Adaptive Resolution-Based Tradeoffs for Energy-Efficient Visual Computing Systems

Robert LiKamWa, Jinhan Hu, Venkatesh Kodukula, and Yifei Liu, *School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ, 85287, USA*

The real world presents interpretable visual detail at different scales in different situations. While empowering face recognition, augmented reality, and other computer vision tasks, mobile systems should be able to dynamically adapt the spatiotemporal resolution of the visual sensing pipeline to capture image frames at high resolutions for task precision and low resolutions for energy savings. Facilitating real-time decisions to reconfigure resolutions will let systems dynamically adapt to the needs of the vision algorithms, as well as the environmental situation of the visual scene. This article will review system challenges and opportunities of image-resolution-based tradeoffs toward energy-efficient visual computing through device driver and media framework optimization.

Computer vision has dramatically transformed the capabilities and possibilities of modern mobile systems. QR scanning can translate barcodes, face recognition can unlock phones, object recognition can search online stores for products, and visual–inertial odometry can place virtual objects on physical surfaces through augmented reality. Continuous vision on mobile systems enables untethered headsets such as the Microsoft HoloLens and the Oculus Quest to track their environment through using cameras to understand their world as the user moves through it. The ability to sense and process visual information allows mobile devices to connect with and respond to the physical world.

However, capturing and processing image frames for computer vision is an energy-expensive operation. This not only poses issues for battery life, but also for comfort with wearable devices. A Google Glass lasts no more than 45 min and raises the surface temperature of the device to 50 °C[1] when running a video chat application. High power consumption is especially common when high precision is required, due to the memory traffic generated by high resolution frames captured at high frame rates.[2]

Fortunately, vision algorithms need not operate on full high-resolution frames all of the time. Object tracking can target operations on bounding boxes that follow detected objects to reduce computational workload.[4] Image-target-based tracking for augmented reality can use smaller frames when targets are nearby,[3] as shown in Figure 1. Temporal redundancy can be exploited to reuse previously detected visual features for object tracking.[5,6] Reducing the number of pixels in the frames opens the opportunity to transmit frames to networked resources for offloaded visual processing.[7] Scalable video coding techniques[8] allow systems to make on-demand decisions to stream and decode frames with lower resolutions and frame rates as needed, scaling the bitrate to suit the quality needs and streaming capabilities of the device.

The variable resolution needs of vision algorithms and applications present a strong opportunity: Mobile systems should support adaptive resolution-based tradeoffs toward energy-efficient visual computing. This can lead to energy-efficient operation; the visual computing pipeline can be reconfigured to capture fewer pixels. Ideally, such adaptivity would occur "early" in the visual sensing pipeline, discarding pixels—or preventing their sensing—before incurring energy costs of analog-to-digital conversion, interface traffic, and memory traffic.

However, building in support for such adaptivity is not without its challenges. While much sensor hardware already supports resolution reconfigurability, current systems have been designed around setting

Digital Object Identifier 10.1109/MPRV.2021.3052528

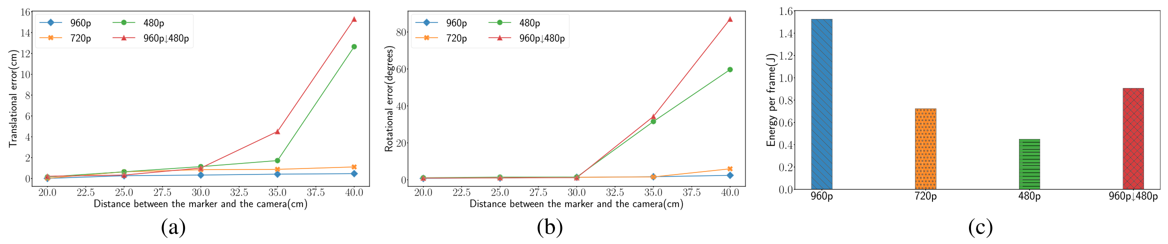

FIGURE 1. Our case study[3] around a marker-based pose estimation app demonstrates that the task accuracy (translation and rotation error) could be maintained with a 70% reduction in system energy consumption (energy per frame), if sensor resolution is reconfigured from 960p to 480p when the camera is approaching the image marker from 35 cm to 20 cm. (a) Low resolution maintains a low translation error when marker is close to camera. (b) Low resolution maintains a low rotation error when marker is close to camera. (c) System energy consumption at 480p is 70% and 50% less than 960p and downsampling, respectively.

resolution for video capture and image capture, not adapting to dynamic visual computing needs. As such, current systems limit the possible energy-efficiency of dynamically changing sensor resolution and introduce substantial delays, including frame drops, while reconfiguring the sensor resolution. Our ongoing research efforts aim to provide system support to overcome these limitations and unlock the benefits of adaptive-resolution sensing.

