WebGazer: Scalable Webcam Eye Tracking Using User Interactions

Alexandra Papoutsaki
Brown University
alexpap@cs.brown.edu

Patsorn Sangkloy[*]
Georgia Institute of Technology
psangkloy3@gatech.edu

James Laskey
Brown University
jlaskey@cs.brown.edu

Nediyana Daskalova
Brown University
nediyana@cs.brown.edu

Jeff Huang
Brown University
jeff@cs.brown.edu

James Hays[*]
Georgia Institute of Technology
hays@gatech.edu

Abstract

We introduce WebGazer, an online eye tracker that uses common webcams already present in laptops and mobile devices to infer the eye-gaze locations of web visitors on a page in real time. The eye tracking model self-calibrates by watching web visitors interact with the web page and trains a mapping between features of the eye and positions on the screen. This approach aims to provide a natural experience to everyday users that is not restricted to laboratories and highly controlled user studies. WebGazer has two key components: a pupil detector that can be combined with any eye detection library, and a gaze estimator using regression analysis informed by user interactions. We perform a large remote online study and a small in-person study to evaluate WebGazer. The findings show that WebGazer can learn from user interactions and that its accuracy is sufficient for approximating the user's gaze. As part of this paper, we release the first eye tracking library that can be easily integrated in any website for real-time gaze interactions, usability studies, or web research.

1 Introduction

Eye tracking is a common method for understanding human attention in psychology experiments, human-computer interaction studies, medical research, etc. Typical eye trackers use an infrared video camera placed at a fixed distance from the user, require explicit calibration and setup, and cost thousands of dollars. Thus, the use of eye tracking technology has primarily been confined to specialized labs with artificial tasks. In essence, current eye trackers relinquish naturalistic studies to more scalable technologies such as web analytics.

Eye tracking using consumer webcams and offline software has been studied before and unsurprisingly has been found to be less accurate than specialized equipment, negating its utility in professional studies. However, several technological advancements have recently arrived that justify webcams as practical technologies. Over 65% of web browsers support the HTML5 functions for accessing the webcam [Deveria, 2015], typical laptop webcams support higher resolutions of capture, and modern computers are fast enough to do eye tracking for video in real time. Nevertheless, these advancements do not solve the problem of poor accuracy due to diverse local environments and human features.

We designed WebGazer, a new approach to browser-based eye tracking for common webcams. WebGazer aims to overcome the accuracy problems that webcams typically face, by adopting user interactions to continuously self-calibrate during regular web browsing. Webcam images during these user interactions can be collected and used as cues for what the user's eyes look like. Future observations of the eyes can be matched to similar past instances as WebGazer collects mappings of eye features to on-screen gaze locations, allowing inference of the eye-gaze locations even when not interacting.

The base WebGazer technology is compatible with three open-source eye detection libraries for locating the bounding box of the user's eyes. The eye detectors that are evaluated in this work are clmtrackr, js-objectdetect, and tracking.js, but it can be generalized to include others. There are two gaze estimation methods in WebGazer, one which detects the pupil and uses its location to linearly estimate a gaze coordinate on the screen, and a second which treats the eye as a multi-dimensional feature vector and uses regularized linear regression combined with user interactions. WebGazer goes beyond using only clicks as user interaction data, to also applying the cursor movements and the eye gaze-cursor coordination delay as modifiers to the basic gaze estimation model. This is where understanding user behavioral habits is helpful in constructing the model.

We evaluate WebGazer through a remote online study with 76 participants and an in-lab study with 4 participants to compare with a low cost commercial eye tracking device. In the online study, we find that two of our regression models outperform existing approaches with an error of 175 and 210 pixels respectively. Compared to the commercial eye tracking device we discover comparable mean errors with an average visual angle of $4.17°$. This demonstrates the feasibility of WebGazer to approximate the gaze in diverse environments.

The two main contributions of this work are: 1) the research, development, and evaluation of a real-time browser-based auto-calibrated webcam eye tracker, WebGazer (with source code and documentation publicly available at `https://webgazer.cs.brown.edu`), 2) investigations of different gaze estimation models enhanced by user interactions.

[*] Research conducted while at Brown University

2 Related Work

2.1 Webcam Eye Tracking

There have been few published attempts at using *webcams* for eye tracking. Such methods typically involve an explicit calibration phase and are unsurprisingly less accurate than infrared eye trackers [Hansen and Pece, 2005]. One of the first appearance-based methods used video images of mobile cameras to train neural networks [Pomerleau and Baluja, 1993]. [Lu *et al.*, 2011] introduced an adaptive linear regression model which needs sparse calibration but is sensitive to head movement. [Lu *et al.*, 2012] overcame this using synthetic images for head poses, but need extensive calibration.

