen the important of sequence over spatial distribution. With the string-editing approach, we saw the benefit of including a substitution cost to give non-uniformity to the substitution of AOIs in a sequence. We will follow a similar line of reasoning by considering the cost, or probability, of transitions by using the Markov property.

4.6.1 Approach

We built and compared four classifiers from the assumption eye movements obey the Markov property:

- MC1-CL: using the posterior probabilities of first-order Markov chains
- MC2-CL: using the posterior probabilites of second-order Markov chains
- HMMM-CL: using the posterior probabilities of a first order Hidden Markov (Mixture) Model, with observations as fixations of (x, y) dimensions and AOIs as the latent state space
- HMMD-CL: using the posterior probabilities of a first order Hidden Markov (Mixture) Model, with observations as fixations of (x, y, t) dimensions and AOIs as the latent state space

Transition matrices were built using the training data across each group *seen* and *unseen* for the Markov chain models (MC1-CL, MC2-CL). Transitions matrices were created empirically with counts, with an adapted version of add-one smoothing to solve the sparse matrix problem [1], where the occurence matrix was pre-populated with the value 0.01. This implies a non-informative uniform distribution for any AOI that was never reached in the training set.

All the HMMs were first-order and trained and fitted using the Baum-Welch algorithm (see section 2.1.1.4). The training data was composed of the fixations and corresponding AOI assignment. A HMM was fitted for both first and second viewings, and classification was the comparison between the log-likelihoods of the test scanpath belong to either HMM.

For MC1-CL and MC2-CL, an aggregated transition matrix was constructed using the training data for each group. The log-likelihood was calculated and compared for classification of each test scanpath.

4.6.2 Results

Performance for all four classifiers aggregated over the whole cross-validation is represented in figure 4.10, with the performance metrics shown in table 4.10a, the confusion matrices in table 4.10d, and the ROC curves in figure 4.10b. The accuracy per category and per participant is shown in table 4.8.

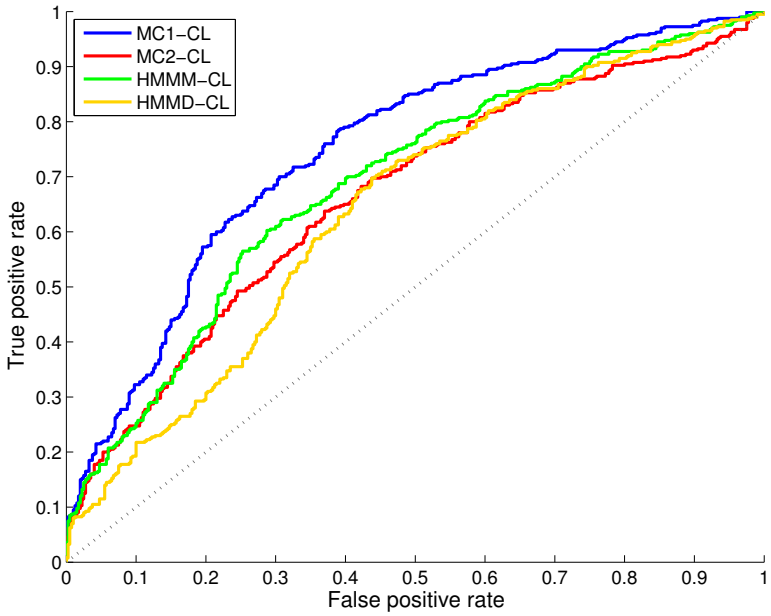
4.6.2.1 Comparison

The performance of all four classifiers is comparable, with the first-order Markov Chain model (MC1-CL) achieving the highest accuracy (67.8%). However, looking closer at the results, there were five test sets (number of tests was 10 in each set) where MC1-CL scored an accuracy of less than 50%, which shows that the model has a clear division in performance. The Hidden Markov Mixture Model (HMMM-CL) performed second best, with an average accuracy of 64.7%. The overall distribution between positive predictions and negative predictions was similar to MC1-CL, with a bias towards positive predictions (P = 427, N = 373) as was the case for MC1-CL (P

Figure 4.10: Summary of results for the MC1-CL, MC2-CL, HMMM-CL and HMMD-CL classifiers

Classifier	Accuracy		Sensitivity	Specificity	Precision	AUC
MC1-CL	0.678	(0.165)	0.623	0.732	0.698	0.749
MC2-CL	0.610	(0.133)	0.557	0.663	0.609	0.665
HMMM-CL	0.647	(0.161)	0.603	0.692	0.661	0.690
HMMD-CL	0.616	(0.149)	0.655	0.577	0.633	0.642

(a) Averaged performance metrics for each model of the four Markov classifiers (parentheses indicate standard deviation)



(b) ROC Curve for MC1-CL, MC2-CL, HMMM-CL and HMMD-CL classifiers (positive: *seen*)

(c)

Classifier	v.s. correct		Classifier	v.s. correct	
	seen	*not seen*		*seen*	*not seen*
MC1-CL			HMMM-CL		
seen	294	141	*seen*	276	151
not seen	106	259	*not seen*	124	249
MC2-CL			HMMD-CL		
seen	265	169	*seen*	235	137
not seen	135	231	*not seen*	165	263

(d) Confusion matrices for the four Markov classifiers - columns are the true conditions, rows are the classifier predictions

= 435, N = 365), and MC1-CL(P = 265, N = 366). Only HMMD-CL predicted more negative predictions over the whole cross-validation (P = 372, N = 428). The second order Markov chain performed the worst, with an average accuracy of 61.0%.

4.6.2.2 Category and Participant Breakdown

Classifier	Participant		Category	
MC1-CL	0.677	(0.109)	0.677	(0.052)
MC2-CL	0.610	(0.096)	0.610	(0.043)
HMMM-CL	0.648	(0.091)	0.647	(0.063)
HMMD-CL	0.616	(0.105)	0.616	(0.062)

Table 4.8: Category-divided and participant-divided means and standard deviation (in parentheses) for the Markov classifiers

Table 4.8 shows that the devation across categories was much less than across participants. The variability of the participants may be explained by individual differences in scanpath patterns, or under the structure of the Markov chain, the difference in transition matrices produced by the training models.

4.6.2.3 Thresholding

We can measure the performance by comparing the accuracy of the classifiers with regards to the difference, or distance, in log-likelihoods. If the model is well-oriented, the greater the difference between the log-likelihoods the greater the accuracy. Any other correlation would detail that our model is wrong (inversely related), or no better than random (no correlation). The ROC captures this behaviour with respect to the probabilities of the positive predictions, but we will study the behaviour against the log-likelihoods.

If we find a threshold value for which the classifer provides both good coverage and good accuracy, we can improve our prediction accuracy without too much cost. It's important to note that the threshold tables used represent aggregate accuracy (ratio of correct predictions over the whole cross validation set, not averaged over the folds).

	Category 1		Category 2		Category 3		Total		
Threshold	P	C	P	C	P	C	P	C	Acc
2	17	17	4	1	73	64	94	82	0.872
1	79	59	55	40	209	168	343	267	0.778
0.5	140	101	122	85	302	231	564	417	0.739
0.05	197	134	194	124	387	285	778	543	0.698

Table 4.9: Prediction accuracy of MC1-CL at different thresholds. Columns are number of (P)redictions, number of (C)orrect predictions and total (Acc)urracy

MC1-CL The first order Markov chain can improve upon its accuracy to 73.9% (originally 69.1% for the aggregated accuracy) while still covering over half the predictions (564) with a threshold value of 0.05. The different threshold values in table 4.9 imply a positive correlation between accuracy and distance.

	Category 1		Category 2		Category 3		Total		
Threshold	P	C	P	C	P	C	P	C	Acc
5	19	8	22	12	34	21	75	41	0.547
2	62	42	68	44	150	108	280	194	0.693
1	118	75	129	75	264	181	511	331	0.648
0.5	157	97	165	94	338	227	660	418	0.633
0.05	195	121	195	112	395	257	785	490	0.624

Table 4.10: Prediction accuracy of MC2-CL at different thresholds. Columns are number of (P)redictions, number of (C)orrect predictions and total (Acc)urracy

MC2-CL For the second order Markov chain to cover a similar number (511) of test cases as the first order Markov chain, the accuracy would only be at 64.8%. The different threshold values in table 4.9 imply a positive correlation between accuracy and distance, except that a threshold too high (5) actually reduces the accuracy. This could indicate the location of outliers, and shows the potential value of thresholding both sides of the difference. If we restrict the predictions to be between 2 and 5, aggregate accuracy will improve to 71.1%.

	Category 1		Category 2		Category 3		Total		
Threshold	P	C	P	C	P	C	P	C	Acc
5	10	9	6	4	30	22	46	35	0.761
2	66	52	36	27	166	127	268	206	0.769
1	130	100	97	66	278	200	505	366	0.725
0.5	175	128	136	83	342	239	653	450	0.689
0.05	199	136	196	113	395	270	790	519	0.657

Table 4.11: Prediction accuracy of HMMM-CL at different thresholds

HMMM-CL For HMMM-CL, performance rivals MC1-CL for most of the thresholds. For a threshold of 1, the accuracy rises to 72.5% and the coverage is 505.

	Category 1		Category 2		Category 3		Total		
Threshold	P	C	P	C	P	C	P	C	Acc
5	16	14	14	9	49	34	79	57	0.722
2	81	60	71	43	208	150	360	253	0.703
1	132	97	134	74	303	208	569	379	0.666
0.5	162	113	162	90	353	234	677	437	0.645
0.05	197	132	199	108	397	255	793	495	0.624

Table 4.12: Prediction accuracy of HMMD-CL at different thresholds. Columns are number of (P)redictions, number of (C)orrect predictions and total (Acc)urracy

HMMD-CL For HMMD-CL to achieve an aggregate accuracy greater than 70% (70.3%), the model is limited to guess only 360 cases out of a total of 800. Otherwise, it also loses accuracy over increased distance, with a 66.6% accuracy for 569 predictions.

Chapter 5

Evaluation

The results from classification show that the primary goal, to classify recognition, was only a mild success. In fact, in comparison to user accuracy, all our classifcation attempts fell just short of the 72.8% average accuracy of the participant.

5.1 Data Collection

Ultimately, the greatest contribution of this investigation may be the creation of a novel dataset targeted towards the relationship between eye movement and memory. We successfully implemented a protocol and conducted twenty trials to collect 1200 scanpaths from twenty different users. We also created a platform for revising and rerunning experiments with a similar setup. We aim to make both available for future use.

The value of the data set is dependent on its intended use. We created the data set to classify recognition and therefore it is catered towards that purpose. However, we aimed to collect and annotate the data set as much as possible in order to make it useful for a variety of applications, as well as a variety of standards. For example, we analyzed all 1200 scanpaths collected; other uses may be more restrictive on the quality and characteristics of the scanpaths. Our purpose was classification so naturally robustness against outliers was a property we sought; noise in the data is useful for testing our models.