In this article, we provide an overview of recent works to discuss the opportunities and challenges of supporting image resolution-based tradeoffs at the sensor level and operating system (OS) level. We first provide a background of visual computing systems and their major components, highlighting adaptive resolution opportunities. We next discuss sensor management techniques to create energy-proportional image capture through driver-level control. We then discuss Banner, an adapted media framework that facilitates rapid resolution reconfiguration with no frame drops during reconfiguration actions. Finally, we discuss future avenues of research to facilitate fine-grained adaptive resolution-based tradeoffs, toward energy-efficient visual computing systems.

MULTIRESOLUTION OPPORTUNITIES IN THE VISUAL COMPUTING PIPELINE

The visual computing pipeline uses: 1) image sensors to capture pixels, 2) image signal processors to perform color conversion and place the data into memory, 3) OS media frameworks to synchronize computational frame processing, and 4) vision libraries to enable developers to invoke algorithmic processes. Each of these elements provides opportunities for resolution-adaptive behaviors and opportunities for commensurate energy-efficiency tradeoffs.

Image Sensor. Image sensors convert physical light into digitally usable data through pixel arrays and read-out circuits that are formed of signal amplifiers, analog-to-digital converters, and interface I/O.

The power consumption of these analog components is nonnegligible, and generally scales with the number of pixels in the sensor. Image sensors often provide the ability to program their spatial resolution, frame rate, and region of interest by setting register values.

On mobile systems, the image sensor is typically connected to the System-on-Chip via a serial interface, usually following a Camera Serial Interface 2 (CSI-2) standard from the MIPI Alliance.

Image Signal Processor (ISP). ISPs conduct quality-improving operations on image frames, such as Bayer transformation, demosaicing, noise reduction, and image sharpening. The architecture of ISP employs fully streaming functional units with the optimization of data-level parallelization. The resulting frames are sent to a DRAM frame buffer for further computational use.

Notably, ISP usage may be significantly reduced for visual computing workloads.

A recent study have shown that a reconfigurable ISP for neural network workloads could save 75% energy without impacting task accuracy, suggesting an opportunity to utilize dynamic resolution for neural network workloads.[9]

Media Frameworks. From the perspective of the OS, the driver of image sensors acquires packed-data in frame buffers and coordinates with the image sensor over I2C or SPI control. To properly operate the image sensor's registers, the system needs software drivers, such as V4L2, executing in kernel mode in combination with the signal processing Application Programming Interface (API) for userspace applications. Media frameworks such as Android Camera 3

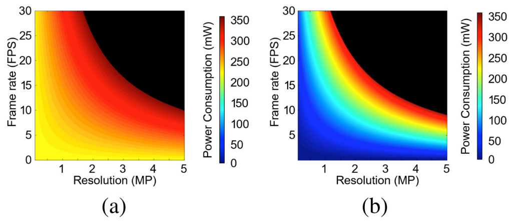

FIGURE 2. (a) Legacy sensor power consumption versus resolution, frame rate. Power consumption does not significantly scale down with reduced frame resolution or frame rate. (b) Improved energy-proportional sensor power consumption versus resolution, frame rate through aggressive standby and pixel clock scaling mechanisms.

and Frankencamera model the camera device as a pipeline that provides programmable image processing APIs, allowing the developer to specify image resolutions and other format needs.

Developers will use the media frameworks to process image frames through visual algorithms. Developers may operate directly on the frames or use vision libraries, such as OpenCV or Apple Core ML APIs, to perform algorithmic operations. For computational performance, developers have the ability to resize and crop image frames before executing vision library processing through these APIs. These abstractions allow developers to creatively produce applications that send frames through a set of stably implemented vision algorithms to deduce visual information from image frames, building in resolution-adaptive behavior as desired.