Another line of research uses image saliency to estimate user gaze for calibration purposes [Sugano *et al.*, 2010], but saliency is only a rough estimate of where a user is looking. [Alnajar *et al.*, 2013] introduced a webcam eye tracker that self-calibrates with pre-recorded gaze patterns, but still require users to view "ground truth" gaze patterns. PACE is a desktop application that performs auto-calibrated eye tracking through user interactions [Huang *et al.*, 2016]. TurkerGaze [Xu *et al.*, 2015] is a webcam based eye tracker deployed on Amazon Mechanical Turk that predicts saliency on images. WebGazer differs from these implementations by being the first browser-based model to self-calibrate on real time via gaze-interaction relationships which are readily available. WebGazer can be incorporated into any website to perform gaze tracking almost instantaneously and without inhibiting the user's natural behavior through explicit calibration.

2.2 Gaze-Cursor Relationship

Lab studies involving eye tracking during web browsing have been commonly used to track visual attention, and our user interaction model partially builds on these findings. Past research has found a correlation between gaze and cursor positions [Chen *et al.*, 2001; Guo and Agichtein, 2010; Liebling and Dumais, 2014; Rodden *et al.*, 2008]. Cursor movements can determine relevant parts of the web page with varying degrees of success [Shapira *et al.*, 2006]. [Buscher *et al.*, 2008] used eye tracking features to infer user interest and show that this can improve personalized search. [Buscher *et al.*, 2009] demonstrated the value of gaze information for building models that predicted salient regions of web pages, while [Cole *et al.*, 2011] built reading models operating on eye tracking data to investigate information acquisition strategies.

Another line of research primarily asks whether cursor movements can be used as a substitute for eye tracking, but not enhance it. An early study by [Chen *et al.*, 2001] showed that the distance between gaze and cursor was markedly smaller in regions of the page that users attended. [Rodden *et al.*, 2008] and [Guo and Agichtein, 2010] identified a strong alignment between cursor and gaze positions and found that the distance between cursor and gaze positions was larger along the x-axis. [Lagun and Agichtein, 2015] created a model that combines user interactions and saliency to predict searchers' attention. Inferring the gaze location using webcams can be more powerful than the above approaches as the cursor can be idle for long periods. Webcam video frames can inform gaze prediction algorithms even when no user interactions take place.

3 WebGazer

We develop WebGazer, a self-calibrating client-side eye tracking library written entirely in JavaScript that trains various regression models that match pupil positions and eye features with screen locations during user interactions. WebGazer can predict where users look within any device with a browser that supports access to the webcam. A few lines of JavaScript can integrate WebGazer in any website and perform eye tracking once the user starts using the web page naturally.

WebGazer is relatively simple from a computer vision point of view—it has no explicit 3D reasoning as found in more sophisticated trackers [Hansen and Ji, 2010]. This simplicity allows it to run in real time through browser JavaScript. Lack of 3D reasoning would typically make brittle predictions, but the primary novelty of WebGazer is being able to constantly self-calibrate based on cursor-gaze relationships. Not only does this eliminate the need for initial calibration sessions, but it means users are free to move and WebGazer will learn new mappings between pupil position, eye features, and screen coordinates.

As WebGazer is agnostic about face and eye detection algorithms, we incorporated and evaluated it using three different facial feature detection libraries: clmtrackr [Mathias, 2014], js-objectdetect [Tschirsich, 2012], and tracking.js [Lundgren *et al.*, 2014]. All three implement different vision algorithms in JavaScipt. Js-objectdetect and tracking.js detect the face and eyes and return rectangles that enclose them. Instead of using the whole video frame we first perform face detection for finer-scale eye detection on its upper half, speeding up gaze prediction and suppressing false positives. Clmtrackr performs a more realistic fitting of the facial and eye contour. To provide consistent input for WebGazer, we use the smallest rectangle that fits the eye contour.

3.1 Pupil Detection

Having detected the eye regions, WebGazer next identifies the location of the pupil, making three assumptions: i) the iris is darker than its surrounding area, ii) it is circular, and iii) the pupil is located at its center. These assumptions are not always true but in practice they hold often enough to get real time results with reasonable accuracy. To identify the pupil, we search over all offsets and scales for the region with the highest contrast to its surrounding area. This exhaustive search is made efficient by using a summed area table or integral image.

3.2 Eye Features

The pupil location as a 2D feature can potentially fail to capture the richness of the eye's appearance. An alternative is to *learn* a mapping from pixels to a gaze location. For this we extend TurkerGaze [Xu *et al.*, 2015] and represent each eye as a 6×10 image patch. We follow up with grayscaling and histogram equalization, resulting with a 120D feature that is input into the linear regression algorithms described below. TurkerGaze uses only clmtrackr but we apply these steps to all three eye detection libraries. Unlike TurkerGaze, WebGazer does not require users to stare at calibration points nor remain motionless. It also does not perform offline post-processing as its goal is live operation instead of image saliency prediction.

