Studying Dry Eye Syndrome with Machine Learning

by

Tejasvi Kothapalli

Electrical Engineering and Computer Science

in the

Undergraduate Division

of the

University of California, Berkeley

Spring 2022

Abstract

Studying Dry Eye Syndrome with Machine Learning

by

Tejasvi Kothapalli

Bachelor's of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Dry Eye Syndrome (DES) is the most common ocular disease. In fact, over 5% of US adults suffer from DES. This thesis is an exploration of applying machine learning to study DES.

Chapter 1 will discuss the use of classic machine learning for diagnoses relating to DES. A large dataset with over two hundred variables was collected for subjects. These variables include demographic information, clinical signs, subjective symptoms, and ocular disease diagnoses. We propose a method that predicts variables and also allows us to analyze the relationship between variables. Logistic Regression is used as the prediction model. A recursive pruning algorithm is used to achieve high accuracy predictions and also chose the top predictor variables. These top predictor variables can be analyzed to check prior clinical findings and also tease out new relationships between variables.

Chapter 2 will describe an image processing technique with implications for DES. We propose a novel method for tracking the spread of the Tear Film Lipid Layer (TFLL). The motion of the TFLL is a determining factor for whether a patient has DES. Specificially, [1] showed that the time required for the lipid layer to stabilize after a blink was longer for Dry Eye subjects. Our method presents a robust way to segment the iris in a video of the eye. In addition, we present a way to track points on the lipid layer using feature tracking.

The focus of this thesis will be on the machine learning methodology rather than the clinical implications of the results.

Contents

List of Figures

Chapter 1

Predicting and Understanding Relationships for Ocular Disease Diagnosis, Signs, and Symptoms

1.1 Introduction

As a complicated and multifactorial disease, Dry eye disease requires a systematic and comprehensive study of relationships of dry-eye-related signs, symptoms and diagnoses. This chapter makes three significant contributions. First, a comprehensive dataset for studying dry-eye related signs, symptoms and diagnoses was collected and will be publicly available for future studies. Second, a machine-learning based method is proposed for interpreting relationships between clinical signs, symptoms and diagnoses. Third, analyzing the relationships of signs, symptoms and diagnoses has illuminated novel observations relating to Dry eye disease. Some observations align with previous works while some challenge results and views from previous works.

1.2 Prior Work

[7] provides a great overview of all the different artificial intelligence methods used to study Dry Eye Disease. In our case, we want to predict different variables from tabular data, which is a supervised machine learning classification task. More important, we have a large dataset with over two hundred variables collected for each subject. The challenge of our work is to process this large dataset and distill out the import variables that can be used to predict other variables. Most of the work done for Dry Eye Disease surrounds analyzing one variable (ie. [2]) in relation with Dry Eye Diagnosis as shown in [7]. However, a few works have studied the relationships between a large number of variables and Dry Eye Disease. We will discuss each of these studies now.

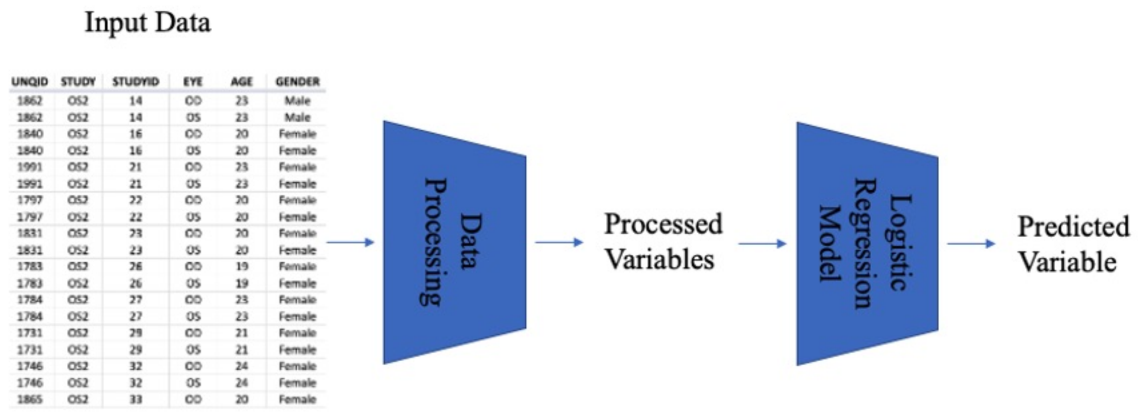
[5] also works with a large dataset with variables relating to demographics and nutrition information. Their goal is to predict Dry Eye Disease with the available variables. The most important features are selected with least absolute shrinkage and selection operator (LASSO). The downside to using LASSO is that the lambda parameter must be tuned for the specific task. Once they have the most important features they train a logistic regression model. Then a point-based scoring system is developed where the total score of each individual is the sum of each of the associated model coefficients for the factors describing the individual. Once again a cutoff must be tuned for determining the best threshold for the individual scores. Its important to note that the logistic regression model is a non-linear model, yet [5] simplifies the model to a linear equation.

[4] also aimed to predict dry eye with physiologic tests. Instead of pruning variables like in [5], [4] instead clusters all the variables with hierarchical clustering. Using the information about the similarity between the variables in each of the clusters, they build a classification tree. However, this method only works with around twenty variables and would require a much stronger variable pruning method to deal with a lot more variables like in our case.

1.3 Methods

The first objective was to use patient information to predict clinical diagnoses, symptoms, and signs. The second objective was to find the best variables used for prediction. Logistic Regression was used to learn from the dataset to build a prediction model and the resulting model weights were used to analyze the top features. The overall pipeline for prediction is illustrated in Figure 1.1.

Figure 1.1: Predicting Variable Pipeline



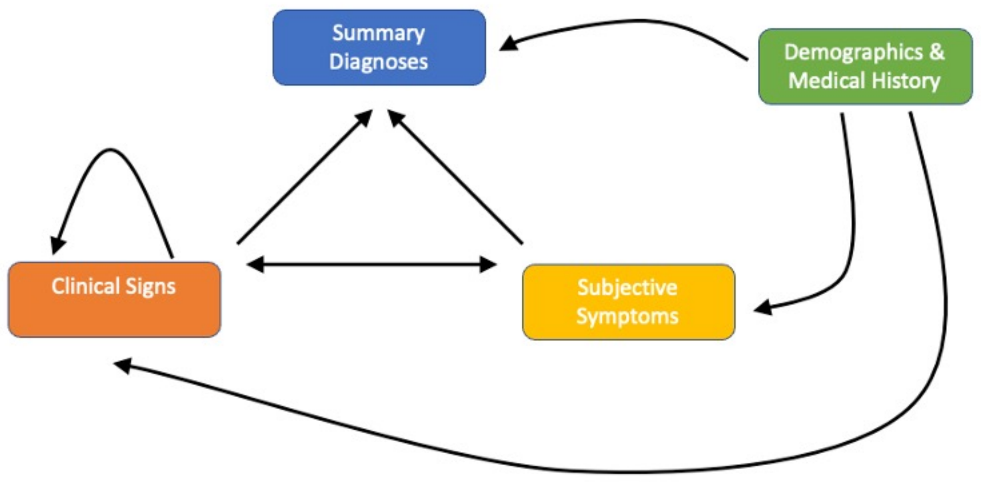
UNQID	STUDY	STUDYID	EYE	AGE	GENDER
1862	OS2	14	OD	23	Male
1862	OS2	14	OS	23	Male
1840	OS2	16	OD	20	Female
1840	OS2	16	OS	20	Female
1991	OS2	21	OD	23	Female
1991	OS2	21	OS	23	Female
1797	OS2	22	OD	20	Female
1797	OS2	22	OS	20	Female
1831	OS2	23	OD	20	Female
1831	OS2	23	OS	20	Female
1783	OS2	26	OD	19	Female
1783	OS2	26	OS	19	Female
1784	OS2	27	OD	23	Female
1784	OS2	27	OS	23	Female
1731	OS2	29	OD	21	Female
1731	OS2	29	OS	21	Female
1746	OS2	32	OD	24	Female
1746	OS2	32	OS	24	Female
1865	OS2	33	OD	20	Female

Pipeline for predicting variables using data processing and logistic regression.