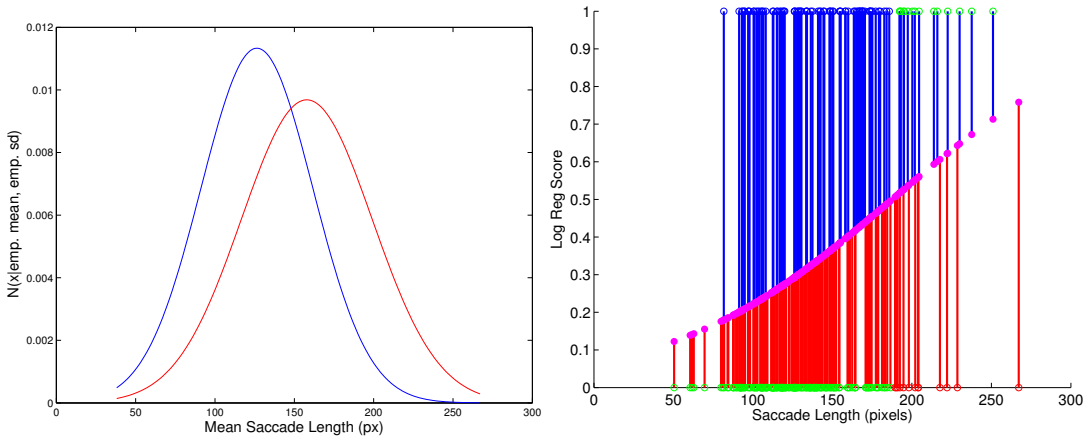
The data set offers recognition on two levels: macroscopic *image* and modular *subimage* recognition. We believe the value in the decomposability of the stimulus will aid future investigations into the nature of eye movement. We only scratched the surface with analysis of *subimage* recognition; however we could potentially recreate or revalidate past experiments into *relational* memory with this same data set.

5.2 Classification

Our principal objective was classification performance, and we built several models for classification of scanpaths.

5.2.1 Discriminative Classifiers

The discriminative classifiers, or classification of the features of the image, did not produce standout results. Most of the feature sets we considered were derived from statistical differences between the groups means of first and second viewing. Difference between means does not



(a) The fitted normal distribution for first (blue) and second (red) image viewings

(b) An example logisitic curve for category 1 (*seen* is 1, *unseen* is 0) - green shows correct prediction

Figure 5.1: The normal distribution and an estimated logistic curve for mean saccade length

translate to 'linearly separable' nor 'easy to discriminate'. We will show its shortcomings using the simplest feature set: mean saccade length. This is a one dimensional feature set using just the mean saccade length of a scanpath to discriminate between *seen* and *unseen* images.

The initial analysis showed that for first and second image viewings, the mean saccade length increased. We showed that this was not mediated by the *task* effect or the *similarity* effect, as the means in both groups decreased. However, if we fit the data to the normal distribution, we will see that there is a large amount of overlap. This is shown in figure 5.1a.

When we compute the conditional using logistic regression, the average accuracy of the MSL-CL classifier is 60.4%. However, if we take a look at the example logistic regression curve in figure 5.1b, we see a similar overlap. The accuracy of example this model with regards to its training data is only 67.8%. As you can see, most points fall on the interval between (0,1) rather than near the asymptotic ends. The large overlap between *seen* and *unseen* cases results in the lower accuracy, and also justifies the difference in performance gained from moving the decision boundary we calculated in section 4.3.2.3.

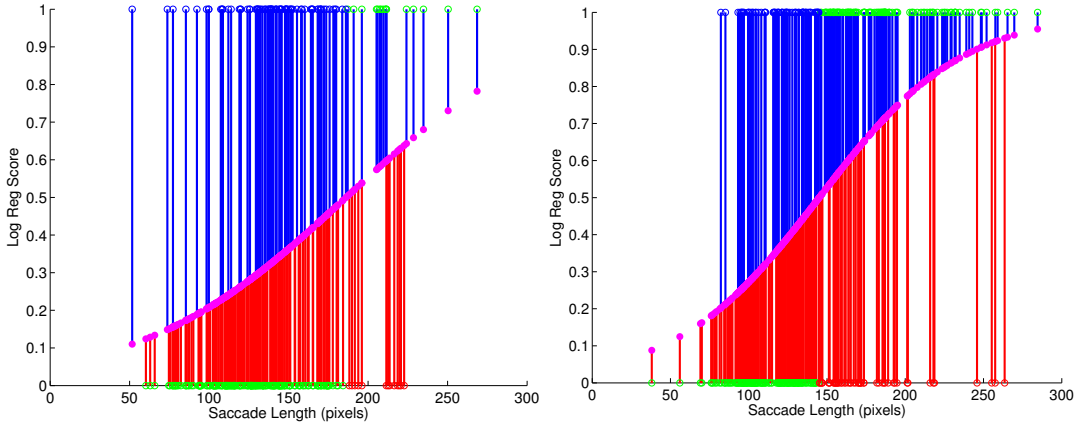
The previous example used a category 1 image; if we examine an example model from category 2 and 3, we will see a similar picture, displayed in figure 5.2a and figure 5.2b respectively.

5.2.2 Spatial Classifiers

Classification of recognition using only the spatial characteristics of scanpaths (in relation to automatically generated Areas-of-Interests) did not succeed. We offer three hypotheses as to why spatial classification did not produce any significant results:

1. The structure of the stimulus, with a small fixed number (5-6) of AOIs, does not provide enough variety for spatial distributions to differ between first and second viewings
2. The clustering algorithms aggregate images for each of the categories, therefore there may be increased noise in comparison to a per-image AOI demarcation
3. Spatial distribution is not a significant indicator of recognition

While we can only point to a similar comparison in the string-edit classifiers (see the pairwise v.s. imagewise comparison in section 4.5.2.1) to counter the second reason, we believe that the root



(a) An example logisitic curve for category 2 (*seen* is 1, *unseen* is 0) - green shows correct prediction

(b) An example logisitic curve for category 3 (*seen* is 1, *unseen* is 0) - green shows correct prediction

Figure 5.2: Two estimated logistic curves for mean saccade length

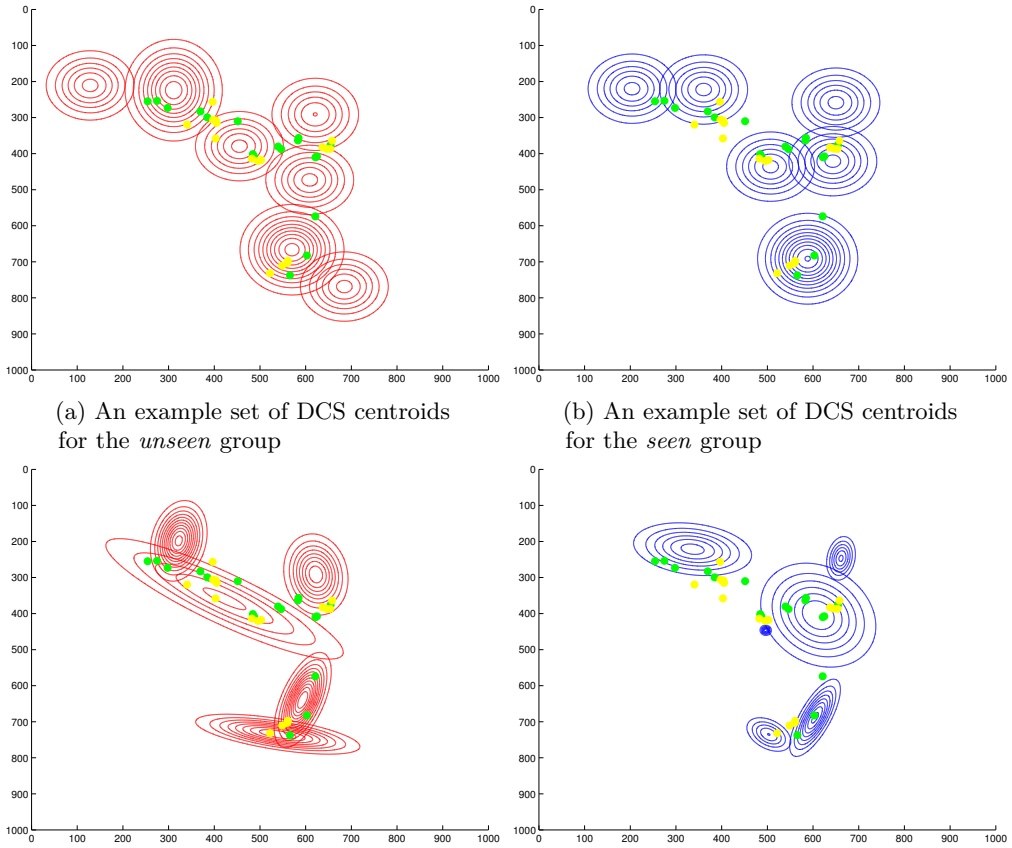
cause of the poor performance is the first reason: with only four or five distinct features and five seconds of viewing time, most participants are able to look at many of the features. Aggregated across all training data, all the features will appear in the AOI demarcation, and the distribution will therefore be similar. The slight success of MOG-CL in comparison to DCS-CL would support this: the only discriminatory feature about the clusters may be the density of fixations in certain AOIs; something better represented by MOGs than the Gaussian centroids defined using the DCS algorithm. Whether the third hypothesis is true is inconclusive given the available evidence.

We can examine the first claim by studying the examples AOIs produced by each classifier. As can be seen in figure 5.3, the difference between the clustering for each groups is marginal, and the difference between the classification of the correct (in green) and incorrect (in yellow) scanpaths seem arbitrary. However, spatial classification does show its value in one regard: the clusters do reveal the underlying structure of the image. This shows the potential value of the two clustering techniques in other applications, including our other use of DCS and MoG clustering: automatic detection of the structure (or AOI) of the images from eye movement.

5.2.3 String-Edit Classifiers

The string-editing representation is a popular approach in the scanpath community, and we found equal success. It performed comparably with all other classification attempts. However, it is clear that the decision to use a weighted or unweighted substitution cost can have noticeable effects on performance; due diligence for any string editing approach is to find the combination which works best.

A potential cause of failure for the string-editing approach with regards to memory is the interchangeabililty of AOIs; namely, each participant may view the specific features of the image differently. A sequence of AOIs $A \rightarrow B \rightarrow C$ for one participant may be synonymous with $C \rightarrow A \rightarrow B$ for another, as their relationship with each AOI is different in terms of semantic or episodic properties. A more abstract definition of the AOIs, rather than a deterministic spatial organization, may be more useful with regards to memory. We did not have time to completely this investigate this idea; however, in regards to our *subimage* analysis in section 3.2.5.5, a representation of AOIs as *seen* and *unseen* features did point to success in both string and Markov representations.



