Rewind: Leveraging Everyday Mobile Phones for Targeted Assisted Recall

Donnie H. Kim, Nicolai M. Petersen, Mohammad Rahimi, Jeff Burke, Deborh Estrin
Center for Embedded Networked Sensing, UCLA
{dhjkim, nmp}@cs.ucla.edu, mhr@cens.ucla.edu, jburke@remap.ucla.edu, destrin@cs.ucla.edu

Lenore Arab
David Geffen School of Medicine, UCLA
larab@ucla.edu

This paper presents an assisted recall system, Rewind, that employs automatic image capture on mobile phones and filtering of images for end-user viewing. The usability of image-based assisted recall systems is limited by the large number of images through which the user must navigate. Rewind is a scalable system of everyday mobile phones and supporting web services that we developed to explore how client- and server-side image processing can be used to both lower bandwidth needs and streamline user navigation. It has been in use since August 2007 as part of a pilot study supervised by an epidemiologist to evaluate its utility for improving recall of dietary intake, as well as other shorter and more exploratory trials. While the system is designed to accommodate a range of image processing algorithms, in this first prototype we rely on simple filtering techniques to evaluate our system concept. We present performance results for a configuration in which the processing occurs on the server, and then compare this to processing on the mobile phones. Simple image processing on the phone can address the narrow-band and intermittent upload channels that characterize cellular infrastructure, while more sophisticated processing techniques can be implemented on the server to further reduce the number of images displayed to users.

Categories and Subject Descriptors
J.3 [Life and Medical Sciences]: Health

General Terms
Design, Experimentation, Human Factor

Keywords
Urban Sensing, Urban Computing, Mobile phones, Automatic image capture, Image processing, Prioritized upload

1. INTRODUCTION
People increasingly carry mobile phones with them everywhere and all the time, across cultures, countries, and demographics. These mobile phones increasingly contain imaging and location capabilities, (i.e., high resolution cameras and GPS). We are exploring innovative applications that leverage the use of image and location data captured by mobile phones as they move around in the environment with their users. In this paper we document and evaluate one such application type - assisted recall.

Assisted recall systems support individuals by recording aspects of the environment around them for later playback. They can assist people with healthy memories who want to track additional details, those with memory impairments or perceptual problems due to injury or illness, or in self-improvement where an individual wants to become more conscious of their habits. Additionally, scientists can use them as tools to study the impact of individual behaviors for which there are no direct measures suitable for longitudinal study, such as foods consumed or numbers of people interacted with [5, 15, 21].

In contrast to lifeblogging systems such as MyLifeBits [9, 10] and SenseCam [13], which attempt to collect as much data as possible for future mining, Rewind, our system for image-based assisted recall, supports data collection on specific behaviors or situations. By targeting recall, we can take advantage of the application’s constraints in capture, presentation, processing, and upload. In particular, our system (i) reduces burden on the end user by employing application-specific signal processing to decrease the amount of information presented, (ii) lowers bandwidth through prioritized or selective upload based on this processing (iii) pursues a conservative, application-specific approach to personal data collection and retention, thus reducing privacy risks.

To evaluate its utility in improving dietary recall, Rewind has been in use since August 2007 as a supplemental tool in an NIH-funded study on dietary intake. We have also designed it to explore other image-based targeted recall applications, such as the PosterSession-Capture Pilot described in Section 4.3. Our use of everyday mobile phones and web services is motivated by low cost and easy availability for the populations in our specific studies, as well as a desire for extensibility to a broad range of applications in the future.

The use model is simple. For example, in the dietary intake pilot, study participants launch the Rewind application on their mobile phone before eating, place the phone around their neck on a lanyard with the camera facing out and down, and then turn it off at the end of the meal. When they visit the study facility on the next day, images captured during their meals will already have been automatically uploaded and tagged with image-quality metrics. Participants log-in to a secure image viewer to browse the best of their own food images while taking a dietary survey.
To be effective for this application, Rewind must sample images at a high enough sampling rate to capture the “dynamics” of eating, such as the participant taking additional servings, leaving items on plates that are returned, etc., and often run for an hour or more at a time several times a day. It must automatically upload and then process the images to select a small representative set for the users to view during their visit. At the rapid sampling rate needed, a single user captures too many images for manual investigation. In the DietaryRecall Pilot, each user generated an average of more than 100 images per meal.

Based on this and other similar scenarios, we designed Rewind to:

- Automatically capture and cache images at a rate that would not be tolerable if the user had to do it manually,
- Perform both handset- and server-side processing of images to generate simple classifications and quality metrics selected and tuned for the specific application target,
- Prioritize images for upload based on mobile-phone-based annotations to mitigate the impact of channel congestion on the rapid availability of representative images to the end user,
- Support rapid experimentation with different image processing algorithms as part of an iterative design process with application experiences,
- Display images for the user in a private on-line gallery, where they are clustered in time and filtered based on the application-specific annotations, and,
- Enable the user to explicitly select images to be stored or shared within the target applications’ privacy constraints, while assuring that all images not marked for sharing are deleted.

**Contributions.** To the best of our knowledge, Rewind is the first targeted assisted recall system using everyday mobile phones. Our contributions include:

- Conceiving, building, and deploying in a “live” medical study this type of platform, with configurability and extensibility enabling other pilots that we will also describe.
- The use of web services and image processing techniques for server-side operations.
- The use of a password protected, user-data workspace model for staging of potentially-private data until it is reviewed and released or deleted by the user.
- Demonstration and evaluation of on-device processing of images to reduce and prioritize upload over bottle-necked channels when real-time and interactive access to the data is needed.

This continues our early experimentation with using mobile phones for monitoring dietary intake [20]. Several significant new challenges are addressed through a new system design with modular and flexible components that will accommodate larger number of phones and improve performance and scalability. Additionally, we gained a wealth of practical experience through collaboration with medical research partners and real deployment. Human subjects research requirements on their study required the development of a model for personal data capture addressing privacy and security issues.

