# iShadow: Design of a Wearable, Real-Time Mobile Gaze Tracker

Addison Mayberry
University of Massachusetts Amherst
140 Governor's Drive
Amherst, MA 01003
amayberr@cs.umass.edu

Pan Hu
University of Massachusetts Amherst
140 Governor's Drive
Amherst, MA 01003
panhu@cs.umass.edu

Benjamin Marlin
University of Massachusetts Amherst
140 Governor's Drive
Amherst, MA 01003
marlin@cs.umass.edu

Christopher Salthouse
University of Massachusetts Amherst
100 Natural Resources Rd.
Amherst, MA 01003
salthouse@ecs.umass.edu

Deepak Ganesan
University of Massachusetts Amherst
140 Governor's Drive
Amherst, MA 01003
dganesan@cs.umass.edu

## ABSTRACT

Continuous, real-time tracking of eye gaze is valuable in a variety of scenarios including hands-free interaction with the physical world, detection of unsafe behaviors, leveraging visual context for advertising, life logging, and others. While eye tracking is commonly used in clinical trials and user studies, it has not bridged the gap to everyday consumer use. The challenge is that a real-time eye tracker is a power-hungry and computation-intensive device which requires continuous sensing of the eye using an imager running at many tens of frames per second, and continuous processing of the image stream using sophisticated gaze estimation algorithms. Our key contribution is the design of an eye tracker that dramatically reduces the sensing and computation needs for eye tracking, thereby achieving orders of magnitude reductions in power consumption and form-factor. The key idea is that eye images are extremely redundant, therefore we can estimate gaze by using a small subset of carefully chosen pixels per frame. We instantiate this idea in a prototype hardware platform equipped with a low-power image sensor that provides random access to pixel values, a low-power ARM Cortex M3 microcontroller, and a bluetooth radio to communicate with a mobile phone. The sparse pixel-based gaze estimation algorithm is a multi-layer neural network learned using a state-of-the-art sparsity-inducing regularization function that minimizes the gaze prediction error while simultaneously minimizing the number of pixels used. Our results show that we can operate at roughly 70mW of power, while continuously estimating eye gaze at the rate of 30 Hz with errors of roughly 3 degrees.

## Keywords

eye tracking; neural network; lifelog

## Categories and Subject Descriptors

H.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems

## 1. INTRODUCTION

An important aspect of on-body sensing is tracking the eye and visual field of an individual. Continuous real-time tracking of the state of the eye (e.g. gaze direction, eye movements) in conjunction with the field of view of a user is profoundly important to understanding how humans perceive and interact with the physical world. Real-time tracking of the eye is valuable in a variety of scenarios where rapid actuation or intervention is essential, including enabling new "hands-free" ways of interacting with computers or displays (e.g. gaming), detection of unsafe behaviors such as lack of attention on the road while driving, and leveraging visual context as a signal of user intent for context-aware advertising. Continuous eye tracking is also useful in non real-time applications including market research to determine how customers interact with product and advertising placement in stores, and personal health, where the state of the eye provides a continuous window into Parkinson's disease progression, psychiatric disorders, head injuries and concussions, and others.

While our understanding of the human eye and gaze has grown through decades of research on the topic [6, 9], eye tracking remains limited to controlled user studies and clinical trials, and has not bridged the gap to daily consumer use. The central challenge is that the sensing and processing pipeline is extremely complex: the eye-facing imager alone requires continuous operation of a camera at tens of frames per second, and compute-intensive image processing for each frame [2, 13, 24]. Unfortunately, these computational demands are very far from what can be accomplished on low-power microcontrollers and resource-limited embedded platforms. In addition, there are complex camera placement considerations and form-factor demands, making the eyeglass design much more intricate. In all, addressing the myriad technical and design challenges of wearable gaze tracking is a daunting task.

To understand the design challenges, consider an eye tracker equipped with two cameras, one facing the eye and one facing the external world. A VGA-resolution eye facing imager sampled at 30Hz generates a data rate of roughly 4 Mbps. Continuous real-time processing of such an image stream would require computational capability and memory comparable to a high-end smartphone, making the eye tracker both bulky and power hungry. An alternative design might be to wirelessly stream data from the eye tracker to a smartphone for leveraging the phone or cloud-based computational resources. However, the bandwidth requirements for such streaming is substantial — most low-power radios cannot support the demanding data rates of eye trackers, and streaming via WiFi is power-hungry and would greatly limit the lifetime of such a device. Perhaps as a result, many state-of-art eye trackers, such as the Tobii glass [24], operate as data recorders and continuously writes data to a disk that the subject carries in their pocket. (Google Glass, while not equipped with an eye tracker, has similar challenges - in continuous video capture mode, the device lasts for only a few hours.)

We argue that the current approach is fundamentally flawed — existing systems separate image acquisition from the eye state processing, and as a consequence are unable to leverage a variety of optimizations that are possible by a more holistic approach that uses application-driven sampling and processing. Consider, for example, recently available smartphones such as the Samsung Galaxy S IV, which track gaze for eye scrolling; here, the entire image is acquired from the camera, after which it is processed through computer vision techniques to estimate gaze direction. The thesis of our work is that by joint optimization of pixel acquisition and gaze estimation, we can enable real-time, continuous eye tracking while consuming only milliwatts of power, thereby enabling real-time continuous gaze based applications.

At the heart of iShadow is a simple idea: individual eye-facing images are extremely redundant, and thus it should be possible to estimate gaze location using only a small subset of pixel values in each frame. In other words — we can leverage knowledge that the imager is looking at the eye, and that the most useful information for gaze tracking is where the iris is located within the eye. Thus, we can estimate gaze coordinates accurately as long as we can extract the location of the iris and sclera of the eye at low power.

Sparse sampling of pixels has a ripple effect on almost all design choices in our system. From a resource perspective, fewer pixels per frame imply less memory needs, and fewer image processing instructions per frame thereby enabling the entire gaze estimation pipeline to execute on a simple microcontroller. From a latency perspective, fewer pixels implies lower latency for image acquisition and gaze estimation, making it possible to achieve real-time high rate gaze computation despite limited processing capability. From a power perspective, we subsample pixels directly at the imager, and not after acquiring the image, thereby reducing power consumption for image acquisition as well as processing. Fewer pixel acquisitions and less processing translates to less energy consumed per frame capture and gaze computation, which enables longer term operation.

The design of a sparse acquisition-based gaze tracking system presents substantial challenges that we address in this work. First, imagers that operate at the milliwatt power levels need to sacrifice pixel quality for power, hence an important question is whether we can achieve high accuracy despite operating with low quality imagers. Second, we ask whether we can design a gaze estimation algorithm that provides a graceful resource-accuracy tradeoff, where the accuracy of gaze estimation gracefully degrades as the energy budget reduces. Third, the computational demands of gaze tracking are often substantial — for example, several gaze estimation algorithms require processing of the eye image to detect the iris and identify its boundary, which requires processing that is considerably higher than what can be accomplished with a microcontroller. Thus, we need to understand how to reduce the processing needs to enable real-time gaze estimation while operating on platforms with tens of kilobytes of memory and no floating point units.

The main contributions of our work are the following.

- First, we design a multi-layer neural network-based point-of-gaze predictor that uses offline model training to learn a sparse pixel-based gaze estimation algorithm. A key novelty of this work is the use of a state-of-the-art sparsity-inducing regularization function for identifying pixel subsets[26]. We show such an approach can reduce pixel acquisition by $10\times$ with minimal impact on gaze prediction accuracy.

- Second, we design a real-time gaze estimation algorithm that implements the model learned by the neural network on a prototype computational eyeglass platform equipped with a low-power microcontroller and low-power greyscale cameras with random access capability. We show that our system can operate at frame rates of up to 30Hz with gaze errors of roughly 3 degrees, while executing in real time and at a power consumption of 72 mW.

- Third, we show that our methods can be easily calibrated to a new individual with a calibration dataset that is only one minute long, and without requiring changes to imager placement or adjustment of our hardware prototype. Our user study with ten users shows that, once calibrated, our system works robustly across individuals.

## 2. BACKGROUND AND RELATED WORK

The gaze tracking problem has a long history and multiple sensing modalities have been developed to address it [25]. In this work, we are specifically interested in gaze tracking using a video stream of eye images, an approach referred to as *video oculography*. We highlight the distinctions between video-based gaze tracking systems that are most relevant to our work in this section and refer readers to the surveys by Young et al. [25], Morimoto et al. [14] and Hansen et al. [7] for further details and references. We organize our review around three important distinctions between gaze tracking systems: (1) whether the hardware system is remote or wearable (typically head mounted), (2) whether the point-of-gaze estimation problem is solved offline or in real time, and (3) the category of algorithm used to solve the gaze inference problem.

