Ayumu: Efficient lifelogging with focused tasks

Ben Stoddard
Duke University
stodds@cs.duke.edu

Kate O'Hanlon
Duke University
cohanlon@cs.duke.edu

Brian Lin
Duke University
brian.lin@duke.edu

Ashwin Machanavajjhala
Duke University
ashwin@cs.duke.edu

Landon P. Cox
Duke University
lpcox@cs.duke.edu

ABSTRACT

Today's lifelogging devices capture images periodically without considering what data is important to users. Due to their small form factors and limited battery capacities, these lifeloggers are bound to miss important data either because they record at a slow rate to conserve power, or because they record at such a high rate that they must frequently recharge. In this paper, we present a new approach to lifelogging that better utilizes a device's battery by integrating knowledge of the specific information that a user wants captured. We have developed the first instance of such a focused-task lifelogging system called Ayumu, which aims to capture the reading material that a user interacts with over the course of a day. Instead of capturing images periodically, Ayumu uses a suite of inexpensive sensors to record only when reading material is present. By recognizing when it would be most beneficial to capture images, Ayumu can achieve superior precision and comparable recall to a conventional, periodic lifelogger while using less energy.

Keywords

Lifelogging; Energy Efficiency; State Estimation

1. INTRODUCTION

Lifelogging, i.e., automatically recording a first-person view of a user's day, has been a tantalizing proposition for wearable-computing systems for several decades [4, 13, 23]. Over that period, smaller and more powerful processors and sensors have led to sleek consumer devices, such as the Narrative Clip and Google Glass. Relative to their forebears these devices are attractive and capable, and they offer wearers the promise of digitizing a wealth of useful information that currently remains unrecorded. And yet despite these advances, form-factor constraints for wearable computers leave these devices inherently resource constrained. In particular, battery density has not improved at the same rate as processor and sensor miniaturization.

Thus, although the promise of capturing and searching first-person recordings remains appealing, lifelogging is unlikely to reach its full potential unless systems can better utilize wearables' limited resources. Today's lifeloggers are particularly wasteful because they are indiscriminate, capturing images without regard to what is important to device wearers. Current systems try to conserve energy by capturing images periodically, but this is a Hobson's choice: a lifelogger must either capture at a high rate and miss interesting data while it recharges, or the device must capture at a low rate and miss important events that occur out of phase with its timer [18].

To help lifelogging reach its full potential, we are investigating a new approach based on *focused tasks* that captures narrow sets of objects. By targeting specific information, a developer can make the most of a wearable system's limited resources. Underlying our approach is the hypothesis that there are large classes of useful information that appear relatively infrequently (e.g., less often than once every minute), and that these appearances can be detected with low-power sensing.

In this paper, we explore our hypothesis by describing the first instance of a focused-task lifelogger, called *Ayumu*[1]. Ayumu's goal is to efficiently capture all of the reading material that a user encounters over the course of a day. Ayumu allows users to search for text they have read from hard-copy sources like restaurant menus and magazines as well as from digital sources like websites and PDF documents. Ayumu supports keyword search over captured images through Google's deep-learning optical character recognition (OCR) service [12].

The primary technical challenge of our work is accurately and efficiently detecting reading material. Our key observation is that users typically read text from a flat, light-colored surface at arm's length and remain still while doing so. Ayumu uses a low-power light sensor to detect changes in reflected light (indicating the appearance or departure of a reading surface) an IR proximity sensor to detect when a user is facing an object at an appropriate distance, and an accelerometer to detect if the user is still. Because these sensors use less power than a camera, they can be sampled at a relatively high rate without draining a device's battery. Ayumu only begins to capture images when its sensors indicate the presence of reading material. As a result, Ayumu conserves energy by suppressing image capture when a user is unlikely to be reading, such as while driving or walking, and captures reading material with high likelihood in settings such as working at a desk.

[1]Ayumu is the name of an eidetic chimpanzee.

We have built an Ayumu prototype and compared how comprehensively and efficiently it captures reading material compared to the simple periodic approach used by existing systems. For a set of representative image traces, Ayumu achieves 20-40% better precision and equivalent recall compared to periodic lifelogging while using less energy.

The rest of this paper is organized as follows. Section 2 provides an overview of our approach, including motivation and design principles. Section 3 describes the design and implementation of Ayumu. Section 4 describes results from experiments with our Ayumu prototype. Section 5 describes related work, and section 6 summarizes our conclusions.

2. APPROACH OVERVIEW

Lifelogging systems periodically capture pictures throughout a user's day to create an archival record of her interactions with the world. While these systems have benefited from smaller and more powerful processors and sensors, battery density has improved much more slowly. Thus, the inherent form-factor constraints of wearable computers create a dilemma. Devices can either capture at a high rate and force users to recharge their devices more often, or devices can capture a low rate and miss useful information that appears out of phase. In this section, we motivate an approach to this problem, called focused-task lifelogging, and describe the design principles underlying a focused-task prototype called Ayumu.

2.1 Motivation

Consider a faculty member who begins her day reading the news, and then spends her work day editing manuscripts, lecturing and meeting with students, and reading books and journals. She may end the day dining at a restaurant with her spouse. Over the course of a typical day, the faculty member will encounter reading material in both hardcopy and digital forms and in a variety of physical settings. It would be useful if the faculty member could perform keyword searches of this material at a later time to quickly remember easy-to-forget details like an obscure citation or items on the restaurant's wine list.

At the moment, the faculty member's best option is to use a lifelogging device like the Narrative Clip. Such a device would allow her to continuously capture images throughout the day, while OCR [17, 30, 19], text extraction [28, 11, 9], key-frame selection [7], and lifelogging search [1] techniques could convert these images into a searchable archive. Unfortunately, all lifelogging systems of which we are aware rely on simple periodic image capture. For example, Narrative Clip devices include a 5MP camera and last approximately 30 hours while capturing images every 30 seconds [23].

The limited battery power of lifelogging devices forces system designers to make a difficult choice. Under periodic image capture, designers must choose a rate that is high enough to create a comprehensive record, but limit the rate so that users needn't frequently recharge their device. Setting a good rate for digitizing reading material is particularly challenging because it is a bursty workload: users may go for long periods without reading anything, and may suddenly flip through pages at fast rate.

To characterize the potential benefit of only capturing images in the presence of reading material, we collected images with a Narrative Clip over two and a half weeks. One of our co-authors put the Clip on the front of his shirt in the morning and wore it all day. A typical day consisted of commuting to and from work on foot or on motorcycle, office work at a desk with a laptop computer, eating meals, and walking a dog. The Clip was charged at night while the co-author slept.