TOWARD ENERGY-PROPORTIONAL IMAGE CAPTURE THROUGH SENSOR CONTROL[10]

Within the sensor, one might expect that lower resolution operation consumes less power than higher resolution operation. Indeed, if one compares the average power consumption of a 640 × 480 sensor to a 4 K sensor, there will often be an order of magnitude disparity in power consumption. However, when using a high-resolution sensor to capture at lower resolutions, we find that the average power consumption of the sensor is still relatively high [see Figure 2(a)]. That is, by default, image sensing is not energy-proportional to resolution at the sensor level.

To understand the sources of energy consumption, we perform a power characterization of five image sensors from two major vendors, spanning power profiles of 250 to 340 mW and maximum resolutions from 768 × 506 to 3264 × 2448. These image sensors are powered with dedicated voltage rails, supplying current to different components: The pixel array, the analog signal chain, the image processor, and the digital controller. We intercept these power rails using a National Instruments DAQ device to measure the power consumption of various sensor components. We also use a programmable clock to study the implications of pixel clock frequency on the sensor power consumption.

Modeling Active and Idle Power

Across all of our measurements, we noted an active period when the power consumption stays high (P_{active}), contrasted with an idle period when the power consumption stays low (P_{idle}). We find that the active period depends on pixel count (N) and clock frequency (f) as shown in (1). That is, every time the clock toggles, one pixel gets read out; this happens for all the pixels during the active period. The idle period is the remainder of the time between the frames, as shown in (2). Energy per frame can be calculated in terms of these parameters, as shown in (3):

$$T_{\text{active}} = \frac{N}{f} \tag{1}$$

$$T_{\text{idle}} = \frac{1}{R} - T_{\text{active}} \tag{2}$$

$$E_{\text{frame}} = T_{\text{active}} * P_{\text{active}} + T_{\text{idle}} * P_{\text{idle}} \tag{3}$$

From the previous equations, there are two system controls—resolution (N) and frame rate (R)—that determine the interplay between active period and idle period. Here, we discuss the implications of these controls on the energy proportionality of the sensor.

Since the active period depends on the number of pixels, scenarios where the system needs to capture at high resolution leads to longer active periods. On the other hand, scenarios where the system needs a low resolution capture or needs to capture only a region within a frame, the active period dramatically reduces. However, in low-resolution cases, the idle period increases significantly, leading to wasteful consumption of idle energy [see Figure 3(a)]. Along similar lines, lower frame rates make the sensor spend more time in idle mode, again consuming wasteful energy [see Figure 3(b)]. As a result, low frame rates and spatial resolutions do not produce commensurate energy savings, thereby limiting the sensor's energy proportionality.

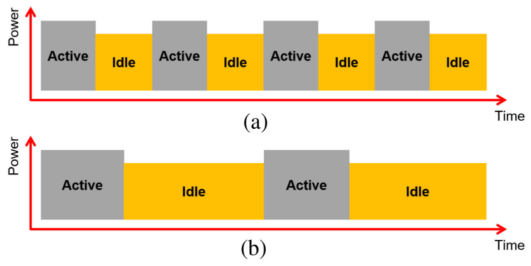

FIGURE 3. Idle period proportion increases with low resolution and frame rate, leading to wasteful consumption of idle energy. (a) Low resolution. (b) Low frame rate.

Driver-Based Energy-Proportional Sensing Techniques

Toward energy proportionality, we introduce two system-level techniques that create desirable tradeoffs between sensor energy and image quality/frame rate. The first technique aims to minimize the idle energy consumption of the sensor through aggressive use of the sensor's standby mode. The second technique aims to minimize the active energy consumption by reshaping the power and time characteristics through clock frequency scaling. These techniques are fairly simple; they can be easily implemented in device drivers for current and future image sensors.

Technique #1: Aggressive Standby

Modern image sensors provide a standby mode, in which the image sensor operates in ultralow-power mode without performing any capture while consuming minimal power, on the order 10μW. We aggressively put the sensor into standby mode in the idle period between active captures. There is a caveat; we cannot push the image sensor in standby mode during the entire idle period, as we need to expose the image sensor for photon collection. Thus, with standby mode, the idle time is now constrained with exposure time instead of frame rate. As a result, we allow the sensor to spend most of the time in standby mode and minimal time in idle mode for exposure for optimizing power savings.

This technique works well for low-quality sensor settings where there is plenty of time between captures for both putting the sensor into deep sleep (standby mode) and exposing (idle mode) the sensor, thereby making it suitable for vision applications. On the other hand, with higher resolutions and frame rates, this window becomes much smaller due to larger active periods, limiting the applicability of this technique.

