Redesigning the Wearable Camera Pipeline to Detect Key Body Parts at Low Power

Pengyu Zhang[1], Bodhi Priyantha[2], Jie Liu[2], Matthai Philipose[2]
[1]University of Massachusetts Amherst, [2]Microsoft Research
April 2015

ABSTRACT

This paper addresses the problem of designing wearable devices suited for continuous mobile vision. Given power budgets, it is infeasible either to stream all video off board for analysis, or to perform all analysis on board. A compromise is to off-load only "interesting" frames for further analysis. We identify windows containing people and parts of their bodies as being interesting. We show that using a careful combination of emerging thermal sensors and ultra-low-power coarse stereo enabled by modern low-power FPGAs, it is possible to detect the presence of these "key" body parts at well within wearable power budgets. We combine our (hardware) binary presence detector with a mobile CPU running a classifier to identify *which* body part is detected. Using this combination, we show how to detect faces, hands (of the video wearer) and bodies of those in the field of view at under 30mW; running on just the phone draws over 1W.

1. INTRODUCTION

We consider the design of a *continuous mobile vision* system based on a wearable device such as Google Glass. The potential benefits of a such a system have been discussed extensively recently [14, 9, 4, 29, 19]. In a nutshell, the variety and detail provided by visual analysis of the wearer's surroundings enables a much larger set of applications than, for instance, those enabled by inertial sensors in popular step-counter-style wearables available today. Given recent progress in visual recognition algorithms, a primary bottleneck to realizing such devices is power consumption. Vision algorithms are computationally demanding, and video streams have heavyweight communications requirements. Reconciling the high resulting demands on energy with the modest battery capacity of wearable devices is challenging. In this paper, we examine techniques for attacking both computational and communication bottlenecks for an important subset of visual recognition tasks (those involving people and their parts) by re-designing the traditional camera pipeline.

Figure 1 illustrates the setting we address. A wireless wearable camera, positioned on the wearer's lapel (or perhaps head) shares the wearer's field of view. The camera continually observes its wearer's activities and their surroundings. Often, of greatest interest are questions about what the user, their social partners and those around them are doing. Much work over the past decade [30] has shown that the objects manipulated by a person and the nature of manipulation are a key to understanding the *tasks* they are performing. On the other hand, analyzing the faces of social partners with a view to identifying them, recognizing their affect and tracking their facial cues are key to decoding *social* interaction [37, 28]. Finally, a large body of work [18, 33] exists on tracking whole-body kinematics as a measure of *coarse human activity*. Accordingly, we designate the wearer's hands, their conversation partners' faces and the bodies of those around them as "key" body parts.

Figure 1: Our wearable camera usage (above) and the key body parts we target (below): the wearer's hands, their conversational partners' faces and the bodies of those around them.

Sensing and compressing the large, high-resolution high-speed streams of image frames required for analyzing entities of interest in the wearable's field of view requires handling a large volume of data. The imager, codec and transmission components required to handle this data exceed the power budget of a typical wearable. Recent pure-offloading-based architectures suited to the relatively large batteries of phones are therefore not adequate for wearables. One solution is to detect and transmit, for further analysis, a small subset of "interesting" windows from the video stream. Traditionally, detection requires a dense windowed scan of the entire frame at several resolutions: a VGA frame can easily contain

tens of millions of windows to be checked. In some cases (e.g., face detection), the best algorithms are extremely efficient, executing a few machine instructions on most windows [6]. In others (e.g., whole-body detection), processing most windows may require hundreds of thousands of instructions. In either case, software implementations of detection algorithms exceed wearable power budgets. Hardware implementations promise adequate efficiency but are available only for front-on face detection; efficient but *general* detection circuits are still well in the future.

In this paper, we present the design, implementation and evaluation of a system for detecting key body parts at wearable power budgets. Intuitively, we exploit the fact that the temperature of human skin is often distinct from the background (our system degrades gracefully if this assumption is violated). Further, the hands of the camera wearer, the faces of their conversation partners and the bodies of people around them tend to be at distinct distances and positions relative to the camera. We show how to exploit low-power, low-resolution imaging channels complementary to the traditional RGB video pipeline to this end. In particular, we show that a carefully designed combination of spatially coarse thermal imaging and focused, coarse-resolution depth imaging optimized for ultra-low-power processing on an FPGA can dramatically reduce both the number of high-resolution RGB frames to be read from the imager, and the number of windows to be checked in each frame. The end result is the first fully realized detection system we know of that, in common use cases of wearable cameras, can detect over 90% of key body parts using tens of mW of power.

2. MOTIVATION AND BACKGROUND

Figure 2 illustrates the structure and key performance characteristics of a wearable camera pipeline comprised of state-of-the art components. A wearable camera must have relatively wide field of view to cover the "front" of its wearers. When combined with the desire to resolve, e.g., faces at 5m, an imager with 4k pixels per side is quite plausible. For such an imager running at 15 frames/second or higher, today's imager, compression chips and wireless transmitters would together easily require several hundred mW to off-load compressed video off-camera for analysis. However, assuming the

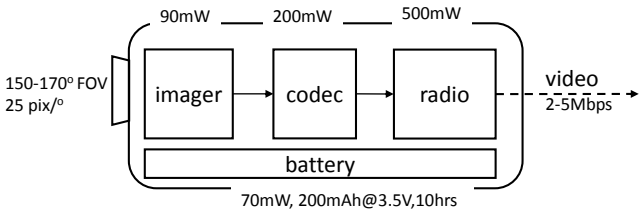
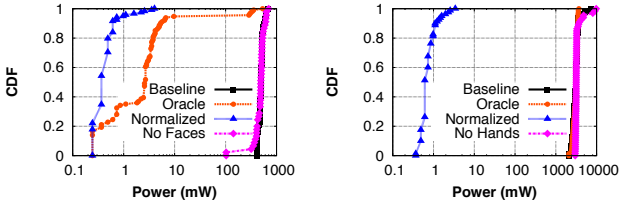
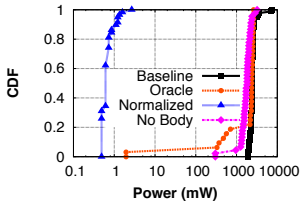


Figure 2: Wearable camera pipeline.



(a) face (b) hand



(c) body

Figure 3: Power consumed for detecting key body parts on a Galaxy Nexus phone.

device needs to run continuously for 10 hours (a "whole day" for its wearers), size and weight constraints on the battery yield a power budget well under 100mW. This "full offload" approach not only breaks the power budget of the wearable device, it places the load of detecting objects of interest on the devices offloaded to, typically the wearer's mobile phone; alternately, the phone could further offload the video to the cloud, requiring *its* radio to stay always on, an infeasible approach given trends in wide-area wireless communication power.

We propose to push the detection of (some) interesting entities into the camera pipeline. Given that most pixels detected by a wearable are uninteresting (they contain no faces, hands, or bodies being monitored), this clearly has the potential to reduce significantly both the amount of data streamed and the amount of data to be analyzed. Doing detection on-camera poses several challenges, which we investigate in this section.

2.1 Detection in Software and Its Limits

Perhaps the simplest approach to incorporating detectors on the camera is to use a microcontroller to run detection algorithms in the camera. To evaluate this approach, we measured the overhead of wearer-hand, face and person detection on a Galaxy Nexus phone with a 750×480 imager running the corresponding OpenCV-based detectors [26]. We target detection at 1Hz. We use data from 14 people moving between 20 and 2 feet from the phone, or moving the hand from 6 inches to 2 feet as a wearer's hand would.