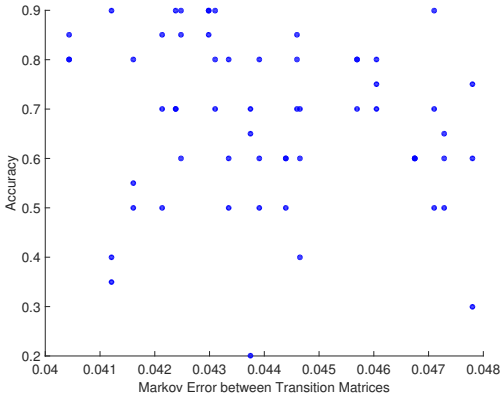
(a) An example set of DCS centroids for the *unseen* group

(b) An example set of DCS centroids for the *seen* group

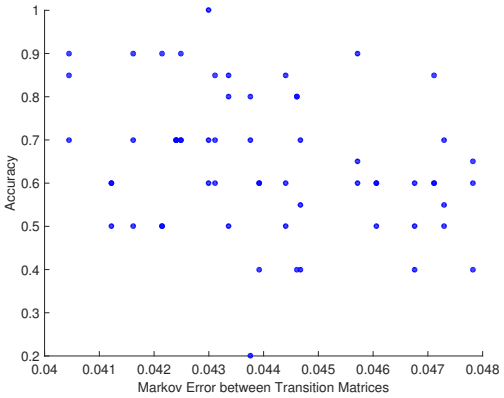
(c) An example MoG for the *unseen* group

(d) An example MoG for the *seen* group

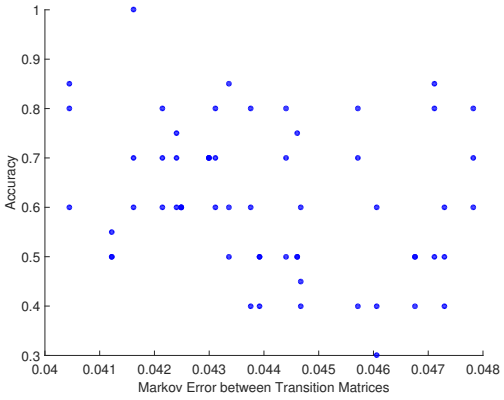
Figure 5.3: Examples of the spatial classifiers clustering from the same training model for category 3 - fixations from a correctly (green) and incorrectly (yellow) predicted scanpath are shown



(a) Markov Error v.s. Accuracy for MC1-CL



(b) Markov Error v.s. Accuracy for HMMM-CL



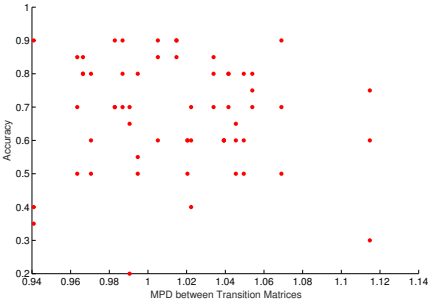
(c) Markov Error v.s. Accuracy for HMMD-CL

Figure 5.4: Scatter plots to show the relationship between accuracy and the Markov Error of the two transition matrices for each training model for (a) MC1-CL (b) HMMM-CL (c) HMMD-CL

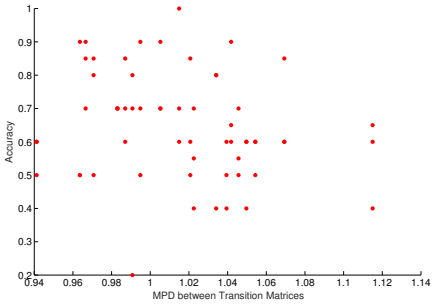
5.2.4 Markov Process Classifiers

As similarly seen from the performance of the string-edit distance classifiers, the inclusion of transition and sequence makes a significant difference to the performance of classification. The four Markov classifiers demonstrate a probabilistic interpretation of sequence, where the weighting of the transitions determine the likelihood of a sequence. For each fold, and for each category, Markov classifiers produce two transition matrices (HMMM-CL and HMMD-CL also produce emission and initial probabilities). We can evaluate our classifiers by examining the transition matrices between the two groups, in terms of the *distance* between them. The distance metrics we can use are *Markov Error* and *Markov-Path-Distance* (defined in section 2.2.5.5).

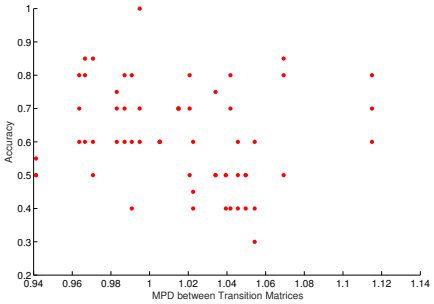
Using the two distance metrics, we can measure the relationship between the distance of the first and second viewing transition matrices and its accuracy. In figure 5.5, we show a scatter plot of the Markov Path Distance (MPD) between the two matrices for MC1-CL, HMMM-CL and HMMD-CL, and in figure 5.4 we show the same except using Markov Error as a distance metric. The table showing the linear correlation between distance and accuracy for the three models is shown in table 5.1. As can be seen both graphically and numerically, the correlation between the distance of the two matrices and its accuracy is not well-correlated. In fact, a greater distance results more often in a worse accuracy. We used a k value of 2 for MPD heuristically as it showed good levels of non-uniformity.



(a) MPD v.s. Accuracy for MC1-CL



(b) MPD v.s. Accuracy for HMMM-CL



(c) MPD v.s. Accuracy for HMMD-CL

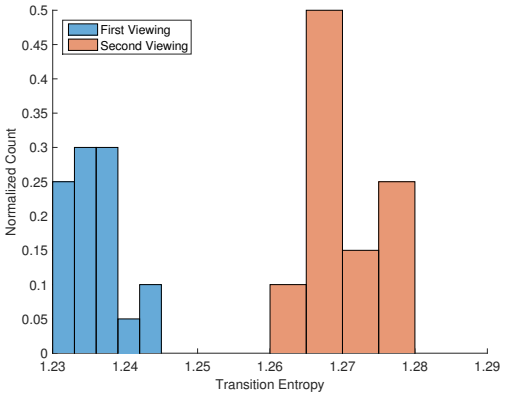
Figure 5.5: Scatter plots to show the relationship between accuracy and the Markov Path Distance of the two transition matrices for each training model for (a) MC1-CL (b) HMMM-CL (c) HMMD-CL

Classifier	Markov Path Distance	Markov Error
MC1-CL	-0.074	-0.183
HMMM-CL	-.210	-0.288
HMMD-CL	-0.153	-0.252

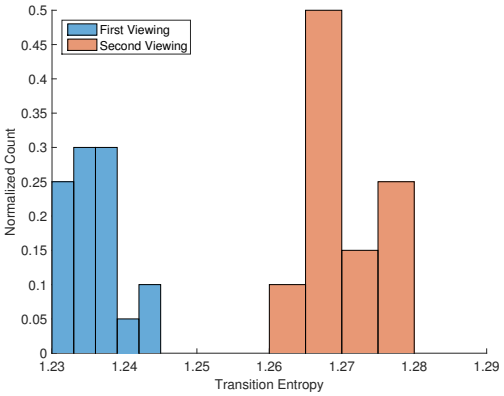
Table 5.1: Correlation between accuracy and MPD and ME between *seen* and *unseen* transition matrix

Another metric of transition matrices we can use is the measure of information entropy, defined in section 2.2.5.5. The entropy of a transition matrix in relation to eye movement can detail the predictability, or inversely the randomness, of scanpaths. The histogram plots aggregating the calculated entropy of the *seen* and *unseen* transition matrices for each of the first order Markov models, MC1-CL, HMMM-CL and HMMD-CL, is shown in figure 5.6. As you can see, the distribution shows that entropy is consistently greater for the *seen* transition matrix over the *unseen* transition matrix for all three models. We believe further investigation of entropy, especially in conjunction with a more abstract representation of AOIs, could result in even greater classification.

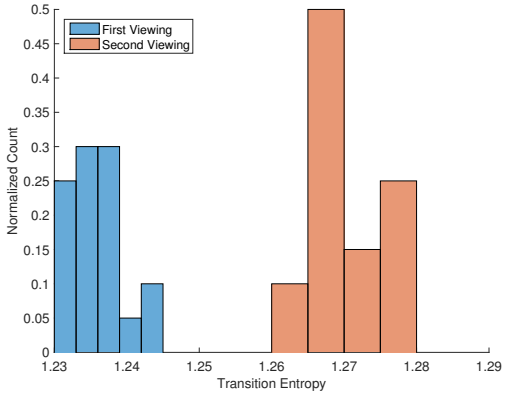
However, the metric of entropy itself does not necessarily translate into good classification performance. We also tested the performance of an entropy-only classifier, using a similar classification technique to the string-edit distance: by comparing the pairwise entropy between the test scanpath and each scanpath of each group, except using the absolute difference of entropies as the measure of distance. The result is an accuracy that does not improve upon the existing classifiers with an accuracy of 60.6%. The ROC curve is in figure 5.7. We include the results here only as a means to demonstrate that while entropy may seem like a discriminative



(a) Entropy of MC1-CL matrices



(b) Entropy of HMMM-CL matrices



(c) Entropy of HMMD-CL matrices

Figure 5.6: Histogram plots of entropy of the transition matrices for (a) MC1-CL (b) HMMM-CL (c) HMMD-CL

feature, more care is needed to translate that into a robust classifier.

5.2.5 Participant Input

As a final evaluation, we can try to compare the performance of the classifiers against participant input to measure how well our classifiers predict the participant's response. However, none of our models will be trained against participant input, but the ground truth. Therefore the models are still separate from participant recognition. To illustrate the performance, we have plotted the ROC curves of the same four sets of classifiers in figure 5.8, except using the participant input as the correct condition.

As you can see, performance is not significantly better than against the ground truth. Without training the models against participant input, we will not claim that the models act as a better or worse classifier of human recognition. However, the similarity in results are interesting. The average accuracy is shown in table 5.2. Accuracy is comparable to the performance when using the ground truth as the correct answer. There are a few noticeable differences however; the highest accuracy (68.1%) is achieved by VEC-CL, the discriminative classifier using mean saccade length and duration as its features.

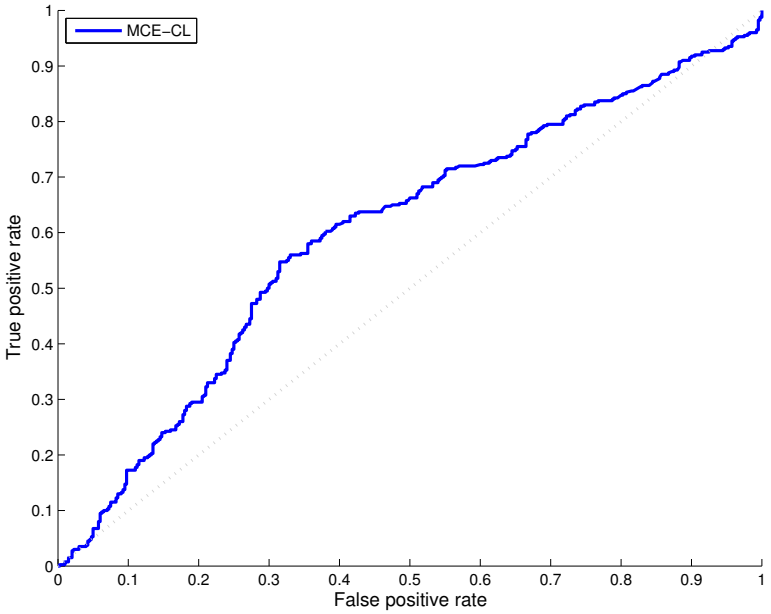
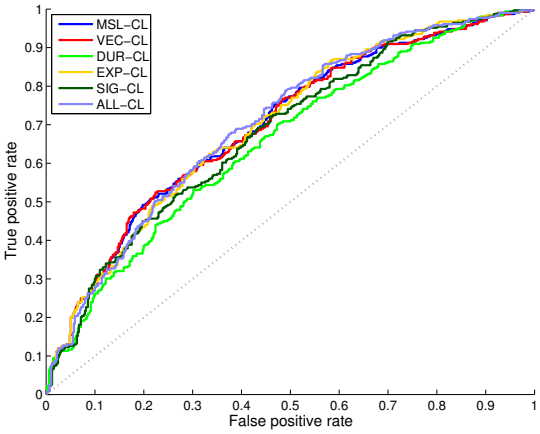