Technique #2: Clock Scaling

Our clock scaling technique is inspired from the clock scaling techniques used in microprocessors for trading power consumption with execution time for a given energy budget. Changing clock frequency has significant implications on the image sensor's efficiency, particularly for vision applications. This can be accomplished with little additional hardware, i.e., a programmable oscillator.

In the context of image sensors, a faster pixel clock leads to lower active times but also raised active and idle power. On the other hand, a slower pixel clock lowers the active/idle power at the cost of increased active periods. The optimal clock frequency, which we analytically derive, is a function of the number of pixels and frame rate of the frame workload, as shown in (4). The optimal frequency also depends on constants (c_1, c_2) are sensor-specific and needs to be calibrated through sensor power characterization:

$$f_{\text{best}} = \sqrt{\frac{c_1 * N}{c_2 * T_{\text{exp}}}} \qquad (4)$$

Integrating Sensor Control Into the OS for Energy Proportional Sensing

These aggressive standby and clock optimization techniques create opportunities to enable energy proportionality in image sensors in the multiresolution computing pipeline. As shown in Figure 2(b), the image sensor consumes higher power (330 mW) for high resolution and high frame rate settings. For lower quality scenarios, the image sensor consumes lower power; at 0.1 Mp at 3 frames per second, the image sensor consumes roughly 10 mW of power.

These techniques can be integrated into systems by using a configurable clock for the sensor and by changing the software drivers of the OS. More challenging, media frameworks will have to trigger the changes to suit the adaptive needs of the vision algorithms and applications, synchronized with the capture patterns of the sensing pipeline to create energy proportional sensor operation.

BANNER: A CAMERA FRAMEWORK FOR SEAMLESS RECONFIGURATION[3]

To utilize the resolution-based accuracy and energy tradeoff at runtime, mobile devices need the ability to reconfigure their sensors' resolution on a per-frame basis in real time.[11] However, any sensor resolution change leads to a substantial pause in frame delivery. As we measured in the Android OS on a Nexus 5X phone, reconfiguring sensor resolution stops the frame delivery to the application for about 267 ms, which is equivalent to dropping 9 frames when the

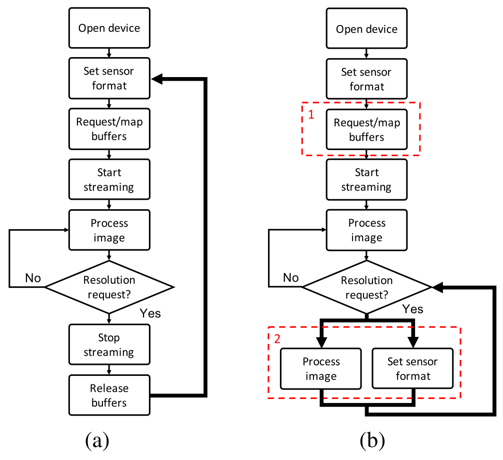

FIGURE 4. In Banner, (1) format-oblivious memory management avoids repeated memory allocation and parallel reconfiguration sets sensor format in parallel with application processing frames (2). (a) Resolution reconfiguration in legacy V4L2. (b) Resolution reconfiguration in Banner.

application works at 30 FPS. Therefore, modern vision applications cannot afford the detrimental effects that come with resolution change at runtime. Specifically, AR requires working on continuous frames to maintain smooth and immersive but fragile user experiences, despite the substantial resolution-based energy savings. More severely, resolution reconfiguration latency can be identified across all systems, including Android, iOS, and Linux.

Bottlenecks of Reconfiguration Latency

Based on our understanding of the current sensor reconfiguration system stack, we found that sensor resolution reconfiguration is bottlenecked by repeatedly invoking a sequence of expensive system calls when new resolutions are requested, as shown in Figure 4(a). This repeated sequence consists of the following steps.

1) The app initializes a new resolution request.
2) Current working streams are turned OFF.
3) The allocated memory is unmapped and released.
4) The new resolution request is processed to set sensor's output format.
5) Based on the requested resolution, video buffers are allocated and mapped.
6) Video channels for the new resolution are ready to be started for streaming. The first frame at the new resolution will be received after a pipeline latency depending on ISP stages.

In this sequence, the sensor resolution is strictly synchronized across the system stack. First, video buffers are allocated based on resolution. Second, the abstraction of video channel is also set based on resolution. As a result, once there is a new resolution request, previously allocated video buffers and channels are required to be cleared and reallocated. Every time, the whole configuration sequence described earlier will be repeatedly invoked, creating a substantial latency.