Design of Input and Output Features

There are four main categories that the collected variables fall under: Demographics and Medical History, Clinical Signs, Subjective Symptoms, and Summary Diagnoses. Of these four categories, certain variables within Signs, Symptoms, and Diagnoses are predicted. Figure 1.2 illustrates which category of variables were used to predict other category of variables. When diagnoses variables were predicted, all other diagnoses variables were not used as input variables. Similarly, Symptoms were not used to predict other Symptoms variables. However, Signs were predicted with other Signs variables.

Figure 1.2: Variables Used to Predict other Variables by Category



Graphic to illustrate which category of variables were used as input variables to predict other variables.

Data Pre-Processing

Sparse Variables

In order to feed the clinical data into the logistic regression model there were several data cleaning steps. We first delt with sparse features or variables where a large percentage of the variables were unknown. Some of the sparse features made no sense to keep in any of the models presented. For example, the variable SJOGREN (Medical History - Sjogren's Syndrome [Yes, No]) has zero percent "Yes" values. Thus, for any variables where any of the values had less than one percent, we completely took it out from all the models. Other sparse features had a large percentage of missing values or a certain value for that feature had a very low percentage of the total data distribution. For example, the variable HIBP (Medical History - High Blood Pressure [Yes, No]) had 1.8% "Yes" and 98.2% "No" values.

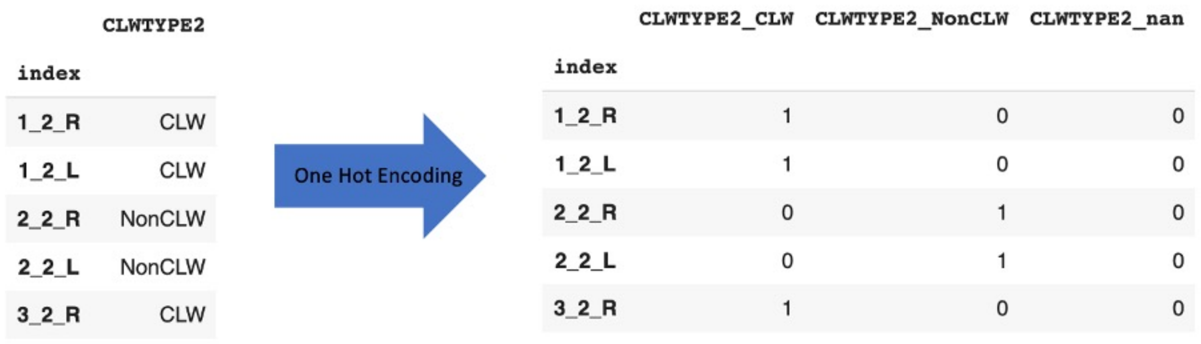
These kinds of sparse features were also removed. Finally, any variables that had more than 90% of null values were removed as well.

It is important to note that allowing the model to use sparse features for prediction increases the prediction accuracy. This is because oftentimes sparse features show up in the top 10 predictor variables when they are allowed to be used in the model. However, when it comes to explaining and interpreting the top features for scientific understanding or clinical utility, removing the sparse features makes the most sense. It is hard to justify (in a practical setting) why a huge lack of data should still allow the variable to be considered important.

Quantitative Variables

Logistic regression requires quantitative features as input. Thus, all categorical variables were converted to quantitative variables through one hot encoding. For example, the variable CLWTYPE2 (Contact Lens Wear Type 2 [CLW, NonCLW]) is a categorical variable. It was converted from one quantitative variable to three categorical variables as shown in Figure 1.3. Note that CLWTYPE2_nan is an indicator variable for when the CLWTYPE2 variable had a missing value.

Figure 1.3: One Hot Encoding Categorical Variables



index	CLWTYPE2
1_2_R	CLW
1_2_L	CLW
2_2_R	NonCLW
2_2_L	NonCLW
3_2_R	CLW

One Hot Encoding

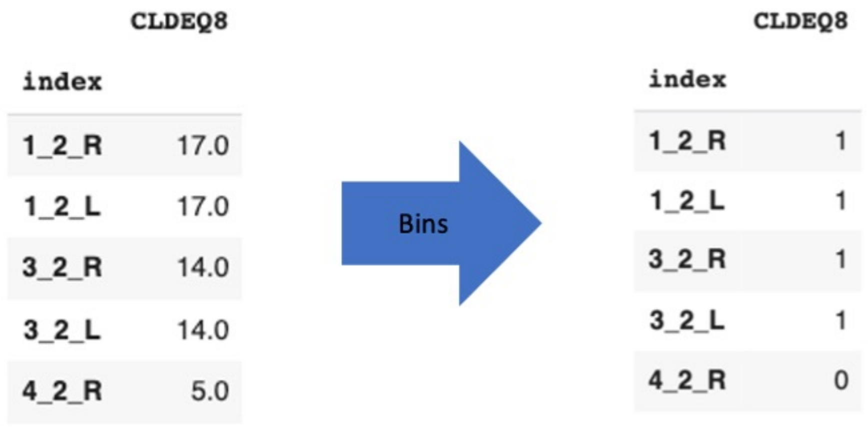
index	CLWTYPE2_CLW	CLWTYPE2_NonCLW	CLWTYPE2_nan
1_2_R	1	0	0
1_2_L	1	0	0
2_2_R	0	1	0
2_2_L	0	1	0
3_2_R	1	0	0

Example of converting a categorical variable into quantitative values through one hot encoding.

When quantitative signs and symptoms variables are predicted they are put into bins based on prior literature to the extent possible. For example, the variable CLDEQ8 (CLDEQ-8 questionnaire score, CL wearers only) is binned with [<12, ≥ 12] as shown in Figure 1.4. Note that when quantitative signs and symptoms are used as predictors, the original values are kept, they are not binned. This method of binning is only used when signs and symptoms variables are predicted because logistic regression only models discrete outputs. It also makes a lot more sense to predict into one of these bins based on prior literature conclusions.

Certain variables that are categorical have an ordinal nature to them and they were converted to frequencies as appropriate. Once all the variables are converted to quantitative

Figure 1.4: Binning Quantitative Variables



index	CLDEQ8
1_2_R	17.0
1_2_L	17.0
3_2_R	14.0
3_2_L	14.0
4_2_R	5.0

index	CLDEQ8
1_2_R	1
1_2_L	1
3_2_R	1
3_2_L	1
4_2_R	0

Example of binning CLDEQ8 based on the following bins: [<12, ≥ 12]

values, any missing data entries are imputed with the mode of that variable. Finally, all variables are normalized such that all values fall in the range of 0 and 1.

Algorithm Design and Prediction

Explanation of Logistic Regression

First, we will briefly discuss binary logistic regression. Binary logistic regression is used for classification tasks where the variable to be predicted has only two values. For example, the diagnosis of Meibomian Gland Dysfunction (variable DIAGMGD) has only two values: Yes and No encoded as 1 and 0, respectively.