### 2. RELATED WORK

There is significant mobile and pervasive computing research that explores wearable image and audio recording devices. Several investigations use wearable image capture devices for creating a multimedia diary of one’s life. Typically, there is no a priori focus on particular situations or behaviors to capture; Rather the goal is to archive the data and, thus, life, in its entirety. The many challenges include: meaningful organization and indexing of the huge image corpus, user interface design, and effective navigation of the information [4, 8, 11]. Notably, SenseCam [13, 10] explores usage of a wearable device worn on a pendant hung from ones neck and has a combination of image, accelerometer, infrared and temperature sensors. It captures images either continuously or based on triggering events from its other sensors and images can be subsequently transferred to a host PC for archival and visualization [12, 14, 10, 19]. Kapur et. al. tested the SenseCam to aid autobiographical memory in patients with severe memory impairment [5]. Subsequent research developed tools to navigate through the SenseCam images and effectively visualize the captured information [6].

Many factors distinguish our work from other assisted recall research. First is the usage of the phone itself. By using mobile phones that are ubiquitously available, we enable future deployments and larger scale experimentation. Therefore we have designed our system to support a large number of phones and users. A second distinction is the usage of the information in our system. In our research, the objective is capturing images for improving a specific, targeted, application. For instance, in the dietary experimentation, the objective is to improve the ability of the participants to recall their daily food intake. This targeted capture also lends itself to the third party usage of the data such as by a medical researcher once the subject has reviewed and released data. The selective sharing of the image data is critical to individual privacy protection and has implications for both system design and evaluation.

Several works in image processing and computer vision literature are relevant to the assessment of application-specific quality of the captured images. This includes characterization of both quality and relevance of the content of the images. The main difficulty in assessment of the image quality is a consistent metric independent of illumination changes. Many of the image processing techniques rely on holistic illumination-independent features such as a histogram of coefficients of Discrete Cosine Transform (DCT) or Harr wavelet transform to assess the quality of the images [18, 23]. There is also a multi-decade research thread in content-based assessment of image information. These research efforts have roots in pattern recognition and object classification to use low-level features in order to find similarity in the content of visual information [2, 7, 17, 24]. In the version of our system described here we have integrated the simplest of image processing filters and hope to use our pilot deployment experiences and image collections to engage image processing and computer vision researchers in exploration and development of more sophisticated and effective techniques.

Finally, we acknowledge the deep challenges in understanding the process of memory and integrating that understanding into the design of an effective assisted recall system. The related work in cognitive psychology to study human autobiographical memory, factors that influence various forms of human memory and effectiveness of multimedia diaries in improving human recall are of significant importance in designing future deployments and experiments [15, 21]. We hope Rewind can serve as an experimental tool for
psychologists to both test and integrate their knowledge in future experiments and applications. With respect to dietary recall mechanisms in particular, several recent studies have indicated the superiority of 24-hour recall over longer-time-period food frequency surveys and we designed our system to work alongside these computerized 24-hour recall applications [22].

3. SYSTEM DESIGN

The system (Figure 1) is comprised of 1) mobile phones, carried by the users, running Campaignr data collection software [16] to capture and upload data 2) a Data Management Server (DMS) managing data flow among system components and providing a user interface to view captured images, and 3) an Image Processing Server (IPS) tasked by the Data Management Server to process, classify, and annotate the images. The Data Management Server interacts with the rest of the components through a RESTful HTTPS interface, and Rewind uses industry standard SSL/TLS encryption, and authenticated transmission, of the data from the phone to the Data Management Server.

Campaignr is configured to capture images (or audio) and meta-data at predefined intervals using a simple XML configuration file. When connectivity is available, it pushes data to the Data Management Server, which stores media files in a secure file system and all other data in a relational database. An asynchronous batch process extracts unannotated images sequentially, requests image analysis and annotation by the Image Processing Server, and generates a thumbnail. The Data Management Server provides a password-protected web interface to each user’s system-annotated data. When users log-in, the Data Management Server uses the image annotations to select which data to display. In the case of the dietary intake pilot, these are the “highest quality” images, as will be explained below. This interface enables users to decide what data is saved permanently and what is shared with selected users of the system. All other data is deleted.

By using low cost and reliable mobile handsets, interconnecting system components through standard web services, and a web-based viewer for end users, we hope to enable future adoption and scaling up in the number of users. By separating the time-consuming and application-dependent image processing tasks from the core data flow services, it is straightforward to add additional Image Processing Servers to support a larger number of images and thus a larger number of mobile phones operating simultaneously. Later in the paper, we explore additional benefits and trade-offs in pushing image processing out to the mobile handsets themselves. The remainder of this section describes each component of Rewind in more detail.

3.1 Mobile Phones

Rewind uses Nokia N80 phones running the Symbian v9.1 Operating System with the S60 3rd Edition User Interface [3]. Each has a 3 mega pixel camera and 802.11b/g connectivity in addition to a GSM modem. These mobile phones run Campaignr [16], an application written for S60 phones that can acquire data from the many hardware sensors on the cell phone, including the camera and microphone. Campaignr can also acquire data from software sensors such as date, time, and text strings. Campaignr stores the data in an internal database on the mobile phone immediately after collecting from the sensors so that no successfully sensed data is lost due to external factors such as bad connectivity. After the data is stored, Campaignr attempts to upload it to the Data Management Server described below. Campaignr reads a few rows from the database, packages them, and attempts to transmit the data over HTTP. Only upon a success signal from the Data Management Server, Campaignr will delete the corresponding data in the database. This may lead to duplicate entries in the on-line database, but they are easy to distinguish by users, or automated processes running on the Data Management Server. The cost of removing duplicates or the additional memory requirements of storing redundant images is insignificant when compared to irrecoverable data lost between the phone and the server. An application-specific XML file is used to specify which sensors to collect data from, and where to upload the data to. Each XML-file captures the details of the specific data collection campaign. This allows for many different types of data collection experiments to be run without having to modify the application itself.