3.3 Mapping to Screen and Self-Calibration

To match the pupil locations and eye features to screen coordinates, we must find a mapping between the 2D and 120D vectors respectively and the gaze coordinates on the device. This relationship is complex—it depends on the 3D position and rotation of the head with respect to the camera and screen, requiring careful calibration and expensive computation. We instead rely on continual self-calibration through user interactions that normally take place in web navigation.

For the self-calibration, we assume that when a user interaction takes place then the gaze locations on the screen match the coordinates of that interaction. [Huang *et al.*, 2012] showed that the gaze-cursor distance averages 74 pixels the moment a user clicks. Since that distance can be task-dependent, we simplify our analysis by assuming that the gaze and cursor align perfectly during clicks. This assumption is general enough to allow any website to use WebGazer for diverse environments and tasks. Here we focus on clicks and cursor movements but WebGazer can be extended to other types of user interactions.

Mapping Pupil Coordinates

Based on the previous assumption, we get a series of training examples. Without loss of generality, we examine the x-axis estimation case. We obtain N pupil location training examples $\mathbf{x} = (x_1, ..., x_N)$ and their corresponding click observations on the display $\mathbf{t} = (D_{x1}, ..., D_{xN})$. These are considered true gaze locations. Using a simple linear regression model, we obtain a function $f(\mathbf{v}) \to D_x$ which given a pupil feature vector $\mathbf{v}$ predicts the location of the gaze on the screen along the x-axis. The function is $f(\mathbf{v}) = \phi(\mathbf{x})^T \mathbf{w}$ where $\phi(\mathbf{x})$ is a basis function and $\mathbf{w}$ is a vector of weights that satisfy:

$$\underset{\mathbf{w}}{\text{minimize}} \sum_{x_i \in \mathbf{x}} ||D_{xi} - f(x_i)||_2^2 \tag{1}$$

Mapping Eye Features

To match eye pixels to gaze locations, we implement a ridge regression (RR) model [Hoerl and Kennard, 1970] that maps the 120D eye feature vector to the display coordinates (D_x, D_y) for each click. With just a few clicks, this regularized linear regression can produce relatively accurate predictions. In addition, its linearity keeps it simple and avoids overfitting due to the regularization, leading to fast-performing evaluation at run time.

Again without loss of generality, we consider the ridge regression function for the x-coordinate prediction: $f(\mathbf{v}) \to D_x$. This function is also $f(\mathbf{v}) = \phi(\mathbf{x})^T \mathbf{w}$ and again depends on a vector of weights $\mathbf{w}$ which is estimated as:

$$\underset{\mathbf{w}}{\text{minimize}} \sum_{x_i \in \mathbf{x}} ||D_{xi} - f(x_i)||_2^2 + \lambda ||\mathbf{w}||_2^2 \tag{2}$$

The last term λ acts as a regularization to penalize overfitting. Here, we set λ to 10^{-5}, the same value that the authors of TurkerGaze used in their model.

The calculation of the weight vector, in matrix notation is:

$$\mathbf{w} = (X^T X + \lambda I)^{-1} X^T Y \tag{3}$$

where X is the design matrix of eye features and Y is the response vector of display coordinates.

Next, we build on the ridge regression model using research on human vision and on the nature of user interactions.

Extra Samples within a Fixation Buffer

Human vision is governed by different types of eye movements that when combined, allows us to examine and perceive targets within our visual field. The two major types of eye movements are saccades and visual fixations, where eyes stabilize on a specific area for an average of 200–500ms [Rayner, 1998]. Perceiving information is activated during fixations, thus they are traditionally used to gain insights into human attention.

In this study, we use the above concepts to inform the ridge regression model. We extend our assumption that gaze and cursor positions align when users click with the assumption that a fixation has preceded the click. Given that, we keep a temporal buffer that stores all eye features within 500ms. When a click occurs, we examine in increasing temporal order the predicted gaze coordinates against the ones corresponding to the moment of the click. Consequently, we add all predictions that occurred within 500ms and at most 72 pixels away from the predicted gaze locations at the moment of the click to the regression. These samples can potentially enrich the accuracy of the predictions made by the ridge regression.

Sampling Cursor Movements

Different studies have shown that there is a strong correlation between cursor and gaze locations when users move their cursor intentionally, e.g. to click on a target [Hauger *et al.*, 2011]. When the cursor remains idle though, the distance between them grows, making it a good signal only when active.