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_m x_m)}} \tag{1.1}$$

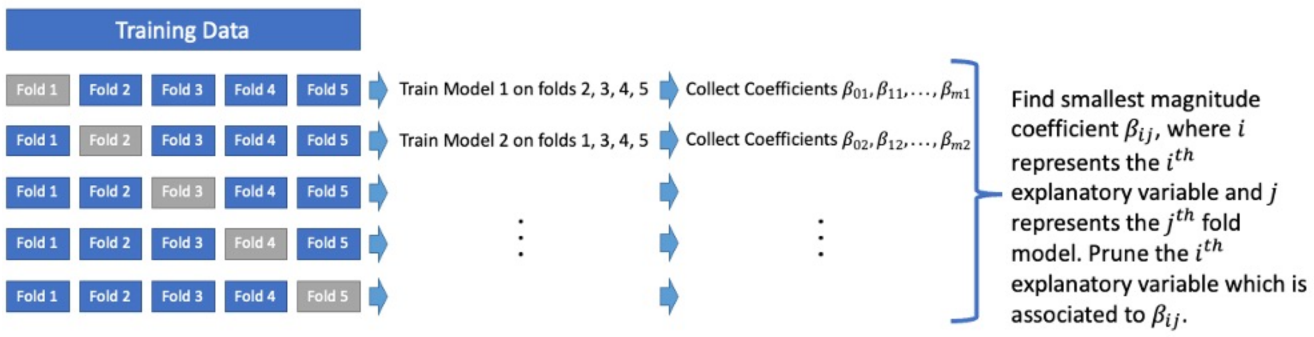
Equation 1 is the standard binary logistic regression equation where m explanatory variables are used to predict a binary class variable. p represents the probability of the prediction being class 1 while 1-p represents the probability that the prediction is class 0. x_i represents the value of the i^{th} explanatory variable and β_i is the associated coefficient. The β_i coefficients will be learned in order to maximize the likelihood of the given training data. Note that when the predicted variables have more than two classes, multinomial logistic regression is used. We will not discuss the details of multinomial logistic regression, but it is a direct extension of binary logistic regression.

Feature Pruning

For each variable to be predicted, there are hundreds of potential explanatory variables that could be used. However, using all the explanatory variables often leads to lower accuracy for prediction. This is because many of the variables in the dataset are redundant. Thus, it is necessary to prune explanatory variables.

The process of pruning variables as illustrated in Figure 1.5 is discussed in more detail as follows. The data is split into five folds randomly at the start of the pruning process. These five folds are kept constant for the rest of the pruning process. Four of the five folds are used to train the logistic regression model and then the fifth fold is used as the validation set. The process of training a logistic regression model is repeated five times, once for each validation fold. Thus, a total of five different logistic regression models will be trained. Each variable will have an associated coefficient for each logistic regression model. The variable with the smallest coefficient magnitude among all the five logistic regression models will be pruned. The average validation fold accuracy will be stored given the total number of features used. This whole process is repeated until there is only one feature used in the logistic regression model. The final model will be the one with the smallest number of features used while also achieving the highest average validation set accuracy.

Figure 1.5: Pruning Least Significant Variable


Training Data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Train Model 1 on folds 2, 3, 4, 5
Train Model 2 on folds 1, 3, 4, 5
Collect Coefficients $\beta_{01}, \beta_{11}, \ldots, \beta_{m1}$
Collect Coefficients $\beta_{02}, \beta_{12}, \ldots, \beta_{m2}$
Find smallest magnitude coefficient β_{ij}, where i represents the i^{th} explanatory variable and j represents the j^{th} fold model. Prune the i^{th} explanatory variable which is associated to β_{ij}.

Graphic illustrating how a feature is pruned after five separate logistic regression models are trained on the five folds.

Top Feature Selection

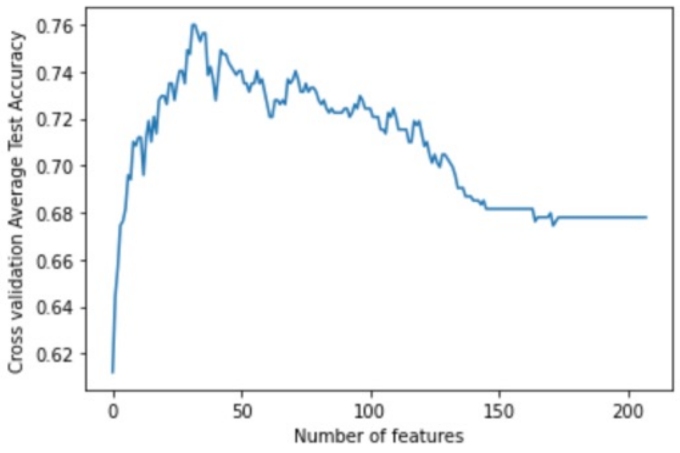
Once the necessary explanatory variables are pruned the best feature predictors for the predicted variable are recorded. The top predictors for the final logistic regression model are the variables with the highest magnitude model coefficients. The feature pruning process described only gives one final model. However, we observed that through multiple runs, where the data folds are different, the top variable predictors would change. In order to reduce the variance of the final top variable predictors we created five separate final models.

The top predictors are then determined based on aggregating the rankings from all five models. Note that the number five is used here as the total number of final models created and it is not related to the five separate folds that are used each time for each final model.

Evaluation Protocol

As mentioned, using all the available features will lead to low accuracies due to redundancy. Thus, features must be pruned. Figure 1.6 demonstrates how the number of features used in the logistic regression model for the variable DIAGMGD affects the average cross validation accuracy.

Figure 1.6: Accuracy Versus Number of Features

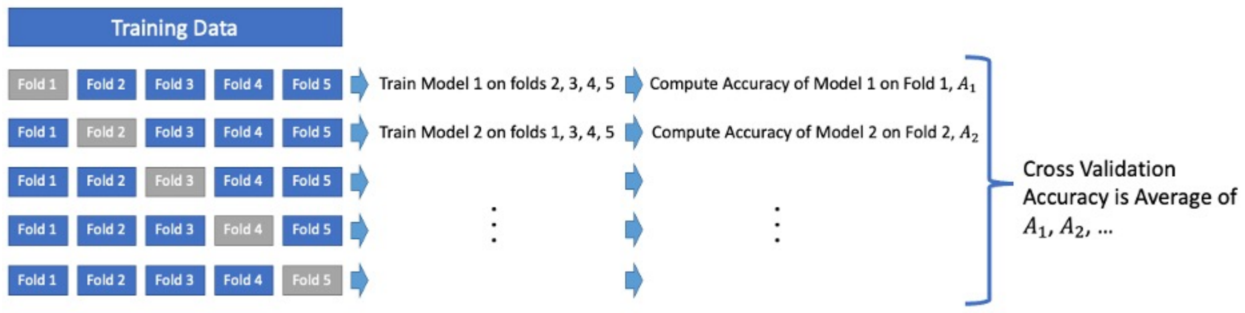


Graphic demonstrating how the number of features used for prediction affects the cross-validation accuracy.

At a given number of features, five separate models are trained, and the cross-validation accuracy is computed as shown in Figure 1.7. The process starts with all the features possible and iteratively prunes variables until only 1 feature remains as shown in Figure 1.8.