To mitigate sensor resolution reconfiguration latency, we introduce the Banner media framework, which employs *parallel reconfiguration* and *format-oblivious memory management*. With these two techniques, the repeated reconfiguration sequence is avoided, as shown in Figure 4(b). As a result, Banner allows vision applications to request different sensor resolutions with seamless frame delivery, i.e., no frames drop during resolution reconfiguration.

Technique #1: Parallel Reconfiguration

Banner reconfigures sensor resolution in parallel with the application processing for frames through thread-level concurrency. Deep inside the sensor system stack, a capture thread is responsible for capturing frames. When this capture thread is awake, it occupies the sensor resources for capturing frames. On the other hand, when there are no capture requests queued, the capture thread is frozen, which sets the sensor free to be reconfigured. Banner deploys a separate reconfiguration thread to occupy sensor resources for processing resolution requests while the application is processing for frames and the capture thread is frozen. Time sharing between capture thread and reconfiguration thread maximizes sensor resources utilization.

This leads to a reconfiguration timing budget T_{budget}, which must be obeyed in order to reconfigure sensor resolution such that the frame delivery speed will not be degraded. T_{budget} is simply the difference between the frame interval T_{interval} and the frame capture time T_{capture}, as shown in (5). In our implementation, we observe that capturing 1080p frames at 30 FPS (33.3 ms T_{interval}) takes less than 20 ms of T_{capture} time. This leaves Banner with a comfortable reconfiguration timing budget T_{budget} of at least 13.3 ms

$$T_{\text{budget}} = T_{\text{interval}} - T_{\text{capture}}. \tag{5}$$

Technique #2: Format-Oblivious Memory Management

To remove resolution synchronization in the sensor reconfiguration system stack such that the repeated reconfiguration procedure could be avoided, Banner manages memory allocation in a format-oblivious fashion, consisting of one-time buffer allocation and format-oblivious frame delivery.

Banner only allocates memory buffers once, for the highest supported resolution. In this way, allocated buffers will have enough space to be reused for any other resolution requested. In particular, one-time buffer allocation removes the system calls to deallocate and allocate buffers, as well as stop and start video streams. Those four system calls contribute most to this long resolution reconfiguration latency.

In current sensor reconfiguration system stack, the amount of bytes for frames to be delivered is determined by the payload calculated based on the format. However, Banner delivers frames based on how many bytes are used because resolution synchronization is removed. With Banner integrated, only the application and the sensor need to know the right format to ensure correct frame capture, delivery, and interpretation.

Implementation/Evaluation

We implemented and evaluated Banner in the Video4-Linux (V4L2) media framework on an NVIDIA Jetson TX2 board running a Linux system (kernel V4.4) with an ON Semiconductor AR0330 image sensor integrated. The Jetson TX2 board is widely used in embedded computing applications. On top of Banner, applications can reconfigure sensor resolution rapidly through only one ioctl(VIDIOC_RECONFIGURE) system call.

We evaluated Banner in three AR use cases integrated with OpenCV. The first application displays camera frames at 25 FPS. The second application offloads frames to a desktop server through a direct connection at 15 and 30 FPS. The third application estimates camera pose on image markers at 15 FPS. All three applications cycle through three resolution formats, i.e., 1920 × 1080 (1080p), 1280 × 720 (720p), and 640 × 480 (480p), at a constant frame rate mentioned earlier accordingly. In addition, we compare Banner against computational downsampling, in which we use OpenCV resize() function to downscale 1080p frames to 480p (represented as 1080p↓ 480p).

Reconfiguration Latency Reduction. We break down the reconfiguration latency into end-to-end (E2E) reconfiguration latency and frame-to-frame (F2F) latency. E2E reconfiguration latency is the time between an application's request to change resolution and the time the application receives a frame of the new resolution. F2F latency is the interval between two frames provided to the application in which the latter frame is at the new resolution.