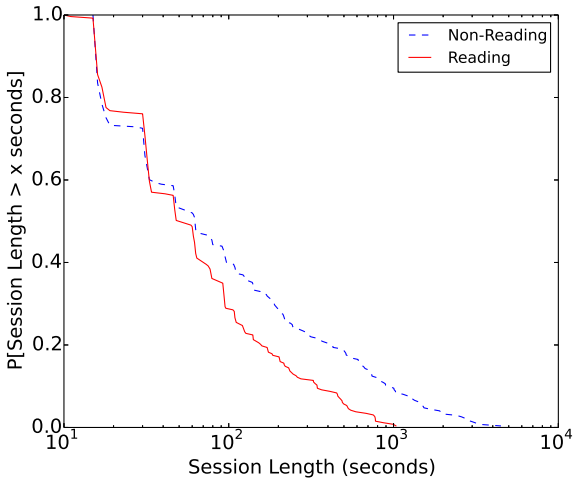


Figure 1: Inverse CDF of lengths (in seconds) of reading (solid line) and non-reading sessions (dashed line) based on trace collected using a Narrative Clip Lifelogger. Non-reading sessions have much longer lengths than reading sessions. Note the log scale on the x-axis.

Using the Clip's default settings, we collected images every 15 to 30 seconds. Due to gaps between images we applied coarse-grained annotations to images indicating only whether they contained reading material or not. For example, we did not differentiate between pages of the same book or screens of the same website.

We manually labeled the collected images based on the following rules: (1) if there was no text in view then the image's label was negative, (2) if the text in view was distant (i.e., out of arm's reach), then we did not consider this reading material and the label was negative, and (3) otherwise, the label was positive since the image contained text that the user was likely interacting with. Overall, we collected approximately 6,100 images, of which 29.7% were labeled positively. This is consistent with our hypothesis that many users spend significant time interacting with reading material, but that the majority of their time is spent doing other things.

We were also interested to know how long a user might typically interact with reading material. To answer this question, we defined a session as a sequence of identically labeled images. The duration of a session was the timestamp of the last image in the sequence minus the timestamp of the first image in the sequence. Our traces yielded 263 reading sessions and 273 non-reading sessions. Figure 1 shows an inverse cumulative distribution function over the lengths (in seconds) of reading and non-reading sessions.

Both reading and non-reading sessions tend to be brief. For reading sessions, 29% of sessions lasted less than 30 seconds (77 sessions), and the median session duration was 48

seconds. For non-reading sessions, 34% of sessions lasted less than 30 seconds (93 sessions), and the median session duration was 62 seconds. However, the distribution of non-reading session lengths exhibited a longer tail, with 25% of non-reading sessions lasting more than four minutes, and the max non-reading session lasting 75 minutes. In comparison, 25% of reading sessions lasted more than one minute, and the max reading session lasted 17 minutes.

These results indicate that (for our data set) reading sessions were relatively brief and frequently interspersed with longer non-reading sessions. This suggests that a lifelogging device may have many opportunities to save energy by not taking pictures, and that taking fewer pictures need not reduce utility. At the same time, the relative rarity of long reading sessions suggests that if a device detects the presence of reading material, then the system may achieve its best utility/Joule by capturing pictures at a relatively high rate during reading sessions. For example, if a user scrolls through a website or flips through pages in a magazine, a device may wish to spend the energy saved during non-reading sessions to increase the likelihood that it captures every word of text.

2.2 Design principles

While the results in Section 2.1 are not definitive, they are consistent with our hypothesis that lifeloggers can better utilize their limited resources by trying to capture only information that users are explicitly interested in. Ayumu's focus is digitizing reading material, and the rest of this section summarizes the design principles that guided our work.

Focus on the task at hand.

The key to efficient lifelogging is spending battery power only when data of interest is available. A *focused task* defines a narrow class of data that a user wants to capture. Current systems capture data periodically and indiscriminately, and offer no way to express what information interests a user. At most, today's devices allow a user to adjust how often they capture images, but this is insufficient for a researcher who wishes to capture only conference-attendee badges, a parent who wishes to capture only pictures of her children, or a dieter who wishes to take only pictures of his meals.

Explicitly narrowing the kind of data that a lifelogger ought to record creates opportunities to save energy wasted by periodic approaches that frequently capture low-utility data. Users and system designers could utilize this potential savings in a number of ways. A user may be happy to charge her device less frequently, or a designer could reallocate the energy savings to capture higher-fidelity data (e.g., by capturing images at a faster rate when useful data is available or by equipping a device with a more powerful, higher-resolution camera). The savings could also be spent shrinking a device's battery so that it has a smaller and more discrete form factor.

Rather than address the question of how energy savings might be used, in this paper we concern ourselves with understanding whether it is feasible to save energy by focusing on specific kinds of data. To this end, we have designed and implemented a lifelogging device for reading material.

Use low-power sensing to trigger recording.

At the heart of Ayumu's design is the observation that reading material can be accurately detected using a set of power-efficient sensors. Our Ayumu prototype relies on three sensors: an IR proximity sensor, a light sensor, and an accelerometer to detect cues indicating that a user is interacting with reading material. The IR proximity sensor has a range of 15 to 150 cm and can detect when a surface is within arm's reach. The light sensor measures ambient-light levels. The accelerometer detects whether the user is moving. These measurements are useful in combination because reading material often resides on nearby flat surfaces that reflect or shine light, such as pieces of white paper or a computer display, and users are typically still when reading.

Although Ayumu targets reading material, we believe that using low-power sensors to detect interesting data will generalize to a wide range of useful data types. For example, audio levels captured by a low-power microphone could be used to detect the presence of people, and voice recognition could be used to detect the presence of friends or strangers. More generally, there may sensors located on other wearable computers, including fitness trackers and smartwatches, that will be helpful for determining when to capture data; certain hand movements may indicate activities such as typing, eating, or writing that are positively or negatively correlated with information the user is interested in. We leave a broader investigation of focused-task lifelogging (i.e., beyond Ayumu) for future work.

Our work thus far is limited to a single task: capturing reading material. Supporting multiple tasks would likely reduce energy savings because useful information would likely be available more often. Our current system does not provide a general programming model for expressing focused tasks, and Ayumu is tightly integrated with the device software. Thus, we leave a thorough investigation of programming abstractions, including efficient implementations of those abstractions, to support general focused-task lifelogging to future work.

Train a classifier instead of hand-tuning parameters.

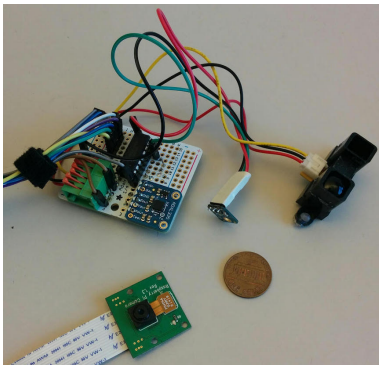
Early in the development of Ayumu, we tried to manually process its light, proximity, and motion sensors, but the results were disappointing. It was difficult to find a good combination of thresholds to indicate when a user was reading. Instead, we found that the most effective approach to using these sensors was to collect in-situ training data and rely on a support vector machine to trigger image capture. Given enough training data, our classifier produced far better results than our hand-tuned parameters, and captured reading material from both hard-copy and digital sources and from upright and angled surfaces. We also found that the computational and memory overheads of runtime classification were minimal.