### *Remote vs Wearable Eye Trackers.*

Gaze tracking has traditionally been studied and applied in the remote tracking setting [25]. The subject sits facing one or more cameras that record video of the subjects eyes. The subject typically performs a task like reading or viewing images or video. The captured eye images are then used to infer the subject's point of gaze, giving information about what they were attending to while performing the different steps of a task. In early remote systems, chin-rests or other restraints were used to ensure the subject's head was completely motionless relative to the camera system. The gaze inference problem was also solved completely offline.

Subsequent advances have led to gaze tracking systems that can reliably infer gaze direction in real time while allowing subjects a much more comfortable range of motion while seated [5, 1, 18]. Modern instantiations of these techniques use either built-in cameras on laptops and smartphones to estimate gaze direction, or use additional add-on hardware [23] that provides more accuracy for gaze tracking. These approaches are particularly useful for measuring user engagement for online ads, and gaze-based gaming or computer interaction. While very useful, the gaze tracking hardware remains static, and doesn't provide continuous gaze information in natural environments when the user is not interacting with a computing device.

In recent years, more eye trackers are being used in a mobile context. The OpenEyes project is a closely related effort to ours at the hardware layer. It involved the development of an open wearable hardware system for recording eye movements as well as associated algorithms for gaze prediction [13]. While the camera system was also an eyeglass form factor device, it was tethered to a laptop computer worn in a backpack. Rantanen et al. also presented an eyeglass-form-factor tracking device, intended for HCI purposes with disabled users [16]. Their system similarly requires the presence of a computer, though it uses a wireless connection as opposed to a tether. The Aided Eyes project is functionally the most similar system to ours [12]. Their tracking device does some local processing, including gaze prediction, but is a bulky system and there is no emphasis on reducing power consumption. Their tracker uses photodiodes for tracking the limbus, yielding an error of $5^{\circ}$.

Current commercially available wearable systems remain prohibitively expensive and in most cases are simply video recording devices with no real-time processing capabilities on-board. For example, the ASL MobileEye XG [2] includes an eye glass frame with eye and outward facing cameras, but is tethered to a recording device that can store or transmit data wirelessly. The system records video at 30Hz. While the eyeglass frame and camera system weigh only 0.17lbs, the recording device weighs 1.7lbs and is 7.56 x 4.65 x 2.0 inches in size. The battery life in recording mode is limited to 3 hours and no data processing is done on-board the device. The recently released Tobii Glasses [24] also include an eyeglass form factor camera system tethered to a recording device. The Tobii Glasses also record video at 30Hz with a combined 0.17lbs eyeglass and camera weight. The recording device weighs 0.44lbs and is 4.84 x 3.27 x 1.3 inches in size. However, the battery life in recording mode is limited to only 70 minutes, and no data processing is done on-board the device.

### *Near-Infrared vs Visible Light.*

Commercial eye trackers largely use Near-Infrared (NIR) illumination of the eye to make the pupil appear darker or lighter [14]. The advantage of NIR illumination is that it creates specific reflection patterns on the cornea and pupil of the eye, and these reflections can be captured by one or two imagers. The resulting images can be processed through advanced image processing techniques to robustly extract various aspects of the state of the eye. The disadvantage of this approach is that the device is considerably more complex, and requires an illuminator and reflector to direct the NIR light into the eye, in addition to one or more imagers. This makes the eyeglass design a much more complex engineering problem, with a variety of active and passive elements mounted at different points on the frame. While visible light based eye trackers have been proposed in the past (e.g. OpenEyes), these are far less advanced than the NIR-based counterparts, perhaps because they are perceived as being much less precise and robust to lighting conditions.

While the use of NIR illumination certainly improves image quality and robustness, we argue for an approach that sacrifices some precision for broader utility and applicability. In comparison to NIR-based eyeglasses, a simple visible-light camera integrated with a passive reflector or polarized glasses to reflect the image of the eye is far easier to mount on a regular pair of spectacles. We also argue that the loss of precision may be mitigated through leveraging advances in machine learning techniques to deal with a variety of noise sources in a more robust manner.

### *Shape-based vs Appearance-based Gaze Estimation.*

In terms of gaze tracking algorithms, three primary approaches have been explored in the literature. They are commonly referred to as shape-based, appearance-based and hybrid algorithms [7]. The shape-based approach uses features of the eye image to fit an ellipse to the boundary between the pupil and the iris [7]. The downside of this approach is that it works best with Near-Infrared (NIR) illumination sources, which, through their positioning relative to the camera, can make the pupil appear brighter or darker, thereby making it easier to detect the boundary [14]. When using visible light, shape-based techniques are harder to use since the boundary of the pupil is harder to detect.

Appearance-based gaze tracking algorithms attempt to predict the gaze location directly from the pixels of the eye image without an intermediate geometric representation of the pupil. This approach essentially treats the gaze inference problem as a regression problem where the inputs are the pixels of the eye image and the outputs are the vertical and horizontal components of the point of gaze in the outward facing image plane. Due to the generality of the gaze inference problem when formulated in this way, predictions can be based on essentially any multivariate regression approach. Two prominent approaches used in the gaze tracking literature are multi-layer neural networks [3] and manifold-based regression [21]. This approach is preferable in our scenario, since it is more robust to artifacts observed in a visible-light based image stream of the eye. While we leverage an appearance-based technique, our primary contribution at the algorithmic level is the development of sparse appearance-based models for gaze prediction that optimize gaze estimation accuracy while minimizing the pixel sampling costs.

## 3. iShadow OVERVIEW

In this section, we provide a brief overview of the iShadow system. The first step in using iShadow is calibration, where a user looks at a few points on a monitor while keeping their head relatively steady, in a manner similar to commercial eye trackers. During this calibration phase, iShadow captures a full image stream from the eye-facing and outward-facing imager, and downloads this data to a computer either via USB or Bluetooth.

The second step is the neural network based sparse pixel selection algorithm. In this stage, the learner divides the calibration dataset into training and testing sets, and sweeps through the regularization parameters to learn a set of models that correspond to different gaze prediction accuracies. Each model specifies both the set of pixels that need to be acquired and the weights on the pixels to use for the activation function that predicts gaze coordinates. Depending on the power constraints of the platform and the accuracy needs of the application, the appropriate model can be downloaded to the iShadow platform for real-time operation.

The third step is the run-time execution of the model that is downloaded onto the iShadow platform. The run-time system acquires the appropriate pixel set and executes the non-linear weighted sum to predict gaze coordinates in real time.

Once gaze coordinates are obtained from the eye-facing imager, they can be used in different ways depending on application needs. For example, it could be used to detect rapid saccades (i.e. rapid eye movements) which correspond to an event in the external field of view of the user. When a saccade is detected, the outward facing imager can be triggered to capture an image or video of the event that could have caused the saccade. Alternately, gaze coordinates

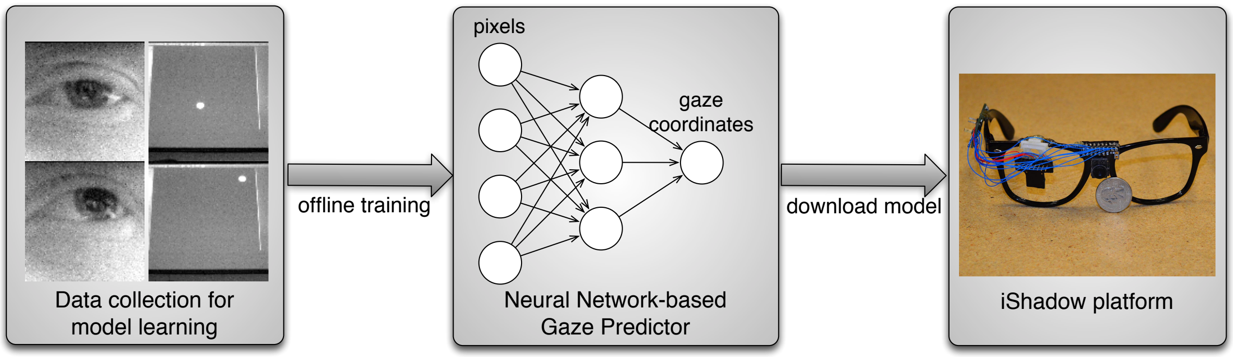

Data collection for model learning
offline training
pixels
gaze coordinates
Neural Network-based Gaze Predictor
download model
iShadow platform

Figure 1: iShadow overview: A user wears the eyeglass and collects a few minutes of calibration data by looking at dots on a computer screen. The calibration data is downloaded from local storage on the eyeglass, and the neural network model is learnt offline. An appropriate model can then be uploaded to the eyeglass for real-time gaze tracking.

could be used to detect fixation and use this information to decide when to trigger the outward facing camera.

