TRACKING THE EYE - A MACHINE LEARNING APPROACH

ABSTRACT. We investigate a supervised machine learning approach to gaze tracking which utilises the SVR and Lasso regression algorithms. The test was run on 50 thousand sample points generated for the purpose of this work and 522 features, which include pupil coordinates, saliency maps and Haar features. Our results in some instances compare well with other theoretical endeavours, but are not convincing for possible practical applications.

1. MOTIVATION

Eye tracking has the potential to become a game changing technology for the way humans interact with devices. It has many applications ranging from user experience and human-computer interaction to more commercial areas such as user identification and behaviour tracking for ad personalisation. Geometrically speaking, eye tracking is a solvable problem that requires exact measurements of eye rotation, head rotation, and face-screen distance. In order to obtain these measurements, however, third-party hardware is required which limits the development and usability of web applications and software that could potentially take advantage of this technology.

On that backdrop we shall regard the problem as mapping a set of eyes to a position on the screen in terms of its pixel coordinates using only commercially available cameras. By constraining the domain to poor quality cameras, the problem becomes more difficult as the necessary measurements must be estimated from a set of images. As a remedy, we thus propose to use machine learning and image processing techniques as an approach to the problem.

2. EXISTING LITERATURE

Little has been done within academia that focuses directly on gaze prediction using an *appearance based* approach as compared to a *model based* approach. The latter takes advantage of third-party hardware to accurately predict the gaze while the former uses non-intrusive methods mainly through computer vision algorithms. While a great deal has been written on user attention tracking using the model based approach as in [1], [2] and [4] , only within the last two decades papers that uses raw image pixels as input have begun appearing. Notably, [11] trained a neural network using low level image features that achieved a mean error of 70px horizontally and

28px vertically for person *calibrated scoring*[1], and a mean error of 280px horizontally and 170px vertically for uncalibrated scoring[2]. In comparison, we implement the same features in traditional regression models, that is Lasso and Support Vector Regression as given in [5], and achieve a worse accuracy. That is to be contrasted with previous regression attempts in [9].

In terms of the low-level feature extraction, [3], [7] and especially [8] provided great reference and reliable results[3].

3. Approach

We collected a total of 50239 samples consisting of an image of a test subject and the gaze position on a 1440x800px display. The collection was done by asking subjects to track a square of 40x40px that moved across the screen during which the built-in 720p camera would take 20 images a second. The testing took place indoors in three different locations with three different types of lightning. Subjects were positioned within 0.3-2m from the screen and asked not to turn their heads under testing, and each subject contributed with close to 700 data samples. No particular instructions were given as to which part of the square to track which constitutes an element of noise on top of the occasional blinking and twitching.

The square movement followed a strict snake pattern to guarantee that the subject's eyes worked around the full screen which. This setup, in combination with the small amount of sample points per subject, renders the data useless for time series models. As a result, we get a uniform data distribution.

For the problem we defined the ϵ-insensitive loss function as the L_2-norm distance between the prediction and the label,

$$L = \max\{||h(x_i) - y_i||^2 - \epsilon, 0).$$

Based on the resolution of our training data we set $\epsilon = 30$ which approximates the test square with which the data was collected. The loss function is clearly bounded by the screen size.

4. Features and Preprocessing

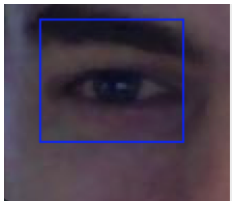
Lending from [11] the features we extracted were 1) pupil positions, 2) two rectangular Haar features, and 3) saliency map. In this respect, we used an image processing pipeline that, for each image, applied the standard face

[1]In their paper they denote this as *person dependent* scoring.

[2]The terms *person calibrated* and *uncalibrated* scoring refers to whether or not the model has been trained on images from the person that is used as the test subject.

[3]All code is available https://github.com/sherl0ck-/eye_track

FIGURE 1. An example of the eye area from which features were extracted.



detection algorithm from OpenCV[4] to narrow down our region of interest to one area around each eye. An example is given in Figure 1.

4.1. **Pupil Features.** To extract pupil positions, we drew on the work of [6] by building on an existing implementation[5]. In simplicity, the algorithm computes the gradient of the image and searches for the most perfect circle which, in the eye region, turns out to be the pupil. The pupil positions were then given in their x and y coordinates relative to the eye region.