The pink and black lines of Figure 3a show the cumulative distribution of average power consumption while processing a single frame, for face detection, over all frames of data collected; sub-figures (b) and (c) examine

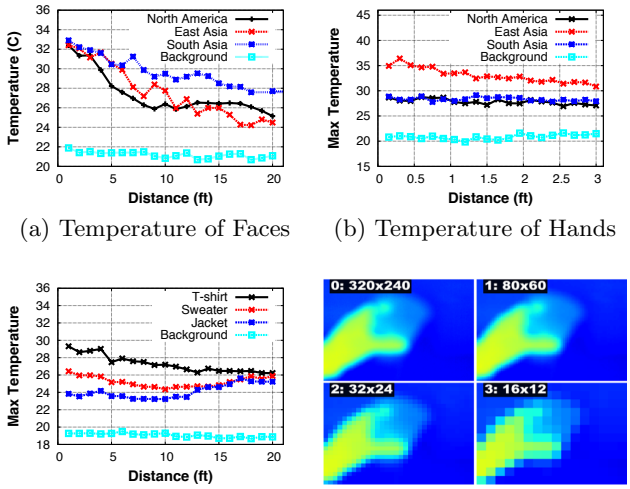
wearer-hand and body detection. The pink line represents power consumption when no body part was in the frame: the detector must still examine and reject ever sub-window within the frame. The black "baseline" represents the case when the relevant body part was within the frame. Power draw is somewhat higher, since the detector tends to expend more cycles on windows when body parts *are* found. Note that *each* detector uses over 1W of power for most frames: naively running detectors in software is prohibitive.

The dotted orange line explores the hypothetical best case where an "oracle" is able to tell the detector where in each frame to look. To generate this line, we mark rectangles around body parts in our data by hand. We run detectors only within these rectangles. As the orange lines show, *if body parts tend to occupy relatively small fractions of the frame*, oracles can reduce net power consumption by orders of magnitude. Faces of conversation partners only take up a relatively small fraction of the frame even when the partners are close by, so that oracles yield high gains (Figure(a)), gaining over 125× in efficiency at the 90th percentile. Wearers' hands, on the other hand, can dominate much of the field of view, thus not gaining much (b). Peoples' bodies dominate the frame even when they are only a few feet away, so serious gains in body detection efficiency are limited to a fraction of the frames (c) (e.g., the 5th percentile).

A fundamental reason that window-based detectors are power hungry is that they search at several (routinely, a dozen or more) scales within their search window. Thus, if the above oracle could provide the scale at which to search, search could be faster. For instance, if the oracle knew the distance to a pixel, and given the typical size of a face, it could provide the detector with a very tight guess of the window sizes to consider. The blue line in Figure 3 shows the effect of of knowing scale during detection. To obtain this curve, we re-scaled every oracle window to the smallest window-size explored by the detection algorithm and ran detection on this window. The impact is dramatic: we now detect all three key parts at roughly 1mW at the 90th percentile. The gains are not at the expense of accuracy: detection after re-scaling found every previously detected face.

To summarize, if an "oracle" system were able to indicate roughly *where* a software-based detector should look, and at *what scale*, even software-based detectors suffice for on-camera detection at low power. Our work can be considered an effort to realize such an oracle, at least for the case of key body parts, and at substantially under the 70mW we target for the total power budget for the wearable device. To cut to the chase, Figure 18 summarizes our success in this direction.

2.2 Far Infrared (FIR) Thermal Imaging



(a) Temperature of Faces (b) Temperature of Hands

(c) Temperature of Body (d) FIR Image of Hand

Figure 4: Temperature of key body parts measured with a FIR imager.

Far infrared (FIR), or thermal, imagers [7] report temperature, rather than traditional light intensity, at every pixel in the field of view. Figure 4d shows FIR views at multiple resolutions for the hand-holding-a-cup from Figure 1. For instance, given that human facial skin preserves a surface temperature of 30-35C, and ambient temperatures are often substantial outside this range (e.g., temperature controlled buildings are kept in the 20-23C range worldwide), thermal imaging is a promising candidate for finding people in the field of view. Traditionally, FIR imagers have cost hundreds to thousands of dollars, consumed substantial power, and been optimized to high-cost high-resolutin settings. However, the past few years have seen the advent of MEMS-thermopile-based technologies developed for the automobile industry, yielding 25mW, low-resolution parts at the 50-dollar price point. We examine this technology from the point of view of on-camera detection below.

We now describe some simple experiments aimed at understanding the intricacies of detecting key body parts with FIR. Figure 4 shows how FIR-measured temperatures of key body parts varies with distance from imager and with possibly relevant auxiliary properties such as clothing and skin color. We generated these numbers by having three people with different skin colors (Northern European, East Asian and South Asian) pose at varying distances from the imager (for face and body data), and move their hands at varying distances in front of the imager to simulate a camera wearer (for hand data). In each case, we report the temperature of the hottest pixel overlapping the body part.

Two related points are most relevant to our setting. First, when parts are close to the imager, bare skin tem-

peratures are close to 30C; temperatures are sometimes higher, but very rarely lower in steady state. Clothing results in a significant, but not catastrophic drop in reported temperature. In temperature controlled buildings, commonly set to 20-23C, key parts are robustly separated by temperature thresholding. At higher temperatures, bare skin remains separable perhaps up to 27C, clothed body parts only up to 24C if a jacket is worn. Skin color exhibits notable trends, but none with direct relevance to thresholdability.

Second, as distance from imager increases, the observed temperature of faces drop rapidly, but hands and bodies remain relatively flat. The explanation lies in the fact that each FIR pixel has a 15^o field of view, and therefore subtends increasingly larger areas at increasing distances from the imager. The reported temperature is the average of body part and background temperature, weighted by the fraction of the field of view subtended by the part. For instance at 20ft, each pixel aggregates temperature measurements from a 1.8×1.8ft square: even if a face of temperature t_f occupied a 1×1ft square, the reported temperature would be $\frac{3}{4}t_b + \frac{t_f}{4}$, where t_b is background temperature. Note, for instance how as resolution decreases (i.e. the angle subtended by pixels increases), the pixels on the edges oof the hand in Figure 4d become cooler

Finally, since wearers' hands are close, they almost always completely subtend the field of view of at least one pixel. Similarly, since torsos are large, unless they are relatively far away, they too usually include such a "subtended" pixel. Since such a pixel is not "weighted down" by the background temperature, its reported temperature will not change with distance and will tend to be maximal among its neighbors. Figure 4, which shows the variation of *maximum* temperature of pixels in each body part will thus show a relatively flat line.

One implication of this averaging effect over the field of view is that FIR imagers with narrower per-pixel field of view will be more robust for temperature based thresholding of relatively distant objects.

2.3 Stereo-Based Depth Imaging

Once a potential object of interest is identified by the FIR imager, we use a depth sensor to determine if that object is located at a distance of interest. While distance sensors such as proximity sensors and laser range finders can measure distance along a particular direction object, we need a sensor that can produce a *depth map* such that we can determine the depth of any given object in the scene.