## 4. GAZE TRACKING ALGORITHM

In this section we describe our framework for energy aware gaze tracking. At a high level, the idea involves setting up the prediction problem as a neural network where the inputs are the pixel values obtained from the imager, and the output is the predicted gaze coordinates. A unique aspect of our work is that in addition to determining the parameters of the neural network, we also determine a smaller subset of pixels to sample to reduce power consumption, while minimizing the loss in gaze prediction accuracy.

To enable subset pixel selection, the neural network learning algorithm uses a regularizer that penalizes models that select more pixels; thus, the optimization involves two terms: a) an error term that captures how well the algorithm predicts gaze coordinates, and b) a penalty term that increases with the number of pixels selected. This optimization is done offline using numerical optimization methods, and the parameters are hard-coded into the microcontroller for real-time execution. Thus, the eyeglass is not making any real-time decision about which pixels to sample, or how to map from pixel values to gaze output — it is simply computing a function of the subsampled pixels based on hard-coded parameters from the learnt neural network model. The online operation is therefore lightweight and easy to optimize in hardware. We describe this process in more detail in the rest of this section.

### *Model Specification.*

Our base gaze prediction model is a feed-forward neural network as shown in Figure 2 [4]. The input layer is a $D \times D$ array of values $I$ representing the eye-facing image. The pixel at row $i$ and column $j$ is given by $I_{ij}$. The desired output of the system is the gaze coordinates in the outward facing image plane $(X, Y)$. The hidden layer of the model consists of $K$ hidden units $H_k$. The model includes input-to-hidden parameters $W^{IH}_{ijk}$ for each pixel location $(i, j)$ in the eye-facing image and each hidden unit $H_k$; a hidden unit bias parameter $B^H_k$ for each hidden unit $H_k$; hidden-to-output parameters $W^{HO}_{kx}$ and $W^{HO}_{ky}$ mapping between hidden unit $H_k$ and the horizontal and vertical gaze coordinates $(X, Y)$; and output bias parameters $B^O_x$ and $B^O_y$ for the horizontal and vertical gaze coordinates $(X, Y)$. The hidden units use a standard hyperbolic tangent $(\tanh)$ activation function. The output units use

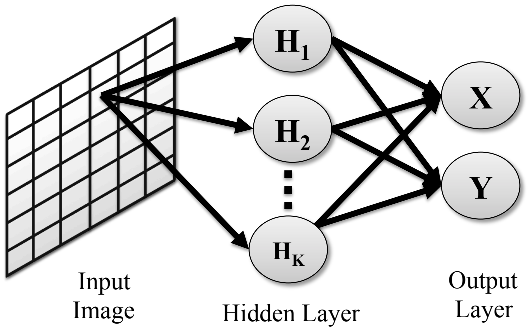


Figure 2: Illustration of the the neural network gaze prediction model.

linear activations. The mathematical formulation of the model is given below.

$$\hat{X} = B^O_x + \sum_{k=1}^{K} W^{HO}_{kx} H_k \quad (1)$$

$$\hat{Y} = B^O_y + \sum_{k=1}^{K} W^{HO}_{ky} H_k \quad (2)$$

$$H_k = \tanh\left(B^H_k + \sum_{i=1}^{D}\sum_{j=1}^{D} W^{IH}_{ijk} I_{ij}\right) \quad (3)$$

### *Model Learning.*

Given a data set $\mathcal{D} = \{I^n, X^n, Y^n\}_{n=1:N}$ consisting of $N$ eye images $I^n$ with corresponding gaze coordinates $(X^n, Y^n)$, our goal is to learn the complete set of neural network model parameters $\theta = \{W^{IH}, W^{HO}, B^H, B^O\}$. We learn the model parameters by minimizing a regularized empirical loss function between the neural network's predicted outputs $(\hat{X}^n, \hat{Y}^n)$ and the true outputs $(X^n, Y^n)$ [4]. In this work, we use squared error as the loss function. The objective function $\mathcal{F}(\theta|\mathcal{D})$ is shown below for an arbitrary regularization function $\mathcal{R}(\theta)$ with regularization parameter $\lambda$.

$$\mathcal{F}(\theta|\mathcal{D}) = \sum_{n=1}^{N} (\hat{X}^n - X^n)^2 + (\hat{Y}^n - Y^n)^2 + \lambda\mathcal{R}(\theta) \qquad (4)$$

The objective function $\mathcal{F}(\theta|\mathcal{D})$ cannot be analytically minimized with respect to the model parameters $\theta$, so numerical methods are required. The gradients of the model parameters with respect to the loss can be efficiently computed using the standard backpropagation algorithm [17]. For standard, smooth regularization functions like the two norm squared $||\theta||_2^2$, the gradients of the regularization function $\mathcal{R}(\theta)$ are also easy to obtain. The base model can be learned using any numerical optimizer such as the limited memory BFGS algorithm [15].

### *Pixel Subset Selection.*

Given that the eye-facing images are extremely redundant, the central hypothesis of this work is that we can drastically sub-sample the eye facing images while preserving much of the accuracy.

The problem of choosing an optimal set of pixel locations is a subset selection or feature selection problem [8]. We refer to the pixels actually selected as *active pixels*. We can represent the set of active pixel locations using a binary mask $A$ where $A_{ij} = 1$ if the pixel is active and $A_{ij} = 0$ if the pixel is not active. Given such an active pixel mask $A$, we can modify our neural network to base its prediction on the active pixel locations only. In Figure 2, this would correspond to simply removing all of the edges between the inactive pixels and the hidden units. The computation and communication complexity of image acquisition and gaze estimation are both linear in the number of active pixels in our framework. This means that we should expect a linear decrease in the energy cost of both image acquisition and prediction as the number of active pixels decreases.

To select a smaller active pixel set, we use a state-of-the-art sparsity-inducing group-$\ell_1$ regularization function [26]. It is well known that the standard squared-error regularizer commonly used in machine learning and statistics shrinks parameter estimates toward zero, but it typically does not result in actual sparsity in the model coefficients. Tibshirani solved this problem for linear regression through the introduction of a method called the *lasso* for optimizing the least squares loss with a regularizer that penalizes the absolute values of the model parameters [22]. For linear models, setting a coefficient to zero is equivalent removing the underlying variable from the model and, as a result, such $\ell_1$ regularization methods have been proven to be very effective at optimally solving subset selection problems.

In the present case, our model is a neural network with one parameter for each pixel in the image and each hidden unit. To solve the subset selection problem we need to simultaneously set all of the outgoing connections from a group of pixels to zero. This is likely not to happen when applying the standard $\ell_1$ regularization function in a randomly initialized neural network model. However, the group-$\ell_1$ regularizer developed by Yuan and Lin is designed specifically to drive groups of parameters to zero simultaneously. There are several different versions of the group-$\ell_1$ regularizer. We make use of the group-$\ell_1/\ell_2$ regularization function as shown below. Note that in our case, we only regularize the input-to-hidden layer weights. The groups consist of all of the parameters from a given pixel to each of the $K$ hidden units.

$$\mathcal{R}(\theta) = \sum_{i=1}^{D}\sum_{j=1}^{d}\left(\sum_{k=1}^{K}(W_{ijk}^{IH})^2\right)^{1/2} \qquad (5)$$

The neural network model parameters can then be learned by optimizing $\mathcal{F}(\theta|\mathcal{D})$ with the choice of regularizer given above.

### *Real-time Inference: Resource Usage vs Accuracy.*

It is important to keep in mind that all model learning described above happens in an offline setting. The real-time component of the gaze estimation system consists only of acquiring an eye facing image and performing a forward pass through the neural network. As shown in Equations (1)-(3), this forward pass requires only floating point addition and multiplication along with the computation of the $\tanh$ non-linearity. These are simple enough that they can be performed efficiently even on microcontrollers that do not have a floating point unit, and are limited in RAM.

One of the key benefits of our approach is that the sparse acquisition-based neural network model naturally lends itself to trading off energy consumption or resource usage for the accuracy of gaze prediction. Setting the regularization parameter to larger values will drive the parameters associated with an increasing number of pixel locations to zero while maintaining as much accuracy as possible. Fewer pixels implies a) less energy for using the imager, b) less computational needs for predicting gaze from the pixels, and c) less memory resources needed for storing the weights. All of these have advantages depending on the energy constraints on the eyeglass or the resource constraints on the microcontroller.