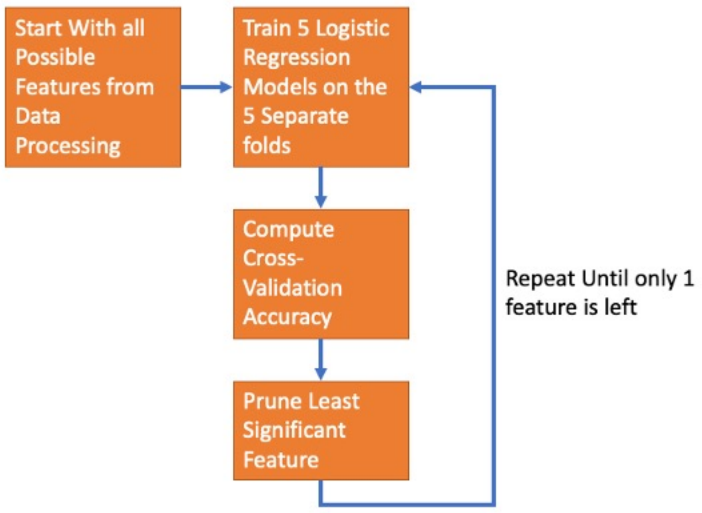
The process shown in Figure 1.8 yields one final model. This process is repeated until five separate final models are trained to predict an outcome variable. Each of these models will have different number of features used and cross validation accuracy. Thus, the median number of features used in the five models is reported. Additionally, the mean accuracy of all five models is reported along with the standard deviation. The mean accuracy will determine the strength of the prediction model while the reported standard deviation measures the variance in the prediction model's training procedure.

Figure 1.7: Computing Cross-Validation Accuracy


Training Data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Train Model 1 on folds 2, 3, 4, 5
Compute Accuracy of Model 1 on Fold 1, A_1
Train Model 2 on folds 1, 3, 4, 5
Compute Accuracy of Model 2 on Fold 2, A_2
Cross Validation Accuracy is Average of A_1, A_2, ...

Graphic illustrating how cross validation accuracy is computed for a set number of features

Figure 1.8: Interatively Pruning Variables

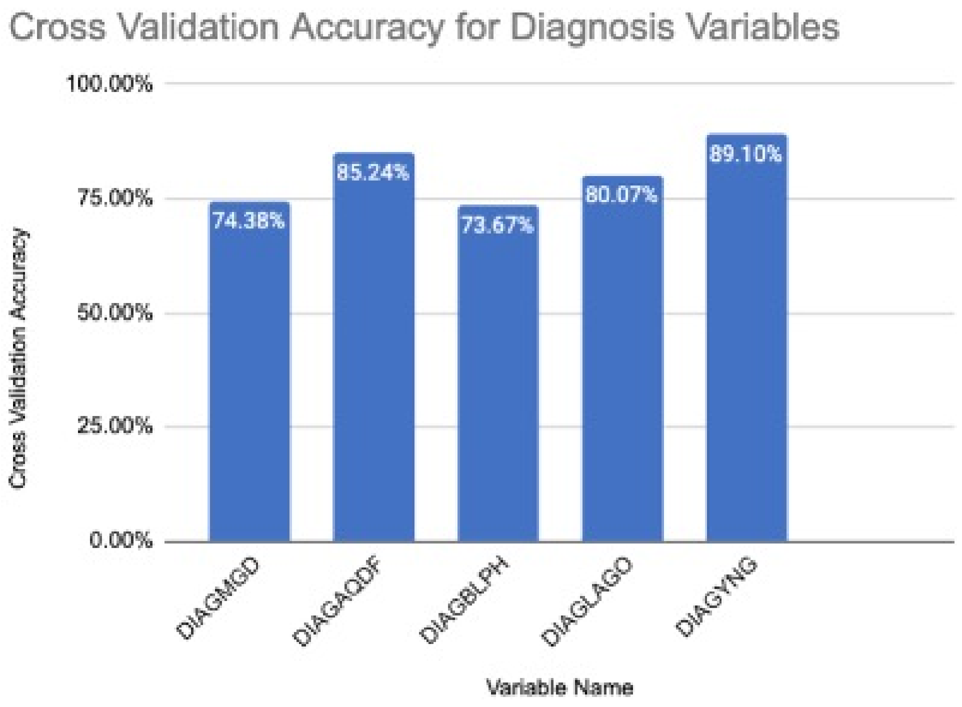

Start With all Possible Features from Data Processing
Train 5 Logistic Regression Models on the 5 Separate folds
Compute Cross-Validation Accuracy
Prune Least Significant Feature
Repeat Until only 1 feature is left

Graphic illustrating how features are pruned, and cross-validation accuracies are collected.

1.4 Results

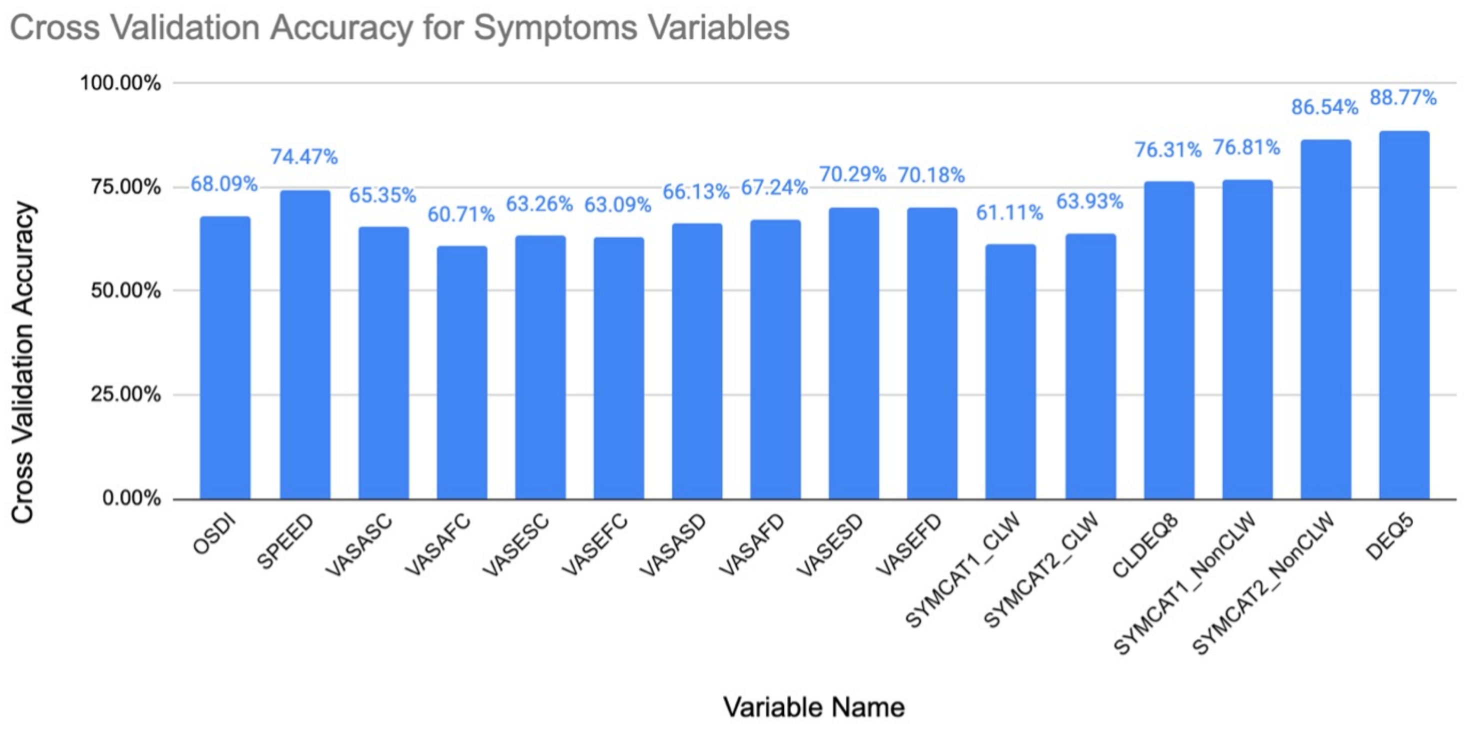
As mentioned in the methods section five separate final models are trained to predict each outcome variable. Each of these models will have different number of features used and cross validation accuracy. Thus, the median number of features used in the five models is reported. Additionally, the mean accuracy of all five models is reported along with the standard deviation. Note that the standard deviation of the cross-validation scores is below 1% for almost all the predicted variables. Thus, the finals models tend to have little variance in terms of accuracy of prediction. Finally, the top five features (aggregated among the five final models) are reported. Along with each of the top predictors we also report the class wise statistics. If the input variable is quantitative, the associated mean of the quantitative variable is reported for each value if the output variable is categorial or bin if the output variable is quantitative. If the input variable is categorical, then the data percentage split is reported for each value if the output variable is categorial or bin if the output variable is quantitative. In addition, Figures 1.9, 1.10, 1.11 summarize the average cross validation for predicting the Diagnoses, Symptoms, and Signs variables respectively.