3. Ayumu

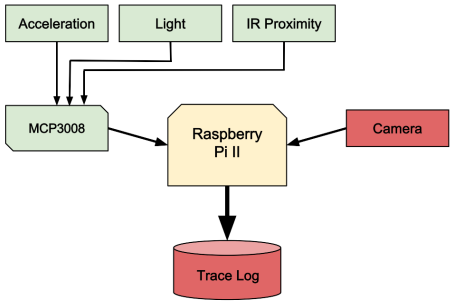
Our Ayumu prototype consists of both hardware and software components. In this section we describe our hardware platform and the software that runs on it.

3.1 Hardware

Because consumer lifelogging devices like the Narrative Clip are closed systems, we built our own lifelogging device using a Raspberry Pi II (RPi) computer [10]. The RPi has a fixed-focus camera module that captures 5MP images of a similar quality to consumer devices. Additionally, we augmented the system with a small suite of sensors, includ-



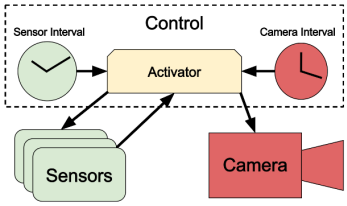
(a) Ayumu Prototype



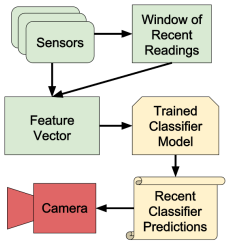
(b) Ayumu Schematic

(c) Ayumu Case

Figure 2: Our Ayumu prototype consists of a Raspberry Pi II, an MCP3008 analog-digital converter connected to an accelerometer, a light, and an IR proximity sensor, as well as a 5MP fixed-focus camera. The case for our sensors can be attached to a standard pair of glasses to be worn by the user. This aligns the camera and proximity sensors with the users line-of-sight.



(a) Ayumu activator



(b) Sensor data-flow

Figure 3: High level overview of Ayumu control inputs and the flow of sensor data

ing a light sensor, an accelerometer, and an IR proximity sensor. Figure 2(a) shows a photograph of our prototype's sensor set, and Figure 2(b) depicts the components of our RPi lifelogger. Since the RPi has only digital inputs, the sensors are connected through an MCP3008 Analog-Digital Converter. This forces Ayumu to sample one sensor at a time, but we have found that readings can be sampled fast enough for our purposes. We use the Picamera Python library to control the camera board. Images are captured at the full resolution of 2592x1944 and stored in jpg format due to the poor performance of writing large images to the SD Card.

These sensors and the camera are housed in a 3D-printed case that can be attached to the right ear-piece of a pair of glasses. This allows the view of the camera and proximity sensor to be more closely approximate the view of the user than a chest-worn badge (which we initially used). Figure 2(c) shows the current form factor. The RPi and battery are kept in a separate case with power and data cables connecting both parts.

3.2 Activating the camera

Ayumu relies on an *activator* to control its camera. As Figure 3(a) shows, the activator uses sensory inputs and timing information to decide when to take a picture. As long as recent sensory input indicates that reading material is present, the activator allows the camera to take pictures at a periodic rate. However, when the activator determines that reading material is absent, Ayumu pauses the camera until reading material reappears.

For simple decisions, the activator could be implemented using hand-tuned rules. For example, strict periodic image capture only requires the activator to handle elapsing timers. However, detecting reading material is complex, and Ayumu's activator relies on a support vector machine (SVM) classifier to control the camera. Section 3.3 describes how we train Ayumu's SVM classifier.

On start up, Ayumu captures sensor readings at a fixed *sensor interval* and feeds those readings to the activator. Our current prototype polls sensors every 200ms. The activator uses new readings and recent history to predict whether reading material is present. Each time new sensor readings arrive, the activator invokes the SVM classifier.

From the device's sensor input, the activator considers 18 features: light level, proximity, acceleration in all three cardinal directions, and acceleration magnitude, as well as the average and standard deviations of these six features over a small evaluation window. After feeding these features to the SVM classifier, the classifier outputs 'yes' or 'no.'

Because individual SVM decisions can be noisy, the activator invokes the SVM classifier multiple times within an evaluation window. Ayumu's default window is 1s. If the number of positive SVM outputs within the window is greater than a pre-defined threshold, the activator can be confident that reading material is present. In this case, the activator turns on the camera, takes an image, and starts a new camera timer.

When a camera timer fires, the activator consults the SVM classifier's most recent set of decisions. If enough decisions are positive, then, as before, the activator takes an image and starts a new camera timer. However, if the SVM has not made enough positive predictions within the evaluation window, the activator keeps the camera off and does not set a new camera timer.

Ayumu continues to poll its sensors as long it believes that reading material is absent. Polling sensor inputs at a much higher rate than the camera interval (e.g., 200ms versus 30s) allows Ayumu to detect reading material far faster than a strictly periodic system. As we will see in Section 4, this

gives Ayumu between 20 and 40% better precision than a strictly periodic system with a 5s camera interval on the data traces we collected.

At the same time, while sensors must be polled continuously as long as reading material is absent, Ayumu can save energy by polling less when a camera timer has been set. In particular, because the activator only uses sensor readings from the evaluation window, Ayumu only needs to poll its sensors for the evaluation period immediately preceding when a camera timer will fire. This can lead to substantial energy savings when there are long stretches without reading material. As we will see in Section 4, for a ten-minute stretch with no reading material Ayumu has an energy overhead of less than 30% of the energy of a system that captures images every 30s.

Figure 3(b) shows how data flows from the sensors to the classifier and camera. The number of recent positive decisions required to power the camera is loosely related to the sensor interval. Generally the longer the sensor interval, the fewer positive readings should be required, since the sensors are read less often. Our default settings are a sensor interval of 200ms, requiring positive predictions within an evaluation window of 1s to trigger image capture. We describe the trade-offs of these settings more in Sections 4.1.3 and 4.2.

Because Ayumu uses its SVM classifier decisions to set camera timers, it will never capture more images than a strictly periodic approach. Both approaches use the camera interval to limit the speed at which images are taken. Thus, the SVM's sensor-based predictions are the sole factor determining whether Ayumu saves energy by taking fewer images.

3.3 Building an Activator

To train Ayumu's SVM classifier, we used our prototype to collect representative traces of sensor readings and images. These traces are described in more detail in Section 4.

To collect traces, a small collection of Python scripts running on the RPi create threads to poll the sensors, trigger the camera, and monitor overall system state. A main control thread detects when the user wants to start tracing through a physical switch on the outside of the device. When a user throws the switch the device operates in tracing mode, and the control thread forks another thread to periodically capture and log images from the camera. While tracing, the RPi periodically collects timestamped images, though images are processed and labeled off-line. Traces must be complete enough to approximate ground truth, and when in tracing mode Ayumu captures images at a much higher rate (e.g., every five seconds) than it would during normal operation. The RPi also periodically collects and logs timestamped sensor readings so that they can be aligned with images captured at the same time.