The DietaryRecall, FullDayDietary, and PosterSessionCapture Pilots described in this paper prompted the addition of configurable options that can be expressed in the campaign XML file. The DietaryRecall Pilot designed for a broad population of untrained users needed a configurable user interface for Campaignr. Elements, such as the font size, notification area, and soft-key options were added to support this. The FullDayDietary Pilot exposed Campaignr database corruption issues resulting data lose when the memory is full because it involved the longest period of continuous collection coinciding with key periods of disconnection. For the PosterSessionCapture Pilot, we added an audio annotation feature to Rewind campaign to let participants record audio annotations about the research posters. The PosterSessionCapture Pilot required user feedback options that are not visual as the phone was being worn in such a way as the screen was not easily accessible. To support this, each time an audio recording was initiated or terminated by the user the phone would vibrate, letting the user know the event had occurred without having to check the phone’s screen. Figure 2 shows images of the mobile phone used for Rewind applications. While there are significant differences in the needs of each campaign, we were able to make only slight modifications to the existing system to accommodate.
3.2 DMS: Data Management Server

In this section, we describe the Data Management Server which provides core functionality of the system including a web interface used for data communications, an asynchronous batch image processing component called Image Handling Service, standard security practices used for data protection and user authentication, and finally a user interface developed for presentation and personal annotation.

3.2.1 Web Interface

The Data Management Server receives images from Campaignr equipped phones over HTTPS as described above. Received images and tagging information (time stamp, audio, etc.) are stored in a secure file system and a relational database. Image handling services that schedule processing based on the data these data also communicate through this web interface to retrieve images and their meta-data from the secure repository, as well as to send back the processing results for storage. All communications are through URLs. The web interface was chosen over a custom designed server protocol because of its simplicity and ease of adaptation and extension.

3.2.2 Image Handling Services

Image handling services periodically check if processing is required for each image. A batch process wakes up and implements the workflow of the system. As shown in Figure 1, three batch processes are currently implemented as cron jobs. The Resizing process generates a thumbnail of each image to allow many images to be displayed in a single web page without stressing the web browser. It also improves the download performance by reducing the bandwidth consumed. The Reaping process permanently deletes images that users marked for deletion through the Image Viewer, as well as images that were never shown to the user due to the image filtering process. The Reaping process requests a feed from the web interface to learn which files should be preserved and assumes all others should be deleted. The Image processing process pushes all not-yet-annotated images to the Image Processing Server and retrieves annotation results for each image. This annotation information is used by the Image Viewer to filter and prioritize the images that are displayed to the users. All three of the currently implemented image handling services interact with the secure repository through DMS’s web interface.

3.2.3 Data Protection/Security/Privacy

Rewind potentially captures significant personal information, from the faces of family members across the breakfast table, to the details of email and e-commerce displayed on the users computer screen, and even the occasional image accidentally taken within a restroom. Therefore, privacy of automatically collected personal data had to be addressed even in the earliest prototyping phases of these systems. Rewind web services adopts standard security practices. All communication uses secure HTTP connections, in particular, HTTP over SSL with a X.509 public key certificate created for the web server. The web-server does not store any identifiable information about the users. Images are viewable only by the individual who owns them, unless they explicitly choose to share the images. In particular, for the DietaryRecall Pilot discussed below, images were deleted automatically if not viewed, or if marked for deletion, within 72 hours after capture.

3.2.4 The Image Viewer

Users are given authenticated access to the Image Viewer (Figure 3) with a user ID and a password. As the camera phone operates in an autonomous collection mode, gathering data without user intervention, many images might not be useful for assisting recall. To support image viewing, annotation, and recall more effectively, the Image Viewer filters out low quality images and displays only a selected set of images to the user. Each image is converted into a thumbnail to enhance download performance and labeled automatically with four image feature metadata tags by the image handling services. These tags are explained in more detail below and are used for image filtering prior to presenting images to the user. The Image Viewer first automatically filters out blurred, over- and under-exposed images, and then temporally clusters the images to reduce redundant images by showing only a few representative images from each time cluster. We have currently implemented a simple uniform time interval clustering for display, but other more sophisticated, and likely more application-specific, clustering methods could be substituted and should be developed as future work to increase the information content of each image displayed to the user. Our modular system design will facilitate exploration and introduction of improved techniques.

3.3 IPS: Image Processing Server
Our manual inspection of the images captured by the mobile phones during the FullDayDietary Pilot indicated a distinctive difference among the quality of the images. In particular, a large subset of the images were under-exposed or over-exposed due to sudden changes in the luminance or non-ideal dark or bright lighting. In addition, there were many cases where the images were blurred due to the movement of individual who was carrying the camera or the subjects captured in the image. We developed the following approach to support this process, but generalized the system so as to enable more experimentation and other applications in the future.

### 3.3.1 Image Processing Server Design

A primary objective in designing the Image Processing Server was to separate the time consuming image processing operation from the core functionality provided by the Data Management Server, and thereby support replication of the server to accommodate higher processing demands or larger number of phones in the future. The second objective in the design of the Image Processing Server was to enable rapid and convenient integration of application specific image processing and computer vision techniques.

The Image Processing Server was designed such that no global memory was needed to synchronize the individual servers, hence eliminating the migration complications inherit in volatile data sharing. Instead the Data Management Server is responsible for scheduling and gathering data. The scheduling function uses a round-robin and asynchronous gathering methodology for communication with the Image Processing Server. This insures the Image Processing Server is not burdened by other tasks and the centralized post-processing eases tuning of the decision algorithms.

To accommodate rapid exploratory integration of application specific processing, we used Matlab as the computation engine for the Image Processing Server. However, since Matlab does not provide an interface for external applications to access its internal functions, we extended it with an internal TCP/IP server. The internal server provided a common interface for the local Java Servlet-based application server to interact with Matlab’s internal function libraries.