4.2. **Haar Features.** Haar features are simple rectangular primitives that takes advantage of intensity patterns in images as explained in [7]. One Haar feature consists of the sum of pixel intensity differences across to adjacent rectangles. For each eye, we computed one vertical Haar feature and three horizontal Haar features as outlined in Figure 2. Presumably these low-level features encode information on the position of the iris by giving away how much of the eye whites that are covered and can thus aid the gaze-tracking. The Haar features were computed anchored around the pupil positions on an image of the eye blurred and decimated by a factor of 2.

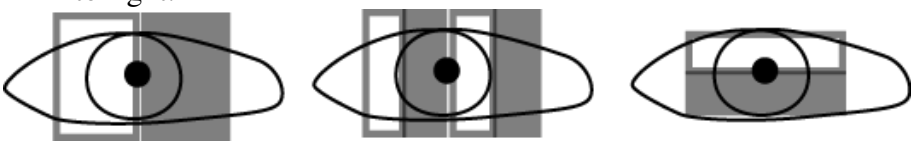
4.3. **Saliency Map Features.** To compute the saliency maps we produced an intensity map, two color maps, and four orientation maps as per [8]. The intensity map consisted of greyscale intensities; the color map, of 2 matrices for red-green and blue-yellow color opponencies respectively. The orientation maps were generated by convolving the image with a Gabor filter for different orientations θ, $\theta \in (0, 180)$, yielding 4 different maps, each emphasising the features that were positioned on the respective angle with the horizontal. The Gabor filter is a linear filter defined by:

$$G_\psi(x, y, \theta) = \exp(-\frac{x'^2 + \gamma^2 y'^2}{2\delta^2}) \cos(2\pi \frac{x'}{\lambda} + \psi)$$

[4]The algorithm is based on Haar cascades and is trained on AdaBoost; more on Haar features in a minute. The OpenCV reference can be found here http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html.

[5]A simple straight-forward implementation of Timm's algorithm was taken from here https://github.com/trishume/eyeLike.

FIGURE 2. The Haar features extracted from the eye images consist of three horizontal and one vertical feature, the former given to the left and in the center and the latter portrayed to right.



Following the methodology from the mentioned papers and [3], we combined these into saliency maps. This was done first by computing the centre-surround across-scale difference, which was normalised by convolution with the difference of Gaussians (DoG) filter. The centre-surround difference was computed across 8 directions $[0, \pi/4...7\pi/4]$, and the minimum was taken, as per [10]. This was performed with square sizes of 3 and 7 for the center and the surround area, respectively. We ran into issues with the centre-surround difference implementation, which waterfalled into other parts of the image processing part. This is why, in the end, we used the SaliencyToolbox from [8] to obtain the saliency maps through a Matlab script.

5. TRAINING

We fed an input consisting of the preprocessed features into the Lasso and the SVR model. Admitting no natural use of kernels, Lasso is a linear regressor that took a minimal amount of time to run and one which promised favorable guarantees. In terms of the generalisation error, it adds a $\sqrt{\log(N)}$ term, which is favorable for the implementations like ours. In the training stage, it is able to select its main parameter, α (also denoted as λ), per [5] to minimize the mean squared error on the training sample. Generally speaking, Lasso selects the number of features in an inversely proportional fashion with respect to α. Such an implementation of coordinate descent algorithm usually converges after a thousand or so iterations. However, we anticipated that the linear fitting would not be sufficient for the problem at hand.

For the Support Vector Regression implementation, we searched for parameter C near the maximum label value, as per [5]. We experimented with the (Gaussian) Radial Basis Function (RBF) kernel and did not experience over-fitting or any problem that would point to the kernel choice. For more optimal results, it could have been beneficial to chose the kernel based on optimal cross validation results. However, this was unfeasible given our time constraint, as the full run would usually take circa 5h to complete.

FIGURE 3. Table overview of the different training settings.

Calibrated	SVR	Cantor
		Ensemble
	Lasso	Cantor
		Ensemble
Uncalibrated	SVR	Cantor
		Ensemble
	Lasso	Cantor
		Ensemble

Selecting ϵ came naturally from our experimental setup and we gave the tightest bound to aim at the precision defined by the 40x40px square. Relaxing this bound and seeing how the algorithm performs on the training set is something we are interested in and plan to test soon after submission of this report.