Figure 5.7: ROC curve for an entropy pairwise classifier

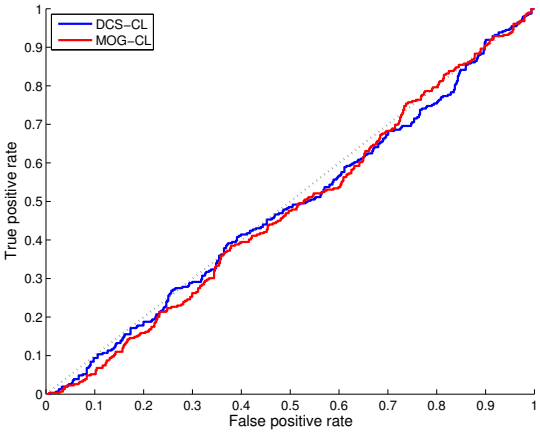
Classifier	Accuracy	
MSL-CL	0.677	(0.117)
VEC-CL	0.681	(0.116)
DUR-CL	0.644	(0.129)
EXP-CL	0.663	(0.116)
SIG-CL	0.662	(0.120)
ALL-CL	0.662	(0.124)
DCS-CL	0.527	(0.142)
MOG-CL	0.469	(0.166)
UC-IA	0.632	(0.160)
WS-IA	0.663	(0.148)
UC-PA	0.587	(0.159)
WS-PA	0.627	(0.158)
MC1-CL	0.662	(0.175)
MC2-CL	0.607	(0.169)
HMMM-CL	0.646	(0.151)
HMMD-CL	0.588	(0.147)

Table 5.2: Accuracy (with standard deviation in parentheses) for all sixteen classifiers using participant input as the ground truth

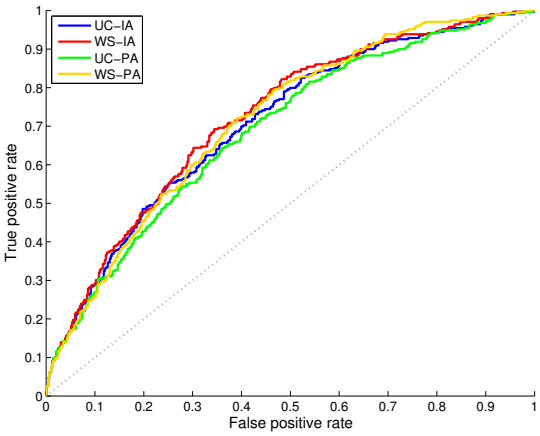
Figure 5.8: ROC curves of classifiers against participant input



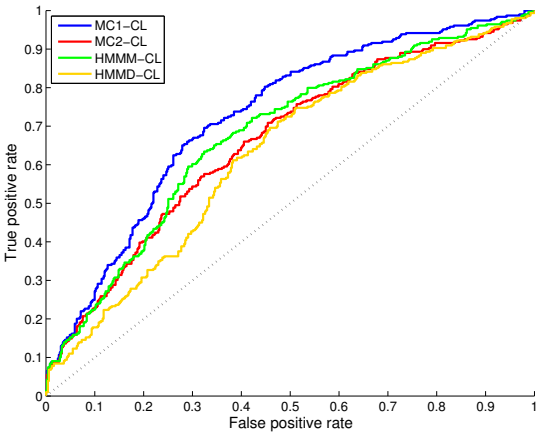
(a) ROC Curves for MSL-CL, VEC-CL, DUR-CL, EXP-CL, SIG-CL, and ALL-CL classifiers against participant input



(b) ROC Curves for DSC-CL and MOG-CL classifiers against participant input



(c) ROC Curves for UC-IA, WS-IA, UC-PA, and WS-PA classifiers against participant input



(d) ROC Curves for MC1-CL, MC2-CL, HMMM-CL, and HMMD-CL classifiers against participant input

Chapter 6

Conclusion

The aim of this project was to build a classifier to identify if a participant has seen an image before based on their eye movement, independent of their own personal awareness of the recognition. In order to achieve this, we built sixteen classifiers covering four representations: a set of discriminating features, a string-editing approach, Markov models and spatial distributions.

We found that spatial distribution did not change between first and second viewing. String editing, which preserves information about the sequence of a scanpath, and Markov models, which use transition matrices to model the probability of the sequence, performed much better.

We found that the metrics used to find statistical significance between first and second viewing did not translate into good classification performance, due to the large overlap in the values.

Our best performing classifiers were the weighted Levenshtein string similarity classifiers (both pairwise and imagewise) in terms of average accuracy with 68.7%, and the first order Markov chain classifer in terms of the area under the ROC curve with an AUC score of 0.741.

However, none of the classifiers performed at a higher average accuracy than explicit recognition, or the average participant accuracy of 72.8%. In this regard, we have not shown that classification of recognition using eye movement can be superior to explicit recognition, which was our original objective. However, we believe we have made progress towards that goal in the following regards:

We produced a data set targeted towards studies investigating the relationship between eye movement and memory. We believe our data set is versatile enough to be used for multiple applications and studies, especially for users who may be interested in an initial data set to explore experimental protocols.

We implemented and evaluated common representations of eye movement as models for classification and compared their effectiveness.

In addition to our attempts at classification, we also provided an analysis of features of eye movement that show statistical differences between first and second image viewing. We also made a brief overture towards *subimage* recognition, or structuring images as a collection of components which may or may not have been seen before.

Finally, we built and used a flexible application for collecting data from eye-movement experiments compatible with commercial eye-tracking hardware.

6.1 Future Work

We believe that this area of research is only at its infancy; the modelling of accurate eye movement has both academic and commercial interests. Understanding how we see things as

well as understanding how we remember unlocks two parts of human physiology and behaviour that could have significant benefits to society.

More specifically, we would like to encourage the following investigations:

Firstly, using more elements of *subimage* recognition to make classification more robust. While our attempts at classification focused on holistic scanpath properties, in our preliminary data analysis we found value in looking at the eye movement in relation to features of the image that have been seen before. Considering that many elements of an image will have either episodic or semantic memory properties, holistic image recognition is difficult without accounting for this decomposability. We made efforts to attempt this, using a HMM where the observations were raw gaze points, and the latent states were the AOIs marked as *seen*, *unseen* or *unknown*; while the investigation seemed promising, we did not have sufficient time to report any findings, and therefore only include it as a potential avenue of investigation.

Secondly, to model a classifier more around each participant, rather than aggregating them altogether. The high variability in the cross-validation results indicate that a per-participant segmentation of models could result in greater accuracy. Intuitively, a per-user segmentation maps to the reality that brains and eyes are specific to individual participants. Therefore, classification which targets finding common memory strategies for clusters of participants, and training a classifier to identify which cluster the individual belongs to, may reveal more about the behaviour of our eyes with regards to memory. This would also follow on from work done by Chuk et. al, who found two recognition strategies among individuals: holistic and analytic [29].

Thirdly, we provided a data set which includes participant input. We modelled our classifiers on the ground truth of whether the participant has seen the image before, but another investigation is the relationship between explicit memory and eye movement. We aim to package the data set and make it publicly available so others may continue down this route without rerunning trials.

Finally, we would like to see the incorporation of memory research into the user-interface design side of the eye-tracking community. Using eye movement as a method of control can fall prey to the Midas Touch problem, where any eye movement can be misinterpreted as a command [47]. A better model of eye movement, including understanding how our eyes react to novel and seen scenes, can help alleviate this problem. Ultimately, a reactive user interface, that can identify the user state of being, can reach the zenith of user experience.

Bibliography

[1] Kevin P. Murphy. Machine learning: a probabilistic perspective. 2012.

[2] Christopher M. Bishop. *Pattern recognition and machine learning*, volume 1. springer, August 2006.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, January 1977.

[4] Pedro Domingos and Michael Pazzani. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29(2-3):103–130, November 1997.

[5] Kenneth P. Burnham and David R. Anderson. Multimodel inference understanding AIC and BIC in model selection. *Sociological methods & research*, 33(2):261–304, 2004.

[6] Anthony Santella and Doug DeCarlo. Robust clustering of eye movement recordings for quantification of visual interest. In *Proceedings of the 2004 symposium on Eye tracking research & applications*, pages 27–34. ACM, 2004.

[7] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, January 1975.

[8] Lee Ann Remington. *Clinical anatomy of the visual system*. Elsevier Health Sciences, 2011.

[9] Andrew Duchowski. Eye tracking methodology: Theory and practice. 2007.

[10] Tobii Eye Tracking - An introduction to eye tracking and Tobii Eye Trackers. Technical report, Tobii Technology AB, January 2010.

[11] Michael F. Land. *The Eye: A Very Short Introduction*. Oxford University Press, May 2014.

[12] Purves. *Types of Eye Movements and Their Functions*.

[13] David E. Irwin. Memory for position and identity across eye movements. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 18(2):307, 1992.

[14] About Tobii Group. http://www.tobii.com/about-tobii/.

[15] Tobii Technology. Developer's Guide: Tobii EyeX SDK for .NET, April 2015.

[16] Dario D. Salvucci and Joseph H. Goldberg. Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the 2000 symposium on Eye tracking research & applications*, pages 71–78. ACM, 2000.

[17] Andrew T. Duchowski, Jason Driver, Sheriff Jolaoso, William Tan, Beverly N. Ramey, and Ami Robbins. Scanpath comparison revisited. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, pages 219–226. ACM, 2010.

[18] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[19] T. Takeuchi and Yoshiko Habuchi. A quantitative method for analyzing scan path data obtained by eye tracker. In *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*, pages 283–286. IEEE, 2007.

[20] Sheree Josephson and Michael E. Holmes. Clutter or Content?: How On-screen Enhancements Affect How TV Viewers Scan and What They Learn. In *Proceedings of the 2006 Symposium on Eye Tracking Research & Applications*, ETRA '06, pages 155–162, New York, NY, USA, 2006. ACM.

[21] F. Galgani, Yiwen Sun, P.L. Lanzi, and J. Leigh. Automatic analysis of eye tracking data for medical diagnosis. In *IEEE Symposium on Computational Intelligence and Data Mining, 2009. CIDM '09*, pages 195–202, March 2009.

[22] LW Stark and SR Ellis. Scanpaths revisited: cognitive models direct active looking. In DF Fisher, editor, *Eye movements: cognition and visual perception*, pages 193–226. Lawrence Erlbaum Associates, 1981.

[23] Selim S. Hacisalihzade, L.W. Stark, and J.S. Allen. Visual perception and sequences of eye movement fixations: a stochastic modeling approach. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3):474–481, May 1992.

[24] Ahmed A. Faisal, Markus Fislage, Marc Pomplun, Robert Rae, and Helge Ritter. Observation of human eye movements to simulate visual exploration of complex scenes. 1998.

[25] Gabriela Ciuperca and Valerie Girardin. On the estimation of the entropy rate of finite Markov chains.

[26] Krzysztof Krejtz, Tomasz Szmidt, Andrew T. Duchowski, and Izabela Krejtz. Entropy-based statistical analysis of eye movement transitions. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 159–166. ACM, 2014.

[27] Rik Pieters, Edward Rosbergen, and Michel Wedel. Visual Attention to Repeated Print Advertising: A Test of Scanpath Theory. *Journal of Marketing Research*, 36(4):424–438, November 1999.

[28] Jaana Simola, Jarkko Salojärvi, and Ilpo Kojo. Using Hidden Markov Model to Uncover Processing States from Eye Movements in Information Search Tasks. *Cogn. Syst. Res.*, 9(4):237–251, October 2008.

[29] T. Chuk, A. B. Chan, and J. H. Hsiao. Understanding eye movements in face recognition using hidden Markov models. *Journal of Vision*, 14(11):8–8, September 2014.

[30] W. B. Scoville and B. Milner. Loss of recent memory after bilateral hippocampal lesions. 1957. *The Journal of Neuropsychiatry and Clinical Neurosciences*, 12(1):103–113, 2000.

[31] Morris Moscovitch. The hippocampus as a" stupid," domain-specific module: Implications for theories of recent and remote memory, and of imagination. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 62(1):62, 2008.

[32] Deborah E. Hannula and Anthony J. Greene. The hippocampus reevaluated in unconscious learning and memory: at a tipping point? *Frontiers in human neuroscience*, 6, 2012.

[33] Deborah E. Hannula and Charan Ranganath. The eyes have it: hippocampal activity predicts expression of memory in eye movements. *Neuron*, 63(5):592–599, 2009.