In our research, we explore the applicability of introducing cursor behavior in the ridge regression model (RR+C). We alter the ridge regression model by adding weights to the samples by introducing to Equation 3 the diagonal matrix K that contains the weights for each sample along the diagonal:

$$\mathbf{w} = (X^T K X + \lambda I)^{-1} X^T K Y \tag{4}$$

We keep the same assumption as before: gaze and cursor locations align when clicks occur. Click events are given a full unit weight. Every time the cursor moves, we assign to its corresponding position a weight of 0.5 and assume it corresponds to the predicted gaze location. We decrease the weight of each cursor position by 0.05 every 20ms. This allows a cursor location to contribute to the regression model for at most 200ms, a duration comparable to a fixation. Thus, when the cursor is idle, this model falls back to the original simple ridge regression (RR) where only clicks train WebGazer.

Combining Fixations and Cursor Movements

We also explore the combination of the two last models (sampling within a fixation buffer and sampling cursor movements) with the simple ridge regression (RR+F+C). As the evaluation of WebGazer is based on the moments that clicks occur, we wanted to have a more accurate and enhanced model that would provide predictions even when the cursor remains idle. As such, we build a regression model that matches gaze with click locations, includes extra samples within a fixation buffer, and uses cursor movements only when the cursor is moving.

4 Experiment Design

4.1 Remote Online Large-Scale Study

We conducted a remote online user study to evaluate the accuracy and feasibility of performing eye tracking with WebGazer.

A consent form was presented to the participants and a compatibility test detected if their browsers could access the webcam. Upon agreement, they were assigned two types of tasks. WebGazer was integrated in all task pages and each user was uniquely identified. All parameters were reset between pages.

The first type of tasks emulated reading and interacting with content, two typical behaviors in web pages. Participants had to fill a quiz with 40 yes/no questions. The answers could be given by selecting one of two radio buttons per question. Each row included 3 questions that spanned across the whole width of the display and resulted in a grid of 14 rows.

The second type of tasks included selecting a target as part of a standard Fitts' Law study using the multidirectional tapping task suggested by the ISO9241-9 standard [Soukoreff and MacKenzie, 2004] which measures usability. Participants had to click on a circular target that appeared in 11 locations on a circular grid as seen in Figure 1. The active target would be shown in red color while the 10 inactive targets were gray. The distance between two consecutive locations of the active target was 512 pixels, while its radius was 12 pixels. For each target selection task, participants had to successfully click on the red target 40 times. Note that Figure 1 is a composite image demonstrating the facial feature detection and predictions made by different regression models. The camera video and the predictions were never shown to the participants.

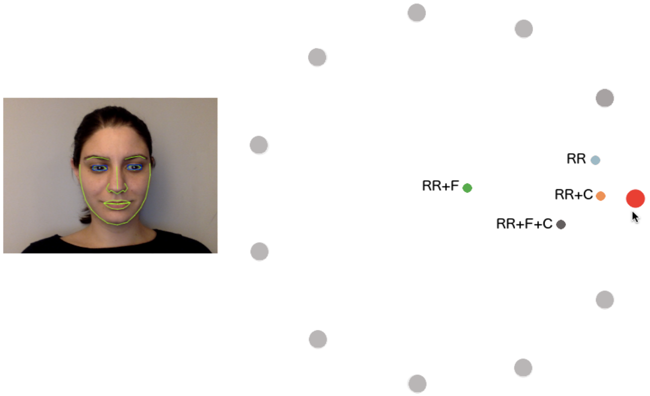

Figure 1: Composite image demonstrating the experimental setup for the target selection task. Clmtrackr is used for facial feature detection. Predictions from different regression models are depicted with different colors. Users aim at the red target.

For both types of tasks, eye detection was performed with one of the following facial feature detection libraries: clmtrackr, js-objectdetect, and tracking.js. This resulted in six trials, as both tasks were assessed using the three eye detection libraries. Each trial was introduced with a verification page showing instructions for the upcoming task along with the video captured by the users' webcam so that they could adjust their position and ambient lighting to ensure that their face, eyes, and pupils were correctly captured. The quiz tasks always preceded the target selection tasks. The order of the facial feature detectors was uniformly and randomly selected.

After all the trials were completed, participants filled a short demographic questionnaire inquiring their age, gender, handedness, vision, any feedback, and optionally their emails so that they could enter a raffle. Participants were free to move and no chin-rest was used. This approach differs from the traditional practices in research employing eye tracking as it allows users to use eye tracking at the convenience of their own space and while behaving naturally.

Participants

82 participants (40 female, 42 male), mainly college students and young professionals, were recruited through university-wide mailing lists, and entered a raffle for ten $50 Amazon gift cards. They were between 18 to 42 years old (*M* = 25.6, *SD* = 4.2); 39 had normal vision, 25 wore glasses, and 18 wore contact lenses. All participants used Google Chrome or Firefox. The experiment lasted an average of 9.9 minutes.

Six participants were excluded due to technical issues. Overall there were 20,251 clicks, 18,657 predictions for the simple linear regression and 19,545 for each ridge regression model.