## 5. iShadow CALIBRATION

One important practical aspect of using a gaze tracking system is calibration of the device to each user. We faced three challenges in designing the calibration procedure for the system. (1) There were many idiosyncrasies with the low-power Stonyman imager since it was an early version of a research prototype, and we had to perform extensive experiments across numerous parameters to understand its characteristics. (2) The image stream from the eye-facing imager changes depending on each individual's eye shape and eyeglass position, hence it was important to calibrate for each individual. (3) We needed to find ways to minimize burden on the participant and any manual overhead including adjustment of the imager position, manual labeling of images for learning, etc. all of which would make iShadow difficult to use for a new user. We detail the process by which we addressed these issues and calibrated the iShadow system for a new user.

**FPN calibration:** One drawback of the Stonyman camera is the noise issues inherent to the logarithmic pixels used by the camera. Such Fixed Pattern Noise (FPN) is typical in digital imagers, and results from small manufacturing imperfections including pixel size, material, interference with circuitry, etc which is unique to each individual imager. However, logarithmic pixels are particularly sensitive to these imperfections, meaning that most pairs of pixels have a noticeably different response to equal incident illumination. This effect yields an extremely noisy image if it is not dealt with.

FPN noise can be easily accounted for under the assumption that the noise remains stationary. In this case, it is simple to correct for the FPN by determining each pixel's response to uniform incident illumination and using this to generate an offset mask over the whole pixel array. The values in this mask are then subtracted from every image captured by the camera, which removes the effects of FPN completely. While FPN will remain constant under consistent lighting conditions and camera configuration settings, the FPN for the Stonyman camera is very sensitive to changes in outdoor lighting conditions. In addition, the dynamic range of imager is relatively low, resulting in saturation effects in bright sunlight.

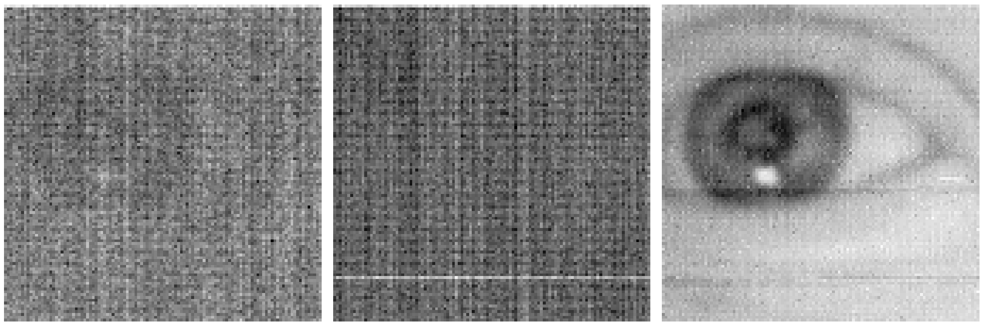

(a) Image with FPN (b) FPN Mask (c) Image After FPN Correction

Figure 3: Stages of the fixed-pattern-noise (FPN) correction process. The raw image is collected with FPN present (a), from which the the static FPN mask (b) is subtracted. The resulting image (c) is mostly free of FPN.

Fortunately, we found that the imager's behavior under indoor lighting illumination tends to be relatively uniform, and the we could learn a mask that was reusable over a moderate range of lighting conditions. Under all indoor conditions, we have been able to use a single FPN mask per camera and generate consistently viable images. While this means that our system is not currently useful in outdoor settings, this is an artifact of the camera that will hopefully be resolved with further improvements to the hardware.

The process to learn a new mask for an imager is straightforward, and needs to be only done once prior to mounting the imager on the eyeglass. The imager must be exposed to uniform illumination so as to determine the relative offsets for each pixel in the mask. This should be done using a light-colored diffuser placed over the camera lens. A diffuser can be anything from a nylon mesh, as is often used in professional photography for illumination diffusion, to a thin sheet of paper. Once the lens is covered, the iShadow driver will capture and download an image to process as the new FPN mask for that camera. This process must be performed once for each of the two mounted cameras, and then FPN calibration is complete. See Figure 3 for an example of an image with and without FPN noise.

**Collecting training data:** Variations between subjects in the shape, position of the eye, and placement of the eyeglass mean that a new neural network model must be trained for each user, and therefore some amount of labeled training data must be generated per user.

The data collection process itself is quite simple. The subject sits in front of a display of some kind for a short period of time. We use a simple script that shows a black field on the display, and a white circle of some small radius on that field. Subjects are fitted with the iShadow glasses and seated in front of the display. To maximize the effectiveness of the training data, the user should be positioned so that the display extends to the edges or just beyond the edges of the scene-facing camera's field of view. This will ensure that there can be training data samples generated over the entire field of view. Checking that the system is in proper position can be done quickly by pulling a few images from the outward-facing camera, and displaying this to the user in real-time so that the user can adjust the position of the eyeglass.

When data collection is triggered to begin, the circle begins moving in a random pattern over the space of the display. Subjects are directed to focus their gaze on the circle as it moves across the screen, and iShadow begins collecting full-frame images from both the eye-facing and world-facing cameras. The exact training time needed depends on how accurate of a model is needed, however, as we show in our evaluation, after accumulating one minute of labeled training data, adding more does not yield a significant increase in the accuracy of the predictor. Thus, the training session can be very brief without compromising on the accuracy of the generated neural network model.

After training data collection is complete, the collected images need to be transferred to a computer to train the neural network model. This can be done by storing the images on an SD card during the session or live streaming via USB.

**Labeling training data:** One potentially time-consuming aspect about learning a model for each user is generating labels from the collected data. Since the user's head position depends on height and how he/she is seated, we cannot just use the pixel positions where the target circle is drawn on the computer screen. Instead, we process the image stream from the outward-facing imager, and use a simple computer vision algorithm to detect a light-colored patch on a dark field (rest of the screen). In cases where this process fails (often because the dot is on the edge of the field of view of the imager), the calibration algorithm asks for assistance from the human, but this is largely unnecessary for the training process. Depending on the amount of training data, the process generally takes only a few minutes.

**Robustness to blinks and head movements:** One question with calibration is whether there is a negative effect of the users blinking or possibly drifting from the target for brief periods. This is not a significant issue since we are able to generate a high volume of data over a short period of time through automatic labeling. As a result, even if there are periods where the user was blinking or the user's gaze moved off-target for a brief of period of time, these are treated as noise by the neural network learner as long as the majority of the samples in the training set involve the user being correctly focused on the target. Our experiments show that training under this assumption yields models that have a high degree of prediction accuracy.

**Learning the model:** In addition to images from the eye-facing camera, and a corresponding set of gaze coordinates from the labeling session, the model learner also requires a set of regularization parameters $\lambda$, each of which will yield a different model with differing sparsity values. The process for choosing the $\lambda$ values depends upon the application. For our study we swept the training across a series of values ranging from very high to very low sparsity to generate curves for the effect of $\lambda$ on gaze prediction accuracy as well as power consumption.

The suitable regularization parameter can be chosen in a few different ways. The default option is to find a good tradeoff between prediction accuracy and power. We generally find that there is a sweet spot for each individual i.e. there is a small range of sparsity that yields good prediction accuracy at a very low power cost, and this may be a good standard target for the model generator. Another option is dynamic generation based on the desired value of a certain system parameter, such as prediction accuracy or frame-rate. In this scenario, the model generator would sweep over a standard set of lambda values, generating a prediction model for each one. The model trainer estimates system error based on cross-validation during training, this value can be used to find an acceptable $\lambda$ value and a corresponding model to meet a prediction accuracy criterion. If the goal is a certain prediction rate, then the same process can be performed using sparsity as the metric.

Once the $\lambda$s have been decided (or if they are decided dynamically), the calibration program trains a neural network model for each and saves the corresponding model parameters. Once a final model has been chosen to be used for online prediction, it is automatically loaded onto the glasses. At this point, the system is prepared to do gaze prediction for the new subject.

## 6. iShadow SYSTEM

Figure 4 shows a system diagram and photos a prototype version of iShadow; our current hardware implements all the design elements described in previous sections. We now briefly describe the key hardware sub-components used in the prototype and describe our optimized real-time gaze tracking implementation.