Two most widely used vision-based depth map estimation techniques are stereo vision and auto focus. Of these two, auto focus requires a image sensor with a dynamic focusing mechanism as well as a large image aperture. Because of these limitations, we use stereo vision-based depth sensing.

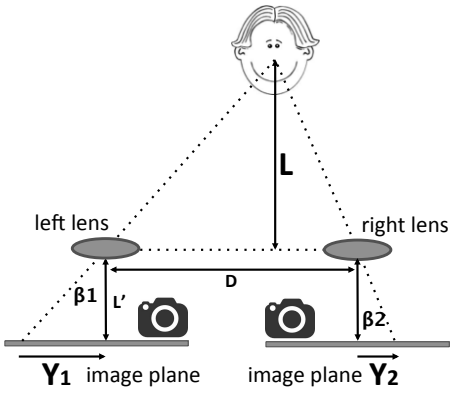


Figure 5: Depth sensing with stereo imagers. L is the distance to be estimated, $L^{'}$ is the lens height, D is the distance between two imager lens, $\vec{Y_1}$ and $\vec{Y_2}$ are projected coordinates of an object on the left and right image planes.

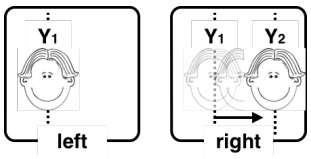


Figure 6: Searching the corresponding object Y_2 on the right image using correlation coefficients.

Depth estimation from stereo vision is based on the relative shifting of the image captured on two image sensors that are placed some distance D apart. As shown in Figure 5, if a point source located at a distance L from the image sensors are projected at coordinates $\vec{Y_1}$ and $\vec{Y_2}$ of the two image sensors, then L can calculated from: $L = \frac{D \times L'}{\vec{Y_1} - \vec{Y_2}}$. Where L' is the height of the image sensor lens from the plain containing the image sensors. The relative shift, $\delta\vec{Y} = \vec{Y_1} - \vec{Y_2}$, is called the *disparity* of the image of the point source.

Conventional stereo vision based depth estimation techniques first compute the disparity map of the whole image by searching for corresponding pixels on the two images captured by the image sensors (Figure 6). A widely cited technique in computer vision literature for finding corresponding pixels computes cross correlation of the blocks of pixels from the two image sensors [24].

Figure 7 shows the power consumption of computing the disparity map of 140 human face images on an ARM Cortex M-4 microcontroller (STM32F417IGH6) at a frequency of 1 image/second. While the exact power consumption varies with the size of the face, the averaged power consumed is 78mW, which much higher than our target power budget for continuous sensing.

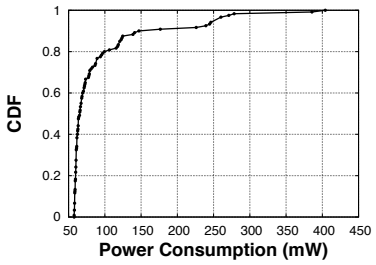


Figure 7: Power consumption of searching Y_2 by computing correlation coefficients.

Apart from the power consumption overhead, computing the cross correlation requires the two images to be read in to some local storage medium. This will require additional memory overhead on a wearable device that aims to capture and process only the relevant regions of the image.

While there exists a large body of work in vision community on how to optimize the disparity map calculation using techniques such as dynamic programming and localized correlations, we observe that we have a much restricted requirements on the disparity map which enable us to use much simpler and energy efficient techniques to estimate the target distance.

First, we are only interested in well defined regions of the image selected based on the FIR sensor readings. Hence, we only need to find correspondence for pixels or a subset of pixels in these regions. Second, we are only interested in coarse grain depth estimation to differentiate among objects within arms length, people within personal and group-wide interaction distances, and far away objects. Such coarse grain estimates can easily tolerate 10% - 20% error in distance estimation. In section 3 we describe the design of an efficient depth estimation solution that take advantage of these restricted requirements.

3. DESIGN

In this section, we introduce the key components of our low-power real-time key body part detection pipeline.

3.1 System Overview

Figure 8 shows the architecture of the body part detection pipeline. The first stage is a FIR sensor that uses the measured temperature of the scene to identify image frames with key human body parts, as well as the location of the body parts within the image frame. The location information is then sent to the second stage which consists of a coarse-resolution depth sensor that estimates the distance to these identified regions.

If the region temperature and distance to the region matches one of the criteria for detecting a body part such as detecting the hand to identify the object the user is holding, the image pixels within the region of interest are captured for post processing such as feeding to a remote object recognizer to identify the particular object.

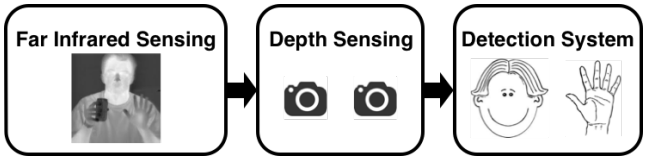


Figure 8: Architecture of the low-power key human body part detection pipeline.

Table 1: Comparison of Melexis and FLIR infrared sensors.

	Power (mW)	Resolution
Melexis (MLX90620)	23.4	16×4
FLIR (LEPTON)	150	80×60

The use of low-power low-resolution FIR sensor and the low-power coarse-resolution depth sensor enables an end-to-end key human body part detector with a power profile suitable for continuous operation on a battery powered wearable device.

3.2 Far Infrared Sensing

While our initial observations showed that a FIR sensor can differentiate between human body parts and the background fairly well, one key design criteria is the resolution of the FIR sensor. For example, as Table 1 shows, a state of the art LEPTON FIR sensor with 80×60 resolution from FLIR consumes 150mW, which is too high for continuous sensing on a wearable device.

A low-resolution image sensor is attractive due to the relatively low power consumption. However such a sensor has two draw backs. First, it cannot accurately detect the edges of objects due the mixing of regions of different temperatures in a single pixel. Second, for relatively small objects at far away distances, the object ad a portion of the surrounding region maps in to a single pixel, reducing the measured temperature of the object, making it difficult to detect the object based on its temperature.

While a high resolution FIR sensor is better at identifying the objects of interest and their boundary accurately, these benefits come at a higher power consumption.

We found the MLX90620 FLIR sensor from Melexis with 16×4 resolution is the right balance between the resolution and power consumption since it has enough resolution to detect human parts at 12 feet from the user while consuming only $\simeq 23$ mW.

From Figures 4a 4b 4c, we observe the measured temperature of human body parts depends on multiple factors such as the skin color and the distance of the person

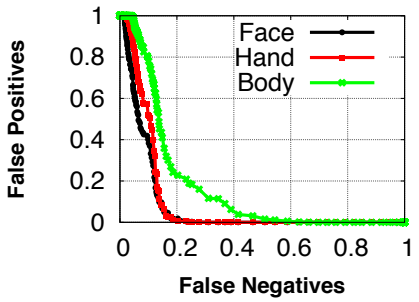


Figure 9: Tradeoff between false positives and false negatives of FIR sensing when the threshold of human temperature changes.

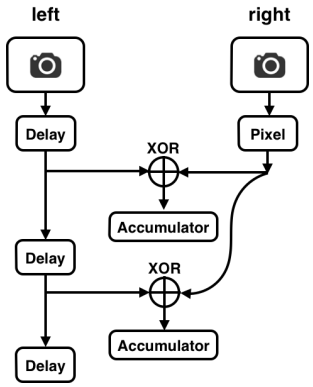


Figure 10: Micro hardware architecture for low-power depth sensing.

from the sensor. Hence, we need to select a temperature threshold for separating human body features from the back ground.