Once we have collected a trace, we use it to train our SVM. A *moment* is a period of time during which useful information can be captured. For Ayumu, a moment consists of a period of time in which a unique *page* is in view of the camera. Ayumu's goal is to capture as many unique pages as possible using the least amount of energy. The notion of a page is straightforward for most hard-copy reading material, because moving to a new page often involves a physical change, such as turning a page in a book. Defining pages for digital text is less well defined. We assume that two images whose text differs by 50% or more represent unique pages. For example, scrolling down two lines on a long web page is not a new page, but opening a new tab or scrolling down the majority of the screen represents a new page.

Before training, we manually label traces to determine when reading material is present. The rules for labeling are identical to those applied to the Narrative Clip data in Section 2.1: (1) if there is no text, then an image's label is negative, (2) if text is present but distant (e.g., on a billboard), then the label is negative, and (3) if text is present and nearby (i.e., at arm's length or closer), then the label is positive. For evaluation purposes we also recorded the unique identifier of any pages contained in the image, but this information is not used to train our SVM.

Using a labeled trace, we train an SVM using a vector of 18 features. These features include raw sensor data (light level, proximity, acceleration in all three cardinal directions, and acceleration magnitude), the average over a time-based window of those readings, and the standard deviation over the same time-based windows.

It should be noted that device developers, not users, are expected to perform trace collection and SVM training. In particular, manually labeling trace images is tedious and error prone, even for experts. However, allowing a device to learn what specific information interests a user is a promising direction for future research. Rather than simply using a set of predefined classifiers, it may be beneficial to allow a user to communicate her interests by training a device herself. It may also be possible to use image-processing techniques to automatically label trace images. A more in-depth investigation of these issues is beyond the scope of this paper.

3.4 Back-end image analysis and search

To use the images captured by Ayumu, we need to complete the following steps: (1) extract text from all images, (2) break this text into searchable terms, and (3) make terms available for keyword search.

For text extraction, we use Google's Cloud-Vision APIs, which are based on work in deep learning [12]. Once extracted from the collected images, text has to be broken down into component terms. For this, Ayumu uses the NLTK library [20] by breaking the text into tokens with the word_tokenize tokenizer. We stem the resulting word tokens with the EnglishStemmer class, allowing us to provide related search results in addition to exact matches.

The resulting set of tokens is saved into a Redis database with two tables. The first table maps the extracted word tokens to a document ID, and the second maps document IDs to saved image files. Users can query the database through through a simple web page implemented with Python's Tkinter library. Search terms entered by a user are tokenized and stemmed via NLTK, and the resulting terms are checked against Redis. Any returned document ids are mapped to the saved image files, and the user is provided with a clickable link for each image result from their search.

4. EVALUATION

In evaluating Ayumu, we sought answers to the following questions:

- How well does Ayumu capture reading material compared to a simple periodic system?

- Does Ayumu save energy compared to the periodic system?
- Will Ayumu allow users to perform keyword searches over their reading material?

In each of our experiments, we compared our Ayumu prototype to a system that captures images periodically. First, our results show that Ayumu achieves similar recall to the periodic system despite capturing fewer images. Second, our results show that even when polling its sensors every 200ms, Ayumu consumes less than a quarter of the energy used by a periodic system with a camera interval of 30s (which is common for consumer devices). Finally, an end-to-end retrieval experiment shows that Ayumu achieves recall that is comparable to taking pictures by hand and significantly outperforms a periodic system with camera interval of 30s.

4.1 Trace-based simulations

To better understand how well Ayumu captures reading material, we collected five traces using our prototype and hand-labeled the images. Each trace covered approximately 30 minutes, with sensor data collected every 2ms and camera images collected every 2s. We collected this data over several representative periods inside and near our department. While collecting traces, the device wearer performed typical office work and interacted with both printed and digital reading material.

Recall from Section 3 that a moment within a trace is a period when useful information can be captured. Recall too that for Ayumu, a moment consists of a period in which a unique page is in view of the camera. Among our traces, the minimum number of moments was 22 and the maximum was 40. The duration of those moments ranged from several seconds to about a minute. The sparsity of a trace is the percentage of time when there is no reading material. For example, a trace with no reading material has 100% sparsity. Sparsity provides opportunities to save energy, and our collected traces exhibited sparsities ranging from 36% to 43%.

To simulate a periodic system, we chose a camera interval and iterated through trace images in the order that they were captured. If a camera interval would have elapsed between the timestamps of consecutive images, we marked the image with a timestamp closest to the elapsed time as having been captured by the periodic system.

Simulating Ayumu was more involved, because it required training an SVM and processing sensor values. To train the SVM, we performed hold-one-out testing, allowing us to train the SVM on four traces and test the resulting model on the remaining trace. Additionally, for our trace-based experiments, we varied the camera interval. However, unless stated otherwise, we used a fixed sensor interval of 200ms and a fixed evaluation window of 1s.

4.1.1 Images taken

Comparing the energy consumption of different life-logger configurations is difficult. First, there is no standard workload. Second, tethering a device like our Ayumu prototype to a Monsoon Power Monitor would not provide meaningful results since the device wearer would not move naturally. Third, because we had one prototype device, even if we could record a device's energy consumption, we could not ensure that different configurations would be exposed to equivalent camera and sensory inputs across trials.

Thus, to approximate the energy that might have been consumed by Ayumu and a periodic system under our traces, we counted the number of images that each would have taken. For both systems, we varied the camera interval between five and 120 seconds. Figure 4(a) shows the average image count for both systems across all five traces. As mentioned previously, we used hold-one-out testing for Ayumu, and our results represent the average count from each trace when the SVM was trained on the other four.

Our results show that Ayumu consistently captured fewer images than the periodic system, with a maximum difference of approximately 150 images for a camera interval of 5s and minimum difference of only two images for a camera interval of 120s. This indicates that Ayumu consistently took advantage of the traces' sparsity.

Both systems took fewer images as the camera interval increased. This was expected since the periodic system only captures images when its camera interval elapses, and Ayumu captures images at the camera interval as long as it believes that reading material is present.

Note that image count is an imperfect approximation of energy consumption because Ayumu will consume more baseline energy than a periodic approach due to its extra sensors. We investigate this issue in Section 4.2. Nonetheless, it is encouraging that these simulation results are consistent with our hypothesis that additional sensors can save energy.