[34] Deborah E. Hannula, Robert R. Althoff, David E. Warren, Lily Riggs, Neal J. Cohen, and Jennifer D. Ryan. Worth a glance: using eye movements to investigate the cognitive neuroscience of memory. *Frontiers in human neuroscience*, 4, 2010.

[35] Alfred L. Yarbus, Basil Haigh, and Lorrin A. Rigss. *Eye movements and vision*, volume 2. Plenum press New York, 1967.

[36] Geoffrey R. Loftus. Eye fixations and recognition memory for pictures. *Cognitive Psychology*, 3(4):525–551, 1972.

[37] Geoffrey R. Loftus and Norman H. Mackworth. Cognitive determinants of fixation location during picture viewing. *Journal of Experimental Psychology: Human perception and performance*, 4(4):565, 1978.

[38] Jackie Andrade, David Kavanagh, and Alan Baddeley. Eye-movements and visual imagery: A working memory approach to the treatment of post-traumatic stress disorder. *British Journal of Clinical Psychology*, 36(2):209–223, 1997.

[39] Marcel Hout, Peter Muris, Elske Salemink, and Merel Kindt. Autobiographical memories become less vivid and emotional after eye movements. *British Journal of Clinical Psychology*, 40(2):121–130, 2001.

[40] Harold H. Greene and Keith Rayner. Eye movements and familiarity effects in visual search. *Vision Research*, 41(27):3763–3773, 2001.

[41] Deborah E. Hannula, Carol L. Baym, David E. Warren, and Neal J. Cohen. The eyes know eye movements as a veridical index of memory. *Psychological science*, 23(3):278–287, 2012.

[42] Charlotte Schwedes and Dirk Wentura. The revealing glance: Eye gaze behavior to concealed information. *Memory & cognition*, 40(4):642–651, 2012.

[43] Estela Càmara and Lluís Fuentemilla. Accessing forgotten memory traces from long-term memory via visual movements. *Frontiers in human neuroscience*, 8, 2014.

[44] Ilker Yildirim and Robert A. Jacobs. Transfer of Object Category Knowledge across Visual and Haptic Modalities: Experimental and Computational Studies. *Cognition*, 126(2):135–148, 2013.

[45] Richard W. Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950.

[46] Pieter Blignaut. Fixation identification: The optimum threshold for a dispersion algorithm. *Attention, Perception, & Psychophysics*, 71(4):881–895, May 2009.

[47] Robert JK Jacob. What you look at is what you get: eye movement-based interaction techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 11–18. ACM, 1990.

Appendices

Appendix A

Analysis Architecture

In order to facilitate the analysis, the raw data was processed into *models* dubbed *Viewing Experience Models*. A global system to create, manage and analyse these models was built. This section outlines this system.

A.1 Viewing Experience Model

As described in section **TBD**, the data is divided per phase, with one file for the study phase and another for the score phase. This means that eye movement data for a series of images is captured in one file, interweaved with data for the neutral images.

Our analysis aims to classify a viewing experience, where viewing experience is defined to be continuous observation of one image. This viewing experience model (VEModel) is the basic structure used throughout the analysis.

A VEModel includes the neutralising data directly before it, which can be used to check whether the VEModel's eye data is valid. Therefore, a phase is broken down into a consecutive series of VEModels.

The basic structure of a VEModel includes the fields:

- imageid (int): a globally-consistent image identifier
- imagesize ([double,double]): the size of the image as shown to the user
- userid (int): a unique user id
- phase ({1,2}): phase number
- index (int): the index of the VEModel relative to the phase
- order ({1,2}): whether this is the first or second time this image has been shown
- input ({-1,0,1}): what key the user pressed {0 for not seen, 1 for seen, -1 for invalid/no input}
- points ([double,double,double]): all the raw gaze data points, in terms of [x, y, t_{start} time started relative to the first point]
- fixations ([double,double,double,double]): processed fixations, in terms of [x, y, t_{start}, *duration*]

- saccades ([double,double,double,double, double, double]): processed saccades, in terms of [x_{start}, y_{start}, x_{dir}, y_{dir}, t_{start}, $duration$]
- calibration ([double,double,double]): raw gaze data points for the previous neutral image

Fixations were calculated using the I-DT algorithm.

A.2 Global IDs

The system keeps track of globally consistent identifiers and mappings using Matlab default matrices data format (.mat). Image IDs map to their string name which can be used to retrieve their full path, if the user has properly configured the global variable *rootdir*. User IDs map to their string representation, a three letter keyname used to store their data files. Accessors to get files based on name, images based on ID, etc. are available as utility functions.

The system also maintains the mapping of image ID to category ID, which is useful for classification.

Appendix B

Example Files

This section contains example files used for the data collection application described in section 3.1.

B.1 phases.txt

Listing B.1: An example phases.txt for the data collection application

```
instruction      Follow the Dot
calibration
eyetrigger
image    image-1.txt
instruction      Follow the Dot
calibration
keytrigger       Freely look at the image. Press Space to Continue.
image    image-1.txt
instruction      Thank you!
```

Listing B.2: The TestScript used in the data collection trials

```
keytrigger       Hello! \nThank You For Taking Part In This Experiment.\
    nPress Green to Move On.
keytrigger       All Instructions Will Appear Here.\nPressing Green Will
    Move To The Next Instructions.
keytrigger       Please Follow All Instructions
keytrigger       Before We Begin\nWe Need To Test The Calibration
keytrigger       Please Follow The Dot
calibration
keytrigger       Calibration Test Complete.
keytrigger       Instructions For Phase 1:
keytrigger       If You See A Cross On the Screen,\n Please Look At It Until
     It Disappears
keytrigger       Otherwise Please Look Freely At The Image
keytrigger       Each Image Will Be Shown For 5 Seconds.
keytrigger       Press Green When You Are Ready To Begin Phase 1
study    experiment-1.txt
keytrigger       Phase 1 Complete.
keytrigger       Instructions for Phase 2:
keytrigger       Phase 2 Is A Game. \nThe Goal Of The Game Is To Maximize
    The Number of Points.
keytrigger       You Get Points For Correctly Identifying Whether:\nThe
    Image Was Shown In Phase 1\nOr\nIs New To Phase 2
```

```
keytrigger      If You Saw The Image In Phase 1, \nPress Green
keytrigger      If You Did NOT See the Image In Phase 1, \nPress Red
keytrigger      Each Image Will Always Be Shown For 5 Seconds.
keytrigger      You Can Press A Button At Any Time, But You Must Press A
    Button Once Per Image
keytrigger      The Screen Will Remain Black Until You Press A Button
keytrigger      You Get +1 Points For A Correct Answer\n-1 For An Incorrect
     Answer
keytrigger      If You See A Cross On the Screen,\n Please Look At It Until
     It Disappears
keytrigger      Your Score Will Only Be Shown At The End. \nA Sound Will
    Play If You Press A Key.
keytrigger      Press Green When You Are Ready To Begin Phase 2
score   experiment-2.txt
keytrigger      Phase 2 Complete.
keytrigger      We Need To Test The Calibration Again.
keytrigger      Please Follow The Dot
calibration
keytrigger      Calibration Test Complete.
end
keytrigger      All Done - Thank You!
```

B.1.1 TestScript

Listing B.3: An example TestScript for the data collection application

```
next     6
next     0        3000
randomnew         8
next     0        3000
next     7
next     0        3000
randomold         5
```

Appendix C

Fribbles Data Set

Information regarding the fribbles data set is found here.

Table C.1: The features of the source set

Image	F1	F2	F3	F4	Body	Species
1.png	1	1	1	1	1	1
10.png	1	3	3	3	1	1
2.png	2	2	1	2	1	1
3.png	3	3	1	3	1	1
4.png	3	1	1	2	1	1
5.png	2	3	2	2	1	1
9.png	3	1	3	1	1	1
7.png	3	2	2	3	1	1
6.png	1	1	2	2	1	1
8.png	2	2	3	2	1	1
11.png	1	1	1	1	1	2
12.png	2	2	2	1	1	2
13.png	3	3	3	1	1	2
14.png	3	3	3	2	1	2
15.png	2	1	2	2	1	2
17.png	1	2	1	2	1	2
16.png	3	3	3	2	1	2
20.png	1	3	3	1	1	2
19.png	1	2	1	2	1	2
18.png	2	1	2	3	1	2
21.png	1	1	1	1	1	3
22.png	2	2	1	2	1	3
23.png	2	3	1	3	1	3
24.png	1	3	1	2	1	3
25.png	1	1	2	3	1	3
26.png	2	2	2	2	1	3
27.png	2	3	2	1	1	3
28.png	2	1	3	3	1	3
29.png	1	3	3	1	1	3
30.png	2	2	3	2	1	3
31.png	1	1	1	1	1	4
32.png	1	2	2	2	1	4
33.png	1	3	3	1	1	4
34.png	1	3	3	2	1	4
35.png	2	1	1	2	1	4
36.png	2	2	1	1	1	4
37.png	2	3	2	1	1	4
38.png	3	3	1	2	1	4
39.png	3	2	3	2	1	4
40.png	3	1	3	1	1	4

Appendix D

Results

This section lists the full results for all 16 classifiers across all categories and participants.

(a) Full results for DCS-CL classifier

Category	Fold	TP	FP	FN	TN	Accuracy
1						
	1	1	3	4	2	0.300
	2	3	4	2	1	0.400
	3	3	2	2	3	0.600
	4	3	3	2	2	0.500
	5	3	4	2	1	0.400
	6	2	1	3	4	0.600
	7	4	4	1	1	0.500
	8	3	1	2	4	0.700
	9	1	3	4	2	0.300
	10	4	2	1	3	0.700
	11	1	2	4	3	0.400
	12	3	1	2	4	0.700
	13	5	5	0	0	0.500
	14	4	4	1	1	0.500
	15	4	2	1	3	0.700
	16	2	3	3	2	0.400
	17	4	0	1	5	0.900
	18	3	3	2	2	0.500
	19	2	4	3	1	0.300
	20	2	1	3	4	0.600
2						
	1	0	4	5	1	0.100
	2	2	5	3	0	0.200
	3	3	4	2	1	0.400
	4	3	2	2	3	0.600
	5	5	5	0	0	0.500
	6	5	4	0	1	0.600
	7	3	3	2	2	0.500
	8	3	4	2	1	0.400
	9	3	3	2	2	0.500
	10	1	2	4	3	0.400
	11	3	2	2	3	0.600
	12	4	4	1	1	0.500
	13	4	5	1	0	0.400
	14	4	5	1	0	0.400
	15	3	3	2	2	0.500
	16	3	3	2	2	0.500
	17	2	1	3	4	0.600
	18	2	2	3	3	0.500
	19	2	4	3	1	0.300
	20	2	4	3	1	0.300
3						
	1	5	8	5	2	0.350
	2	7	5	3	5	0.600
	3	8	8	2	2	0.500
	4	7	8	3	2	0.450
	5	5	8	5	2	0.350
	6	7	6	3	4	0.550
	7	5	8	5	2	0.350
	8	4	3	6	7	0.550
	9	4	3	6	7	0.550
	10	2	0	8	10	0.600
	11	1	2	9	8	0.450
	12	6	8	4	2	0.400
	13	7	5	3	5	0.600
	14	10	6	0	4	0.700
	15	7	5	3	5	0.600
	16	5	7	5	3	0.400
	17	7	6	3	4	0.550
	18	7	9	3	1	0.400
	19	6	7	4	3	0.450
	20	7	5	3	5	0.600