To analyze the images, the Data Management Server negotiates access to the services of the Image Processing Server through a standardized HTTP-based interface. The interface accepts RESTful encoded function calls and the results were returned as a JSON encoded text string. The URL encodes the name of the requested functionality and a unique ID for each requested image. The image processing process, one of the Data Management Server’s image handling services, retrieves the image from local disk, and sends the image with the processing request of its choice, indicated by a function name, to the Image Processing Server. The Image Processing Server parses the request, saves the file to the local disk for accessing by Matlab, and passes the function name to Matlab’s internal TCP/IP server. Matlab looks up the function and either returns the result of the processing if the function is available or otherwise returns an error message to the Data Management Server.

The Data Management Server can be configured for two independent scenarios. In one scenario, new images are handled at scheduled intervals; preferably at times when the system would not be occupied by intensive tasks. In the second, a semi-real time scenario, incoming images are fed to the Image Processing Server in a FIFO manner. In either case, the logic on the Data Management Server handles all filtering and segmentation. The file descriptors and data provided by the Image Processing Server are used as input to the filtering and segmentation logic.

### 3.3.2 Image Processing Techniques

To filter out images before presenting through the Image Viewer, the Image Processing Server processes each image in the gallery and determines its quality by classifying it into four categories of Clear, Blurred, Exposed and BlurExposed. The resulting category is recorded back as an annotation to the image and subsequently used by the Data Management Server to select images for presentation to the user.

To determine the class of each image, we use four well-known features: mean of intensity, standard deviation of intensity, the number of edges in the image, and finally the sum of the high frequency coefficient of Discrete Cosine Transform (DCT) of the image. We refer to these features as the mean feature, standard deviation feature, edge feature and DCT feature respectively throughout the paper. While these features are widely used individually in previous work [18, 23] to determine the image quality, we use their combination to enhance the performance of the image quality estimation. The Image Processing Server uses a decision tree classifier that based on the above four features determines the class of each image. We evaluate the performance of the classifier in terms of classifying the images and the latency cost of extracting the four features and determining the class of the images in Section 5.1.

### 3.4 Mobile Phone Processing

In this section, we describe extensions to the system design to support local image processing on the mobile phone as a means of increasing the efficiency and interactivity of recall applications. Our early pilot image sets had many low quality images, as well as some congestion on the wireless upload link. Therefore we were motivated to reduce the congestion by using in situ filtering of the extremely low quality images that were in any case filtered out on the back-end server once uploaded. Our pilots also prompted interesting requirements for prioritized and real-time upload, particularly when participants connected to the system after an extended period of disconnected operation. The functionality added includes uploading and viewing the most recently captured images first, and rapid viewing of a subset of images that represents the entire day, rather than an equal number of images that covers just a small portion of the day. To facilitate this, we have modified the functionality of the Campaignr application running on the mobile phone to classify the captured images based on the extracted features, and implement reverse chronological order upload and prioritized upload policies. This section describes the simple image processing and prioritized upload mechanisms implemented on the mobile phone.
3.4.1 Image Processing on Mobile Phones

Simple image features can be used to determine the quality of the image as described in Section 3.3.2. However, image processing overhead is a more serious concern on the mobile phone than it was on the back-end server because of the device’s relatively constrained resources.

To extract image features and classify the acquired images on the mobile phones, we have extended the Campaignr application that runs on the mobile phone. The new Image Annotation module extracts image features and the Image Classification module classifies the images based on the features extracted from the Image Annotation module. The modules are designed to easily accommodate additional or alternative image processing functions where the binary decision on the feature can improve classification. The software was designed so that application designers can conveniently select the features to be extracted from the captured images and configure the thresholds of the chosen decision tree used for classification via the local Campaignr XML configuration file. Campaignr currently supports extracting mean and standard deviation of intensity, the number of edges in the image using well-known edge detection algorithms (i.e. Sobel, Canny, and Prewitt), and the JPEG image size of the captured image. We have also implemented a Clustering module that clusters the images based on the capture time of the image. This module can also be easily extended to use other available information, such as image features for more sophisticated clustering algorithms. Lastly, an Upload Ranker module is implemented to implement different types of upload policies.

The data flow of each captured image is shown in Figure 4. Whenever Campaignr captures an image, it temporarily stores the image on the database along with other sensing data including the capture time. If activated by the XML configuration file, the Image Annotation module fetches the stored image, extracts the selected features, and stores the results in the database. Then, the Image Classification module reads the computed features and classifies the image using the configured decision tree. The Clustering module generates the cluster ID using the capture time. Lastly, the Upload Ranker module ranks the images based on the configured upload policy. The policy uses the annotations provided by the earlier modules for discarding or uploading the images. The Upload Ranker module may also modify other previous image ranks as they could be changed depending on the subsequent images.

We have implemented a decision tree on the mobile phone that classifies the images into the four categories of Clear, Blurred, Exposed, and BlurExposed, as we did in the back-end Image Processing Server described earlier. However, to minimize the cost of classification on the phone, we profiled the cost of extracting each feature and we only use the subset of features, that provide maximum classification performance with a minimum overhead. We discuss the performance and cost of the classifier we used on the phone in Section 5.2.

3.4.2 Upload Policy for Mobile Phones

As expected, mobile phones continuously collecting multiple images per minute can easily congest the relatively narrow and intermittent upload channels that characterize cellular infrastructure. Simple image processing on the mobile phone can address this by selectively uploading informative and high quality images, rather than uploading every single image.

Prioritizing the upload order can also improve the usability of the system. When mobile phones are disconnected from the network, automatically collected images are stored in the phone’s memory until network connectivity becomes available and the phone resumes upload. In the event of extended disconnection, uploading the accumulated images takes a significant amount of time. This upload delay is not a concern when users do not attempt to view images until a later time. However, if the usage model is such that users want to review their images rapidly after starting the application or after reconnecting to the wireless infrastructure, selectively uploading images can provide enhanced interactivity and responsiveness.