4.2 In-Person Small-Scale Study

WebGazer's ability to infer the gaze location is based on the assumption that the gaze and cursor locations match during a click. To evaluate this claim and assess the accuracy of WebGazer throughout the interaction of a user with a page, we ran a smaller-scale study to gain a better insight on webcam eye tracking and how it compares to commercial eye trackers. We repeated the same procedures as with the remote large-scale user study, but in a lab using Tobii EyeX, a 50Hz commercial interaction eye tracker primarily used for development of interactive applications. We recorded the predictions made by Tobii EyeX and WebGazer throughout the duration of the user study even when clicks were not occurring.

The experiment was conducted with a chin rest on a desktop PC with Windows 7 and Google Chrome in a maximized window. A Samsung SyncMaster 2443 24-inch monitor with a resolution of 1920 by 1200 pixels was used with a Logitech Full HD C920 USB webcam, from a distance of 59cm from the user. We recruited 5 college students (2 female, 3 male) that performed the same study as described earlier. Their ages ranged from 19 to 30 years (*M* = 23, *SD* = 4.3). Four had normal vision, and one wore contact lenses. The study lasted 7.2 minutes on average.

One participant was excluded due to technical problems. Overall there were 962 clicks with 802 predictions derived from the simple linear regression model and 866 from all the rest. Predictions were tracked throughout the study, even when the user would not interact with the page.

5 Results

5.1 Evaluating Predictions From Online Study

To measure the performance of WebGazer, we compute the Euclidean distance between the location of a click and the corresponding predicted gaze location. This distance is measured in pixels as we cannot control the positioning of the online users or know the resolution of their monitors.

As part of the study we required that for a user to complete a task, they would have to perform at least 40 clicks. This number increases when accidental clicks happen. We normalize the results across all participants and map them to 50 clicks.

	Quiz/ Clmtrackr		Target/ Clmtrackr		Quiz/ Js-object.		Target/ Js-object.		Quiz/ Tracking.js		Target/ Tracking.js		All Tasks/ Libraries	
Model	***M***	***SD***	***M***	***SD***	***M***	***SD***	***M***	***SD***	***M***	***SD***	***M***	***SD***	***M***	***SD***
Linear	245.4	82.5	227.9	37.5	271.2	82.8	230.0	36.1	311.9	112.2	260.7	29.6	256.9	75.0
RR	207.6	87.5	165.8	71.7	248.5	80.8	197.3	58.8	309.8	111.7	275.1	47.1	232.4	92.3
RR+F	247.4	91.3	207.4	78.7	257.8	87.1	218.6	65.4	308.1	111.9	275.7	49.1	251.5	89.0
RR+C	118.7	55.4	104.5	55.1	167.3	67.8	160.7	54.6	258.7	115.8	251.8	55.8	174.9	91.6
RR+F+C	180.8	75.5	157.0	69.7	208.7	82.5	206.1	61.3	263.0	114.2	255.4	52.4	210.6	86.3

Table 1: Mean (*M*) and standard deviation (*SD*) of prediction errors measured in pixels across all regression models and combinations of tasks and libraries. The mean is obtained from averaging across the 76 online participants.

Simple Linear vs. Ridge Regression

First, we compare the accuracy of the simple linear regression model that maps the pupil location to display coordinates against the ridge regression model that maps the 120D eye feature vector. Across all clicks, the mean error from the linear regression model is 256.9 pixels (*SD* = 75.0). Similarly, the mean distance between the location of a click and the prediction made by the ridge regression model is 233.4 pixels (*SD* = 92.3). The means and standard deviations of each combination of task type and eye detection library are reported in Table 1.

We average the error across all clicks for each participant. A Mann-Whitney U test showed that the mean error was greater for the simple linear regression ($p < 0.005$).

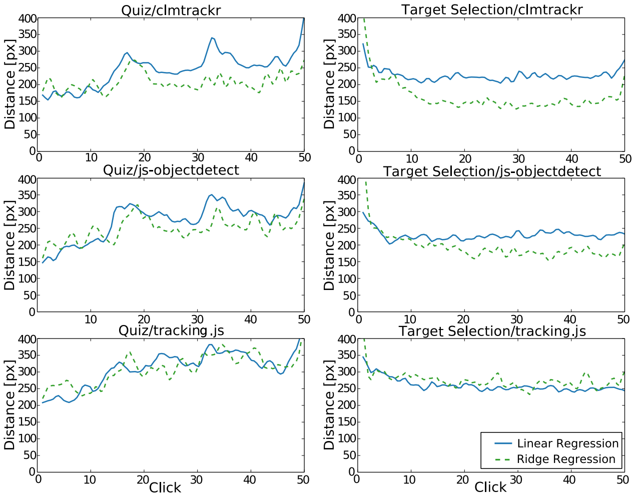

Figure 2: Average Euclidean distance in pixels between the click location and the predictions made by the simple linear (solid blue) and the ridge regression model (dashed green).