As shown in Figure 9, selecting this threshold is a tradeoff between the false positives and false negatives in FIR-based detection.

A high threshold results in close to zero false positives, while causing high false negatives for far away objects whose measured temperature is lower. A low threshold results in a large number of false positives, while the false negatives become smaller.

While this threshold should be selected based on the specific application needs, we selected 23.7^{o}C as the threshold since this minimizes the sum of false positives and false negatives.

3.3 Depth Sensing

Once the Far Infrared sensor detects potential regions containing objects of interest based on the temperature profile, depth sensing module is triggered for coarse-resolution distance measurement. This section describes design of our low-power depth sensing module.

3.3.1 Hardware Architecture

The main design goals of the depth sensing module are to minimize the processing and storage overhead while providing coarse-granularity depth information of specific regions of the image.

To minimize the processing overhead, the depth sensing module only operates on the regions of interest that are identified by the FIR sensor. To further reduce computational complexity, we use the bit-wise XOR operation of pixel values as a measure of similarity between them (smaller results corresponds to high similarity).

After capturing an image, a typical image sensor outputs the pixels of the image row-by-row as a serial bit stream. To minimize storage requirements, instead of reading the two images from the image sensors to some storage medium, our depth sensor directly operates on the two pixel streams coming from the image sensors.

Here we use the key observation that finding the correspondence between two rows of spatially separated pixels in the two image sensors translates to finding the correspondence between the two pixel streams delayed with respect to each other.

Figure 10 shows the hardware architecture of the of depth sensor design. This is composed of three HW building blocks: delay, XOR, and accumulator.

Delay: Delay modules are used to delay the pixel stream coming from the left image sensor because the right picture is a delayed version of the left image due to spatial disparity. The delayed pixels from the left image are XORed with the pixels from right image to compute disparity.

In our design, delays are implemented with D-flip-flops, that temporarily store the pixels. For example, if each pixel is represented by an 8-bit value, each delay module is an 8-bit D-flip-flop.

XOR: We use bit-wise XOR to compare the difference between two pixels instead of computing the correlation coefficients to reduce power consumption. One potential drawback using XOR is the lower accuracy. However, our evaluations show that XOR is accurate enough for coarse-resolution depth sensing for human detection.

Accumulator: Accumulator aggregates the results of the bit-wise XOR operation and outputs the similarity of the two image blocks. Our design contain multiple XOR-Accumulator blocks that are connected to multiple tap points of the delayed pixel stream of the left image sensor.

Each of these XOR-Accumulator blocks measures the similarity of pixels delayed by a number of pixels corresponding to the length of the delay line up to that block. Hence, these multiple accumulators correspond to different distance estimates of the target object.

For example, in our implementation we use 4 accumulators that correspond to {4, 8, 12, ∞} feet. Once the image blocks pass through all the accumulators, the accumulator values are compared and the object distance is associated with the accumulator holding the smallest sum (bit-wise XOR of two identical pixels is zero).

Hardware complexity: Our hardware depth sensing architecture with its data-flow style processing on multiple simple arithmetic blocks is particularly suitable for direct implementation on an FPGA. We can directly compute the FPGA resource requirements as follows.

Given N delays and M accumulators, the number of D flip-flops (single bit storage elements) needed for implementing this logic is $(8N + 24M)$ because each delay element is an 8-bit register which buffers an 8-bit pixel and a 24-bit register is needed for each accumulator to prevent overflow.

For example, in section 4.3, we show that our design for detecting if an object of interest is at X feet, where $X \in \{4, 8, 12, \infty\}$, can easily fit inside a tiny FPGA or CPLD such as Igloo Nano which only consumes several mW [23]. In contrary, traditional depth sensing, such as Capella [1], needs to be operated on a powerful CPU and consumes several Watts, orders of magnitude higher than our micro hardware architecture.

4. IMPLEMENTATION

We implemented our low-power key body part detector design described in section 3. While, the current prototype measures 8.7cm by 11.5cm, we believe that this can be shrunk to fit a small wearable device.

4.1 Hardware Platform

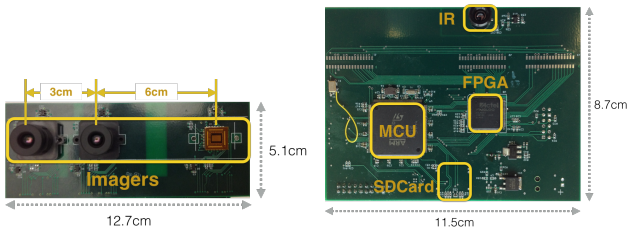
Figure 11 shows our hardware prototype. This prototype is implemented as two modules: an image sensor daughter board and a processor mother board.

The image sensor daughter board is equipped with three Aptina image sensors (MT9V034). As shown in Figure 11a, this enables us to evaluate the accuracy of depth sensing under image sensor separations of 3cm, 6cm, and 9cm.

The motherboard contains FIR sensor (MLX90620) for measuring the scene heat map. The motherboard also contains a low-power FPGA (Igloo Nano AGLN250) and a ARM Cortex M-4 microcontroller (STM32F417IGH6). The low power depth sensing module is implemented in the FPGA while microcontroller coordinates and manages various sensing subsystems.

The microcontroller can also capture and store full images from the image sensors for offline analysis and system debugging.

4.2 Processing Pipelines



(a) Imager daughter board (b) Processor mother board

Figure 11: Prototype of our hardware platform.

Figure 12 shows how different processing pipelines are mapped to various processing elements on the motherboard. The mother board runs three processing pipelines: FIR and image processing pipelines on the MCU and the depth sensing pipeline on the FPGA.

FIR sensor pipeline: The FIR sensor pipe line is more involved than simply reading the temperature values directly from the sensor. The Melexis MLX90620 FIR sensor does not directly output the object temperature. Instead, it compares the amount of radiation sensed against the temperature of the sensor die and returns this relative value.

To determine the object temperature, these readings need to be compensated according to the die temperature. This compensation needs to be done for each pixel because each pixel has different compensation coefficients. Each pixel compensation involves significant amount of floating points calculations with some coefficients of magnitude 10^{-8}.

We managed to reduce duration of this processing pipeline to 3.68ms for a 16×4 pixel IR sensor when the STM32F417IGH6 micro controller runs at 168MHz. To achieve this, we precompute most of the compensation coefficients and store them in RAM. With these optimizations, the MCU only needs to be on for 4% of the time even when the FIR sensor is operated at 10Hz.