For example, during the DietaryRecall Pilot, which we describe in more detail in the next section, we received 3502 images that were classified as low quality image, equivalent to 33% of the total number of captured images. We also experienced many cases where the participants were disconnected from the network for an extended period of time at the end of their recording day, and needed to upload the previously captured images after they arrived at the lab to participate in the dietary recall experiment the next morning. This introduced a considerable amount of waiting time before they could actually run the web-based assisted recall process. Also during the image capture part of this pilot, users or experiment-facilitators needed to occasionally view captured images right away to deter-
mine whether the mobile phone was being worn properly (e.g., at a height and angle that allowed the device to capture images of food for the particular user’s stature and posture.)

As another example of the use of this feature, consider the benefit of displaying a subset of images that represent the users entire day, rather than an equal number of images that cover just a small portion of the day. This can be achieved by showing only the representative and quality images from each cluster of images collected during a time slot. Thus, uploading first the best quality images uniformly distributed over the entire capture time provides a more interactive interface for the user, as compared to sequentially uploading all quality images from each time cluster in chronological order. To provide better interactivity we implemented the ability to selectively upload images based on different upload policies, rather than naïvely uploading all unfiltered images based on chronological order. We have implemented two upload policies.

The reverse chronological order upload policy uploads the most recently captured images first. Configuring this policy requires activating the Image Annotation module and Image Classification module to discard low quality images. However, other modules are deactivated so that Campaign can simply upload the images based on the capture time of each image.

The prioritized upload policy improves the responsiveness that the users experience after an extended period of disconnection (Figure 5). The images on the phone are clustered by the Clustering module (step 1). Subsequently, the images within each cluster are classified by the Image Classification module and ranked by the Upload Ranker (step 2). The Upload Ranker assigns a rank to the images based on the combination of their cluster ID and class membership and records the rank to the phone database. The ranking system guarantees that higher quality images are uniformly taken from each cluster (starting from the most recent cluster). Finally, in the event that the phone connects to the system, the Upload Controller gets a rank-sorted list of the images from the local database and uploads them sequentially to the server.

4. USE CASES, DATA COLLECTED
Rewind was tested in a real epidemiological research setting to evaluate its utility for assisted recall. We also performed other short-term trials. Results gathered from the first three pilots were used to evaluate the server-side operations, while a fourth was specially designed to evaluate on-device processing.

4.1 Dietary Recall Study Pilot (DietaryRecall)
To test Rewind’s feasibility in assisting previous-day dietary recall, we distributed camera phones to subjects who agreed to participate in the DietaryRecall Pilot study under the direction of Dr. Lenore Arab. The participants began using the system in August. To date, 10 have consented and received phones. Our first concern was that subjects might not find wearing the phone around their neck acceptable. However, all subjects, save two, consented to the add-on camera phone study. They were provided with phones on a lanyard so that the camera would be directed downwards and in front of the subject. This is the first work of its kind that uses worn mobile phones angled downward to autonomously capture food items and it established the importance of proper phone positioning and end-to-end system interactivity. Consent to conduct the study was received from both the UCLA medical IRB and the General Clinical Research Center at UCLA.

Ten users ran the system during the course of the initial phase of the DietaryRecall Pilot. Participants were asked to wear the device turned on only when eating. Across those users there were 110 distinct eating episodes and a total of 11090 images uploaded. There was on average of 101 images per episode, ranging from a maximum of 775 to a minimum of 1; reflecting the diversity of short snacks and longer meals across study participants. The images were uploaded using the phone’s GSM modem.

Dr. Arab wished to keep the participant experience as simple as possible. Based on her experience with participant compliance, the system was configured to display a single screen with a maximum of 35 images per eating episode; thus eliminating any scrolling by the user while completing their dietary recall survey. Rewind selected images to present based on image quality and uniqueness features that could be automatically determined by the image processing software. The system deleted images that were outside of a specified quality threshold. These images were considered of such a low quality that they would not provide any useful information. The Reaper function deleted 3502 such images, equivalent to 33% of the total number of captured images. Due to privacy concerns it was not possible to access the files for later evaluation of the image filtering techniques. This large number of unusable images is likely exaggerated in this first study because we did not have extensive experience in training users on device placement and had not optimized the lanyard used. It will be interesting to observe in future deployments if we see a significant reduction in this percentage as training techniques are improved.

4.2 Full-day Dietary Pilot (FullDayDietary)
Fourteen volunteers from our research lab ran the Rewind application to capture images during their meals to test its feasibility before the study described above. This pilot was conducted during a workday while the participants were outside the home, capturing six images a minute throughout their day.

All of the phones captured and transferred images as expected. Both the secure image repository and image processing algorithms handled the full load of incoming images without incident at the end of the pilot day. We were also able to test a primitive version of our Image Viewer and verify that both the image deletion and the eating-episode-identification mechanisms functioned; but at the same time we were able to obtain feedback from the local users that lead to design refinement during the early development phase of the system. Unlike the DietaryRecall Pilot participants who had to remain anonymous and whose images we could not analyze due to study design and Institutional Review Board restrictions, the local users were given the opportunity to release all or some of their images for use by the system designers. Six participants consented to release total of 14958 images. The users also provided us with detailed verbal feedback on the experience.

A nutritionist analyzed the released images to determine the number of images taken of each individual food reported during these days. The median of number of images per food was 28 for this population and the range was 1 to 211. Most foods had much duplication.

Significant information on phone use, management of the Image Viewer and presentation of images was obtained from the FullDayDietary Pilot study. For example, users experienced extended upload time when phones were disconnected during operation for significant periods of time and could not obtain rapid feedback once
connected on the quality of their uploaded images. The users also suggested providing a feedback mechanism for correct data capture and upload (as in messages on the screen or audible sounds), and more stable and convenient physical mounting on the lanyard. We have also provided access to the images that were approved for sharing to computer vision researchers interested in category recognition to begin developing targeted algorithms for identifying images that contain food items.