4.1.2 Precision and recall

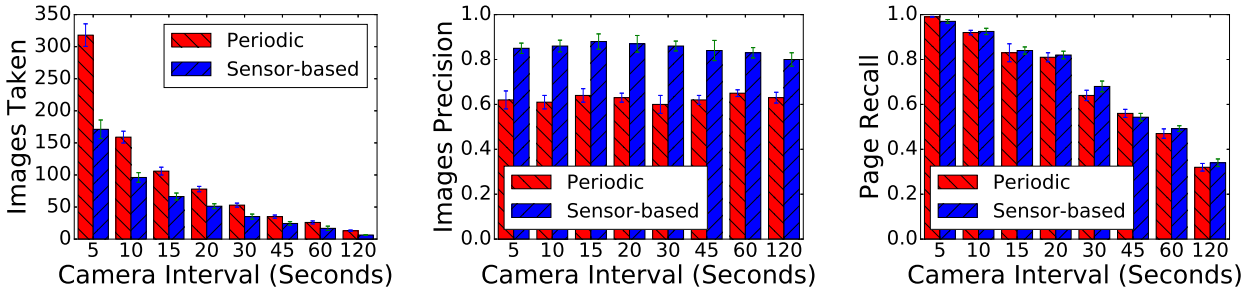
Next we wanted to evaluate the precision and recall of Ayumu and the periodic approach. The *image precision* for a set of images is the percentage of images captured that contain reading material. This suggests how likely a captured image was useful and reflects how much energy a system wasted by capturing low-utility images. Figure 4(b) shows the precision of both systems, with varying camera intervals.

The results show that Ayumu provides far superior precision compared to a periodic approach with the same camera interval. For a camera interval of 5s, Ayumu achieves a precision of 85% while the periodic approach manages a precision of only 62%. These numbers reflect the sparsity we observed in our traces (i.e., between 36% and 43%). That is, given that approximately 60% of the images in our traces contained reading material and the remaining did not, it is not surprising that periodically sampling images resulted in precisions of approximately 60%. On the other hand, Ayumu was able to use its sensors to mostly suppress image capture when reading material was absent.

We can also see that at different camera intervals, the average image precision for periodic capture fluctuates slightly. This is due to the exact alignment of images with the current camera interval. Some of the shown intervals are more "in phase" with the reading material, leading to these slight differences.

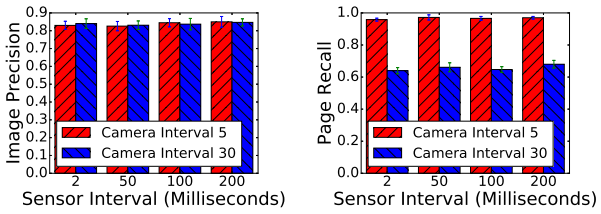
Ayumu achieved a high precision because its SVM classifier produced a low number of false positives. And even when there were false positives, they did not occur with great frequency within the evaluation window.

Of course, a life-logger that captures a small subset of useful images but misses many more would not be practical. Thus, the *page recall* of a set of images is the percent of moments that have been recorded in at least one image. This metric can only be computed when there is ground-



(a) Average # Images Taken (b) Image Precision (c) Page Recall

Figure 4: A comparison of the average number of images captured, the image precision and page recall of Ayumu and the periodic systems as the camera interval is varied from 5s to 120s.



(a) Image precision (b) Page recall

Figure 5: Ayumu's image precision and page recall are not significantly affected by the rate at which sensors are polled.

truth knowledge of how many times reading material was present.

Figure 4(c) show that Ayumu and a periodic approach exhibit similar recall for all intervals. Note that for a 5s interval the periodic system obtains near perfect coverage because it took images faster than almost all moments occurred. As the capture interval increased both systems missed more moments, and recall declined accordingly.

At larger camera intervals Ayumu occasionally exhibited better recall than the periodic system due to better timing. For a 5s interval, Ayumu missed a few moments due to misclassification. This prevented Ayumu from achieving the same near perfect coverage, since a small number of moments were not recorded. Overall, the rate at which page recall declined was similar for both systems as camera intervals increased. This was essentially the best one could hope to achieve with either system, since a higher interval will prevent both systems from collecting enough images to cover moments that occur in quick succession.

In summary, for the same camera interval, Ayumu consistently provides superior precision and similar recall relative to a periodic system. These results indicate that Ayumu's sensor-based approach is likely to waste less energy on capturing useless images.

4.1.3 Sensor sampling rates

For the previous trace-based experiments Ayumu was configured with a sensor interval of 200ms and an evaluation window of 1s. Under these settings Ayumu demonstrated better precision and comparable recall to a periodic system for the same camera interval. A natural question would be how much better Ayumu might perform by polling its sensor more often, even if doing so would cost more energy. We consider sensor intervals of $2, 50, 100$ and 200ms. Recall that the activator consulted recent predictions from an evaluation window of 1s to decide whether or not to turn on the camera. Varying the sensor interval also requires varying the threshold on the number of positive predictions in the 1s window. At sensor intervals of $2, 50, 100$ and 200ms, we use thresholds of $300, 12, 6$ and 3, respectively. Note that these thresholds are equivalent to 60% of the observations in the evaluation window being positive.

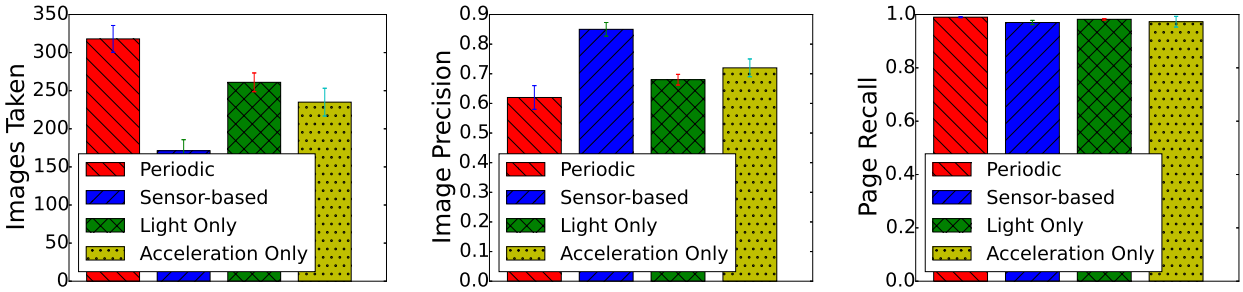
Figure 5(a) shows Ayumu's image precision on our traces by varying sensor interval and fixed camera intervals of 5s and 30s. Similarly, Figure 5(b) shows Ayumu's page recall on our traces for the same sensor and camera intervals.

The results indicate that for our traces, there is no demonstrable reason to poll sensors faster than every 200ms. Doing so would consume far more energy, and neither precision nor recall is changed by polling Ayumu's sensors more often. There is slight variation in both image precision and page recall as the sensor interval increased from 2ms to 200ms, but it was not significant. These fluctuations were due to slight differences between when Ayumu collected an image, leading to slight differences in what sensor readings were available during testing. This also shows that across camera intervals a sensor rate of 200ms is still appropriate.