Figure 1.9: Diagnoses Results



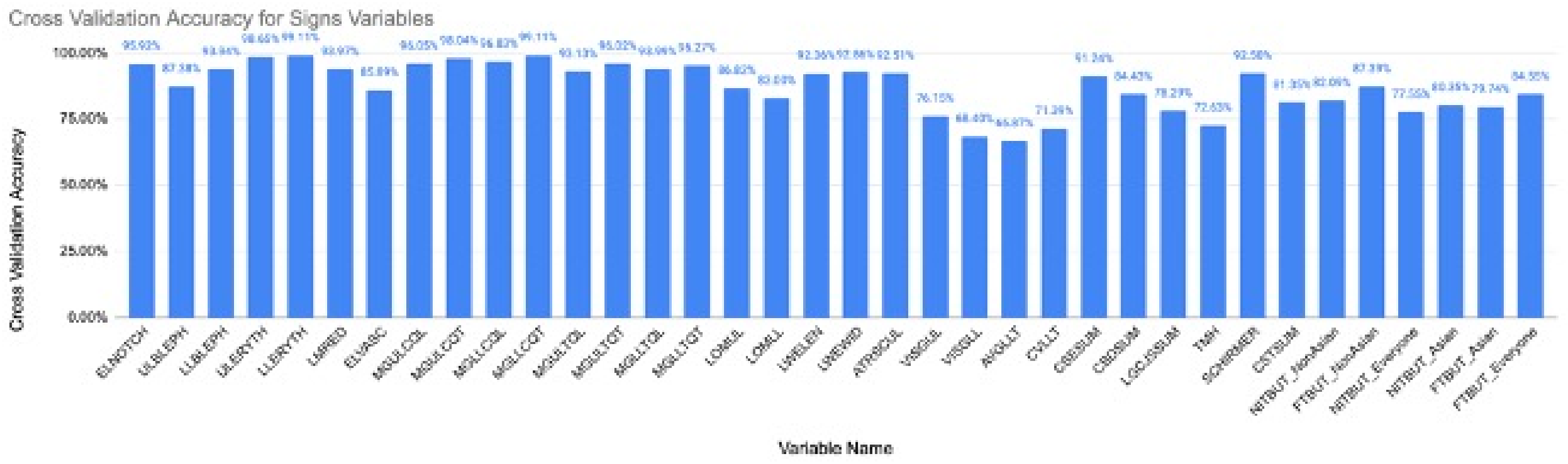
Bar Chart of the cross-validation accuracies for predicting Diagnoses Variables

Figure 1.10: Subjective Symptoms Results



Bar Chart of the cross-validation accuracies for predicting Symptoms Variables

Figure 1.11: Clinical Signs Results

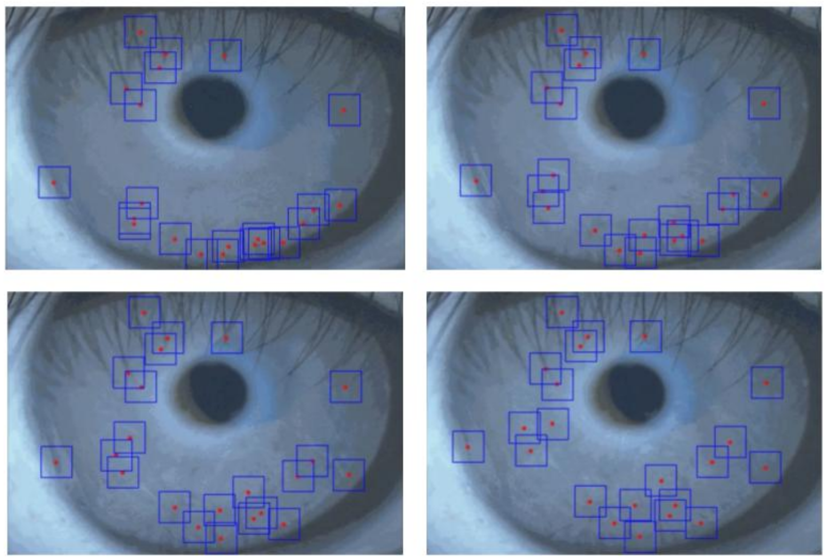


Bar Chart of the cross-validation accuracies for predicting Symptoms Variables

Chapter 2

Tracking the Motion of the Tear Film Lipid Layer

Figure 2.1: Feature Tracking



The above four pictures demonstrate the tracking of the lipid layer. The red dots represent the feature points tracked through the frames and the blue squares represent the patches associated with each feature point. The top left picture is right after a blink and as it can be seen all the feature points start out at the bottom of the iris. The top right is a few frames after and as it can be seen the feature points are moving up the iris. Similarly the chronological order is followed by the bottom left and the bottom right frames. The feature points move up the iris through the frames.

2.1 Introduction

Figure 2.1 shows snapshots of the lipid layer being tracked over multiple frames. The rest of this chapter will describe how the tracking demonstrated in Figure 2.1 is accomplished.

The overview of the method is as follows. We start by detecting blinks in the video so that we can analyze each set of frames between blinks separately. We then align the frames in order to get an accurate average frame. This average frame is used to segment the iris with clustering based segmentation. The feature vectors used for clustering are a combination of the LUV color space value and the position of the pixel. Finally, random points are picked within the segmented iris region and tracked with Sum of Squared Differences (SSD) feature tracking. We show the significance of this procedure by analyzing the vertical displacement of the lipid layer through time and also the lipid layer thickness across the iris region.

2.2 Prior Work

Tracking the motion of the lipid layer can be partitioned into two sub-problems. The first problem is to finding the iris region of the eye. The second problem is tracking the lipid layer motion.

First we will discuss the prior work relating to finding the iris region. Finding the iris region is an image segmentation task. [3] finds this region of interest by finding the darkest spot in the image and performing a flood fill algorithm. Then, roughly speaking, the pixels found from the flood fill algorithm and edges detected from Canny Edge detection are used to determine the iris region. However, it is important to note that this method uses a lot of priors that are specific to their dataset. These priors rely on the exact positioning of the iris for the camera and the scale of the iris in the image. Their videos capture a lot more of the upper and lower eyelids whereas the videos we work with do not show the eyelids at all. Thus their method is not robust to different camera set-ups. Furthermore, their method simplifies the shape of the iris to be circular when in reality it is far more ovular in our camera perspective.

While [3] uses classical image processing techniques, deep learning segmentation models can be used as well like in [8]. However, the problem with deep learning models is that they are often only successful at predicting on test data that falls within the distribution of the training data. In this example it means that the images of the eyes must look similar to the training data that was used to train a specific network. Similar to [3], [8] uses eye images where the upper and lower eyelid are visible, which not true for the data we work with. We have not found any ready to use deep learning models that were trained on data that visually matches that data we are working with. Thus, we have opted to use classic image processing to find the iris.