FPGA depth sensing pipeline: Figure 12 shows the hardware architecture of depth sensing implemented on an FPGA. The implementation includes three components. First, a window control module is implemented

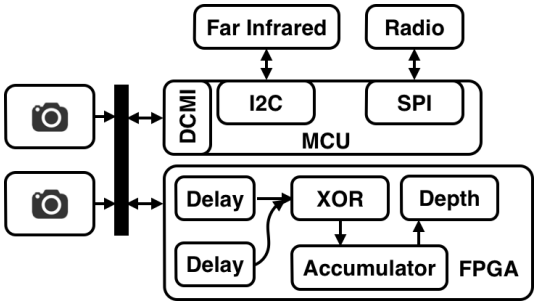
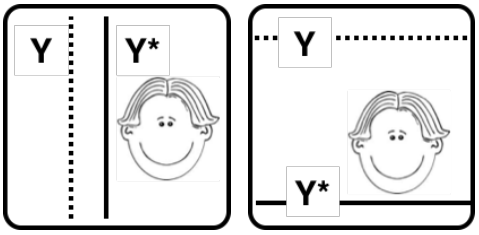
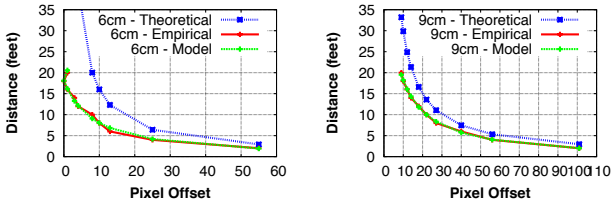


Figure 12: Architecture of our hardware platform.



(a) Horizontal errors (b) Vertical errors

Figure 13: Horizontal and vertical manufacture errors where the observed picture Y^* deviates from its ideal position Y.



(a) 6cm lens separation (b) 9cm lens separation

Figure 14: Theoretical, empirical, and modeled curves for depth estimation using coordinates offset $Y_2 - Y_1$.

for interfacing Aptina image sensors. The window control only pass pixels in a specific region where depth needs to be estimated to the following modules.

Next, delays modules, which are composed by D-flip-flops, are used for achieving synchronization between left and right images. Delay module uses 8-bit D-flip-flops since each pixel is represented as an 8-bit value. Last, accumulators with XOR and summation logic are used for comparing similarity of blocks on left and right images. Depth is estimated based on the content of the accumulator at the end of the image.

MCU image processing pipeline: The image processing pipeline built on an MCU is used to capture and store raw images for system debugging and evaluation. DCMI interface, which is a standard image sensor interface, is used in combination with processor DMA module to capture and transfer images directly to MCU memory.

Once a complete image is captured into the RAM, it is written into a SD card with a FatFS file system. Because SDIO is used for interfacing the SD card, the images can be written in a short time. We also implemented correlation-based depth computation on the MCU for comparing with our low-power FPGA implementation.

4.3 Handling Manufacturing Errors

One challenge in using stereo image sensors for depth sensing is handling manufacture errors. These errors are introduced by stereo image sensor mis-alignment, lens mounts that deviates from the center of image sensors, and other factors. Figure 13a and Figure 13b show two types of errors: horizontal and vertical manufacture errors. The observed object position Y^* deviates from its ideal position Y in horizontal and vertical directions.

4.3.1 *Horizontal manufacturing errors*

Horizontal manufacturing errors shown in Figure 13a are introduced by the horizontal lens deviation from the center of an image sensor plane. Figure 14 shows the theoretical and empirical curves of our hardware platform where stereo image sensor lens are separated by 6cm and 9cm. The theoretical curves are calculated using $L = \frac{DL'}{Y_2 - Y_1}$ with parameters calibrated. For both 6cm and 9cm lens separation, a significant gap between the theoretical and empirical curves is observed. The 6cm lens separation has slightly larger gap than the 9cm case because smaller lens separation is more sensitive to manufacturing errors. This is a result of the smaller pixel offset range (2∼57) under 6cm separation compared to the larger range (1∼102) of 9cm lens separation. We come up a model to compensate these horizontal manufacturing errors.

The model shown in equation 1 is used for capturing the horizontal manufacturing errors, where parameter b deals with errors introduced by lens mount that horizontally deviates from the center of an image sensor, and parameter c deals with the rest of errors within the vision system.

$$D = \frac{a}{Y_2 - Y_1 - b} - c \tag{1}$$

By calibrating parameters a, b, and c as shown in equation 2, both modeled 6cm and 9cm curves fit the empirical curves well as shown in Figure 14, and therefore can be used for practical and accurate depth estimation.

$$D = \frac{160.1026}{Y_2 - Y_1 - 8.5671} - 0.5894 \tag{2}$$

$$D = \frac{298.7770}{Y_2 - Y_1 - 5.7169} - 0.7962 \tag{3}$$

4.3.2 *Vertical manufacturing errors*

Another manufacturing error is the vertical error shown in Figure 13b which is usually introduced by vertical lens deviation from the center of an image sensor plane. This error is not a significant issue for conventional stereo image systems because left and right images are stored in RAM and post-processed, where these vertical errors can be compensated in software. However, vertical manufacture errors are detrimental to our depth estimation solution.

As shown in Figure 13b, the targeted block Y^* will be outputted much later than it is expected because of the vertical offset. While this error can be compensated by adding extra delay elements, the power consumption and HW complexity of the solution increases dramatically due to the large number of storage elements required.

Instead, we use the *windowing* capability of the image sensor to define the vertical boundary of the image windows to compensate the vertical manufacturing errors. The built-in windowing capability of the image sensor can horizontally or vertically trim the captured image and outputs only a *window* of the image. This capability allows us to compensate the vertical manufacture errors at the image sensor itself without extra hardware resources.

For example, when the vertical error of the right image sensor is X rows, which implies that the targeted object will be outputted X rows earlier than expected, we instruct the left image sensor to skip its first X rows and starts its output from the X+1 row. As a result, the images of the left and right image sensors become vertically aligned.

4.4 Mapping between FIR and depth sensors

One particular problem that we need to address is the mapping between FIR and depth sensors. Once the FIR sensor detects the presence of "hot" objects, depth sensor will be triggered to measure the distance to that object. To enable this pipeline, we need to map the object observed by the FIR sensor to the filed of view of the depth sensor. Establishing the mapping is non-trivial due to two reasons. First, mapping needs to be done for sensors with different resolution. The resolution of a FIR sensor is 16×4 pixels while each imager of the depth sensor has 752×480 pixels. Second, mapping needs to be done for sensors with different angle of view. The horizontal and vertical angle of view of a FIR sensor is 20^o and 5^o while depth sensor has 26^o and 18^o.

The mapping between FIR and depth sensors is built following two principles. First, depth is computed only for pixels that are horizontally and vertically covered by both FIR and depth sensors. As a result, some pixels of the depth sensor are not included in the mapping because of the narrow field of view of the FIR sensor. Second, a single FIR pixel is mapped to 35×32 pixels of the depth sensor. Such linear transformation is feasible because both FIR and depth sensors use linear optical systems for imaging.

5. EVALUATION

We evaluate three aspects of our system. How accurate are FIR and depth sensing, both separately and jointly? How much power does our system consume? When coupled with a software body-part detector to confirm parts detected by our front-end, how much power does the system save compared to the pure software approach? In the terminology of Section 2.1, how well does our system serve as an oracle?

5.1 Accuracy of the Hardware Detector

5.1.1 Accuracy of FIR sensing

To evaluate the accuracy with which the FIR sensor detects key body parts, we tested it in a *static* and a *mobile* configuration at *home* and in a university (*"school"*) setting. The settings were empty except for at most a single experimental subject. In the static home case we placed the sensor overlooking the living room; in the static university case we placed it facing a corridor. In the mobile case, we moved around with the sensor as with a wearable, traversing the same living room and corridor. We collected footage for 10 minutes in each configuration, sampling at 8Hz. To test detection accuracy, we had one subject behave "naturally" in each setting, while the experimenter stayed behind the camera (and moved it, in the mobile case). The footage was labeled by hand as to whether a face, wearer-hand or body was showing.