### 6.1 iShadow Platform

The iShadow platform features a standard glasses frame with two low-power cameras, an inertial sensor, microcontroller, and bluetooth, as seen in Figure 4. One camera is mounted in front of the user's right eye for acquiring eye-facing images. The other is mounted in the center of the frame facing outward to capture the center of the user's field of view. We use the standard optics on the image sensors, which give a 36° field of view.

Our hardware is designed with the constraint that needs to be thin and lightweight enough to be mounted on the side frame of a pair of eyeglasses, and roughly the same form-factor as a Google Glass. This required several hardware revisions and optimizations to make all components fit in the appropriate size. Our latest prototype is shown in Figure 4 and the components are described in Table 1. Of the components listed, the focus in this paper is primarily on the eye-facing imager, and computation of gaze on the MCU. Since the imager is central to our algorithm and implementation, we now turn to a more detailed description of its innards.

| | |
|---|---|
| Eye-facing imager | Stonyman 112x112 greyscale [20] |
| World-facing imager | Stonyman 112x112 greyscale |
| Inertial Motion Unit | Invensense 9-axis IMU [10] |
| Processor | STM32 Arm Cortex M3 microcontroller [19]. 32 MHz processor; 48KB memory; 384KB flash storage |
| Storage | microSD card, 64GB max |
| Radio | Bluetooth |

Table 1: iShadow platform components

**Image Sensors:** Our hardware framework is built around the Stonyman Vision Chip produced by Centeye, Inc.[1] This device fits our research purposes for several reasons. First, it is a low-power embedded camera, consuming approximately 3 mW when operating at full power. (see Table 2 for details). Second, the design of the pixel array allows for random access to individual pixel values. The combination of low-power operation and random access capability makes the Stonyman unique among commercially available image sensor chips.

The Stonyman features a 112x112 square grid of pixels. These pixels are characterized by their logarithmic voltage response to lighting conditions. This allows for a greater dynamic range compared to a pixel that operates linearly with respect to lighting conditions. The use of logarithmic pixels allows a random-access interface, which the Stonyman provides via a register-based control scheme. It is this feature specifically that enables the energy-accuracy trade-offs that we explore in this work.

Like many contemporary mobile image sensors, the Stonyman sensor provides a sleep mode. We exploit this feature in iShadow to use power more efficiently. The majority of the power drawn by the camera while it is active comes from the pixel acquisition circuitry, and this can be powered down using a control register. In this low-power state there is only a small power draw - less than a half a microwatt - for the digital control circuitry that maintains the values in the control registers and allows the camera to be switched back to full-power mode. There is a small delay time, on the order of a few microseconds, for the camera to power up and back down. These delay times, as well as the power consumption of the camera in wake and sleep modes, are given in Table 2.

Finally, while we made the choice to use the Stonyman imager as the outward-facing camera in addition to the inward facing one, the outward facing imager can be replaced with a higher-end device if better quality images are needed for vision processing. The insight in this paper is that gaze coordinates can be obtained with a very low-power and low-resolution eye-facing imager such as the Stonyman camera.

| | |
|---|---|
| Active to Sleep Delay | 4 $\mu$s |
| Sleep to Active Delay | 10 $\mu$s |
| Active Power Consumption | 3.13 mW |
| Sleep Power Consumption | 0.041 $\mu$W |

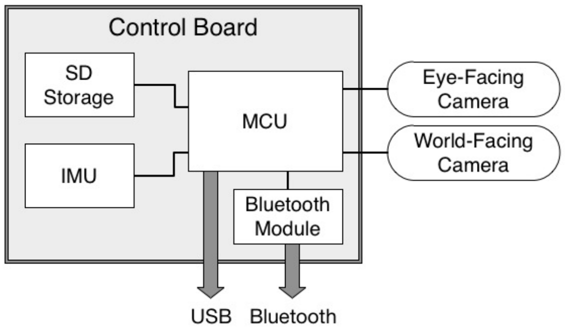
Table 2: Stonyman Power Consumption and Transition Times in and out of Sleep Mode

### 6.2 Basic Operation Modes

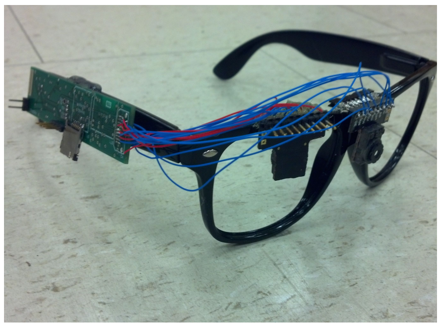
At the lowest level, iShadow supports three basic modes of system operation — full image capture, real-time gaze tracking, and life logging. However, the system's functionality is not limited to the modes outlined here, as one of the primary benefits of iShadow over existing commercial eye trackers is its programmability. Users who have a specific application in mind that might benefit from a more complex operating scheme can implement and run it on the system themselves, allowing for a much broader set of usage scenarios than those outlined here.

**Full image capture:** In this mode, iShadow is continuously capturing full-frame images from the inward and outward-facing imagers and storing it into on-board flash. This mode is intended primarily for offline data analysis, and adjustment of parameters

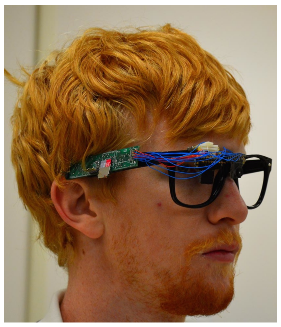
[1]http://centeye.com/products/stonyman-vision-chip-breakout-board



(a) Diagram



(b) Platform



(c) On Head

Figure 4: Figures show an architecture diagram and different views of the third-generation iShadow prototype. The prototype has two cameras, one at the center front and one facing the eye, with the electronics mounted on the control board on the side. Batteries, while not shown, are mounted behind the ear.

during the calibration phase (described above). By tightly optimizing the image capture method and interleaving control signals with ADC delays, we are able to drive both cameras at the same capture rate as if we were only driving one. The full-image capture rate for both cameras is 10 Hz.

These images can be stored to an onboard SD card or transmitted to another device via USB or bluetooth. For storage, we have designed our system to use a double-buffered scheme that allows the SD communication peripheral to read and transmit already-collected pixel data while more data is written to the second buffer. There is still some delay introduced by the SD card writes, as there are a small number of operations needed to trigger the SD card write process. However, this improvement hides the majority of the SD write latency. Since our available RAM is too small to hold an entire frame, let alone two double-buffered frames, we write data at regular intervals during image collect to prevent overflow.

**Real-time gaze tracking mode:** The most useful operating mode of iShadow is real-time gaze prediction, where it is continuously processing the pixels from the eye-facing imager to output gaze coordinates. In this mode, iShadow is solely processing image streams from the eye-facing imager. Once a gaze prediction has been completed and a gaze estimate generated, it can be used in several ways. One option is to store values for some form of post processing later - this is especially useful when run the eye-facing imager runs in conjunction with the world-facing imager, as in the final operating mode below. The gaze data can also be used for triggering more complicated on-board processes or for making higher-level inferences about the state of the eye or the wearer, as discussed in the following section.

**Lifelogging mode:** iShadow's third operating mode is a simple extension of real-time gaze-tracking. Simply put, it is capturing what in the outside world the user is looking at. By running the real-time gaze inference algorithm using the eye-facing imager and taking full images from the world-facing imager, the system records where the user is looking and what they are looking at. This type of "lifelogging" is becoming more prevalent as wearable computing grows cheaper and more accessible.

In this mode, the bottleneck is the outward-facing image capture. By doing interleaving of prediction operations with ADC reads the predict time can be completely hidden, however, there is no way to significantly reduce the time needed to collect a full-frame image from the outward-facing camera. This operating mode is a simple but straightforward example of the types of applications that iShadow facilitates, and does not require any additional logic on top of the existing firmware.

## 6.3 Gaze-triggered applications

While the focus of this paper is not on designing gaze-driven applications, we briefly describe a few example applications that can be designed over the above operating modes. Several higher-level inferences are possible from a continuous stream of gaze coordinates: a) Eye fixations can be detected by looking for windows when the gaze coordinates are relatively steady with small changes due to head movements, b) Smooth Pursuit can be detected by looking for slowly changing gaze coordinates, and c) Sudden events can be detected by looking for a large saccade or change in gaze. These higher-level inferences could suggest different things about the visual state of the outward-facing imager — for example, eye fixations could be during conversation with an individual, smooth pursuit could be while reading from a book or screen, and sudden events could be an unexpected change in the surroundings. In the case of disease progression, abnormal gaze patterns may be detected, for example, abnormally frequent saccades. Once such a high-level inference detects an event of interest, it quickly triggers the outward-facing imager to capture the state of the world, and stores this data into the local SD card or streams the data over bluetooth to a mobile phone.