Since our label space is two-dimensional, in one instance we transformed the x, y labels into a single dimensional space via the *Cantor pairing function*, while in the other we trained a model for the x labels and a model for the y labels in an *ensemble fashion*. The Cantor pairing function is a bijective mapping, $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ defined by

$$\pi(x, y) := \frac{1}{2}(x + y)(x + y + 1) + y.$$

Set up this way, it allowed for unique mapping and retrieval of x and y with the clear distinction between the two coordinates. It was computationally inexpensive (constant time for both encoding and decoding), with the down-side of increasing the range of label values (maximum value was around 2.5 million, or twice the widthxheight). This made tuning C by cross-validation slightly harder.

Although we conjectured that the Cantor pairing models would yield better results by maintaining the correlation between x, y, the ensemble methods, as we shall see, overall performed better. All models were run on person-calibrated data and uncalibrated data. In the former case the model was tested on data from subjects who had already been seen in the training data. In the latter case, the test subjects were novel to the model. See Figure 3 for an overview.

Generally, we maintained a 3:1 ratio between training and test set cardinality. Having mapped the label space Y taking values in $[0, 2.49M]$, we seek a hypothesis $h : X \to Y$ from which we can deduce the labels by inverting the Cantor pairing function. Let $y' \in Y$, then (x, y) can be found

FIGURE 4. Table over the results. *These results were obtained with only a saliency for one eye and not two eyes due to time constraints.

	Accuracy (%)	**Mean Loss (px)**
Person Calibrated		
Lasso Cantor	0.275	591.503
Lasso Cantor Ensemble	0.558	317.254
SVR Cantor	0.235	549.893
SVR Cantor Ensemble	0.235	276.835
Uncalibrated		
Lasso Cantor	0.128	624.585
Lasso Cantor Ensemble	0.443	359.501
SVR Cantor	0.150	600.437
SVR Cantor Ensemble*	0.180	392.613

by letting

$$w = \left\lceil \frac{\sqrt{8z+1}-1}{2} \right\rceil \quad \text{and} \quad t = \frac{w^2+w}{2},$$

Then $y = z - t$ and $x = w - y$. For the ensemble approach, we trained two models separately, where one predicted the x label and the other (independently) predicted the y-coordinate of the labels.

6. RESULTS

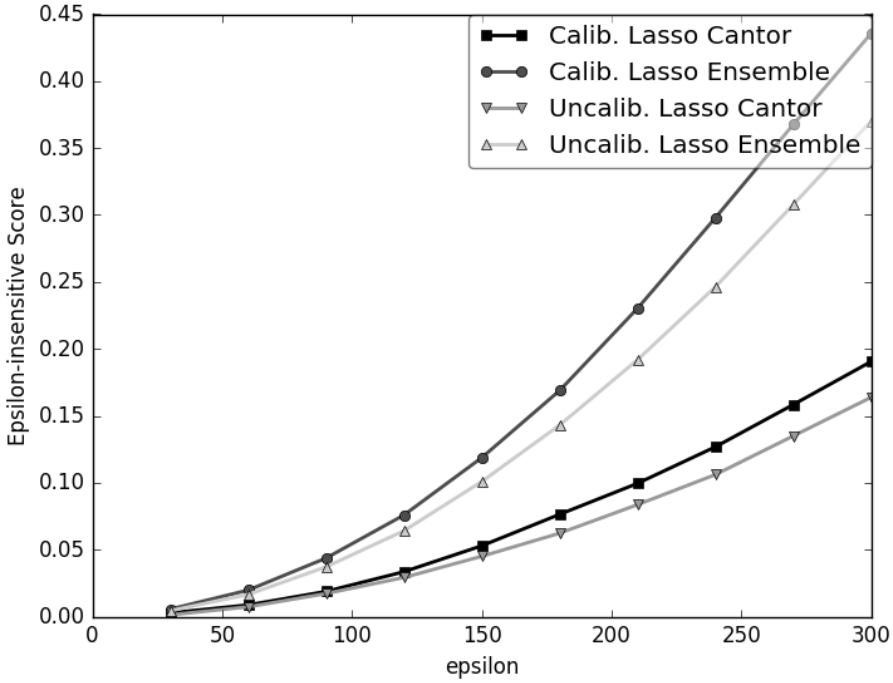
The results of the runs are reported in Figure 4. As seen, our models only performs slightly better than the random baseline which has an accuracy of 0.14 percent[6]. As expected the person calibrated models achieve roughly double the accuracy of the uncalibrated scoring, except from the Lasso Cantor ensemble model which does almost as good uncalibrated as calibrated. Moreover, the Lasso generally outperforms the SVR model by a factor of 2. Consistent with theory, for the relatively large number of samples points, the SVR also takes much longer to train, around 5 hours, while the Lasso only takes a couple of minutes.