We define the *recall* of our system with respect to a part as the fraction of frames labeled as containing the part where the system correctly detects a part (note FIR-thresholding does not report *which* part). We define the *false positive* with respect to a part as the fraction of all the frames where the FIR detector claims a part is visible where that part was not labeled as visible.

Table 2 shows the results. Perhaps unsurprising, recall rates are quite high, since the background temperature in these indoor settings is at the conventional 20^o level. Encouragingly, however, false positives are relatively low as well. They are higher in the mobile case than fixed case, because when walking around, more lighting fixtures, computer monitors and heating vents come into the field of view. They are also somewhat higher in faces than for hands and significantly higher for bodies. This is because (Figure 4) while wearer-hands stay close to the imager and maintain a relatively high temperature, faces tend to report lower temperatures further away due to the background-averaging effect reported in Section 2.2 and bodies, being covered clothes report even lower temperatures. The lower the reported temperature of a body part, the more objects in the surroundings it may be confused with. It is worth noting that even a relatively high false positive rate does not cause incorrect detection of body parts (since later stages of our system will verify the pixels flagged), but will "only" diminish the power saved by our system.

5.1.2 Accuracy of depth sensing

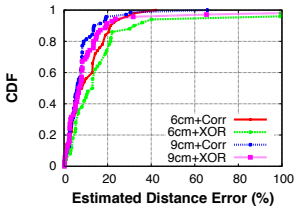
Table 2: Accuracy of detecting key body parts using FIR.

	Detecting human faces				Detecting human hands				Detecting human bodies			
	Static sensor		Mobile sensor		Static sensor		Mobile sensor		Static sensor		Mobile sensor	
	Recall	FP	Recall	FP	Recall	FP	Recall	FP	Recall	FP	Recall	FP
School	91.8%	2.4%	99.7%	13.9%	83.9%	0.9%	99.6%	6.4%	98.8%	36%	99.8%	58.7%
Home	96.8%	0.2%	94.1%	6.3%	95.2%	0.2%	92.9%	6.2%	99.6%	1.1%	98.9%	22.3%

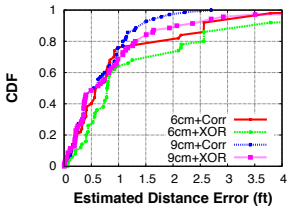
Fine Grained Depth Estimation: We wish to measure how accurately we can estimate depth, and the effect of lens separation and distance measure (correlation versus XOR (Section 3.3.1)) on this accuracy. To this end, we had 14 persons each stand at distances of $2, 4, 6 \ldots 20$ft from the imager, and for each configuration, compared the system-estimated depth with the ground truth. We estimate depth using both 3- and 6-cm lens separation (recall that our board has both separations built in), and using both correlation and XOR-based matching. Figure 15 shows accuracy in both (a) relative (depth error÷ground truth distance) and (b) absolute terms.

Accuracy of estimation is quite good, given the relatively coarse needs for finding key body parts: e.g., for 80% of data points, the error of depth estimation is smaller than 20% and 2 feet. Large errors occur for points a the 2- and 4-ft ranges, as per the limitations of stereo. Further, 9cm lens separation has higher accuracy than the 6cm lens separation: as expected, higher depth resolution is obtained with larger lens separation. Finally, for both 9cm and 6cm lens, our XOR-based matching is less accurate than the conventional correlation-coefficient-based approach. However, the degradation in accuracy is less than 5%, while the gains in reduced power consumption are considerable, as shown later.

Coarse Grained Depth Estimation: We use the 14-person dataset just discussed to estimate coarse distance, rather than fine distance. We group the distances into bins as shown in the first column of Table 3. The bins are intended to represent distances corresponding to "touched objects", "conversation partners" , "people possibly performing relevant actions" and "everything else". We classify the dataset into one of these four categories by running the micro hardware architecture. *Recall* is the fraction of true instances of the class correctly inferred. The *false positive* fraction (FP) is the fraction of claimed instances of a class that were incorrect. As the table indicates, our depth sensor excels at coarse estimation, with both 6- and 9-cm baselines.



(a) Error in percentage



(b) Error in feet

Figure 15: Accuracy of fine-grained depth estimation.

Table 3: Accuracy of coarse depth estimation.

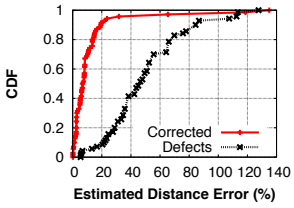
	9cm lens		6cm lens	
Depth (ft)	Recall	FP	Recall	FP
(2,4)	93%	0%	83%	0%
(6,8,10)	83%	0%	88%	2%
(12,14,16)	91%	5%	83%	4%
(18, 20, ∞)	100%	5%	91%	10%

Impact of Correcting Manufacturing Errors: As mentioned in section 4.3, manufacturing errors can be divided into horizontal and vertical errors. We address these by using imager hardware windowing and a correction model. Figure 16 illustrates the impact of error correction on the 14-person dataset above. When errors are corrected, only 10% of depth estimates have errors above 2ft absolute, or 20% relative to ground truth distance. Uncorrected, over 70% of estimates have errors above 10ft absolute/60% relative. Correction is therefore essential. It is worth noting here that our error correction has low power overhead: neither the hardware windowing nor the horizontal correction model requires additional computation, since they require just a one-time configuration.

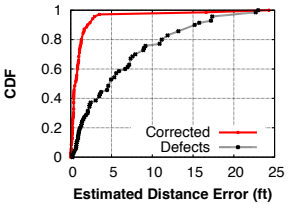
5.2 Power Draw of the Hardware Detector

5.2.1 *Power draw for platform components*

Table 4 shows the static and active current consumption of each component in our hardware platform. The static current draw of the whole hardware platform is 4mA, which is measured when the IR sensor is not initial-



(a) Relative error.



(b) Absolute error.

Figure 16: Impact of correcting manufacture errors.

Table 4: Component-wise power consumption. × indicates that the corresponding state does not exist.

Power (mA)	Component active	asleep
Static Current Draw	×	4
16×4 IR sensor	9	×
752×480 Stereo Imagers	73	11*
MCU	55	0.002
FPGA	2.58	0.01

ized, stereo imagers are not mounted, the MCU stays in STANDBY mode and the FPGA clock is disabled. The IR sensor consumes 9mA after initialization and unfortunately, cannot be turned into sleep mode due to its lack of duty cycling mechanism. The stereo imagers, which are used for depth estimation, consume 11mA in STANDBY mode and 73mA in active mode. The MCU consumes 55mA and 2.3mA for the FPGA, which follow the corresponding data sheet specification.

Table 5 shows the timing of tasks executed on the hardware platform. Some tasks need to be done only once, such as stereo imager initialization and SD card initialization. Other tasks, such as FIR and depth sensing, need to be done frequently. FIR sensing takes 3.68ms and depth estimation, including exposure and sensing, takes 16ms. Although operating the MicroSD is time-consuming, it is designed for debugging and not used in runtime system.