(b) Full results for MOG-CL classifier

Category	Fold	TP	FP	FN	TN	Accuracy
1						
	1	3	2	2	3	0.600
	2	3	2	2	3	0.600
	3	1	4	4	1	0.200
	4	3	3	2	2	0.500
	5	2	1	3	4	0.600
	6	3	4	2	1	0.400
	7	3	4	2	1	0.400
	8	1	4	4	1	0.200
	9	3	5	2	0	0.300
	10	4	1	1	4	0.800
	11	2	5	3	0	0.200
	12	2	4	3	1	0.300
	13	3	3	2	2	0.500
	14	2	2	3	3	0.500
	15	4	4	1	1	0.500
	16	3	2	2	3	0.600
	17	1	2	4	3	0.400
	18	2	0	3	5	0.700
	19	2	2	3	3	0.500
	20	3	4	2	1	0.400
2						
	1	1	0	4	5	0.600
	2	1	0	4	5	0.600
	3	0	1	5	4	0.400
	4	1	0	4	5	0.600
	5	1	0	4	5	0.600
	6	1	0	4	5	0.600
	7	0	0	5	5	0.500
	8	0	0	5	5	0.500
	9	0	0	5	5	0.500
	10	0	0	5	5	0.500
	11	0	0	5	5	0.500
	12	0	0	5	5	0.500
	13	0	0	5	5	0.500
	14	0	0	5	5	0.500
	15	2	0	3	5	0.700
	16	0	0	5	5	0.500
	17	0	0	5	5	0.500
	18	2	1	3	4	0.600
	19	1	1	4	4	0.500
	20	0	0	5	5	0.500
3						
	1	2	4	8	6	0.400
	2	8	9	2	1	0.450
	3	6	7	4	3	0.450
	4	9	6	1	4	0.650
	5	9	6	1	4	0.650
	6	9	10	1	0	0.450
	7	8	9	2	1	0.450
	8	7	10	3	0	0.350
	9	5	7	5	3	0.400
	10	1	4	9	6	0.350
	11	6	7	4	3	0.450
	12	8	5	2	5	0.650
	13	7	3	3	7	0.700
	14	5	7	5	3	0.400
	15	8	6	2	4	0.600
	16	8	6	2	4	0.600
	17	9	8	1	2	0.550
	18	9	7	1	3	0.600
	19	7	4	3	6	0.650
	20	9	10	1	0	0.450

(a) Full results for MC1-CL classifier

Category	Fold	TP	FP	FN	TN	Accuracy
1						
	1	5	3	0	2	0.700
	2	4	2	1	3	0.700
	3	1	1	4	4	0.500
	4	4	4	1	1	0.500
	5	3	2	2	3	0.600
	6	3	1	2	4	0.700
	7	5	1	0	4	0.900
	8	4	2	1	3	0.700
	9	5	1	0	4	0.900
	10	3	2	2	3	0.600
	11	1	0	4	5	0.600
	12	5	3	0	2	0.700
	13	5	2	0	3	0.800
	14	4	5	1	0	0.400
	15	5	1	0	4	0.900
	16	3	5	2	0	0.300
	17	4	1	1	4	0.800
	18	3	0	2	5	0.800
	19	4	4	1	1	0.500
	20	5	2	0	3	0.800
2						
	1	5	3	0	2	0.700
	2	4	1	1	4	0.800
	3	4	2	1	3	0.700
	4	5	4	0	1	0.600
	5	4	0	1	5	0.900
	6	1	1	4	4	0.500
	7	3	4	2	1	0.400
	8	2	5	3	0	0.200
	9	4	0	1	5	0.900
	10	3	2	2	3	0.600
	11	4	4	1	1	0.500
	12	3	0	2	5	0.800
	13	3	0	2	5	0.800
	14	3	2	2	3	0.600
	15	4	2	1	3	0.700
	16	4	3	1	2	0.600
	17	3	1	2	4	0.700
	18	1	1	4	4	0.500
	19	4	3	1	2	0.600
	20	5	5	0	0	0.500
3						
	1	9	1	1	9	0.900
	2	10	4	0	6	0.800
	3	9	1	1	9	0.900
	4	9	3	1	7	0.800
	5	9	2	1	8	0.850
	6	7	0	3	10	0.850
	7	4	7	6	3	0.350
	8	6	3	4	7	0.650
	9	9	2	1	8	0.850
	10	5	3	5	7	0.600
	11	4	2	6	8	0.600
	12	6	1	4	9	0.750
	13	8	1	2	9	0.850
	14	7	3	3	7	0.700
	15	10	4	0	6	0.800
	16	9	4	1	6	0.750
	17	7	0	3	10	0.850
	18	2	0	8	10	0.600
	19	10	7	0	3	0.650
	20	9	8	1	2	0.550

(b) Full results for MC2-CL classifier

Category	Fold	TP	FP	FN	TN	Accuracy
1						
	1	5	3	0	2	0.700
	2	4	3	1	2	0.600
	3	3	2	2	3	0.600
	4	5	5	0	0	0.500
	5	3	2	2	3	0.600
	6	3	2	2	3	0.600
	7	5	1	0	4	0.900
	8	4	4	1	1	0.500
	9	4	3	1	2	0.600
	10	2	3	3	2	0.400
	11	2	1	3	4	0.600
	12	4	1	1	4	0.800
	13	5	4	0	1	0.600
	14	3	4	2	1	0.400
	15	5	2	0	3	0.800
	16	4	2	1	3	0.700
	17	3	1	2	4	0.700
	18	2	0	3	5	0.700
	19	3	3	2	2	0.500
	20	4	4	1	1	0.500
2						
	1	3	2	2	3	0.600
	2	4	4	1	1	0.500
	3	2	2	3	3	0.500
	4	5	3	0	2	0.700
	5	3	2	2	3	0.600
	6	3	2	2	3	0.600
	7	2	4	3	1	0.300
	8	4	5	1	0	0.400
	9	2	1	3	4	0.600
	10	3	3	2	2	0.500
	11	2	4	3	1	0.300
	12	4	3	1	2	0.600
	13	3	0	2	5	0.800
	14	1	2	4	3	0.400
	15	4	1	1	4	0.800
	16	3	2	2	3	0.600
	17	3	0	2	5	0.800
	18	0	0	5	5	0.500
	19	4	3	1	2	0.600
	20	5	4	0	1	0.600
3						
	1	5	3	5	7	0.600
	2	7	4	3	6	0.650
	3	6	4	4	6	0.600
	4	9	3	1	7	0.800
	5	8	2	2	8	0.800
	6	5	5	5	5	0.500
	7	6	4	4	6	0.600
	8	5	4	5	6	0.550
	9	7	1	3	9	0.800
	10	4	3	6	7	0.550
	11	5	4	5	6	0.550
	12	6	3	4	7	0.650
	13	8	3	2	7	0.750
	14	4	3	6	7	0.550
	15	9	4	1	6	0.750
	16	9	5	1	5	0.700
	17	8	1	2	9	0.850
	18	2	1	8	9	0.550
	19	10	6	0	4	0.700
	20	9	9	1	1	0.500

(a) Full results for HMMM-CL classifier

Category	Fold	TP	FP	FN	TN	Accuracy
1						
	1	4	2	1	3	0.700
	2	5	1	0	4	0.900
	3	3	2	2	3	0.600
	4	5	5	0	0	0.500
	5	4	2	1	3	0.700
	6	4	0	1	5	0.900
	7	3	2	2	3	0.600
	8	4	1	1	4	0.800
	9	5	0	0	5	1.000
	10	3	2	2	3	0.600
	11	1	0	4	5	0.600
	12	5	4	0	1	0.600
	13	5	1	0	4	0.900
	14	4	5	1	0	0.400
	15	4	2	1	3	0.700
	16	3	4	2	1	0.400
	17	5	2	0	3	0.800
	18	2	1	3	4	0.600
	19	4	4	1	1	0.500
	20	5	1	0	4	0.900
2						
	1	4	2	1	3	0.700
	2	4	3	1	2	0.600
	3	4	3	1	2	0.600
	4	4	1	1	4	0.800
	5	3	1	2	4	0.700
	6	2	2	3	3	0.500
	7	5	5	0	0	0.500
	8	1	4	4	1	0.200
	9	2	0	3	5	0.700
	10	2	3	3	2	0.400
	11	2	2	3	3	0.500
	12	3	3	2	2	0.500
	13	3	1	2	4	0.700
	14	3	1	2	4	0.700
	15	3	2	2	3	0.600
	16	4	3	1	2	0.600
	17	2	3	3	2	0.400
	18	2	3	3	2	0.400
	19	4	2	1	3	0.700
	20	4	2	1	3	0.700
3						
	1	6	2	4	8	0.700
	2	7	4	3	6	0.650
	3	7	0	3	10	0.850
	4	10	3	0	7	0.850
	5	9	1	1	9	0.900
	6	4	4	6	6	0.500
	7	3	1	7	9	0.600
	8	8	4	2	6	0.700
	9	6	4	4	6	0.600
	10	5	5	5	5	0.500
	11	7	0	3	10	0.850
	12	6	4	4	6	0.600
	13	8	1	2	9	0.850
	14	2	1	8	9	0.550
	15	9	2	1	8	0.850
	16	7	4	3	6	0.650
	17	10	4	0	6	0.800
	18	7	5	3	5	0.600
	19	9	8	1	2	0.550
	20	7	7	3	3	0.500

(b) Full results for HMMD-CL classifier

Category	Fold	TP	FP	FN	TN	Accuracy
1						
	1	4	2	1	3	0.700
	2	3	0	2	5	0.800
	3	4	1	1	4	0.800
	4	4	3	1	2	0.600
	5	3	2	2	3	0.600
	6	3	0	2	5	0.800
	7	1	1	4	4	0.500
	8	4	1	1	4	0.800
	9	4	2	1	3	0.700
	10	2	2	3	3	0.500
	11	3	0	2	5	0.800
	12	4	3	1	2	0.600
	13	5	2	0	3	0.800
	14	4	5	1	0	0.400
	15	3	2	2	3	0.600
	16	3	1	2	4	0.700
	17	2	2	3	3	0.500
	18	2	2	3	3	0.500
	19	3	3	2	2	0.500
	20	5	0	0	5	1.000
2						
	1	3	2	2	3	0.600
	2	1	2	4	3	0.400
	3	3	3	2	2	0.500
	4	1	1	4	4	0.500
	5	2	1	3	4	0.600
	6	3	1	2	4	0.700
	7	3	3	2	2	0.500
	8	1	2	4	3	0.400
	9	3	1	2	4	0.700
	10	1	2	4	3	0.400
	11	2	2	3	3	0.500
	12	2	4	3	1	0.300
	13	2	1	3	4	0.600
	14	2	1	3	4	0.600
	15	3	1	2	4	0.700
	16	4	1	1	4	0.800
	17	2	2	3	3	0.500
	18	2	2	3	3	0.500
	19	1	2	4	3	0.400
	20	4	2	1	3	0.700
3						
	1	8	3	2	7	0.750
	2	4	0	6	10	0.700
	3	8	1	2	9	0.850
	4	8	1	2	9	0.850
	5	7	5	3	5	0.600
	6	4	2	6	8	0.600
	7	2	1	8	9	0.550
	8	6	4	4	6	0.600
	9	8	4	2	6	0.700
	10	2	2	8	8	0.500
	11	7	3	3	7	0.700
	12	4	6	6	4	0.400
	13	9	2	1	8	0.850
	14	2	3	8	7	0.450
	15	6	0	4	10	0.800
	16	7	5	3	5	0.600
	17	9	4	1	6	0.750
	18	5	7	5	3	0.400
	19	9	7	1	3	0.600
	20	9	7	1	3	0.600