4.3 Poster Session Interactions Pilot (PosterSessionCapture)

Fifteen visitors participated in a technical feasibility exercise designed to test the ability of our system to handle uploaded images collected by many simultaneous and co-located users for an hour during a research conference poster session in October 2007. Participants were asked to wear a phone configured to capture six images a minute on average, while they were visiting student research posters, amidst a large number (~200) of conference attendees. Participants were also asked to make audio annotations occasionally to provide additional context for later discussion. Audio samples were collected and uploaded by the Campaignr on the same phones that were used to capture images. The captured images and audio samples were used in a plenary session as a vehicle for prompting discussion of the student posters and associated research. In addition we presented system usage statistics such as a ranking of the 15 participants in terms of number of images uploaded and filtered, as well as number of audio annotations provided.

In this pilot, the 15 users uploaded 4342 images during the course of one hour. Each participant uploaded 289 images on average, ranging from a maximum of 351 to a minimum of 162. As we wanted to present the results from the pilot and have discussions with visitors immediately following the poster session, we ran two Image Processing Servers in parallel to handle all the images in semi-real time. The filtering rate was fairly low relative to the DietaryRecall Pilot. Only 2.64% of each user’s images were filtered out on average, ranging from a maximum of 17.41% to a minimum of 0%. The low filtering rate was in part due to the fact that we ran only the standard deviation filter in order to process all images in semi-real time without applying significantly more computational resources. In addition, we configured the filter thresholds to be the same as those used in our earlier pilot studies, but found the lighting conditions and continuous movement of the participants was quite different. The average file size of the captured 800 x 600 images was 35 kB yielding an average total data rate of 3.5 kB/s. 118 audio samples were collected ranging from 1 to 40. The images were uploaded using the phone’s 802.11 radio. In this setup, the system was able to handle 1.5 images/sec with a small safety margin. This is by no means an upper bound on the capacity of the system; the re was able to handle 1.5 images/sec with a small safety margin. This is indicative of the overall latency of the system, and ultimately can be used to determine the number of required Image Processing Servers.

5. Evaluation

We evaluated system performance with respect to the following metrics: the ratio of images classified as “high quality” compared to the total number of images, which is a measure of the efficacy of applying image processing filters on the server and mobile device, the performance of Image Processing Server in identifying and annotating low quality images compared to manually annotated “ground truth”, and the time required to determine image quality. Since the latter is the major latency of the system, it is indicative of the overall latency of our system and ultimately can be used to determine the number of required Image Processing Servers.

We furthermore evaluate the performance of the in situ processing of the images on the phone with respect to the following metrics: performance of the local image classifier, its latency and its energy consumption. The evaluation of energy consumption of the local classifier determines the impact of the in situ image processing on the lifetime of the phone. Finally, we investigate the advantages of our prioritized image upload in improving the interaction of the user with the system.

5.1 Image Processing Server Performance

The performance evaluation was performed on three individual machines. A central Data Management Server maintained a MySQL 5.0 DBMS with access to all the image information. The server also ran the orchestration application that delegated the image processing tasks to the Image Processing Servers and updated the local database. A high-performance setup, a Dual-Core AMD Opteron, 1800 MHz with 2048 MB DRR RAM, ensured a minimal turnaround time for any requests to the management server. The Image Processing Servers were run in parallel on the local Ethernet network.

5.1.1 Server-side Image-Quality Classification

To investigate performance of server-side image quality classification, we used 200 randomly selected images from the FullDay Dietary Pilot for classifier training and evaluation. To determine the
Figure 6: This Figure illustrates sample images from the FullDayDietary Pilot. (a) and (b) Clear images from the food intake of participants, (c) a Blurred image due to the motion of individual carrying the phone, (d) a Blurred image due to the motion of the subjects, (e) an under-image due to poor lighting (Exposed), (f) an exposed and blurred image (BlurExposed).

Figure 7: Percentage of each image class in the groundtruth gallery from the FullDayDietary Pilot

ground truth class membership of each, we asked three people to browse the data and mark the category of each image from the four categories of Clear, Blurred, Exposed and BlurExposed. We then picked the popular vote as the category of each image in the ground truth data. We furthermore divided the data set into two equally-sized and randomly selected parts for training and evaluating the performance of the classification.

Figure 6 illustrates a sample set of images taken during the Full-DayDietary Pilot. As described earlier many images were blurred either due to the motion of the individual carrying the phone or due to the motion of the subjects in the images, and many other images were deemed to be either under- or over-exposed due to the poor lighting conditions. Figure 7 illustrates the percentage of each class in the ground truth data as marked by the individuals. In this case, only 45% of the images were marked as clear images by the users and more than 24% of the images were marked as blurred and 31% of the images were marked as exposed. The high number of blurred, under-, or over-exposed images motivated us to design the in situ image processing and prioritized upload features on the mobile phone to filter out the images that would otherwise be uploaded and filtered in the Image Processing Server. We also note that the number of blurred, under-, or over-exposed images is application specific. For instance, the FullDayDietary Pilot resulted in a higher potential filtering rate since images were captured throughout the entire day where mobile phone’s cameras were more exposed to private situations where people intentionally blocked the camera or activities requiring many movements, whereas the PosterSession-Capture Pilot resulted in a lower filtering rate since images were taken during a short period of time (an hour) in a very public setting where participants often stayed stationary when visiting the research posters.

Figure 8: The decision tree classifier on the Image Processing Server, classifies the images into four different classes.

Figure 8 illustrates the decision tree classifier that runs on the Image Processing Server. The classifier uses four features including mean and standard deviation of intensity, the number of edges in the image, and finally the sum of a subset of Discrete Cosine Transform (DCT) coefficients of the image to classify each image. To determine the structure of the tree we used the ground truth data from the FullDayDietary Pilot as marked by the individual.

The confusion matrix shown in Table 1 summarizes the performance of the Image Processing Server classifier.

<table>
<thead>
<tr>
<th>Truth</th>
<th>Clear</th>
<th>Blurred</th>
<th>Exposed</th>
<th>BlurExposed</th>
<th>FN</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear</td>
<td>45</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Blurred</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Exposed</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BlurExposed</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1: Confusion Matrix for the back-end Image Processing Server classifier. (FN: False Negative, FP: False Positive)
the test images. Since the emphasis is recognizing the clear images, an instance was considered a False Positive if a clear image was detected when it was not clear and was considered a False Negative if a clear image was not detected when it was clear. It illustrates that more than 83% of images were correctly classified. It also shows that 93% of the clear images correctly classified as clear images and only 2% of low quality images were incorrectly classified as clear and hence potentially shown to the user.