These results initially surprised us. We suspected that sampling the sensors more often would improve precision and recall. However, given the slow rate at which moments occurred, i.e., on the order of seconds, sampling more frequently than every few hundred milliseconds was overkill. This was particularly true when the evaluation period was also on the order of hundreds of milliseconds.

4.1.4 Necessary sensors

A point raised during our investigation was if a single feature from a cheap sensor could work as well as Ayumu. To answer this we reran our experiments on the trace set with a five second interval, but with two additional baseline systems for comparison. We trained alternative activators using single sensor features, the light-level and acceleration magnitude. We found that while the single-sensor systems did manage to filter out a subset of instances where an image should be skipped, they did not do as well as the full version of Ayumu. This led to a comparable page recall (Figure 6(c)), but with more images (Figure 6(a)) and a less favor-



(a) Average # Images Taken (b) Image Precision (c) Page Recall

Figure 6: A comparison of the average number of images captured, the image precision and page recall of Ayumu and the periodic systems as the decision method for capturing images is changed while using a capture interval of 5 seconds.

able image precision (Figure 6(b)) compared to Ayumu.

4.2 Energy measurements

The second question we wanted to answer in evaluating Ayumu is whether it is likely to save energy compared to the periodic approach. Our primary goal was to save energy by using sensors to suppress the camera, but if the camera interval is long enough the energy savings may not materialize. In particular, if Ayumu's sensor polling consumes more energy than a periodic system's capturing low-utility pictures, then the periodic approach will be more energy efficient. Furthermore, users may prefer to capture extra images if using sensors does not save energy, since battery capacity is more constrained than device storage.

To investigate this issue, we used a Monsoon Power Monitor to measure the energy consumed by our Ayumu prototype under different system configurations. We did this by powering the RPi from the Monsoon system to directly measure voltage and current draw.

The RPi consumes more energy than alternatives with less hardware flexibility. As such, all of the following energy measurements are reported as the difference between the energy consumed in the test and the energy that would be consumed if the RPi was allowed to idle without using the camera or sensors. When idling for 10 minutes, the RPi consumed an average of 731 Joules.

Our energy experiments were performed by running our prototype for exactly 10 minutes with nothing in front of the sensors. Thus, the workload for our experiments is a ten-minute period without reading material. As before, Ayumu ran with a 200ms sensor interval and a 1s evaluation interval. Figure 7 shows our results. One thing to note is that the y-axis is log-scale to allow us to show the three blue lines for Ayumu, indicating the average energy consumed as well as the standard deviation.

First, the periodic approach consumed less energy over idle as camera intervals increased. This is unsurprising since the system spends less energy taking pictures at low camera intervals. Ayumu's power during the experiment is represented by the dotted line and is a constant 6.7 Joules, shown across all camera intervals. This is also unsurprising since Ayumu's sensors never allow the camera to capture an image. This energy consumption represents the additional energy consumed by Ayumu over idle.

As expected, the biggest savings was for a camera interval of 2s, when the periodic approach consumed nearly 240 Joules above idle compared to Ayumu's 6.7. For a camera interval of 15s, Ayumu required approximately one-third the energy of the periodic approach above idle. For a camera interval of 30s, Ayumu required approximately a quarter the energy of the periodic approach above idle. Ayumu's energy consumption above idle intersected the periodic system at a camera interval of 90s.

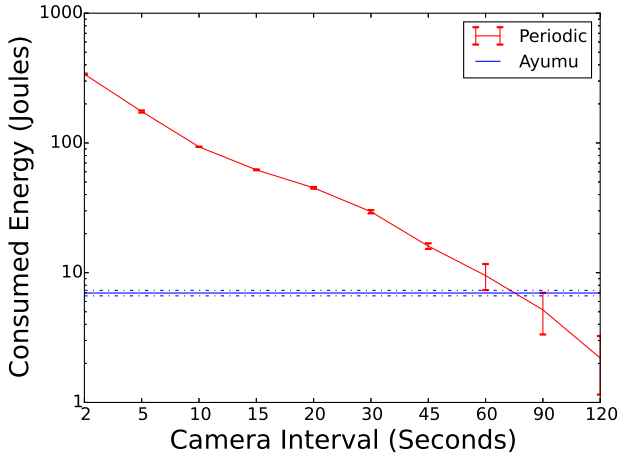


Figure 7: Energy of Camera Use vs Ayumu

Extrapolating from these results is not straightforward since Ayumu will consume a nearly identical amount of energy as the periodic approach with the same camera interval when reading material is always present. This is because Ayumu only polls sensors for the evaluation window preceding image capture after a camera timer has been set. Nonetheless, these results strongly suggest that for sparse workloads Ayumu will save energy compared to a periodic system when both are configured with a typical camera interval, such as 15s or 30s. In addition, as we saw in our other experiments, Ayumu provides superior precision and comparable recall for these intervals.

4.3 End-to-end text extraction

To understand how well Ayumu will allow a user to use

keyword search to locate an image, we performed experiments with four short stories: The Gift of the Magi by O. Henry, The Little Match Girl by Hans Christian Andersen, To Build a Fire by Jack London, and An Occurrence at Owl Creek Bridge by Ambrose Bierce. We selected these texts because they were the first four items in a list of American short stories [2]. These texts collectively yielded 6,503 words, of which 2,612 are unique. Figure 8(a) shows the occurrence distribution for the unique words.

To train Ayumu, we used a different dataset than we used in the trace experiments. This training data was collected in and around the co-authors' office while working. We labeled this data by hand, as described previously.

To model how a user might search these texts, we derived four groups of possible search terms: the head (words appearing more than 10 times), the torso (more than five times and at most 10), the tail (two to five times), and the singletons (only once). These groups contain 84, 147, 780, and 1601 words, respectively.

We copied the stories from the web into a word-processing program (preserving the online formatting), and printed the resulting document. A co-author then read the printed stories while wearing Ayumu. While reading, the device wearer performed other office activities, such as getting coffee, walking through the halls, and talking with graduate students. The setting for the experiment was around and immediately outside of the department. Locations included offices, couches, hallways, kitchenettes, and exterior seating. We repeated this procedure twice: once with our device configured for 5s periodic capture and once as Ayumu. For each session, the user spent approximately forty minutes reading and doing other normal activities with both systems. Finally, to get a better idea of what levels of recall are achievable an additional set of stable images were taken to compare to both systems. We collected the stable image set by taking two pictures of each page in the test documents by hand.

We uploaded the images captured during each session to the back-end for image analysis. Following text extraction and indexing we computed recall numbers for words in all four query-groups across Ayumu, the periodic systems and the stable image set. Figure 8(b) shows the recall for all four query groups when run on the collected data. The recall of a single query is the fraction of pages correctly identified by the search to the total number of pages that should have been identified by the search.

We note that the recall for both systems never reaches 100%. This missed text could be due to difficulties in processing noisy images or a missed moment where an additional image could have been captured. When running the query groups across the stable image set we see an approximate recall of 96% across all query groups. This is our optimal baseline for text extraction when given a complete set of clear images.