Figure 2 shows the average Euclidean distance in pixels across all 50 normalized clicks. We observe different error trends across the two task types. Filling the quiz seems to introduce more error with more clicks, perhaps because users need to scroll to reach all questions and thus they move more. On the other hand, when selecting targets, the error drops until it stabilizes—no scrolling happens in this type of task.

As ridge regression has generally a lower error, we base our subsequent analysis only on the ridge regression model that matches eye feature vectors to display coordinates.

Comparison of All Ridge Regression Models

We compare the accuracy of all prediction models that use ridge regression: the simple ridge regression (RR), the regression when adding extra samples within the fixation buffer (RR+F), when sampling cursor activity outside of clicks (RR+C), and when combining all the above (RR+F+C). Figure 3 shows the average Euclidean distance in pixels across all clicks for all combinations of tasks, libraries, and regression models. Again we observe the same upward trend for the quiz task across all prediction models. On the other hand, for the target selection task, we notice that for the clmtrackr and js-objectdetect detection libraries, the error decreases during the first half of the task and increases during the second half. Performing the study online leaves room for the interpretation of the observed variability. The speed of the external libraries can have a significant effect on WebGazer's ability to match correctly frames and locations on screen. Head movement can also affect the detected images that correspond to eye patches.

Note that WebGazer achieves a significant increase in the accuracy of the predicted gaze. Sampling cursor activity (RR+C) has the smallest average error of 174.9 pixels, followed by the model that combines fixations and cursor activity (RR+F+C) with an error of 210.6 pixels. Contrary to our expectations, the error of WebGazer using extra samples within a fixation buffer (RR+F) increased (*M* = 251.5 pixels). There are a couple of factors that could have an effect, e.g. blinks within the fixation buffer or temporal and spatial ranges being too lenient. The number of extra samples within a fixation buffer also depended on the performance of the eye detection library in conjunction with the increased cost of extra training data.

5.2 In-Person Study Results

The in-person user study data was captured via log files from the Tobii EyeX eye tracker and server logs for WebGazer. Both datasets were parsed down to a time series of predictions and grouped into 10 millisecond bins. The error was computed as the average distance between each regression and Tobii EyeX for the equivalent bin. Tracking.js data were excluded due to performance issues running both eye trackers simultaneously.

Table 2 contains the mean prediction errors and standard deviations measured in pixels for all 4 participants. We note that the mean errors reported here are comparable with the ones in Table 1 from the online study. The errors are naturally higher as the population size differs significantly. The mean errors are computed even when the user is not clicking. That further supports our assumption for matching gaze and click coordinates.

	Quiz/ Clmtrackr		**Target/ Clmtrackr**		**Quiz/ Js-object.**		**Target/ Js-object.**		**All Tasks/ 2 Libraries**	
Model	***M***	***SD***	***M***	***SD***	***M***	***SD***	***M***	***SD***	***M***	***SD***
RR	214.7	170.6	173.8	140.7	286.7	201.2	212.7	132.0	226.7	175.3
RR+F	234.7	190.3	189.3	146.4	269.5	192.5	205.3	135.9	231.3	178.8
RR+C	128.9	115.6	152.4	139.0	197.1	159.0	195.9	133.6	169.0	147.1
RR+F+C	164.0	156.7	177.6	160.1	200.1	150.7	197.4	137.1	187.4	159.3

Table 2: In-person study mean (*M*) prediction errors and standard deviation (*SD*) measured in pixels across all regression models and tasks against the EyeX eye tracker. Only clmtrackr and js-objectdetect are reported.

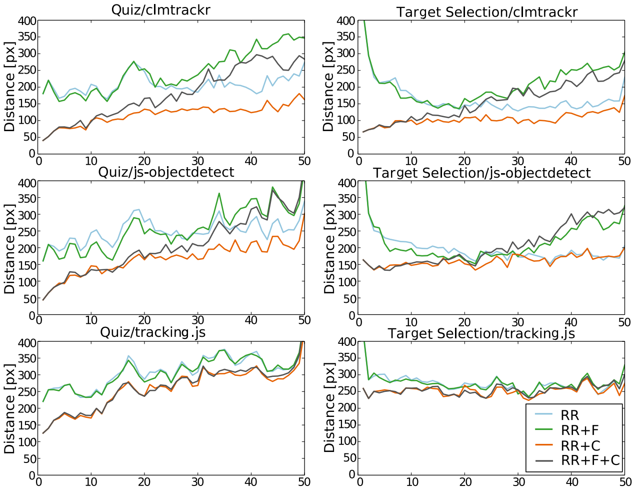

Figure 3: Average prediction error in pixels made by the ridge regression model (blue), with extra sampling within fixation buffer (green), with sampling cursor activity (orange), and the combination of all three (black).