## 7. EVALUATION

To test the effectiveness of iShadow at accurately and efficiently predicting the wearer's gaze location, we collected sample data from ten different subjects - eight men and two women from the ages of 22 to 31. We ran each subject through the calibration process outlined in section 5, excluding the FPN mask generation as it does not affect the per-individual results. We generated at least five minutes of labeled data for each subject in full-image-capture mode. To generate the ground-truth labels, we use the approach described in section 5. Running at the maximum framerate of 10 Hz, the resulting dataset includes at least 3000 images per user. We used this data to perform a variety of experiments to determine whether we had reached our design goals.

We present our evaluation in three sections. Section 7.1 details our experiments to determine whether the neural network model and the $\ell_1$-subsampling method are capable of accurately estimating the wearer's gaze coordinates. In section 7.2, we evaluate the

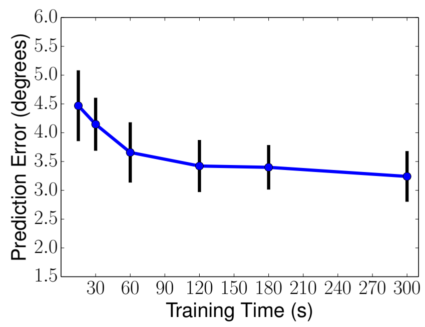


Figure 5: Amount of training time for the system versus the resulting gaze prediction accuracy.

tradeoffs between the model parameters and the resulting performance of the hardware, especially the image capture pipeline.

## 7.1 Evaluation of Neural-Network-Based Pixel Selection

We first evaluate the benefits of our neural network-based method to select a sparse set of pixels. Using the data collected, we performed a number of experiments to investigate the accuracy of the neural network gaze prediction model as a function of the number of active pixels. We generated a set of 10 values of the regularization parameter $\lambda$, and trained each user's data over this same set of $\lambda$s. Our per-user, per-$\lambda$ experimental framework is based on a standard five-fold random re-sampling protocol. For each fold, we divided the available data into a training set and a test set completely at random using an $80/20$ split.

We use the training set to estimate the parameters of the model and use the learned model parameters to predict gaze locations for each test image. We compute the gaze prediction error for a given test fold in terms of the average squared distance ($\ell_2$ distance) between each true gaze location in the test set and the predicted gaze location given each eye image in the test set. We average the prediction error over the five test folds to generate an error value for that user and $\lambda$. In addition, the size of the pixel selection mask varies with each fold as the optimizer finds slightly different solutions when using different training data sets. To account for this, we also average the size of the pixel mask generated over the five folds.

After generating per-user data, we average the prediction error and pixel mask sizes across all ten users to generate our results below. While there is moderate variation in the results between users due to differences in eye position and shape, the general trends across users are consistent and can be seen in the results we present here. For all of our results, we present the mean value and the 90% confidence interval unless otherwise indicated.

*Accuracy – Model Size tradeoffs.*

One of the benefits of our algorithmic framework is that it is able to provide a variety of models that offer different tradeoffs between the overall complexity of the model (i.e. number of pixels sampled, and number of weights for computation) and the accuracy of the model (i.e. the precision in degrees). This tradeoff is enabled by using different choices of the regularization parameter, which varies the penalty for model complexity compared to the accuracy.

First, we look at whether the regularization parameter is a useful knob for tuning model size and accuracy. Figures 6a and 6b show that varying the regularization parameter enables us to select a spectrum of models with different prediction accuracies and different fractions of selected pixels, respectively.

Figure 6c shows just the prediction accuracy vs model size — interestingly, we see that varying the percentage of activated pixels from 100% down to about 10% only has a minor effect on the prediction accuracy. This shows that there is substantial redundancy in the eye image, and the neural network is able to predict gaze just as well with 10% of the pixels activated as 100% of the pixels. On our imager, this means that sampling 10K pixels per image vs sampling 1K pixels per image has roughly the same prediction error, which in turn can translate to substantial reduction in power consumption.

We also see that the accuracy of the neural network model does not drop below about $2^\circ$ even when the entire image is sampled. In contrast, high-end mobile eye trackers advertise accuracies of as low as $0.5^\circ$. The primary reason for this gap is that the eye-facing imager lens is mounted at a fixed position on our eyeglass in our current iShadow prototype, and has a limited field of view of $36^\circ$. As a result, for some individuals, the pupil was rarely completely within the field of view of the imager.

This can be seen in Figure 7, which compares sample eye images between users for whom the predictor performed well, average, and poorly compared to the rest of the set. In general, having the entire range of the pupil's motion visible to the camera, as well as a strong contrast between the iris and the white of the eye (sclera) seem to be the most significant indicators of good predictive performance.

Despite the gap, we argue that a $3^\circ$ accuracy error is acceptable for a variety of real-world applications, particularly when the object being viewed is in relatively close proximity, for example, detecting the individual with whom a user is engaged in face-to-face conversation. Figure 8 provides a better understanding of the error in the outward image plane. Given that the imager already has a small field of view, gaze coordinates with roughly $3^\circ$ error is reasonable enough to get a good idea of where the user is looking.

To get a better idea of how the neural network selects weights, we look at some example weights learnt by the hidden units. Figure 9 shows the weights for each hidden unit. Each example has approximately 10% of pixels active We can also see that the group-$\ell1$ method learns to focus on the region of the eye image where the pupil and the white of the eye are most likely to appear, as one would expect.

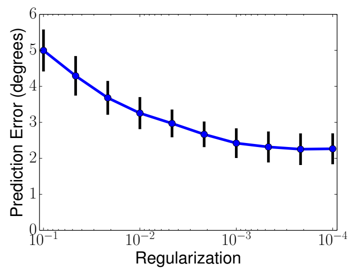
*Calibration time.*

To evaluate how much training data is needed to generate an effective model, we look at how quickly the gaze prediction converges as the amount of data that we use for training increases. Figure 5 shows the results for a particular choice of the regularization parameter ($\lambda = 0.01$). We see that the convergence is very fast — even if there is only 60 seconds of data used for training, that is sufficient for the algorithm to determine the appropriate parameters with little to no improvement as the amount of training data increases. Similar results were seen for other values of $\lambda$. Thus, the time for calibration is not a bottleneck in system operation.

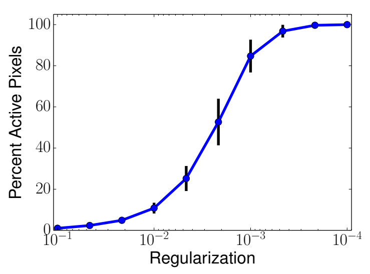
## 7.2 iShadow System Evaluation

We now turn to an evaluation of the iShadow platform predicting gaze in real-time with models trained using the neural network-based sparse sampling algorithm and appropriate regularization parameters, $\lambda$, as described earlier.

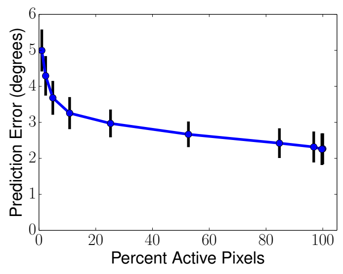
We wish to evaluate three key performance metrics — gaze tracking rate, prediction accuracy, and energy consumption. While gaze prediction accuracy and energy consumption are self-explanatory, gaze tracking rate requires a bit more explanation. Unlike a traditional eye tracker with an active pixel imager that involves ex-



(a) Lambda vs Accuracy

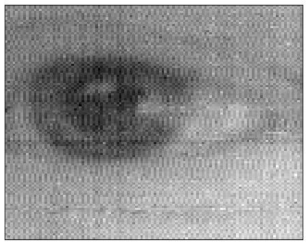


(b) Lambda vs Model Size

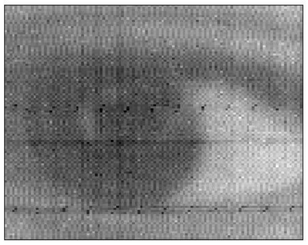


(c) Model Size vs Accuracy

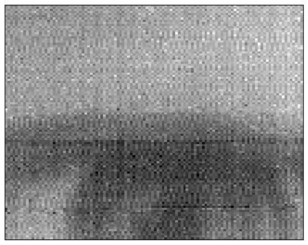
Figure 6: Plots (a) and (b) show the effect of regularization parameter on gaze prediction accuracy and model size respectively. Plot (c) shows the net result, which is how the number of pixels acquired can be reduced dramatically (up to 10×) with minor effect on gaze prediction accuracy.