(a) Full results for MSL-CL classifier

Category	Fold	TP	FP	FN	TN	Accuracy
1						
	1	5	5	0	0	0.500
	2	5	2	0	3	0.800
	3	4	2	1	3	0.700
	4	5	5	0	0	0.500
	5	5	5	0	0	0.500
	6	5	5	0	0	0.500
	7	5	4	0	1	0.600
	8	5	5	0	0	0.500
	9	4	4	1	1	0.500
	10	5	3	0	2	0.700
	11	5	2	0	3	0.800
	12	5	4	0	1	0.600
	13	5	4	0	1	0.600
	14	4	5	1	0	0.400
	15	4	5	1	0	0.400
	16	5	5	0	0	0.500
	17	4	4	1	1	0.500
	18	4	4	1	1	0.500
	19	4	5	1	0	0.400
	20	5	5	0	0	0.500
2						
	1	5	5	0	0	0.500
	2	5	5	0	0	0.500
	3	5	5	0	0	0.500
	4	5	4	0	1	0.600
	5	5	4	0	1	0.600
	6	5	3	0	2	0.700
	7	5	5	0	0	0.500
	8	5	5	0	0	0.500
	9	5	5	0	0	0.500
	10	4	3	1	2	0.600
	11	4	3	1	2	0.600
	12	4	5	1	0	0.400
	13	5	3	0	2	0.700
	14	5	5	0	0	0.500
	15	5	5	0	0	0.500
	16	5	4	0	1	0.600
	17	5	2	0	3	0.800
	18	4	1	1	4	0.800
	19	5	5	0	0	0.500
	20	5	5	0	0	0.500
3						
	1	6	2	4	8	0.700
	2	8	2	2	8	0.800
	3	7	2	3	8	0.750
	4	8	1	2	9	0.850
	5	5	3	5	7	0.600
	6	10	3	0	7	0.850
	7	8	7	2	3	0.550
	8	8	2	2	8	0.800
	9	9	3	1	7	0.800
	10	6	3	4	7	0.650
	11	1	1	9	9	0.500
	12	7	3	3	7	0.700
	13	8	2	2	8	0.800
	14	8	8	2	2	0.500
	15	10	6	0	4	0.700
	16	6	5	4	5	0.550
	17	7	1	3	9	0.800
	18	4	2	6	8	0.600
	19	10	6	0	4	0.700
	20	8	5	2	5	0.650

(b) Full results for VEC-CL classifier

Category	Fold	TP	FP	FN	TN	Accuracy
1						
	1	5	5	0	0	0.500
	2	5	3	0	2	0.700
	3	4	3	1	2	0.600
	4	5	5	0	0	0.500
	5	5	5	0	0	0.500
	6	5	5	0	0	0.500
	7	5	4	0	1	0.600
	8	5	5	0	0	0.500
	9	5	4	0	1	0.600
	10	5	3	0	2	0.700
	11	5	2	0	3	0.800
	12	5	4	0	1	0.600
	13	5	4	0	1	0.600
	14	4	5	1	0	0.400
	15	4	5	1	0	0.400
	16	5	5	0	0	0.500
	17	4	4	1	1	0.500
	18	4	4	1	1	0.500
	19	4	5	1	0	0.400
	20	5	5	0	0	0.500
2						
	1	5	5	0	0	0.500
	2	5	5	0	0	0.500
	3	5	5	0	0	0.500
	4	5	4	0	1	0.600
	5	5	4	0	1	0.600
	6	5	3	0	2	0.700
	7	5	5	0	0	0.500
	8	5	5	0	0	0.500
	9	5	4	0	1	0.600
	10	5	4	0	1	0.600
	11	4	3	1	2	0.600
	12	4	5	1	0	0.400
	13	5	3	0	2	0.700
	14	5	5	0	0	0.500
	15	5	5	0	0	0.500
	16	5	4	0	1	0.600
	17	5	2	0	3	0.800
	18	4	1	1	4	0.800
	19	5	5	0	0	0.500
	20	5	5	0	0	0.500
3						
	1	6	2	4	8	0.700
	2	8	3	2	7	0.750
	3	7	2	3	8	0.750
	4	8	1	2	9	0.850
	5	5	2	5	8	0.650
	6	10	4	0	6	0.800
	7	8	7	2	3	0.550
	8	8	2	2	8	0.800
	9	9	3	1	7	0.800
	10	6	3	4	7	0.650
	11	1	1	9	9	0.500
	12	7	3	3	7	0.700
	13	9	2	1	8	0.850
	14	8	8	2	2	0.500
	15	10	6	0	4	0.700
	16	6	5	4	5	0.550
	17	7	1	3	9	0.800
	18	4	2	6	8	0.600
	19	10	6	0	4	0.700
	20	8	5	2	5	0.650

(a) Full results for DUR-CL classifier

Category	Fold	TP	FP	FN	TN	Accuracy
1						
	1	5	5	0	0	0.500
	2	5	5	0	0	0.500
	3	5	5	0	0	0.500
	4	5	5	0	0	0.500
	5	5	5	0	0	0.500
	6	5	5	0	0	0.500
	7	5	5	0	0	0.500
	8	5	5	0	0	0.500
	9	5	5	0	0	0.500
	10	4	3	1	2	0.600
	11	5	5	0	0	0.500
	12	5	5	0	0	0.500
	13	5	5	0	0	0.500
	14	5	5	0	0	0.500
	15	5	5	0	0	0.500
	16	5	5	0	0	0.500
	17	4	5	1	0	0.400
	18	5	5	0	0	0.500
	19	5	5	0	0	0.500
	20	5	5	0	0	0.500
2						
	1	5	5	0	0	0.500
	2	5	2	0	3	0.800
	3	5	5	0	0	0.500
	4	5	4	0	1	0.600
	5	5	5	0	0	0.500
	6	4	5	1	0	0.400
	7	5	5	0	0	0.500
	8	5	4	0	1	0.600
	9	5	3	0	2	0.700
	10	5	3	0	2	0.700
	11	5	5	0	0	0.500
	12	5	5	0	0	0.500
	13	4	4	1	1	0.500
	14	5	5	0	0	0.500
	15	5	5	0	0	0.500
	16	5	5	0	0	0.500
	17	5	5	0	0	0.500
	18	4	3	1	2	0.600
	19	5	5	0	0	0.500
	20	5	5	0	0	0.500
3						
	1	8	3	2	7	0.750
	2	2	0	8	10	0.600
	3	4	0	6	10	0.700
	4	6	1	4	9	0.750
	5	1	3	9	7	0.400
	6	8	5	2	5	0.650
	7	6	3	4	7	0.650
	8	6	3	4	7	0.650
	9	6	1	4	9	0.750
	10	3	2	7	8	0.550
	11	6	1	4	9	0.750
	12	8	4	2	6	0.700
	13	4	0	6	10	0.700
	14	3	1	7	9	0.600
	15	10	6	0	4	0.700
	16	8	6	2	4	0.600
	17	5	3	5	7	0.600
	18	4	4	6	6	0.500
	19	6	4	4	6	0.600
	20	6	3	4	7	0.650

(b) Full results for EXP-CL classifier

Category	Fold	TP	FP	FN	TN	Accuracy
1						
	1	5	5	0	0	0.500
	2	5	3	0	2	0.700
	3	5	5	0	0	0.500
	4	5	4	0	1	0.600
	5	5	4	0	1	0.600
	6	5	5	0	0	0.500
	7	5	4	0	1	0.600
	8	5	5	0	0	0.500
	9	4	4	1	1	0.500
	10	5	3	0	2	0.700
	11	5	3	0	2	0.700
	12	5	3	0	2	0.700
	13	5	3	0	2	0.700
	14	4	5	1	0	0.400
	15	4	5	1	0	0.400
	16	5	5	0	0	0.500
	17	4	4	1	1	0.500
	18	3	2	2	3	0.600
	19	4	5	1	0	0.400
	20	5	5	0	0	0.500
2						
	1	5	5	0	0	0.500
	2	5	3	0	2	0.700
	3	5	5	0	0	0.500
	4	5	3	0	2	0.700
	5	5	4	0	1	0.600
	6	4	3	1	2	0.600
	7	5	5	0	0	0.500
	8	5	4	0	1	0.600
	9	5	3	0	2	0.700
	10	5	3	0	2	0.700
	11	5	3	0	2	0.700
	12	4	4	1	1	0.500
	13	5	3	0	2	0.700
	14	5	5	0	0	0.500
	15	4	5	1	0	0.400
	16	4	4	1	1	0.500
	17	5	2	0	3	0.800
	18	3	1	2	4	0.700
	19	5	5	0	0	0.500
	20	5	5	0	0	0.500
3						
	1	7	1	3	9	0.800
	2	6	2	4	8	0.700
	3	6	1	4	9	0.750
	4	7	1	3	9	0.800
	5	2	2	8	8	0.500
	6	8	4	2	6	0.700
	7	8	6	2	4	0.600
	8	8	3	2	7	0.750
	9	7	3	3	7	0.700
	10	5	2	5	8	0.650
	11	2	1	8	9	0.550
	12	8	3	2	7	0.750
	13	8	1	2	9	0.850
	14	7	7	3	3	0.500
	15	10	7	0	3	0.650
	16	8	6	2	4	0.600
	17	6	1	4	9	0.750
	18	2	1	8	9	0.550
	19	9	3	1	7	0.800
	20	8	5	2	5	0.650

(a) Full results for SIG-CL classifier

Category	Fold	TP	FP	FN	TN	Accuracy
1						
	1	5	5	0	0	0.500
	2	5	5	0	0	0.500
	3	5	5	0	0	0.500
	4	5	5	0	0	0.500
	5	5	5	0	0	0.500
	6	5	5	0	0	0.500
	7	5	5	0	0	0.500
	8	5	5	0	0	0.500
	9	5	5	0	0	0.500
	10	5	5	0	0	0.500
	11	5	5	0	0	0.500
	12	5	5	0	0	0.500
	13	5	5	0	0	0.500
	14	5	5	0	0	0.500
	15	5	5	0	0	0.500
	16	5	5	0	0	0.500
	17	5	5	0	0	0.500
	18	5	5	0	0	0.500
	19	5	5	0	0	0.500
	20	5	5	0	0	0.500
2						
	1	5	5	0	0	0.500
	2	5	5	0	0	0.500
	3	4	4	1	1	0.500
	4	5	5	0	0	0.500
	5	4	5	1	0	0.400
	6	4	4	1	1	0.500
	7	5	5	0	0	0.500
	8	5	4	0	1	0.600
	9	5	5	0	0	0.500
	10	5	5	0	0	0.500
	11	5	5	0	0	0.500
	12	5	5	0	0	0.500
	13	5	4	0	1	0.600
	14	5	3	0	2	0.700
	15	5	5	0	0	0.500
	16	5	5	0	0	0.500
	17	5	5	0	0	0.500
	18	3	5	2	0	0.300
	19	5	4	0	1	0.600
	20	5	5	0	0	0.500
3						
	1	7	3	3	7	0.700
	2	6	2	4	8	0.700
	3	5	1	5	9	0.700
	4	6	2	4	8	0.700
	5	5	4	5	6	0.550
	6	6	4	4	6	0.600
	7	5	3	5	7	0.600
	8	5	3	5	7	0.600
	9	6	1	4	9	0.750
	10	3	1	7	9	0.600
	11	5	0	5	10	0.750
	12	8	2	2	8	0.800
	13	5	0	5	10	0.750
	14	8	2	2	8	0.800
	15	10	5	0	5	0.750
	16	9	5	1	5	0.700
	17	7	2	3	8	0.750
	18	4	2	6	8	0.600
	19	6	1	4	9	0.750
	20	5	7	5	3	0.400