The two models with the lowest error were RR+C with M = 169 pixels and RR+F+C with M = 187 pixels. The average visual angle was $4.17°$ or 1.6 inches. In other words, non-click features that are based on an understanding of gaze-cursor relationships can improve the accuracy. For practical applications, the accuracy of the better eye detector (clmtrackr) with the best model (RR+C) achieved about 130 pixels of error.

5.3 Comparison with Other Webcam Eye Trackers

Two webcam eye trackers that take a similar approach to WebGazer are TurkerGaze and PACE. TurkerGaze is a crowdsourcing platform that guides users through games to collect information on image saliency. Its goal is not real-time gaze prediction and thus contains phases of explicit calibration and offline training with more sophisticated machine learning algorithms. Their pipeline also uses the ridge regression model (RR) with the same input and parameters that WebGazer uses and enhances with user interactions. As discussed in previous sections, our use of cursor activity improves the accuracy.

PACE is auto-calibrated and uses a variety of user interactions, reporting an average error of $2.56°$ in a lab study. A direct comparison with WebGazer is not appropriate as their functionality and focus differ significantly: i) WebGazer presents an in-browser webcam eye tracking library that can be incorporated in any website while PACE is a desktop application that performs general gaze prediction on a workstation and without focusing on the Web, ii) WebGazer provides gaze predictions instantaneously after a single interaction takes place while PACE relies on sophisticated training algorithms that need several hundreds of interactions to reach such accuracy, making it impractical for the context we have designed WebGazer—continuous gaze prediction on any website.

6 Discussion

There are numerous applications that webcam eye tracking can enable: large-scale naturalistic usability studies, identifying successful advertisements, integrating eye movements into online gaming, online newspapers can learn what their users read, clinicians can remotely perform attention bias tests, people with motor impairments can navigate websites with their eyes, researchers can better understand web user attention, and search engines can know more precisely which search results are examined to improve future result rankings.

Online webcam eye tracking has obvious privacy concerns that must be balanced with its benefits. This eye tracking procedure is opt-in as browsers request access to the webcam, and the website is able to use the data if the user agrees. The benefit of doing the eye tracking in real time on the client-side is that the video stream does not have to be sent over the web. We believe local processing is a critical requirement of any webcam eye tracker; otherwise, users risk unintentionally sending private information somewhere out of their control.

7 Conclusion

We presented WebGazer, a new approach for scalable and self-calibrated eye tracking on the browser using webcams. WebGazer can be added on any webpage and aims to democratize eye tracking by making it accessible to the masses. Our findings showed that incorporating how gaze and cursor relate can inform a more sophisticated eye tracking model. Using the best of the 3 tracking libraries and with our best regression model, the mean error is 104 pixels in a remote online study.

In its current state, WebGazer is useful for applications where the approximate location of the gaze is sufficient. The best arrangement for WebGazer mimics the abilities of a consumer-grade eye tracker, but can be deployed on any website. Its utility will improve as devices gain more powerful processing capabilities and webcams capture better images in poor lighting. We believe that this work takes a step towards ubiquitous online eye tracking, where scaling to millions of people in a privacy-preserving manner could lead to innovations in web applications and understanding web visitors.

8 Acknowledgments

This research is supported by NSF grants IIS-1464061, IIS-1552663, and the Brown University Salomon Award.

References

[Alnajar *et al.*, 2013] Fares Alnajar, Theo Gevers, Roberto Valenti, and Sennay Ghebreab. Calibration-free gaze estimation using human gaze patterns. In *Proc. ICCV*, pages 137–144, 2013.

[Buscher *et al.*, 2008] Georg Buscher, Andreas Dengel, and Ludger van Elst. Eye movements as implicit relevance feedback. In *Proc. CHI Extended Abstracts*, pages 2991–2996, 2008.

[Buscher *et al.*, 2009] Georg Buscher, Edward Cutrell, and Meredith Ringel Morris. What do you see when you're surfing?: using eye tracking to predict salient regions of web pages. In *Proc. CHI*, pages 21–30, 2009.

[Chen *et al.*, 2001] Mon Chu Chen, John R. Anderson, and Myeong Ho Sohn. What can a mouse cursor tell us more?: correlation of eye/mouse movements on web browsing. In *Proc. CHI Extended Abstracts*, pages 281–282, 2001.

[Cole *et al.*, 2011] Michael J. Cole, Jacek Gwizdka, Chang Liu, Ralf Bierig, Nicholas J. Belkin, and Xiangmin Zhang. Task and user effects on reading patterns in information search. *Interacting with Computers*, 23(4):346 – 362, 2011.

[Deveria, 2015] Alexis Deveria. Can I use getUserMedia/ Stream API. http://caniuse.com/#feat=stream, 2015. [Online; accessed 2014-10-08].