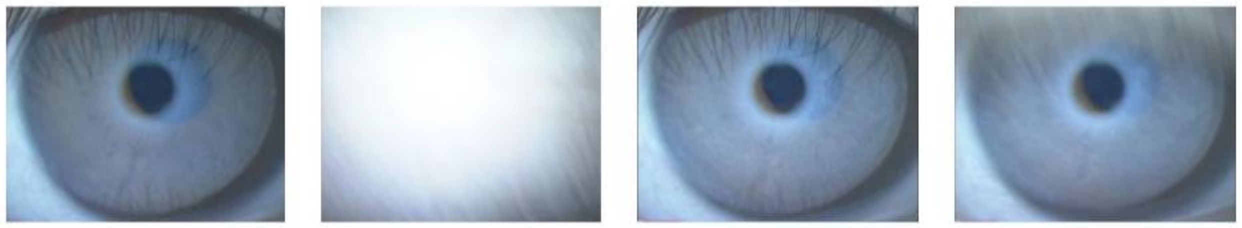
Next we will discuss the previous work relating to tracking the lipid layer motion. [9] uses a video-interferometer as a means to measure the vertical displacement of the lipid layer. Our method aims to use a camera setup rather than the much more expensive video-interferometer. To the best of our knowledge, our method is the first to use a direct video of the eye to measure the lipid layer spread.

2.3 Methods

Blink Detection

The video of the subject eye is captured so that multiple eye blinks are included. After each eye blink the tear film is restructured. The goal is to track the tear film's motion in between blinks. Right after a blink, the tear film will rapidly restructure and move but it will slowly stagnate in motion and eventually settle in place until the next blink. It is thus important that the tear film motion is tracked separately for each blink.

Figure 2.2: Blinking Frames Sequence



The above four pictures demonstrate what a blink appears as in the video. The first picture is right before a blink. The second picture is right when the blink is starting. The third picture is when the eyelid has completely occluded the eye. The fourth picture is right after the blink is complete.

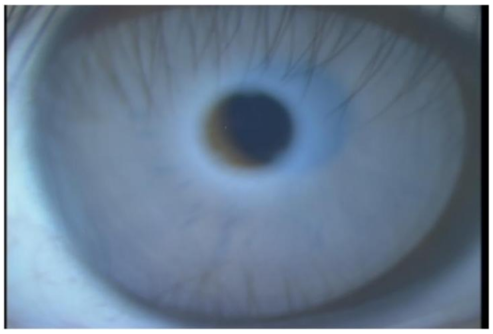
Figure 2.2 illustrates what the frames of the video look like during a blink. The goal of blink detection is to find the frames where the eyelid is partially or completely occluding the eye. We then want to collect the frames in between blinks and analyze each set of frames separately.

We observe that the video of the eye looks relatively constant in nature aside from when the blinks appear. Thus we compute the average frame of the video. This is done by adding all the frames in the video together in the RGB colorspace and dividing by the total number of frames. Figure 2.3 illustrates what the average frame looks like.

Every frame is subtracted by the average frame and the frobenius norm is computed as a measure of distance from the average frame. Then the z-score of each frame's distance from the average frame is computed. Figure 2.4 shows the distance of each frame from the average frame and the thresholding used to determine blink frames.

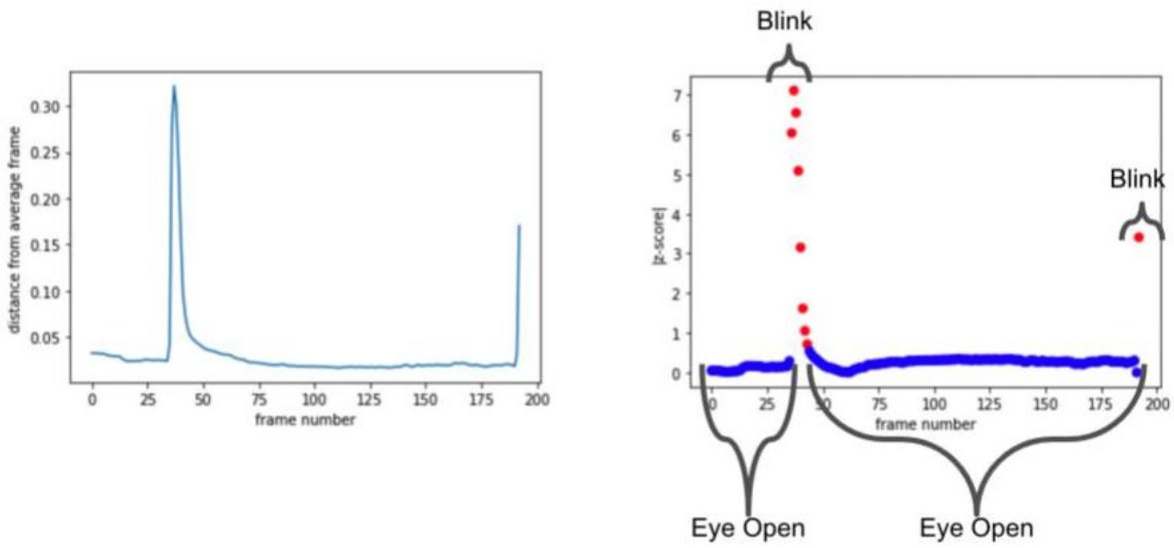
With blink detection each set of frames between blinks will be analyzed separately for tear film motion. In Figure 2.4 there are two separate sets of open eye frames that are separated by a blink.

Figure 2.3: Average Frame



The average frame of the video.

Figure 2.4: Blink Frame Outlier Detection

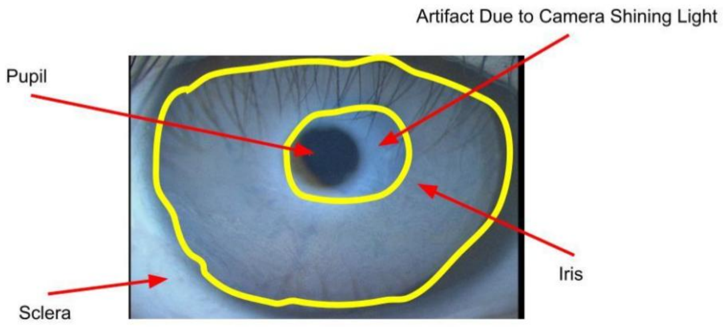


The plot on the left shows the distance of each frame from the mean frame. The plot on the right shows the absolute value of the z-score of each frame. The plot shows when the absolute value of the z-score is greater than 0.6 it will be considered a blink frame. If the frame's absolute value z-score falls below 0.6 it will be considered as the frame where the eye is open.

Iris Segmentation

When tracking the lipid layer, it is important to only do it over the iris region. Thus it is essential to discriminate all other pixels from the iris. This means we want a segmentation map where the iris is detected. Figure 2.5 shows the iris labeled on the eye, which is the region of interest.

Figure 2.5: Parts of Eye Labeled

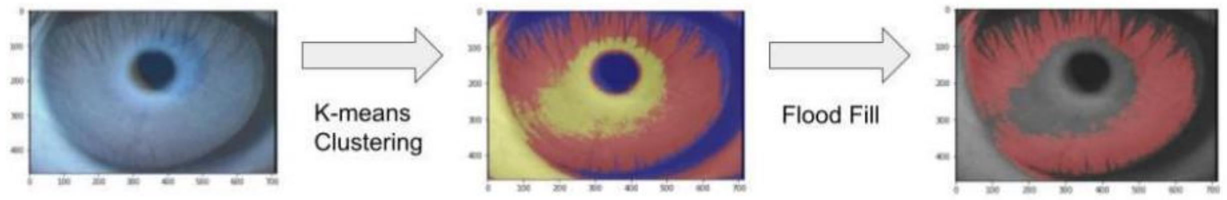


This picture labels the different parts of the eye that are of significance when trying to segment the eye.