5.2.2 *Power draw for estimating depth*

The power consumption of depth sensing depends on the physical distance of targeted objects because closer objects occupy more pixels. Figure 17 compares the distribution of average power drawn for processing frames comprising our 14-person dataset when our FPGA-based architecture is used (red line labeled "Micro Arch") to using a low-power microcontroller ("MCU"). Critically, even when frames are *filled* with body pixels (e.g. at 100% of the CDF), power draw tops off at 5.1mW (median 3.1mW). Thus, although this evaluation was on detecting bodies, we expect wearer-hands, which often dominate the field of view, to draw acceptable power.

Our efforts in adapting our algorithms to the low-power FPGA seem worthwhile. Power consumed by the FPGA-based architecture (the black line in Figure 17) is roughly 18× lower than when an MCU (the STM32F417IGH6) is used. There are three reasons for this reduction. First, interfacing the imager with an FPGA avoids the software driven DCMI interface on an MCU, and is thus significantly cheaper. Second, the FPGA processes pixel streams in real time and as a natural result, avoids pixel storage overhead, such as SRAM. Third, some key operations, such as XOR, are simply cheaper on an FPGA.

Table 5: Timing of tasks executed on the platform.

	Time (ms)
64 pixels FIR sensing	3.68
depth sensing	15
Stereo imager initialization	1.2
Stereo imager exposure	1
MicroSD Card initialization	2400
MicroSD Card logging	125

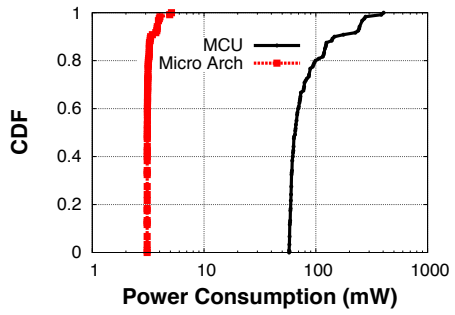
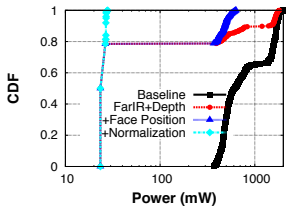
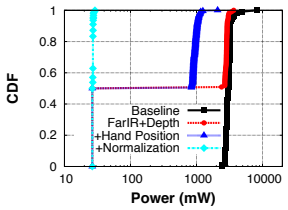


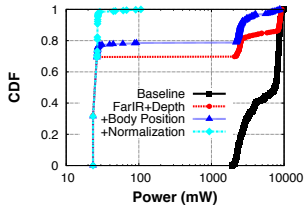
Figure 17: Power consumption for estimating depth.



(a) Detecting Faces



(b) Detecting Hands



(c) Detecting Bodies

Figure 18: Power consumed for detecting key body parts using our system (compare Figure 3).

5.3 Overall System Evaluation

In Section 2.1, we pointed out that the overall goal of our system is to realize an oracle that indicates where and at what scale in the imager frame a software parts-detector should search for key body parts. Figure 3 (cyan line) pointed out that such an oracle, if it ran at zero power, could then detect body parts at 1mW *in software*. We now examine to what extent our FIR/depth-based architecture achieves this goal. Similar to Section 2.1, we collected footage of subjects from 3 to 15 feet away in a corridor, and a hand moving from 6 inches to 2 feet. We then used our system to detect parts, bound their pixel windows and scales, and executed OpenCV-based body-part detectors on these

windows on a Galaxy Nexus. We added the measured energy consumed on our detector to that of detecting parts within those windows on the phone and divided by total execution time to get average power draw.

The cyan lines on Figures 18a-c illustrate the performance of our oracle. The entire system runs at roughly 24mW when no body part is detected (this corresponds to running the 9mA FIR sensor at 2.6V), and roughly 28mW when a body part is detected (an additional mW for the on-phone calculation and 3mW to run the depth unit). Given our 70mW budget, **we believe these numbers provide compelling evidence that an architecture cascading FIR and depth sensing via today's low-power off-the-shelf components (such as our FPGA) could enable key body part detection on wearable devices**, at least in areas where background temperature is outside the 30-35C range, and in temperature controlled buildings.

6. RELATED WORK

Body parts detector: There is an extensive body of work on algorithms for identifying key human body parts, including faces [36, 31, 27], torsos [38, 22], and hands [15, 16]. These algorithms have been used in various applications, including human computer interfaces [39], smart rooms [43], and intelligent robots [40]. Our work does not target specific algorithmic improvements. Instead, we investigate how multiple sensing modalities can be used as pre-filters to limit the invocation of expensive algorithms.

Low-power imagers: Ultra low-power imagers, such as [13, 5], consume less than 1mA of current when active. In addition, various mechanisms have been proposed by [17, 21, 12] for reducing the power consumption of imagers. Therefore, imagers will not likely be the bottleneck for achieving low power depth estimation. Instead, reducing the processing overhead is going to be the key to achieving low power depth estimation.

Low-power depth sensing: Computer vision community has investigated accurate disparity map computation using stereo web cameras [35], reducing the computational overhead of disparity map calculation [11], 3D image construction based on disparity map [20], etc. These approaches focus on improving computer vision algorithms while our focus is to improve the hardware architectures. We see these as complementary efforts.

Some previous work use FPGAs for efficient depth computation. [34] presents a hardware architecture for computing depth map which avoids modifications of depth algorithms. [32] designed an optimized hardware architecture for computing depth map in 1D. [25, 3] discuss low cost FPGA implementation of specific depth algorithms. These approaches focus on computing the complete disparity map while our focus is on estimating the depth of specific objects within the scene.

[41, 42, 35] propose several algorithms for stereo vision calibration when manufacture defects are present. These algorithms are computationally expensive to run continuously on a battery power device. We use a simplified correction model for achieving low power consumption at the cost of small degradation of depth estimation performance.

Stereo imager platform: Stereo camera hardware platforms have been on the market for a long time. [2] is a commercial stereo camera that supports capturing of two images simultaneous at different sensitivity, magnification, and color tonality. Unfortunately, this camera is not a programmable device which prevents its use in low-level investigations. [1] is a stereo camera development board that allows low-level firmware configuration of imagers. However, detailed power benchmarking is hard on this platform because of its deep coupling between OS and stereo image processing, and additional on board hardware components (Ethernet, USB, HDMI). In contrast, our hardware platform allows detailed power benchmarking of individual hardware modules.

Far Infrared sensing: [8] uses FarIR sensors to determine the occupancy of zones within a building. [10] exploits low power 8×8 FarIR sensor array to monitor human activities in home and office environments. While both these work provide valuable insight into using FarIR for human detection, none of them investigate the possibility of combining depth sensor for detecting objects of interest.

7. CONCLUSION

In this paper, we present a low-power human body part detector that exploits Far Infrared (FIR) and depth sensors to gate the traditional vision pipeline. We make two main contributions. First, we design and implement a low-power hardware architecture which is able to obtain depth at low power. Second, we show that the combination of FIR and depth sensors serves as a means to detect (parts of) people their in the field of view. When combined with software classifiers that can identify the body parts, our system can identify "key" body parts at identical accuracy to the pure software solution but at much lower power. We believe our paper only scratches the surface of developing gating circuitry for vision, and fully expect both more general and lower power gating systems to emerge in the near future.

8. REFERENCES

[1] Capella - stereo vision camera reference design. `http://www.e-consystems.com/stereo-vision-camera.asp`.