(a) $1.32°$ Error



(b) $2.19°$ Error



(c) $3.40°$ Error

Figure 7: Comparison of eye images from multiple users, giving the average prediction error for that user. Notice that the position of the eye in the image and iris / sclera contrast have a prominent effect on prediction accuracy.

posure followed by frame capture and image processing, we use a logarithmic pixel imager where the pixels can be continuously read out at any time. Thus, we refer to gaze tracking time as the amount of time after sampling the first pixel of the pre-selected sparse set until we obtain the results of the gaze. Gaze tracking rate is the number of such gaze predictions we obtain per second.

**Tracking rate vs prediction accuracy:** By tuning the regularization parameter, $\lambda$, we can tradeoff between the three performance metrics. Let us first consider gaze tracking rate vs accuracy while fixing the power budget. Lets start with the case when all pixels are sampled — here, accuracy is high since all pixels are available, but gaze tracking rate is poor since fewer images are sampled per second. By increasing the level of regularization, we can progressively decrease the time needed to make a single prediction since: a) fewer pixels need to be read from the camera and there are fewer calculations to perform, and b) the number of feature weights is proportional to the number of pixels, hence the memory footprint of the model also decreases. Thus, we can increase gaze tracking rate to capture rapid gaze changes, but we suffer in the accuracy of tracking due to a coarser model with fewer pixels.

Figure 11 provides an empirical evaluation of the rate vs accuracy tradeoff on iShadow. We run iShadow in always-on mode and progressively reduce the model size from large (low error, low rate) to small (high error, high rate). The results are as expected — for large models, we get prediction errors of roughly $3°$ but low gaze tracking rates around 10 Hz. As the models reduce in size, gaze prediction errors increase to roughly $4°$, but gaze tracking rates increase to 30+ Hz as well. For reference, commercial eye trackers operate at 30 Hz or higher, therefore, these results show that with a small reduction in accuracy, iShadow can achieve sampling rates comparable to commercial eye trackers.

**Tracking accuracy vs Energy consumption:** We now turn to gaze tracking accuracy vs power consumption. When evaluating energy consumption, we need to consider two hardware components, the micro controller (MCU) and the imager, and two software subsystems on the MCU, pixel acquisition from the camera, and the cost of running the gaze prediction algorithm. Thus, we do a three-way breakdown of energy: a) energy consumed for pixel capture by the imager, b) energy consumed for pixel acquisition by MCU, and c) energy consumed for executing the gaze prediction algorithm on the MCU.

Figure 10a shows the results for a fixed gaze tracking rate of 4 Hz. The system is duty-cycled whenever it is not actively performing gaze acquisition or prediction. We measured power between our on-board voltage regulator and the relevant circuit at a rate of 30 KHz, and report the energy breakdown for each gaze prediction across the different components. (Power consumption is just 4× the reported energy number since gaze tracking rate is 4 Hz.)

The general trend in the graph is as expected — a smaller model means less energy consumption but higher error. More interesting is the breakdown across the hardware and software components. The results show that the energy consumed for prediction is roughly the same as the energy consumed by the camera. But the main cost of acquisition is at the MCU, which needs to set appropriate control lines to select pixels, and read-out the pixels over a serial bus. The cost of gaze prediction is about a third or less of acquisition cost.

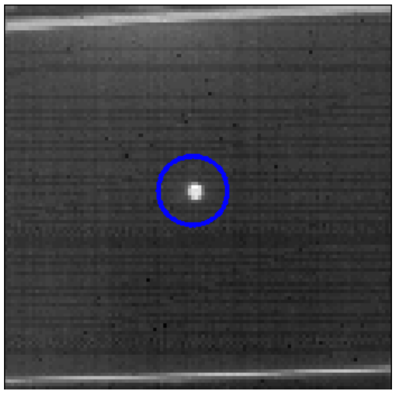


Figure 8: The circle gives an example of 3° of error in the outward imager plane around the white dot. The error is less if the eye is fully within the field of view of the imager, and higher when not fully in the field of view.

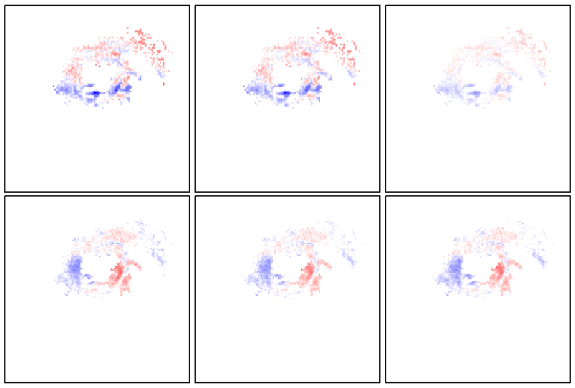


Figure 9: This figure shows the weights learned by each hidden unit in the neural network model for subsets of approximately 10% of pixel locations.

Figure 10b shows the time spent in each subsystem per gaze prediction — since a gaze tracking output is obtained every 250 ms, this shows what portion of this time period is spent per subsystem. Since the camera ON time is the same as acquisition time, we show only the plot for acquisition in the figure. The result shows that the least amount of time is spent in the gaze prediction operation, and more time is spent in pixel capture and acquisition by the MCU.

Overall, these results demonstrate that sub-sampling pixels is extremely important to reduce power consumption of the iShadow platform — gaze prediction consumes less time and energy compared to pixel capture and acquisition, clearly justifying the benefits of sparse acquisition. This is facilitated in hardware by the Stonyman's random-access pixel capability, without which the time and energy for the MCU to acquire all of the pixel data would dwarf the benefits of decreased gaze prediction time and camera sleep.

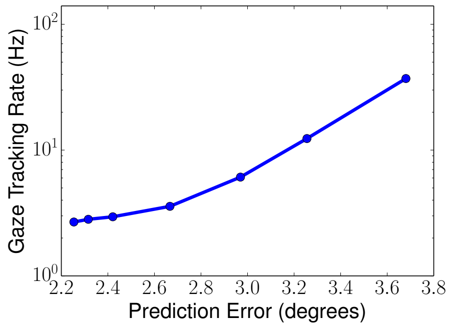


Figure 10: Smooth tradeoff between gaze tracking rate and prediction error as $\lambda$ is varied. The MCU is always active in this experiment.

## 8. DISCUSSION

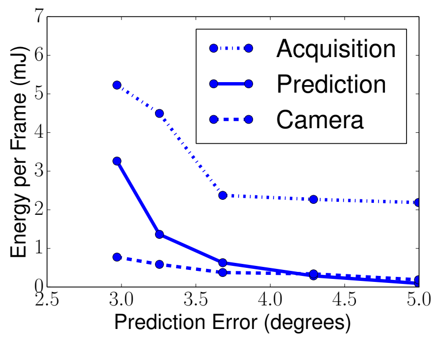
The design of a computational eyeglass platform is extremely complex due to the complexity of the sensors, the positioning and mounting on the eyeglass frame, and the demanding computational needs for processing image streams. While our results are promising, there are several steps to take before such platforms are ready for more widespread use. We discuss some of these steps in this section, as well as the types of applications that our platform is suitable for.

**Imager + IMU fusion:** While the techniques that we use in this paper can be used to identify gaze direction, we need additional information about head movement to precisely determine whether the individual is fixated at the same location in the external world. For example, the head might move about during conversation but the eye might continue to track the individuals face. Our platform incorporates an IMU in addition to imagers to enable such fusion, but our algorithms have not yet taken advantage of this capability.

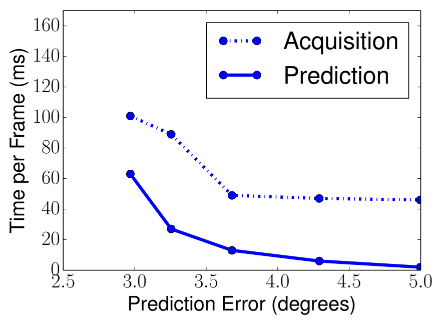
**Image field of view:** As described in §7, one of the reasons why our gaze tracking performance varies across individuals is because of the field of view of the eye-facing imager. We currently use a lens that has a field of view of 36°, as a result of which the entire eye is not in the visual field of the imager for some subjects. We believe that this issue can be fixed using a lens with a wider field of view, such as a fisheye lens. One important advantage of the neural network-based gaze tracking algorithm used in iShadow is that it is more robust to distortions caused by different lens than canonical vision-based techniques. Even though a fisheye lens may be expected to distort the image, the neural network technique is not sensitive to such distortions compared to an algorithm that does shape fitting, hence our approach can be expected to retain or improve performance.