Single Frame Iris Clustering

We first try segmentation on a single frame. Specifically, the frame right after a blink is completed will be segmented for the iris. We use clustering, which is a form of unsupervised segmentation. The idea is to assign a cluster to every single pixel in the image. K-Means clustering is used for its efficiency and speed. Figure 2.6 illustrates the steps for segmenting the iris in the image.

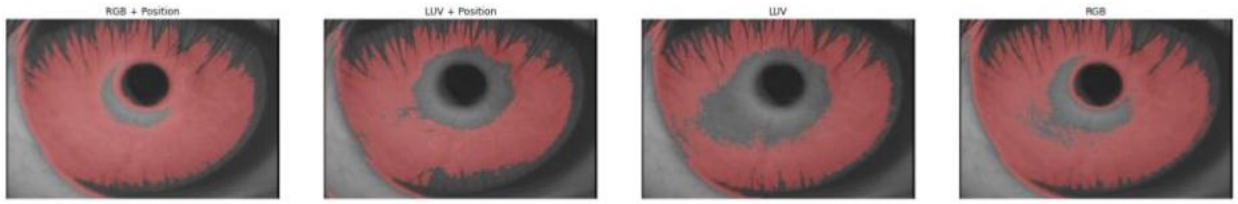
Figure 2.6: Iris Segmentation Process



This graphic shows the pipeline for segmenting the iris. Each pixel feature vector is assigned a cluster with K-means. Then the cluster with the largest concentration of pixels near the center of the image is picked and flood filled to get the iris.

Several feature vectors for each pixel were considered. For the color value both the RGB and LUV colorspace were tested. In addition, the x, y coordinate values were concatenated to the colorspace feature as well to encode position of the pixel. These position pixels were multiplied by a gaussian kernel centered at the center of the image. The motivation behind multiplying by a gaussian kernel was to encode the natural circular shape of the eye into the pixel feature vectors.

Figure 2.7: Segmenting Iris from a Single Frame with different Feature Vectors



These pictures show the clustering segmentation results for different feature vector choices in the single frame setting. The title on top of each picture states the feature vector choice. We found that the "LUV + Position" feature vector was the best for single frame segmentation.

Figure 2.7 demonstrates the qualitative results of segmentation with clustering using different feature vector choices. We find qualitatively that using the LUV color space seems to do a better job for single frame clustering. In addition we find that encoding position into the feature vector results in a better segmentation of the iris. Ultimately, however we found that the segmentations for a single frame alone were too noisy and changed too much between frames.

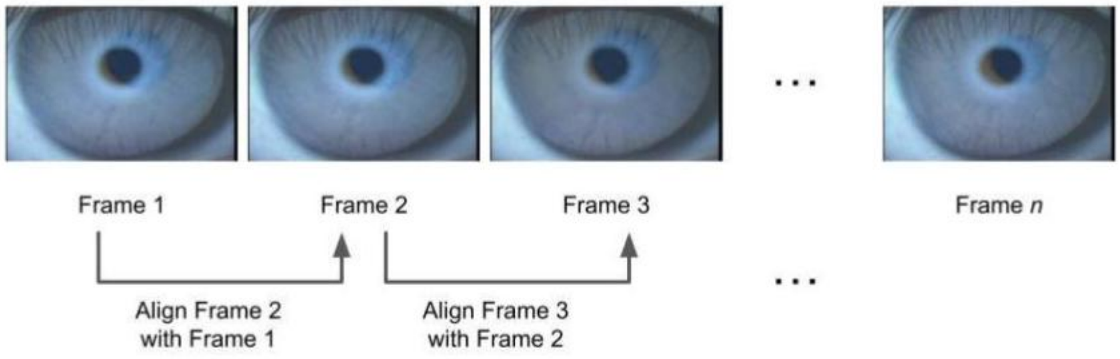
Aligning Frames with Sum of Squared Differences

In order to address the noisy results of segmentation frame by frame we decided to start by aligning all the frames. When the subject's eye is recorded there will be some inherent movement in the eye. This will change the overall position of the iris between frames. Thus each frame will be aligned with its previous frame. The first valid frame will serve as template for every consecutive frame to match. Frame $n+1$ will be aligned with frame n. Frame $n+1$ will slide across frame n, and the alignment score will be computed with Sum of Squared Differences (SSD). The borders of frame $n+1$ will be cut out when computing the alignment. Thus only 90% of the frame $n+1$ will be considered when aligning to frame n. Figure 2.8 illustrates the recursive process of aligning all the frames.

Aligned Mean Frame Iris Clustering

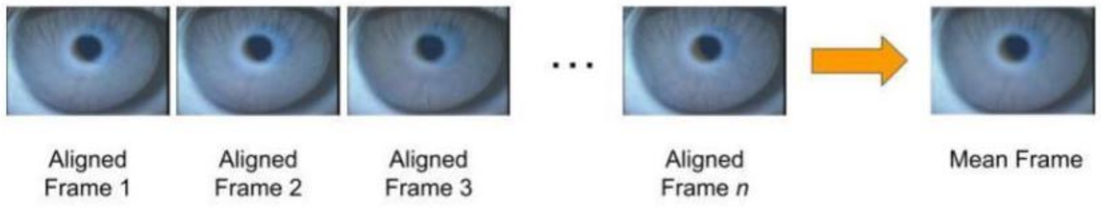
After all the frames of interest have been aligned the mean frame is computed as shown in Figure 2.9. Since all the frames have been aligned to this mean frame we can segment the iris in the mean frame and use the same segmentation map for all the aligned frames. Figure 2.10 shows the qualitative results of segmenting the iris in the mean frame for different choices of feature vectors. We find that the "LUV + Position" feature vector has the best qualitative results for segmentation. We can compare the results of segmentation on a single frame versus the average frame by looking at Figure 2.7 and Figure 2.10, respectively. We observe that the results for segmentation on the average frame is better across all the feature vector options.

Figure 2.8: Aligning Frames



This graphic illustrates the process of aligning all the images together. The process starts with aligning frame 2 with frame 1 and every consecutive frame will be aligned with its previously (already aligned) frame. This recursive process will continue until frame n is aligned with frame n-1. Note when a frame is being aligned with its previous frame, the borders of the current frame will be cut out.

Figure 2.9: Mean Frame of Aligned Frames



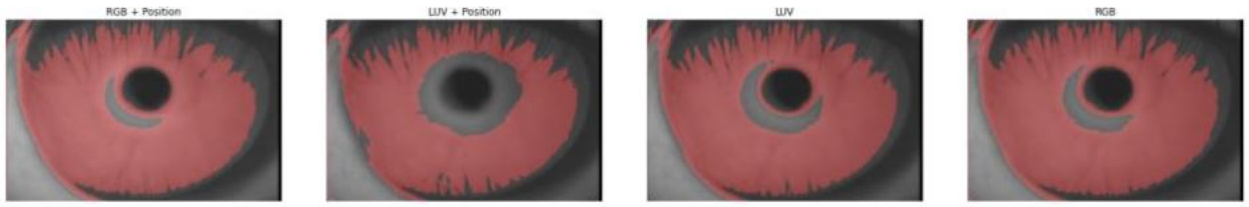
This graphic illustrates how the mean frame is computed. Once all the frames have been aligned they are averaged out in the RGB colorspace to yield the mean frame.