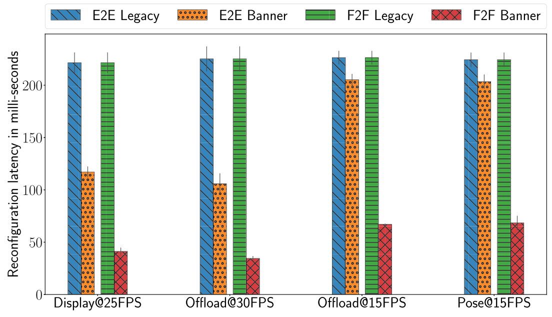

FIGURE 5. Banner reduces E2E resolution reconfiguration latency and eliminates F2F latency compared with legacy V4L2.

Results in Figure 5 show that Banner completely eliminates the F2F latency in all three use cases and could halve the E2E reconfiguration latency compared with the legacy V4L2 media framework. In particular, the E2E reconfiguration latency is reduced by 47%, 9%, 54%, and 10% and the F2F latency is reduced by 82%, 70%, 85%, and 70%, accordingly in display-only at 25 FPS, cloud-offloading at 15 FPS, cloud-offloading at 30 FPS, and pose estimation at 15 FPS.

Power Efficiency Improvement. Our measurements, shown in Table 1, indicate that Banner enables a substantial power efficiency improvement by allowing vision applications to reconfigure sensor resolution dynamically. First, by allowing the sensor resolution to be reconfigured from 1080p to 480p, the total system power consumption could be reduced by 62%, 60%, and 42% in display-only, cloud-offloading, and pose estimation accordingly. Second, comparing Banner with computational downsampling, physically reconfiguring the sensor resolution dynamically from 1080p to 480p with Banner consumes 43%, 49%, and 16% less total system power than 1080p↓ 480p in display-only, cloud-offloading, and pose estimation accordingly.

Banner maintains frame delivery while the application requests changes to the resolution. This enables vision applications the opportunity to change resolutions dynamically to adapt to environmental changes without creating visual gaps in the user experience. Paired with other energy-efficient system optimizations through resolution-based tradeoffs, the seamless reconfiguration of Banner overcomes an important barrier toward energy-efficient visual computing.

FUTURE WORK

We have thus far been motivated by the issue that using traditional imaging architectures at the

TABLE 1. Total system power consumption in MW is reduced by 62%, 60%, and 42% if sensor resolution could be reduced from 1080p to 480p (colored in blue), in each use case accordingly in legacy V4L2.

Use Case	Resolution@FPS	Legacy V4L2					Banner				
		SoC	GPU	DDR	CPU	**Total**	SoC	GPU	DDR	CPU	**Total**
Display-only	1080p@25	1149	910	1869	2460	**6388**	1149	918	1839	2190	6096
	720p@25	1073	611	1727	1076	4487	1073	559	1704	1100	4436
	480p@25	617	306	826	691	**2440**	669	306	860	681	**2516**
	1080p↓480p@25	1078	613	1690	1035	**4416**	N/A	N/A	N/A	N/A	N/A
Cloud-offloading	1080p@15	1073	536	1629	1967	**5205**	1146	544	1638	2027	5355
	720p@15	688	230	863	617	2398	700	230	856	630	2416
	480p@15	617	230	702	540	**2089**	630	230	687	554	**2101**
	1080p↓480p@15	1073	382	1606	1032	**4093**	N/A	N/A	N/A	N/A	N/A
Pose estimation	1080p@15	1281	1701	1940	2524	**7446**	1149	1448	1875	2527	6999
	720p@15	1230	1058	1814	1271	5373	1225	987	1795	1203	5210
	480p@15	1210	589	1635	928	**4362**	1150	540	1637	916	**4243**
	1080p↓480p@15	1227	850	1727	1260	**5064**	N/A	N/A	N/A	N/A	N/A

Physically reconfiguring sensor resolution to 480p with banner consumes 43%, 49%, and 16% less total system power than downsampling 1080p↓ 480p in legacy V4L2 (colored in orange), in each use case accordingly.

necessary high resolutions and frame rates leads to an overwhelming data rate from the sensor interfaces, to the memory, to the computational units. The strategies presented earlier allow systems to reconfigure resolution on a frame-by-frame basis.

Fine-Grained Multiresolution Visual Sensing. We observe that many visual computing tasks do not require high resolution across the entire frame, nor is it necessary at all times; for visual tracking, high resolution may only needed near certain visual features in the scene and on an occasional basis to maintain precision.

An alternative visual computing paradigm could leverage this observation by focusing high-resolution capture around regions of the image where there is predicted need. For example, visual features that are far from the camera or finely textured may require raised resolution in their neighboring area to capture sufficient visual information. Meanwhile, visual features closer to the camera can often suffice with lower resolutions.