(b) Full results for ALL-CL classifier

Category	Fold	TP	FP	FN	TN	Accuracy
1						
	1	5	5	0	0	0.500
	2	5	3	0	2	0.700
	3	4	5	1	0	0.400
	4	5	4	0	1	0.600
	5	5	5	0	0	0.500
	6	5	5	0	0	0.500
	7	5	4	0	1	0.600
	8	5	5	0	0	0.500
	9	4	4	1	1	0.500
	10	5	2	0	3	0.800
	11	5	3	0	2	0.700
	12	5	3	0	2	0.700
	13	5	3	0	2	0.700
	14	4	5	1	0	0.400
	15	4	5	1	0	0.400
	16	5	5	0	0	0.500
	17	4	4	1	1	0.500
	18	3	2	2	3	0.600
	19	4	5	1	0	0.400
	20	5	5	0	0	0.500
2						
	1	5	5	0	0	0.500
	2	5	3	0	2	0.700
	3	5	5	0	0	0.500
	4	5	4	0	1	0.600
	5	5	4	0	1	0.600
	6	4	3	1	2	0.600
	7	5	5	0	0	0.500
	8	5	4	0	1	0.600
	9	5	3	0	2	0.700
	10	5	3	0	2	0.700
	11	5	3	0	2	0.700
	12	4	5	1	0	0.400
	13	5	3	0	2	0.700
	14	5	3	0	2	0.700
	15	5	5	0	0	0.500
	16	4	4	1	1	0.500
	17	5	2	0	3	0.800
	18	2	1	3	4	0.600
	19	5	4	0	1	0.600
	20	5	5	0	0	0.500
3						
	1	6	0	4	10	0.800
	2	6	4	4	6	0.600
	3	6	1	4	9	0.750
	4	7	0	3	10	0.850
	5	3	3	7	7	0.500
	6	8	4	2	6	0.700
	7	6	7	4	3	0.450
	8	7	3	3	7	0.700
	9	7	2	3	8	0.750
	10	5	1	5	9	0.700
	11	2	1	8	9	0.550
	12	10	4	0	6	0.800
	13	8	0	2	10	0.900
	14	7	6	3	4	0.550
	15	10	3	0	7	0.850
	16	8	5	2	5	0.650
	17	7	0	3	10	0.850
	18	1	1	9	9	0.500
	19	9	3	1	7	0.800
	20	8	8	2	2	0.500

(a) Full results for UC-IA classifier

Category	Fold	TP	FP	FN	TN	Accuracy
1						
	1	4	3	1	2	0.600
	2	3	2	2	3	0.600
	3	5	0	0	5	1.000
	4	4	3	1	2	0.600
	5	4	2	1	3	0.700
	6	3	1	2	4	0.700
	7	2	2	3	3	0.500
	8	3	0	2	5	0.800
	9	4	3	1	2	0.600
	10	1	1	4	4	0.500
	11	1	0	4	5	0.600
	12	5	2	0	3	0.800
	13	5	4	0	1	0.600
	14	5	5	0	0	0.500
	15	5	0	0	5	1.000
	16	4	2	1	3	0.700
	17	2	2	3	3	0.500
	18	3	2	2	3	0.600
	19	3	3	2	2	0.500
	20	4	2	1	3	0.700
2						
	1	4	2	1	3	0.700
	2	3	1	2	4	0.700
	3	4	1	1	4	0.800
	4	5	0	0	5	1.000
	5	5	0	0	5	1.000
	6	2	1	3	4	0.600
	7	3	1	2	4	0.700
	8	5	5	0	0	0.500
	9	3	4	2	1	0.400
	10	3	1	2	4	0.700
	11	4	2	1	3	0.700
	12	3	4	2	1	0.400
	13	4	1	1	4	0.800
	14	1	2	4	3	0.400
	15	5	2	0	3	0.800
	16	3	1	2	4	0.700
	17	4	4	1	1	0.500
	18	1	1	4	4	0.500
	19	5	4	0	1	0.600
	20	5	2	0	3	0.800
3						
	1	9	2	1	8	0.850
	2	9	0	1	10	0.950
	3	6	3	4	7	0.650
	4	7	3	3	7	0.700
	5	5	1	5	9	0.700
	6	5	1	5	9	0.700
	7	4	4	6	6	0.500
	8	7	4	3	6	0.650
	9	8	3	2	7	0.750
	10	3	2	7	8	0.550
	11	6	2	4	8	0.700
	12	8	3	2	7	0.750
	13	5	5	5	5	0.500
	14	9	4	1	6	0.750
	15	8	2	2	8	0.800
	16	4	2	6	8	0.600
	17	6	2	4	8	0.700
	18	2	2	8	8	0.500
	19	10	7	0	3	0.650
	20	9	7	1	3	0.600

(b) Full results for WS-IA classifier

Category	Fold	TP	FP	FN	TN	Accuracy
1						
	1	5	3	0	2	0.700
	2	4	2	1	3	0.700
	3	5	2	0	3	0.800
	4	5	4	0	1	0.600
	5	5	4	0	1	0.600
	6	4	1	1	4	0.800
	7	4	3	1	2	0.600
	8	4	4	1	1	0.500
	9	5	3	0	2	0.700
	10	2	2	3	3	0.500
	11	3	0	2	5	0.800
	12	5	3	0	2	0.700
	13	5	4	0	1	0.600
	14	5	5	0	0	0.500
	15	5	2	0	3	0.800
	16	4	2	1	3	0.700
	17	3	3	2	2	0.500
	18	3	2	2	3	0.600
	19	5	5	0	0	0.500
	20	5	3	0	2	0.700
2						
	1	4	1	1	4	0.800
	2	4	0	1	5	0.900
	3	4	1	1	4	0.800
	4	4	0	1	5	0.900
	5	5	1	0	4	0.900
	6	2	0	3	5	0.700
	7	3	2	2	3	0.600
	8	5	5	0	0	0.500
	9	4	2	1	3	0.700
	10	3	1	2	4	0.700
	11	4	1	1	4	0.800
	12	2	3	3	2	0.400
	13	5	2	0	3	0.800
	14	3	2	2	3	0.600
	15	5	2	0	3	0.800
	16	2	0	3	5	0.700
	17	4	3	1	2	0.600
	18	1	0	4	5	0.600
	19	5	4	0	1	0.600
	20	5	2	0	3	0.800
3						
	1	10	2	0	8	0.900
	2	9	0	1	10	0.950
	3	6	3	4	7	0.650
	4	6	1	4	9	0.750
	5	7	2	3	8	0.750
	6	5	2	5	8	0.650
	7	4	2	6	8	0.600
	8	5	4	5	6	0.550
	9	6	1	4	9	0.750
	10	4	1	6	9	0.650
	11	6	2	4	8	0.700
	12	6	0	4	10	0.800
	13	4	1	6	9	0.650
	14	7	2	3	8	0.750
	15	9	2	1	8	0.850
	16	5	5	5	5	0.500
	17	6	1	4	9	0.750
	18	3	3	7	7	0.500
	19	8	3	2	7	0.750
	20	8	5	2	5	0.650

(a) Full results for UC-PA classifier

Category	Fold	TP	FP	FN	TN	Accuracy
1						
	1	3	0	2	5	0.800
	2	3	2	2	3	0.600
	3	3	0	2	5	0.800
	4	5	3	0	2	0.700
	5	3	2	2	3	0.600
	6	2	1	3	4	0.600
	7	1	1	4	4	0.500
	8	2	0	3	5	0.700
	9	4	2	1	3	0.700
	10	1	0	4	5	0.600
	11	1	0	4	5	0.600
	12	3	1	2	4	0.700
	13	4	2	1	3	0.700
	14	3	3	2	2	0.500
	15	5	0	0	5	1.000
	16	2	1	3	4	0.600
	17	2	1	3	4	0.600
	18	1	1	4	4	0.500
	19	3	3	2	2	0.500
	20	4	1	1	4	0.800
2						
	1	3	1	2	4	0.700
	2	3	1	2	4	0.700
	3	2	1	3	4	0.600
	4	4	0	1	5	0.900
	5	3	0	2	5	0.800
	6	1	0	4	5	0.600
	7	2	1	3	4	0.600
	8	3	4	2	1	0.400
	9	3	2	2	3	0.600
	10	1	2	4	3	0.400
	11	4	4	1	1	0.500
	12	2	2	3	3	0.500
	13	4	0	1	5	0.900
	14	3	0	2	5	0.800
	15	5	2	0	3	0.800
	16	3	2	2	3	0.600
	17	3	3	2	2	0.500
	18	0	1	5	4	0.400
	19	5	3	0	2	0.700
	20	5	1	0	4	0.900
3						
	1	8	1	2	9	0.850
	2	7	0	3	10	0.850
	3	4	0	6	10	0.700
	4	6	3	4	7	0.650
	5	5	0	5	10	0.750
	6	4	0	6	10	0.700
	7	2	0	8	10	0.600
	8	6	2	4	8	0.700
	9	5	2	5	8	0.650
	10	2	0	8	10	0.600
	11	2	2	8	8	0.500
	12	5	3	5	7	0.600
	13	4	0	6	10	0.700
	14	4	0	6	10	0.700
	15	6	1	4	9	0.750
	16	3	2	7	8	0.550
	17	3	3	7	7	0.500
	18	2	1	8	9	0.550
	19	8	1	2	9	0.850
	20	8	7	2	3	0.550

(b) Full results for WS-PA classifier

Category	Fold	TP	FP	FN	TN	Accuracy
1						
	1	4	4	1	1	0.500
	2	3	2	2	3	0.600
	3	5	0	0	5	1.000
	4	4	3	1	2	0.600
	5	3	2	2	3	0.600
	6	3	1	2	4	0.700
	7	3	3	2	2	0.500
	8	3	2	2	3	0.600
	9	3	2	2	3	0.600
	10	1	0	4	5	0.600
	11	2	0	3	5	0.700
	12	4	2	1	3	0.700
	13	5	2	0	3	0.800
	14	4	3	1	2	0.600
	15	5	1	0	4	0.900
	16	2	1	3	4	0.600
	17	2	2	3	3	0.500
	18	2	0	3	5	0.700
	19	4	4	1	1	0.500
	20	5	3	0	2	0.700
2						
	1	4	1	1	4	0.800
	2	4	0	1	5	0.900
	3	4	1	1	4	0.800
	4	5	0	0	5	1.000
	5	4	1	1	4	0.800
	6	1	0	4	5	0.600
	7	3	1	2	4	0.700
	8	5	5	0	0	0.500
	9	4	2	1	3	0.700
	10	3	1	2	4	0.700
	11	4	1	1	4	0.800
	12	2	1	3	4	0.600
	13	5	2	0	3	0.800
	14	3	2	2	3	0.600
	15	5	2	0	3	0.800
	16	2	1	3	4	0.600
	17	3	2	2	3	0.600
	18	0	0	5	5	0.500
	19	5	4	0	1	0.600
	20	5	2	0	3	0.800
3						
	1	8	1	2	9	0.850
	2	8	0	2	10	0.900
	3	3	2	7	8	0.550
	4	6	1	4	9	0.750
	5	5	1	5	9	0.700
	6	5	0	5	10	0.750
	7	3	1	7	9	0.600
	8	5	2	5	8	0.650
	9	4	1	6	9	0.650
	10	4	1	6	9	0.650
	11	7	1	3	9	0.800
	12	6	0	4	10	0.800
	13	4	0	6	10	0.700
	14	3	0	7	10	0.650
	15	7	1	3	9	0.800
	16	4	2	6	8	0.600
	17	5	1	5	9	0.700
	18	2	1	8	9	0.550
	19	6	2	4	8	0.700
	20	8	5	2	5	0.650