[2] Fuji finepix real 3d w3. `http://www.fujifilm.com/products/3d/camera/finepix_real3dw3/`.

[3] Stereographic depth mapping on an fpga. https://courses.cit.cornell.edu/ece576/FinalProjects/f2010/pfk5_jk459/pfk5_jk459/index.html.

[4] Sharad Agarwal, Matthai Philipose, and Paramvir Bahl. Vision: The case for cellular small cells for cloudlets. In *International Workshop on Mobile Cloud Computing and Services*, 2014.

[5] Suat U Ay. A 1.32 pw/frameě pixel 1.2 v cmos energy-harvesting and imaging (ehi) aps imager. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, pages 116–118. IEEE, 2011.

[6] Junguk Cho, Shahnam Mirzaei, Jason Oberg, and Ryan Kastner. Fpga-based face detection system using haar classifiers. In *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*, pages 103–112. ACM, 2009.

[7] E. L. Dereniak and G. D. Boreman. *Infared Detectors and Systems*. Wiley, second edition, 1996.

[8] Varick L Erickson, Alex Beltran, Daniel A Winkler, Niloufar P Esfahani, John R Lusby, and Alberto E Cerpa. Toss: Thermal occupancy sensing system. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, pages 1–2. ACM, 2013.

[9] Kiryong Ha, Zhuo Chen, Wenlu Hu, Wolfgang Richter, Padmanabhan Pillai, and Mahadev Satyanarayanan. Towards wearable cognitive assistance. In *MobiSys*, 2014.

[10] Peter Hevesi, Sebastian Wille, Gerald Pirkl, Norbert Wehn, and Paul Lukowicz. Monitoring household activities and user location with a cheap, unobtrusive thermal sensor array. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 141–145. ACM, 2014.

[11] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 807–814. IEEE, 2005.

[12] Steve Hodges, Lyndsay Williams, Emma Berry, Shahram Izadi, James Srinivasan, Alex Butler, Gavin Smyth, Narinder Kapur, and Ken Wood. Sensecam: A retrospective memory aid. In *UbiComp 2006: Ubiquitous Computing*, pages 177–193. Springer, 2006.

[13] K Kagawau, Sanshiro Shishido, Masahiro Nunoshita, and Jun Ohta. A 3.6 pw/frameâ· pixel 1.35 v pwm cmos imager with dynamic pixel readout and no static bias current. In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 54–595. IEEE, 2008.

[14] Takeo Kanade and Martial Hebert. First-person vision. *Proceedings of the IEEE*, 100(8):2442–2453, 2012.

[15] Mathias Kolsch. *Vision based hand gesture interfaces for wearable computing and virtual environments*. University of California, Santa Barbara, 2004.

[16] Mathias Kölsch and Matthew Turk. Robust hand detection. In *FGR*, pages 614–619, 2004.

[17] Robert LiKamWa, Zhen Wang, Aaron Carroll, Felix Xiaozhu Lin, and Lin Zhong. Draining our glass: An energy and heat characterization of google glass. *arXiv preprint arXiv:1404.1320*, 2014.

[18] Shai Litvak, Tomer Yanir, and Eran Guendelman. Tracking body parts by combined color image and depth processing, July 22 2014. US Patent 8,787,663.

[19] Haowei Liu, Matthai Philipose, Martin Pettersson, and Ming-Ting Sun. Recognizing object manipulation activities using depth and visual cues. *Journal of Visual Communication and Image Representation*, 25(4):719–726, 2014.

[20] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

[21] WW Mayol, B Tordoff, and DW Murray. Towards wearable active vision platforms. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 3, pages 1627–1632. IEEE, 2000.

[22] Antonio S Micilotta, Eng-Jon Ong, and Richard Bowden. Real-time upper body detection and 3d pose estimation in monoscopic images. In *Computer Vision–ECCV 2006*, pages 139–150. Springer, 2006.

[23] Microsemi. Igloo nano fpgas.

[24] Karsten Mühlmann, Dennis Maier, Jürgen Hesser, and Reinhard Männer. Calculating dense disparity maps from color stereo images, an efficient implementation. *International Journal of Computer Vision*, 47(1-3):79–88, 2002.

[25] Chris Murphy, Daniel Lindquist, Ann Marie Rynning, Thomas Cecil, Sarah Leavitt, and Mark L Chang. Low-cost stereo vision on an fpga. In *Field-Programmable Custom Computing Machines, 2007. FCCM 2007. 15th Annual IEEE Symposium on*, pages 333–334. IEEE, 2007.

[26] OpenCV. Haar feature-based cascade classifier for object detection.

[27] Edgar Osuna, Robert Freund, and Federico Girosi. Training support vector machines: an application to face detection. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 130–136. IEEE, 1997.

[28] Hamed Pirsiavash and Deva Ramanan. Detecting activities of daily living in first-person camera views. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2847–2854. IEEE, 2012.

[29] Moo-Ryong Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan. Odessa: Enabling interactive perception applications on mobile devices. In *Mobisys*, 2011.

[30] Swati Rallapalli, Aishwarya Ganesan, Krishna Chintalapudi, Venkat N Padmanabhan, and Lili Qiu. Enabling physical analytics in retail stores using smart glasses. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 115–126. ACM, 2014.

[31] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1):23–38, 1998.

[32] Yuki Sanada, Katsuki Ohata, Tetsuro Ogaki, Kento Matsuyama, Takanori Ohira, Satoshi Chikuda, Masaki Igarashi, Tadahiro Kuroda, Masayuki Ikebe, Tetsuya Asai, et al. Fpga implementation of a memory-efficient stereo vision algorithm based on 1-d guided filtering.

[33] Yale Song, David Demirdjian, and Randall Davis. Tracking body and hands for gesture recognition: Natops aircraft handling signals database. In *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pages 500–506. IEEE, 2011.

[34] Ngo Huy Tan, Nor Hisham Hamid, Patrick Sebastian, and Yap Vooi Voon. Resource minimization in a real-time depth-map processing system on fpga. In *TENCON 2011-2011 IEEE Region 10 Conference*, pages 706–710. IEEE, 2011.

[35] Roger Y Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *Robotics and Automation, IEEE Journal of*, 3(4):323–344, 1987.

[36] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[37] He Wang, Xuan Bao, Romit Roy Choudhury, and Srihari Nelakuditi. Insight: recognizing humans without face recognition. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, page 7. ACM, 2013.

[38] Fengliang Xu, Xia Liu, and Kikuo Fujimura. Pedestrian detection and tracking with night vision. *Intelligent Transportation Systems, IEEE Transactions on*, 6(1):63–71, 2005.

[39] Ming-Hsuan Yang and Narendra Ahuja. *Face detection and gesture recognition for human-computer interaction*, volume 1. Springer Science & Business Media, 2001.

[40] Woo-han Yun, DoHyung Kim, and Ho-Sub Yoon. Fast group verification system for intelligent robot service. *Consumer Electronics, IEEE Transactions on*, 53(4):1731–1735, 2007.

[41] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.

[42] Zhengyou Zhang. Camera calibration with one-dimensional objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(7):892–899, 2004.

[43] Zhenqiu Zhang, Gerasimos Potamianos, Ming Liu, and Thomas Huang. Robust multi-view multi-camera face detection inside smart rooms using spatio-temporal dynamic programming. In *Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on*, pages 407–412. IEEE, 2006.