Similarly, it would be beneficial to provision high temporal resolutions around regions of fast moving objects, especially to identify gestural interactions, whereas low temporal resolutions would suffice for unchanging static parts of a scene, or regions where motion adaptation could smooth visual movement. These patterns would necessitate new forms of data representations and control toward a nonuniform multiresolution visual sensing paradigm.

Successful implementation and usage of multiresolution visual sensing architecture will require cross-layer optimization across the mobile computer system, from sensor I/O hardware interface to OS control, to visual computing algorithm, to developer library.

Other Approaches to Efficient Visual Computing. Reconfigurable image sensing pipelines would pair well with several recent efforts to reduce the energy footprint of visual computing. This includes integrating with works that 1) offload workloads to the cloud or a nearby cloudlet/edge device, 2) reuse computations to reduce overhead, and 3) leverage hardware acceleration for visual computing.

Offloading workloads from a mobile device to a networked device has become a popular paradigm to reduce computational burden on resource-constrained devices. For visual workloads, this presents challenges, as the high data rate of visual frame streams creates networking overhead. Glimpse[12] reduces this overhead by offloading only a few key frames to the server for expensive object detection and run cheaper tracking for other frames on the mobile device. For high trackability, Glimpse maintains an active cache of frames on the mobile device and estimates object location based on hints arriving from the server. On the other hand, Neurosurgeon[13] reduces the networking overhead by dynamically partitioning an inference workload, sending the reduced output from a CNN layer as opposed to offloading the entire frame. In addition to workload partitioning, Odessa[14] also adapts the degree of parallelism for the workload, leveraging processor frequencies and recent histories of network requirements to determine an optimal partitioning to maximize performance, especially for interactive perceptual applications. Multiresolution visual sensing pipeline can integrate with these systems by capturing visual content at appropriate resolutions before the network-based partitioning.

Toward energy-efficiency, some architectural designs use computation results from previous frames to estimate the results for the next frame. Instead of executing CNN inference on every single frame, Euphrates[5] relaxes computation by reusing motion vector information, which is already precomputed in the denoising stage of the ISP module, to periodically estimate the CNN inference results for a set of frames. Along similar lines, EVA[26] exploits temporal redundancy across frames to approximate CNN results rather than compute them on each frame. These computational reduction efforts would pair well with the sensing and memory efficiency of multiresolution visual computing by directing the capture of redundant frames at lower resolutions or skipped altogether.

Domain specific accelerators have also appeared, using novel systolic array based architectures for efficient and performant vision on mobile systems. Apple's NPU integrated on their latest iPhone is capable of performing expensive vision tasks, such as FaceID among others in an efficient manner. Google's Tiny TPU on their Pixel phones can perform real-time object detection tasks. Additionally, image sensor companies recently integrate VPU inside the sensor itself to significantly reduce the memory traffic. Sony's recent 3-D stacked imager integrates an AI processor in one of the layers inside the camera enabling a wide variety of in-camera processing tasks, such as traffic detection. These works can integrate with multiresolution visual computing toward further energy-efficiency by selectively reducing the frame data processed by the VPU, whether ON or OFFthe sensor chip. The low-power VPU computations can also lead to efficient decision-making for real-time sensor configuration.

Study Multiresolution Effects on User Experience. On mobile devices and headsets, high quality demands reasonably high spatial resolution for the visual display for annotations, and precision placement of immersive augmented reality content. It is imperative that the quality of the displayed frames remains high, with high resolutions and low latencies. We foresee that our work on multiresolution sensing could enable a stream of high quality frames to be delivered to the user while constructing overlays construed from a set of lower resolutions frames captured at faster frame rates for continuity and/or a set of higher resolution frames captured occasionally for precision of placement. Such investigations could entail a deeper analysis of the visual needs of users as resolution changes, depending on various contextual situations, e.g., lighting conditions, motion patterns, and visual overlay richness. Such findings would drive the adaptive system to capture frames that suit the users' visual expectations while also capturing frames that are sufficient for visual computing tasks. Operating at the lower bound of these requirements, such an adaptive system would yield energy-efficient augmented reality system with high usability.