5.1.2 Server-side Image Processing Performance

To determine the image processing overhead of the overall system when processing occurs on the server-side, we benchmarked the latency of each component of our system for processing each image. This includes 1) the overhead of the Data Management Server to look up the local database for images and send the image to the Image Processing Server 2) the latency of the web interface in Image Processing Server and 3) the latency of the Matlab environment to load the image and convert it to gray-scale for processing. In addition, we benchmarked the amount of time it takes for extracting features of the images.

The parallel system design allowed us to use two low-performance servers for the image processing. Although these machines were equipped with relatively outmoded hardware, the service was able to run in a semi-real time. A quadro Pentium III machine running at 700 MHz and a Dual-Core AMD Opteron, 1000 MHz were initially chosen as experimental machines. The throughput rate made it unnecessary to upgrade the machines as more users were connected to the system.

Figure 9 illustrates the image processing latency of our system. It shows that latency is dominated by image processing computation namely extracting the DCT feature, edge feature and mean/standard deviation feature. Furthermore, it illustrates a very small deviation in processing latency of each image. Consequently, we were able to process all images within a roughly fixed predetermined time period.

The Image Processing Servers were used in two scenarios; a real-time application and a scheduled application. We required the real-time application to handle 1.5 images/sec. In this case, the expected processing time per image exceeds the rate at which images were acquired. A large array of IPS machines would have been required to maintain real-time latency. Instead, we chose to reduce the number of processing steps to include only standard deviation and mean intensity. We estimated the total expected end-to-end latency as 0.95 seconds. In the scheduled application setting the off-line process was allowed to run 7 hours. The large time-span allowed us to apply all filters to each image. The average total latency per image was estimated to 8.02 seconds. Consequently, two Image Processing Server machines running in parallel could process 6300 images each night. The approximated throughput rate far exceeded the actual demand per user, each of whom had an upper-bound of 1000 images/day, and enabled the support for multiple users a day.

5.2 Image Processing On Mobile Phones

In our study, we use the Emulator Debugger from the Nokia S60 Development Tool to measure the classifier performance, and the Nokia N80 itself to profile the cost of the classifier. The emulation enables scalable collection of training data and measurement of the classification performance on the large set of the images, and the performance measurements from the Nokia N80 phone enables accurate measurement of the classification cost on the phone hardware. To implement image processing tasks on the device, we used the NokiaCV library (Nokia Computer Vision Library) which is designed to support computer vision applications in a Symbian environment by Nokia Research Center [1].

5.2.1 Client-side Image Processing Performance

We considered the constraints of the phone in designing the local on-device classifier. The latency overhead of the image processing and further classification should be reasonably less than the rate of capturing the images. As described previously, in the DietaryRecall Pilot, Rewind captures the images at 10 second intervals; hence the local classification latency should be reasonably less than the 10 second image capturing intervals.

To design a low cost classifier on the phone, we first benchmarked the overhead of extracting each feature of interest on the phone. Table 2 illustrates the overhead of extracting mean/standard deviation of intensity, the number of edges based on the Sobel edge detection algorithm, and the JPEG image file size on the device. In addition, for mean/standard deviation of intensity and edge count features, there is an additional fixed delay for converting the images from the RGB to Gray-scale format. Although we implemented the edge count feature on the phone, we did not consider it in our final local classifier due to its high overhead of 32 seconds. In addition, since our server side benchmarking of the DCT feature (Figure 9) indicated an even higher computation latency compared to the edge count feature, we did not consider the DCT feature for the local device classifier since it was unlikely that it would end up being an effective choice due to overhead, and thus did not warrant the development time.

In our scheme, we used the JPEG image file size (which is readily available when the image is captured) instead of using the expen-

<table>
<thead>
<tr>
<th>Color Conversion</th>
<th>Mean and Std.</th>
<th>Edge Count</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.91 (sec)</td>
<td>2.97 (sec)</td>
<td>32.13 (sec)</td>
<td>0 (sec)</td>
</tr>
</tbody>
</table>

Table 2: The latency of extracting each feature on the phone.
Figure 10: This figure illustrates a strong relationship between the DCT based feature used on the Image Processing Server and the JPEG file size used on the phone. It also indicates the distribution of image classes vs. the features.

Table 3: Confusion Matrix for the classifier used on the phone. (FN: False Negative, FP: False Positive)

5.2.2 Client-side Image-Quality Classification

The most discriminative feature, as shown in Figure 8 is the “number of edges”. However, due to its significant latency cost of computing this feature (as well as the DCT feature), we built a low latency classifier (Figure 12) that only uses the mean, standard deviation and JPEG file size and compared the performance and latency. To train and evaluate the performance of the local classifier, we used the same gallery of images from the FullDayDietary Pilot study that we used for training the server side classifier.

The confusion matrix shown in Table 3 summarizes the performance of the in situ classifier in classifying and labeling the test images. The results illustrate that more than 82% of the images were correctly classified. More significantly, there are only 14% of low quality images which were incorrectly classified as clear images on the phone.

5.3 Upload Policy Evaluation

In this section, we first evaluate the effectiveness of reverse chronological order upload on improving the waiting time before viewing the most recently captured image. Secondly, we evaluate the effectiveness of prioritized upload on improving the waiting time before seeing at least one image from every time cluster at the Image Viewer. Lastly, we investigate how prioritized upload can improve the quality of the first uploaded image on each time cluster.

5.3.1 Reverse Chronological Order Upload

We compare the waiting time of chronological order upload versus reverse chronological order upload in Table 4. The latency provided here is measured from the time when Campaignr starts uploading images until the back-end server receives the most recently captured image. As the chronological order upload needs to upload the pre-captured images first, the waiting time was proportional to the disconnection time. As expected, the reverse chronological order upload showed a negligible waiting time irrelevant to the disconnection time.