The periodic system's recall is clearly related to the number of moments missed due to the current camera interval. With a 5s interval the periodic system performs about 3% worse than the stable images. For the 15s interval we see that the periodic system still does well, but with a slight loss of accuracy due to less frequent capture. At 30s and 60s intervals we see a dramatic reduction in recall because the user is reading pages faster than images are captured. It should be noted that these recall numbers are higher than in Figure 4(b). This is a result of our ground-truth text from the short stories. The text resides on clear, plain printed pages, and were often in view for around 20 seconds (the time it took to read through a single page).

Ayumu achieves a slightly lower recall for all tested camera intervals. Since Ayumu takes images only up to as frequently a periodic system with the same interval, this is expected. The specific cause for this difference could be that Ayumu missed a chance to take an image right as text came into view, or that the exposures in the Ayumu trace were slightly noisier, leading to less accurate OCR.

Figure 8(c) shows that Ayumu provide superior recall than the periodic system with far fewer images. Specifically, Ayumu with a camera interval of 5s takes approximately as many pictures as the periodic system at 10s. Ayumu with a camera interval of 15s has as high a recall (>80%) as a periodic system with a camera interval of 5s or 15s, but takes approximately as many images as the periodic system with a camera interval of 30s. This shows that when using Ayumu we can achieve high recall on a reduced set of images, which should save energy in practice.

5. RELATED WORK

The consumer market for cheap, stylish, and small lifeloggers has been growing steadily over the last 10 years, taking its initial leap with the introduction of the Sensecam system. More recently, similar devices such as the Autographer and Narrative Clip have found their way into the hands of users.

5.1 Consumer Devices

Below we explain the design of such systems and their current operation.

5.1.1 Hardware

The main design goal of consumer lifelogging systems is to allow operation over at least the duration of a full waking day (or approximately 12 hours), additionally striving to take regular images across that period with minimal blurring and a good light-balance. To achieve this, most available devices come equipped with a basic set of sensors, generally including an accelerometer and a light sensor. The use of these sensors though is limited to increasing the quality of captured images and toggling power-saving sleep modes when a devices is placed face-down or in a bag or pocket.

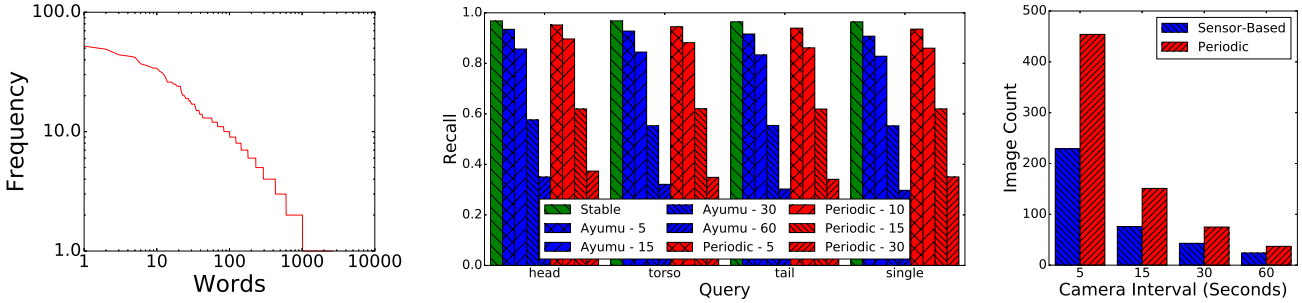
Generally, images are set to be taken consistently at some predetermined interval. The default on the Narrative for example is to take one image every 30 seconds. Over the course of one day of operation this can amount to as many as 1,440 images evenly distributed over a 12 hour period.

5.1.2 Image Analysis

Back-end software and algorithms have been developed for a number of information extraction tasks. One section of this work focuses on what can be extracted from the images collected. This includes work on event segmentation [6], key frame extraction [7], and lifestyle trait recognition [8].

Additional work has been done to incorporate sensors readings that are available to a lifelogging system. This includes analysis of GPS readings in the context of food consumption [3], more efficient memory mediation [14], and efficient retrieval of lifelog data based on indices created over sensors readings, audio, and collected video [1].

[2]https://americanliterature.com/twenty-great-american-short-stories



(a) Word frequencies in documents

(b) Query workload recall

(c) # Images captured

Figure 8: Results from the Ayumu end-to-end evaluation. The frequency of word occurrence in our documents follows a power law. We also show the document recall (under different query workloads) and the number of images captured for Ayumu and periodic systems.

As of the writing of this paper, the Narrative Clip system provides (with subscription) access to their server to select "moments" from the total set of collected images. This requires that users upload all of their images to Narrative servers. Moments are curated and presented to the users through a mobile application, but the process of selecting moments is hidden from view.

5.2 Human Activity Recognition

The recognition of human activity has been studied in a number of ways, and recent work will be useful in the context of this project.

Traditionally research in human activity recognition has been led by researchers in computer vision. This includes work for gesture recognition (recognizing sign language) [26] and action recognition in video [29].

More recently, human activity recognition has been achieved via the use of body-worn sensors. Directly measuring the movement of human subjects has led to work in medical diagnosis and rehabilitation [5], as well as the tracking of basic health actions (such as when the user takes their medication or brushes their teeth) [2].

Beyond work in custom body-worn sensors, research has turned to how current generation smartphones can be used for human activity recognition. This shift into mobile-based online recognition has been driven by the fact that mobiles are easy to obtain, come pre-equipped with sensors, have the ability to perform basic computation, and are able to communicate via Wi-Fi or cell networks to offload data and computation. Research on using mobiles as a sensing and computing platform have led to work ranging from basic activity recognition [16] to systems for real-time assistive feedback [15].

5.3 Visual Information Extraction

Finally there exists a large amount of work in extracting information from visual data. Standard computer vision research in this area covers tasks such as text detection and extraction [28, 9], face detection and tracking [27], as well as scene change detection [21]. For most of these interesting visual information extraction tasks, data must be captured at a higher rate than consumer lifelogging systems achieve during normal operation.

For the task of text extraction our first effort consisted of standard image binarization techniques [24, 22]. The results were then fed into the Tesseract OCR engine [25] to extract available text. We found the resulting quality of the extracted text to be unsatisfactory for our system due to the fact that most basic image binarization techniques required much parameter tuning, and were still fragile with respect to different text sizes/formats. Following this we moved to the Google Cloud-Vision service shortly after its public beta release [12]. Due to the sophisticated deep-learning model built by Google underlying this service we were able to acheive much better text extraction results without image pre-processing, even in the presence of slight noise in our images.

6. CONCLUSION

In this paper, we present our idea for a focused lifelogger for the detection of user interaction with text versus the indiscriminate capture of content. Having reviewed the available commodity systems, we know that there is not a current consumer device that attempts to assess what is happening before it collects an image.