CONCLUSION

In this article, we have discussed the need for system support for visual sensing at adaptive spatiotemporal resolutions for energy-efficient visual computing. We covered two recent works that in tandem provide such support. The first work uses driver-based sensor control to manage the activity of the image sensor, enabling low-power behavior at low spatiotemporal resolutions. The second work, Banner, provided techniques to redesign the camera media framework of mobile OSs to facilitate adaptive resolution changes with no frame drop or other interruption to the user experience. We also discussed the potential for adaptive-resolution visual computing at a fine granularity, considering nonuniform spatiotemporal resolutions across the frame. This requires substantial investigation across the system stack, including in the sensor/interface architecture, pixel data representation, programming language visual data specification, and driver/media framework. Such system support would unlock the potential for adaptive multiresolution visual computing to satisfy visual computing needs with high precision where needed and low power where possible.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under Grants 1942844 and 1909663, in part by a Samsung Research Grant, and in part by the Google Faculty Research Award.

REFERENCES

1. R. LiKamWa, Z. Wang, A. Carroll, F. X. Lin, and L. Zhong, "Draining our glass: An energy and heat characterization of Google glass," in *Proc. 5th Asia-Pacific Workshop Syst.*, 2014, pp. 1–7.
2. J. Hu, J. Yang, V. Delhivala, and R. LiKamWa, "Characterizing the reconfiguration latency of image sensor resolution on android devices," in *Proc. 19th Int. Workshop Mobile Comput. Syst. Appl.*, 2018, pp. 81–86.
3. J. Hu, A. Shearer, S. Rajagopalan, and R. LiKamWa, "Banner: An image sensor reconfiguration framework for seamless resolution-based tradeoffs," in *Proc. 17th Annu. Int. Conf. Mobile Syst., Appl., Serv.*, 2019, pp. 705–706.
4. O. Iqbal *et al.*, "Design and FPGA implementation of an adaptive video subsampling algorithm for energy-efficient single object tracking," in *Proc. IEEE Int. Conf. Image Process.*, 2020, pp. 3065–3069.

5. Y. Zhu, A. Samajdar, M. Mattina, and P. Whatmough, "Euphrates: Algorithm-SoC co-design for low-power mobile continuous vision," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Architecture*, 2018, pp. 547–560.
6. M. Buckler, P. Bedoukian, S. Jayasuriya, and A. Sampson, "Eva^2: Exploiting temporal redundancy in live computer vision," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Architecture*, 2018, pp. 533–546.
7. S. Naderiparizi, P. Zhang, M. Philipose, B. Priyantha, J. Liu, and D. Ganesan, "Glimpse: A programmable early-discard camera architecture for continuous mobile vision," in *Proc. 15th Annu. Int. Conf. Mobile Syst., Appl., and Serv.*, 2017, pp. 292–305.
8. H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.
9. M. Buckler, S. Jayasuriya, and A. Sampson, "Reconfiguring the imaging pipeline for computer vision," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 975–984.
10. R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl, "Energy characterization and optimization of image sensing toward continuous mobile vision," in *Proc. 11th Annu. Int. Conf. Mobile Syst., Appl., Services*, 2013, pp. 69–82.
11. A. Adams *et al.*, "The Frankencamera: An experimental platform for computational photography," in *Proc. ACM SIGGRAPH Papers*, 2010, pp. 1–12.
12. T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "GLIMPSE: Continuous, real-time object recognition on mobile devices," in *Proc. 13th Assoc. Comput. Machinery Conf. Embedded Networked Sensor Syst.*, 2015, pp. 155–168 .
13. Y. Kang *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *Proc. 22nd Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2017, pp. 615–629.
14. M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: Enabling interactive perception applications on mobile devices," in *Proc. 9th Int. Conf. Mobile Syst., Appl., Services*, 2011, pp. 43–56.

ROBERT LIKAMWA is currently an Assistant Professor with the School of Arts, Media, and Engineering and the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA. His research interests include augmented reality, virtual reality, and visual computing systems, frameworks, and applications. He is the corresponding author of this article. Contact him at likamwa@asu.edu.

JINHAN HU is currently working toward the doctoral degree with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA. His research interests include designing novel systems for supporting emerging continuous mobile vision applications. Contact him at jinhanhu@asu.edu.

VENKATESH KODUKULA is currently working toward the Ph.D. degree in computer engineering with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA. His research interests include hardware–software co-design for efficient computer vision. Contact him at vkoduku1@asu.edu.

YIFEI LIU is currently a Graduate Student with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA. His research interest includes accelerator-rich heterogeneous architecture. Contact him at yliu740@asu.edu.