[Guo and Agichtein, 2010] Qi Guo and Eugene Agichtein. Towards predicting web searcher gaze position from mouse movements. In *Proc. CHI Extended Abstracts*, pages 3601–3606, 2010.

[Hansen and Ji, 2010] Dan Witzner Hansen and Qiang Ji. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE TPAMI*, 32(3):478–500, 2010.

[Hansen and Pece, 2005] Dan Witzner Hansen and Arthur EC Pece. Eye tracking in the wild. *Comput. Vis. Image Und.*, 98(1):182–210, April 2005.

[Hauger *et al.*, 2011] David Hauger, Alexandros Paramythis, and Stephan Weibelzahl. Using browser interaction data to determine page reading behavior. In *User Modeling, Adaption and Personalization*, pages 147–158. 2011.

[Hoerl and Kennard, 1970] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[Huang *et al.*, 2012] Jeff Huang, Ryen W. White, and Georg Buscher. User see, user point: Gaze and cursor alignment in web search. In *Proc. CHI*, pages 1341–1350, 2012.

[Huang *et al.*, 2016] Michael Xuelin Huang, Tiffany C.K. Kwok, Grace Ngai, Stephen C.F. Chan, and Hong Va Leong. Building a personalized, auto-calibrating eye tracker from user interactions. In *Proc. CHI*, 2016.

[Lagun and Agichtein, 2015] Dmitry Lagun and Eugene Agichtein. Inferring searcher attention by jointly modeling user interactions and content salience. In *Proc. SIGIR*, pages 483–492, 2015.

[Liebling and Dumais, 2014] Daniel J Liebling and Susan T Dumais. Gaze and mouse coordination in everyday work. In *Proc. UbiComp*, pages 1141–1150, 2014.

[Lu *et al.*, 2011] Feng Lu, Yusuke Sugano, Takahiro Okabe, and Yoichi Sato. Inferring human gaze from appearance via adaptive linear regression. In *Proc. ICCV*, pages 153–160, 2011.

[Lu *et al.*, 2012] Feng Lu, Yusuke Sugano, Toshiya Okabe, and Yuuki Sato. Head pose-free appearance-based gaze sensing via eye image synthesis. In *Proc. ICPR*, pages 1008–1011, 2012.

[Lundgren *et al.*, 2014] Eduardo Lundgren, Thiago Rocha, Zeno Rocha, Pablo Carvalho, and Maira Bello. tracking.js: A modern approach for Computer Vision on the web. http://trackingjs.com/, 2014. [Online; accessed 2015-05-15].

[Mathias, 2014] Audun Mathias. clmtrackr: Javascript library for precise tracking of facial features via Constrained Local Models. https://github.com/auduno/clmtrackr, 2014. [Online; accessed 2015-07-08].

[Pomerleau and Baluja, 1993] Dean Pomerleau and Shumeet Baluja. Non-intrusive gaze tracking using artificial neural networks. In *AAAI Fall Symposium on Machine Learning in Computer Vision, Raleigh, NC*, pages 153–156, 1993.

[Rayner, 1998] Keith Rayner. Eye movements in reading and information processing: 20 years of research. *Psychological bulletin*, 124(3):372, 1998.

[Rodden *et al.*, 2008] Kerry Rodden, Xin Fu, Anne Aula, and Ian Spiro. Eye-mouse coordination patterns on web search results pages. In *Proc. CHI Extended Abstracts*, pages 2997–3002, 2008.

[Shapira *et al.*, 2006] Bracha Shapira, Meirav Taieb-Maimon, and Anny Moskowitz. Study of the usefulness of known and new implicit indicators and their optimal combination for accurate inference of users interests. In *Proc. SAC*, pages 1118–1119, 2006.

[Soukoreff and MacKenzie, 2004] R. William Soukoreff and I. Scott MacKenzie. Towards a standard for pointing device evaluation, perspectives on 27 years of fitts's law research in hci. *Int. J Hum.-Comput. St.*, 61(6):751 – 789, 2004.

[Sugano *et al.*, 2010] Yusuke Sugano, Yasuyuki Matsushita, and Yoichi Sato. Calibration-free gaze sensing using saliency maps. In *Proc. CVPR*, pages 2667–2674, 2010.

[Tschirsich, 2012] Martin Tschirsich. Js-objectdetect: Computer vision in your browser - javascript real-time object detection. https://github.com/mtschirs/js-objectdetect, 2012. [Online; accessed 2015-08-15].

[Xu *et al.*, 2015] Pingmei Xu, Krista A Ehinger, Yinda Zhang, Adam Finkelstein, Sanjeev R Kulkarni, and Jianxiong Xiao. Turkergaze: Crowdsourcing saliency with webcam based eye tracking. *arXiv preprint arXiv:1504.06755*, 2015.