Tracking Lipid Layer with SSD Feature Tracking

With the Iris region segmented, we can now track points on the tear film within the iris region. We take the starting frame and randomly pick feature points in the region of interest as shown in Figure 2.11. Each feature point will have a 75 x 75 pixels patch associated with it where the feature point is at the center. These patches from the previous frame will be used to compute the SSD across the whole image for the next frame to find the most similar patch. Once the most similar patches are found for each feature point, the new patches are used for the next frame tracking. This process is repeated through all the frames. Note that we found randomly picking the feature points for the last frame and tracking backwards in time performed better.

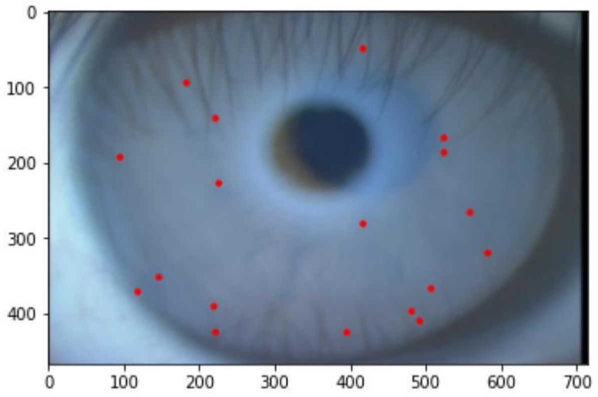
The inspiration for tracking feature points comes from [6]. Their method tracks feature points on the retina but the same principles hold since they are also dealing with the noisy nature of eye movement. Their method uses Normalized Cross Correlation (NCC) instead

Figure 2.10: Aligned Average Frame Iris Segmentation with different Feature Vectors



These pictures show the clustering segmentation results for different feature vector choices in the mean frame setting. The title on top of each picture states the feature vector choice. We found that the "LUV + Position" feature vector was the best.

Figure 2.11: Feature Points on Iris



In order to track points on the tear film we randomly pick points on the segmented region of the iris. These feature points will be tracked across frames.

of SSD. We found that SSD performs a lot better because it respects the intensity value of the pixels whereas NCC is invariant to local average intensity and contrast.

2.4 Clinical Results

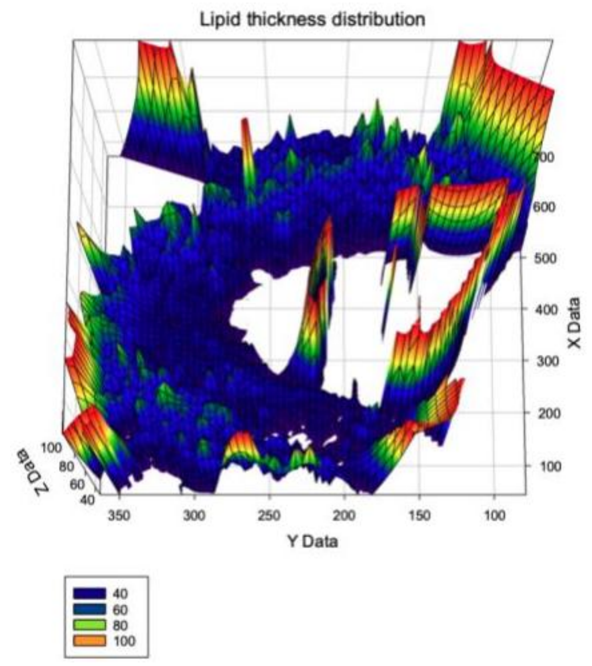
Lipid Layer Thickness Heatmap

Our method is able to successfully segment out the iris region in each frame. Using the technique described in [3] we can compute the lipid layer thickness using the RGB values of the frame.

Exponential Decay Curve for the Vertical Displacement

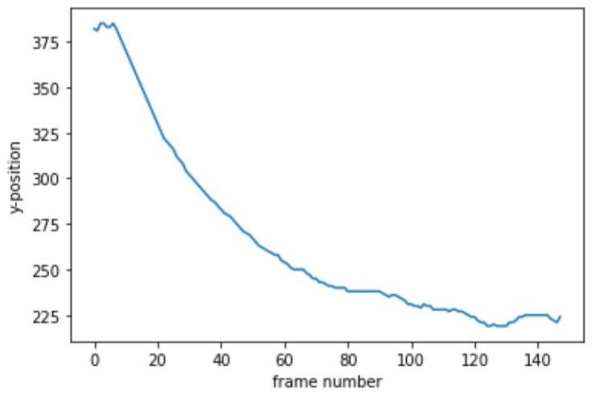
Our lipid layer tracking confirms the findings in [9]. They find that they can model the motion of the vertical displacement with an exponential decay curve. Figure 13 shows the

Figure 2.12: Lipid Layer Thickness Heatmap



This graph shows the Lipid Layer thickness for one frame in time.

Figure 2.13: Lipid Layer Displacement



This graph shows the vertical displacement of the lipid layer is an exponential decay curve.

vertical displacement of the lipid layer through the frames. As it can be seen our tracking method confirms that it is in fact an exponential decay curve.

2.5 Conclusion and Future Work

This work presents a first iteration to track lipid layer motion for direct video. Our work confirms previous findings and analysis of lipid layers. While we provide a robust method we intend to iterate on this method. In particular we want to improve the iris segmentation method by implementing contour finding. Currently the only prior of the eye shape that is used right now is the Gaussian kernel used to weight the position in the feature vector for clustering. However, more priors about the shape of the eye can be used to improve the segmentation results.

We also want to improve the tracking of the lipid layer. We want to try to use a band-pass filter for the color space of the image so that only the lipid layer appears in the image without too much noise from the iris. In addition, we want to try to use optical flow to further improve the tracking of lipid layer.

Bibliography

[1] Eiki Goto and Scheffer CG Tseng. "Kinetic analysis of tear interference images in aqueous tear deficiency dry eye before and after punctal occlusion". In: *Investigative ophthalmology & visual science* 44.5 (2003), pp. 1897–1905.

[2] Franz-H Grus and Albert J Augustin. "Analysis of tear protein patterns by a neural network as a diagnostical tool for the detection of dry eyes". In: *ELECTROPHORESIS: An International Journal* 20.4-5 (1999), pp. 875–880.

[3] Hyeonha Hwang et al. "Image-based quantitative analysis of tear film lipid layer thickness for meibomian gland evaluation". In: *Biomedical engineering online* 16.1 (2017), pp. 1–15.

[4] William D Mathers and Dongseok Choi. "Cluster analysis of patients with ocular surface disease, blepharitis, and dry eye". In: *Archives of ophthalmology* 122.11 (2004), pp. 1700–1704.

[5] Sang Min Nam et al. "Explanatory model of dry eye disease using health and nutrition examinations: Machine learning and network-based factor analysis from a national survey". In: *JMIR medical informatics* 8.2 (2020), e16153.

[6] Jay Shenoy et al. "R-SLAM: Optimizing Eye Tracking from Rolling Shutter Video of the Retina". In: *ICCV*. 2021.

[7] Andrea M Storås et al. "Artificial intelligence in dry eye disease". In: *The ocular surface* 23 (2022), pp. 74–86.

[8] Caiyong Wang et al. "Towards Complete and Accurate Iris Segmentation Using Deep Multi-Task Attention Network for Non-Cooperative Iris Recognition". In: *IEEE Transactions on Information Forensics and Security* 15 (2020), pp. 2944–2959.

[9] Norihiko Yokoi et al. "Rheology of tear film lipid layer spread in normal and aqueous tear–deficient dry eyes". In: *Investigative ophthalmology & visual science* 49.12 (2008), pp. 5319–5324.