For the focused task of user-text interaction we have shown that a suite of three cheap and energy efficient sensors (light, proximity, and acceleration) is able to accurately differentiate between when the focused task is and isn't occurring, resulting in a high percentage of relevant collected images for the desired recording task. In the end, reducing the system's energy consumption and the total number of images collected.

Additionally we have shown that the rate at which the sensors must be used to accurately assess when images should and should not be taken can be very low, resulting in energy savings even at extremely long camera intervals.

7. REFERENCES

[1] K. Aizawa, D. Tancharoen, S. Kawasaki, and T. Yamasaki. Efficient retrieval of life log based on context and content. In *Proc. 1st ACM Work. Contin. Arch. Retr. Pers. Exp. - CARPE'04*, page 22, New York, New York, USA, oct 2004. ACM Press.

[2] O. Amft, H. Junker, and G. Troster. Detection of eating and drinking arm gestures using inertial body-worn sensors. In *Ninth IEEE Int. Symp. Wearable Comput.*, pages 160–163. IEEE, 2005.

[3] A. H. Andrew, K. Eustice, and A. Hickl. Using location lifelogs to make meaning of food and physical activity behaviors. *Int. Conf. Pervasive Comput. Technol. Healthc.*, pages 408–411, 2013.

[4] Autographer. Autographer home page and description, 2015. http://www.autographer.com.

[5] J. Chen, K. Kwong, D. Chang, J. Luk, and R. Bajcsy. Wearable sensors for reliable fall detection. *Conf. Proc. ... Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. IEEE Eng. Med. Biol. Soc. Annu. Conf.*, 4:3551–4, jan 2005.

[6] A. Doherty and A. Smeaton. Combining Face Detection and Novelty to Identify Important Events in a Visual Lifelog. In *2008 IEEE 8th Int. Conf. Comput. Inf. Technol. Work.*, pages 348–353. IEEE, jul 2008.

[7] A. R. Doherty, D. Byrne, A. F. Smeaton, G. J. Jones, and M. Hughes. Investigating keyframe selection methods in the novel domain of passively captured visual lifelogs. In *Proc. 2008 Int. Conf. Content-based image video Retr. - CIVR '08*, page 259, New York, New York, USA, jul 2008. ACM Press.

[8] A. R. Doherty, N. Caprani, C. Ó. Conaire, V. Kalnikaite, C. Gurrin, A. F. Smeaton, and N. E. O'Connor. Passively recognising human activities through lifelogging. *Comput. Human Behav.*, 27(5):1948–1958, sep 2011.

[9] C. Dorai and R. Bolle. Automatic text extraction from video for content-based annotation and retrieval. In *Proceedings. Fourteenth Int. Conf. Pattern Recognit. (Cat. No.98EX170)*, volume 1, pages 618–620. IEEE Comput. Soc, 1998.

[10] R. P. Foundation. Raspberry pi foundation home page, 2015. https://www.raspberrypi.org/.

[11] Y. Y. Hasan and L. J. Karam. Morphological text extraction from images. *IEEE Trans. Image Process.*, 9(11):1978–83, jan 2000.

[12] K. M. Hermann, T. Kočiský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching Machines to Read and Comprehend. page 14, jun 2015.

[13] S. Hodges, L. Williams, E. Berry, S. Izadi, J. Srinivasan, A. Butler, G. Smyth, N. Kapur, and K. Wood. *SenseCam: a retrospective memory aid*, volume 4206 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, sep 2006.

[14] V. Kalnikaite, A. Sellen, S. Whittaker, and D. Kirk. Now let me see where i was. In *Proc. 28th Int. Conf. Hum. factors Comput. Syst. - CHI '10*, page 2045, New York, New York, USA, apr 2010. ACM Press.

[15] N. D. Lane, M. Lin, M. Mohammod, X. Yang, H. Lu, G. Cardone, S. Ali, A. Doryab, E. Berke, A. T. Campbell, and T. Choudhury. BeWell: Sensing Sleep, Physical Activities and Social Interactions to Promote Wellbeing. *Mob. Networks Appl.*, 19(3):345–359, jan 2014.

[16] S. L. Lau, I. Konig, K. David, B. Parandian, C. Carius-Dussel, and M. Schultz. Supporting patient monitoring using activity recognition with a smartphone. In *2010 7th Int. Symp. Wirel. Commun. Syst.*, pages 810–814. IEEE, sep 2010.

[17] H. Li, D. Doermann, and O. Kia. Automatic text detection and tracking in digital video. *IEEE Trans. Image Process.*, 9(1):147–56, jan 2000.

[18] R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl. Energy characterization and optimization of image sensing toward continuous mobile vision. In *Proceeding 11th Annu. Int. Conf. Mob. Syst. Appl. Serv. - MobiSys '13*, page 69, New York, New York, USA, jun 2013. ACM Press.

[19] C. Liu, C. Wang, and R. Dai. Text detection in images based on unsupervised classification of edge-based features. In *Eighth Int. Conf. Doc. Anal. Recognit.*, pages 610–614 Vol. 2. IEEE, 2005.

[20] E. Loper and S. Bird. NLTK: The Natural Language Toolkit. page 8, may 2002.

[21] J. Meng, Y. Juan, and S.-F. Chang. Scene change detection in an MPEG-compressed video sequence. In A. A. Rodriguez, R. J. Safranek, and E. J. Delp, editors, *IS&T/SPIE's Symp. Electron. Imaging Sci. Technol.*, pages 14–25. International Society for Optics and Photonics, apr 1995.

[22] S. Milyaev, O. Barinova, T. Novikova, P. Kohli, and V. Lempitsky. Image binarization for end-to-end text understanding in natural images. In *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, pages 128–132, 2013.

[23] Narrative. Narrative clip home page and description, 2015. http://getnarrative.com/.

[24] F. Shafait, D. Keysers, and T. Breuel. Efficient implementation of local adaptive thresholding techniques using integral images. *SPIE Doc. Imaging Retr.*, pages 1–5, 2008.

[25] R. Smith. An overview of the tesseract OCR engine. In *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, volume 2, pages 629–633, 2007.

[26] T. Starner, J. Weaver, and A. Pentland. Real-time American sign language recognition using desk and wearable computer based video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(12):1371–1375, 1998.

[27] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. 2001 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition. CVPR 2001*, volume 1, pages I–511–I–518. IEEE Comput. Soc, 2001.

[28] F. M. Wahl, K. Y. Wong, and R. G. Casey. Block segmentation and text extraction in mixed text/image documents. *Comput. Graph. Image Process.*, 20(4):375–390, dec 1982.

[29] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden Markov model. In *Proc. 1992 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 379–385. IEEE Comput. Soc. Press, 1992.

[30] Q. Ye, Q. Huang, W. Gao, and D. Zhao. Fast and robust text detection in images and video frames. *Image Vis. Comput.*, 23(6):565–576, jun 2005.