Mimicking the features of [11], except from the spatiogram feature, the regression models do comparatively worse on the person calibrated tests. The paper achieves a mean error distance of 148px in a euclidean space[7]

[6]The random baseline was calculated by dividing the area of pixels of the blue test square with the total area of the screen, which makes up the probability that a random guesser would pick a point therein.

[7]The paper gives the error measured in terms of the eye angle and provide a transformation into pixel distance of 0.5 corresponding to 20 pixels. Moreover, we combine the horizontal and vertical errors into one by taking the root of the sum of the squares.

FIGURE 5. An overview of how the accuracy increases as the ϵ-sensitive loss allows for a higher error margin.



while our models achieve no loss smaller than 290px. However, for the uncalibrated scoring, the paper presents a mean error distance of 654px which our Lasso Ensemble implementation beats with a mean distance error of 379px. While these results are not completely comparable as Zhang uses data collected in a strictly controlled setting, our results suggest nonetheless that: 1) the Lasso model with our features offers a decent alternative to the regression neural network and 2) there is space for improvement on the feature extraction and selection to improve on the accuracy of the predictions.

If we relax our error requirements to outside the 40x40px area in determining the loss on the test set, Figure 5 portrays how the accuracy increases for the Lasso Cantor and the Lasso Ensemble model in both the calibrated and uncalibrated scenario. As seen, they nearly predict half of the gazes correctly, which corresponds to predicting which side of the screen the subject is looking at. This is hardly impression and suggests that either the features have to be revamped or that neural networks, more specifically convolutional networks, works better on predictions uses images than traditional regression models.

7. CONCLUSION

In summary, we tested SVR and Lasso algorithms on the uniformly distributed dataset of 50 thousand samples and with 522 features each. We ran each in person-dependent and person-independent fashion. Feature extraction involved preprocessing to obtain pupil positions, saliency maps and Haar features. The results suggest that these algorithms provide a good alternative to neural network approach, but remain outside of accuracy bounds for practical applications. Further work will be done to improve upon the feature selection and test online counterparts of the algorithms run.

REFERENCES

[1] G. Buscher, E. Cutrell, and M. R. Morris. What do you see when you're surfing?: using eye tracking to predict salient regions of web pages. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 21–30. ACM, 2009.

[2] L. A. Granka, T. Joachims, and G. Gay. Eye-tracking analysis of user behavior in www search. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 478–479. ACM, 2004.

[3] L. Itti and C. Koch. Feature combination strategies for saliency-based visual attention systems. *Journal of Electronic imaging*, 10(1):161–169, 2001.

[4] T. Judd, K. Ehinger, F. Durand, and A. Torralba. Learning to predict where humans look. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2106–2113. IEEE, 2009.

[5] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

[6] F. Timm and E. Barth. Accurate eye centre localisation by means of gradients. *VISAPP*, 11:125–130, 2011.

[7] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[8] D. Walther and C. Koch. 2006 special issue: Modeling attention to salient proto-objects. *Neural Networks*, 19(Brain and Attention):1395 – 1407, 2006.

[9] O. Williams, A. Blake, and R. Cipolla. Sparse and semi-supervised visual mapping with the s3gp. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 230–237, June 2006.

[10] K. Xie, K. Fu, T. Zhou, J. Zhang, J. Yang, and Q. Wu. Small target detection based on accumulated center-surround difference measure. *Infrared Physics and Technology*, 67:229 – 236, 2014.

[11] Y. Zhang, A. Bulling, and H. Gellersen. Towards pervasive eye tracking using low-level image features. In *Eye Tracking Research and Applications Symposium (ETRA)*, number Proceedings - ETRA 2012: Eye Tracking Research and Applications Symposium, pages 261–264, Lancaster University, 2012.