5.3.2 Prioritized Upload

To evaluate the effectiveness of Rewind’s prioritized upload on reducing this waiting time, we define completion delay as the time duration from when one of the cluster receives a image for the first
Table 4: Waiting time before viewing the most recently captured image. Comparison of chronological order upload versus reverse chronological order upload following different disconnection times (1, 2, 3 hours).

<table>
<thead>
<tr>
<th>Time</th>
<th>Chronological</th>
<th>Reverse Chronological</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 hour</td>
<td>201 (sec)</td>
<td>5 (sec)</td>
</tr>
<tr>
<td>2 hours</td>
<td>408 (sec)</td>
<td>4 (sec)</td>
</tr>
<tr>
<td>3 hours</td>
<td>561 (sec)</td>
<td>5 (sec)</td>
</tr>
</tbody>
</table>

(a) 802.11b/g

<table>
<thead>
<tr>
<th>Time</th>
<th>Chronological</th>
<th>Reverse Chronological</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 hour</td>
<td>1022 (sec)</td>
<td>25 (sec)</td>
</tr>
<tr>
<td>2 hours</td>
<td>2192 (sec)</td>
<td>24 (sec)</td>
</tr>
<tr>
<td>3 hours</td>
<td>7399 (sec)</td>
<td>29 (sec)</td>
</tr>
</tbody>
</table>

(b) GSM modem

Table 5: Completion delay comparison of chronological order upload versus prioritized upload following different disconnection times (1, 2, 3 hours).

<table>
<thead>
<tr>
<th>Time</th>
<th>Chronological</th>
<th>Prioritized</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 hour</td>
<td>191 (sec)</td>
<td>25 (sec)</td>
</tr>
<tr>
<td>2 hours</td>
<td>394 (sec)</td>
<td>99 (sec)</td>
</tr>
<tr>
<td>3 hours</td>
<td>545 (sec)</td>
<td>121 (sec)</td>
</tr>
</tbody>
</table>

(a) 802.11b/g

<table>
<thead>
<tr>
<th>Time</th>
<th>Chronological</th>
<th>Prioritized</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 hour</td>
<td>953 (sec)</td>
<td>124 (sec)</td>
</tr>
<tr>
<td>2 hours</td>
<td>2126 (sec)</td>
<td>242 (sec)</td>
</tr>
<tr>
<td>3 hours</td>
<td>7282 (sec)</td>
<td>380 (sec)</td>
</tr>
</tbody>
</table>

(b) GSM modem

Table 6: The quality distribution of the first uploaded image from 80 clusters

<table>
<thead>
<tr>
<th>Quality</th>
<th>BurrExposed</th>
<th>Exposed</th>
<th>Blurred</th>
<th>Clear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Prioritized</td>
<td>0</td>
<td>12</td>
<td>3</td>
<td>65</td>
</tr>
<tr>
<td>Prioritized</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>75</td>
</tr>
</tbody>
</table>

6. CONCLUSION AND FUTURE WORK

We developed a targeted assisted recall system providing image capture, upload, filtering, and presentation of images from Symbian S60 handsets, and deployed it in several tests including a “live” epidemiological study on dietary intake. The system was designed to handle many simultaneous users by distributing the image processing logic to network-accessible servers using standardized web interface, and in later designs leveraged image processing on the mobile phones themselves to reduce wireless-channel congestion. Image filtering was first implemented on the server to reduce image browsing overhead for the user. In the enhanced version of the system, a time-efficient classification algorithm was implemented on the mobile phones to filter out poor-quality images by characterizing the mean intensity, standard deviation intensity, and JPEG file size of the image. Processing on the phones also enabled the system to adapt to the varying connectivity experienced with wireless infrastructure by using prioritized upload mechanisms once connectivity is reestablished.

The system’s web-based processing framework supports rapid prototyping of server-side image processing. The core application framework was built on top of Matlab using well defined interfaces that can be extended in the future to plug in to R or custom processing routines. The scalability of the service-side image handling framework was addressed by centralizing decision processes onto a dedicated Data Management Server. The image processing itself was delegated in a round-robin fashion by the Data Management Server to multiple instances of the Image Processing Server. The need for global cache migration was eliminated by centralizing the decision processes to allow more Image Processing Servers to be added to the system as need arises.

The system was successfully used in a dietary intake study in collaboration with UCLA’s medical school as part of an ongoing NIH study on the effectiveness of web-based dietary recall survey techniques. The high sensitivity of the data gathered in this study mandated the implementation of firm security policies to avoid data leakage and empower the research subject to fully control their own information. Privacy protection must be a foremost concern for designers and users of systems that capture personally identifiable data. Images in particular are a highly revealing and therefore sensitive form of documentation. We designed our system to require the end-user to approve any information before it could be released to the personnel conducting and coordinating the study, let alone into the public domain. Additionally, all communication was encrypted and data was automatically reapplied if no user action was taken to explicitly share an image. Two internal pilots identified the performance metrics of the system and proved the scalability and
Our future work will focus on:

- More extensive evaluation of the systems effectiveness in dietary and other targeted recall applications
- Larger scale systematic studies to identify where additional technical advances are most needed
- Simple mechanical improvements such as the addition of a fish-eye lens and an improved lanyard
- Integration of our system with other existing image browsing and navigation tools such as Picasa and Flickr
- Development of more sophisticated image filtering and clustering mechanisms, including those that take advantage of the specificity of particular target applications

7. ACKNOWLEDGEMENT

We acknowledge Joe Kim, August Joki, and Haleh Tabrizi for their implementation, and Teresa Ko and Brian Fulkerson from Vision Lab at UCLA for their insightful comments and opinions. We also acknowledge Sasank Reddy, Andrew Parker, and Josh Hyman for their contributions to the design of the initial prototype of DietSense. This work was supported partially by CENS Cooperative Agreement #CCR-0120778, NSF-FIND, Nokia Research Center, and UC Micro.

8. REFERENCES