**Inward-facing imager placement:** An obvious consideration from an aesthetic perspective is how to place the inward facing imager such that it does not obstruct the field of view of the user. Several possibilities present themselves — for example, the pixels of an imager may be mounted all along the rims of the eyeglass, or the imager may be mounted on the side frame and observe a reflection of the eye on the spectacles lenses. While such design aspects are beyond the scope of this work, we note that a key benefit of the techniques presented in this paper is that it can be easily adapted to new eyeglass designs.

For example, consider the case where pixel imagers are mounted throughout the eyeglass rim. Depending on the shape of a user's eye, different sets of pixels may carry information that is relevant to gaze. These minimal set of pixels can be selected by applying techniques described in this paper. The same is true for the second example, using a reflected eye image. Thus, our methods generalize to a variety of other designs where subsampling of pixels may be useful.



(a) Accuracy vs Energy



(b) Accuracy vs Time Elapsed

Figure 11: Breakdown of energy and time for different subsystems during the capture and predict process. (a) is the amount of energy consumed for pixel acquisition by the MCU, the gaze prediction computation, and cameras, and (b) is the amount of time spent by the MCU in acquisition and prediction.

**Robustness across users:** We have described the variability of per-user performance and the improvements we plan to implement to make performance better and more consistent across subjects that it has been trained on. However, our end goal is to provide a system that works with little or no per-user training required. This is critically important for accessibility and ease of use. The first step towards this goal is addressing the issue of eye localization in the image, either by altering the hardware to allow repositioning of the camera or by automatically adjusting for the position of the eye in the image. Once that issue has been addressed, we plan to collect data from a much larger number of subjects and begin building and testing "universal" models that will work at least moderately well on a new subject without requiring prior training.

**Comparison to other eye tracking devices:** The development of specialized eye tracking hardware has accelerated rapidly over the past decade, to the point where remote (non-mobile) devices are expected to have only a few tenths of a degree of predictive error [7]. Even mobile trackers report errors of $1^\circ$ or less [2]. The fact that the error rates we report are moderately higher than this immediately begs the question of why we chose to introduce this system. Part of the discrepancy is the immaturity of the system - iShadow is still in early development, and accuracy can be expected to improve noticeably as we refine the system and tracking algorithm. It would be overly optimistic, however, to assume that such progression will bring us to the same order of magnitude of accuracy as industrial tracking devices.

However, current mobile trackers remain too bulky to be deployed comfortably in real-life scenarios and have too short of a battery life for even moderate-length studies of 4+ hours. Because of iShadow's low-power design, it offers a significantly longer run time and smaller form factor than existing mobile devices. Once the design of the system has been refined, we envision it being used for long-running experiments in the wild. For these types of applications it is unlikely that extremely fine-grained accuracy will be needed. iShadow's current accuracy is easily enough to identify, for example, what object the user is looking at in a scene. It is not enough to distinguish gaze between lines of text, but tasks of that nature are generally better suited to static test environments.

In short, all eye tracking hardware represents a tradeoff between accuracy and mobility. The current generation of mobile eye trackers give moderate mobility at the cost of some accuracy. We present iShadow as a further step in that same direction, giving excellent mobility at a slight accuracy cost over and above that of current industrial mobile devices. For tasks that require such mobility, we believe that iShadow is the best option currently in existence.

## 9. CONCLUSIONS

We present a first-of-its-kind low power gaze tracker that is designed to predict gaze in real-time while operating with a power budget of a few tens of milliwatts. This paper presents a soup-to-nuts implementation of such a system, including a full hardware prototype, a new neural-network based algorithm for sparse pixel acquisition, a full implementation of a real-time gaze predictor on a microcontroller, and evaluation on several subjects. Our approach exploits the unique properties of random access pixel cameras to achieve a flexible energy-accuracy trade-off in the wearable/real-time setting. Our results show that we can dramatically reduce power consumption and resource needs by sampling only 10% of pixel values, without compromising accuracy of gaze prediction. These results are highly significant in that they offer a clear path toward ubiquitous gaze tracking for a variety of applications in computer vision, behavioral sensing, mobile health, and mobile advertising. For videos on the working of iShadow, see [11].

## Acknowledgements

This paper has benefited from extensive discussions with several people. In particular, we would like to thank Russ Belawski and Prabal Dutta, who have been instrumental in getting earlier prototypes of our platform up and running, and enabling several iterations of initial data collection that helped us bootstrap this work. We would also like to thank Jeremy Gummeson for much needed help with the hardware prototype. We are also very grateful to our shepherd, Prof. Lin Zhong, for extensive interactions and insightful suggestions that helped us improve presentation of results in the paper. This research was partially supported by NSF grants #1239341, #1217606, and #1218586.

## 10. REFERENCES

[1] A neural-based remote eye gaze tracker under natural head motion. *Computer Methods and Programs in Biomedicine*, 92(1):66 – 78, 2008.

[2] Applied Science Laboratories. NeXtGeneration Mobile Eye: Mobile Eye XG. http://www.asleyetracking.com/Site/Portals/0/MobileEyeXGwireless.pdf, 2013. Online; accessed April 7, 2013.

[3] S. Baluja and D. Pomerleau. Non-intrusive gaze tracking using artificial neural networks. Technical report, Pittsburgh, PA, USA, 1994.

[4] C. M. Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[5] D. Cheng and R. Vertegaal. An eye for an eye: a performance evaluation comparison of the lc technologies and tobii eye trackers. In *Eye Tracking Research & Application: Proceedings of the 2004 symposium on Eye tracking research & applications*, volume 22, pages 61–61, 2004.

[6] A. T. Duchowski. *Eye Tracking Methodology: Theory and Practice*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[7] D. W. Hansen and Q. Ji. In the eye of the beholder: A survey of models for eyes and gaze. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(3):478–500, 2010.

[8] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[9] K. Holmqvist, M. Nyström, R. Andersson, R. Dewhurst, H. Jarodzka, and J. Van de Weijer. *Eye tracking: A comprehensive guide to methods and measures*. OUP Oxford, 2011.

[10] Invensense-9150. MPU-9150 Nine-Axis (Gyro + Accelerometer + Compass) MEMS MotionTracking Device. http://www.invensense.com/mems/gyro/mpu9150.html, 2013.

[11] iShadow. iShadow Videos. http://sensors.cs.umass.edu/projects/eyeglass/, 2013.

[12] Y. Ishiguro, A. Mujibiya, T. Miyaki, and J. Rekimoto. Aided eyes: eye activity sensing for daily life. In *Proceedings of the 1st Augmented Human International Conference*, page 25. ACM, 2010.

[13] D. Li, J. Babcock, and D. J. Parkhurst. openeyes: a low-cost head-mounted eye-tracking solution. In *Proceedings of the 2006 symposium on Eye tracking research & applications*, pages 95–100. ACM, 2006.

[14] C. Morimoto and M. Mimica. Eye gaze tracking techniques for interactive applications. *Computer Vision and Image Understanding*, 98(1):4–24, 2005.

[15] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Science+ Business Media, 2006.

[16] V. Rantanen, T. Vanhala, O. Tuisku, P. Niemenlehto, J. Verho, V. Surakka, M. Juhola, and J. Lekkala. A wearable, wireless gaze tracker with integrated selection command source for human-computer interaction. *Information Technology in Biomedicine, IEEE Transactions on*, 15(5):795–801, 2011.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(Oct):533–536+, 1986.

[18] W. Sewell and O. Komogortsev. Real-time eye gaze tracking with an unmodified commodity webcam employing a neural network. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, pages 3739–3744. ACM, 2010.

[19] STM32. STM32 32-bit ARM Cortex MCUs. http://www.st.com/web/en/catalog/mmc/FM141/SC1169, 2013.

[20] Stonyman. Stonyman Vision Chip. http://centeye.com/products/stonyman-vision-chip-breakout-board/, 2013.

[21] K.-H. Tan, D. Kriegman, and N. Ahuja. Appearance-based eye gaze estimation. In *Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision, 2002.*, pages 191–195, 2002.

[22] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[23] Tobii. Tobii EyeX Controller. http://www.tobii.com/eye-experience/, 2013.

[24] Tobii Technology. Tobii Glasses Eye Tracker. Online, 2013. Online; accessed April 7, 2013.

[25] L. Young and D. Sheena. Survey of eye movement recording methods. *Behavior Research Methods*, 7(5):397–429, 1975.

[26] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.