Mobile Vision: How we must augment APIs to enable a new reality

Alon Or-bach*
Samsung R&D Institute UK

1. Introduction

Since the advent of the smartphone, mobile devices have incorporated increasingly capable cameras, and in recent years a multitude of sensors to provide accurate movement and positioning information. Using feeds from the camera and other sensors for applications such as computer vision (CV) and augmented reality (AR) is strongly compelling. However, there is still work to be done to achieve a smoother user experience and avoid significant power drain.

2. Current mobile API capabilities and limitations

Mobile devices are increasingly incorporating computer vision functionality. Features such as face detection are now common, but tend to require dedicated hardware support from the camera's image signal processor (ISP) to achieve good power efficiency, but lack flexibility. Many third-party AR and CV applications therefore still process on the CPU. With the high resolutions of today's mobile displays and cameras, this is becoming more costly – with copies, cache flushes and color space conversions becoming more expensive [Lanman D. et al. 2014].

The answer to our quest for responsive yet power efficient CV and AR applications lies with the increasing capabilities of other units within mobile SoCs – the growing compute capabilities of the GPUs and dedicated programmable hardware for vision. However, we are still limited by multiple issues yet to be tackled which this talk will explore. An overview of current mobile capabilities will be demonstrated to highlight what performance and power usage are presently achievable on high-end smartphones GPUs.

3. Obstacles – and how to navigate around them

There are several key concerns that developers should consider. For each issue, we will explain the constraints on current devices, how to tackle it and how this could be improved in the future.

3.1 Don't forget, memory matters

Achieving the best performance across different GPU architectures still requires significant customization on host side, as well as customizing kernels running on the device. For example, targeting a GPU with OpenCL, you can expect significant variation depending not only on which format is used, but also what memory object is chosen, and how the memory is allocated. Tips on good practice for application developers to manage these stumbling blocks will be presented, including examples in the Khronos standard for compute, OpenCL.

*e-mail: alon.orbach@samsung.com

SIGGRAPH 2015 Talks, August 09 – 13, 2015, Los Angeles, CA.
ACM 978-1-4503-3636-9/15/08.
http://dx.doi.org/10.1145/2775280.2792581

Figure 1. *We foresee improved responsiveness from augmented reality applications in the coming years*

3.2 Show me your layouts

The GPU's internal data format is often intentionally kept opaque from the developer, permitting optimizations such as proprietary layouts and tiling. However, algorithms in CV/AR tend to require access to unprocessed sensor data. This poses a challenge for applications that require efficient memory handover between domains. The often-conflicting demands of camera ISPs, Digital Signal Processors (DSPs), GPUs and CPUs result in a plethora of formats and layouts, each optimized for a particular hardware architecture of a specific component. In most cases, memory allocation APIs have insufficient information regarding additional usage beyond their remit. Android's Gralloc is a notable exception to this, but this is at the expense of precluding the use of proprietary layouts, thereby potentially reducing performance.

3.3 It's all in the timing

Optimizing the code for each target device is a good start, but one must also consider the bigger picture. Latency must be minimized to get the most compelling experience, and any unnecessary interaction with the user process avoided. We will explain how server-side synchronization can be used to reduce this with current mobile devices, and how future devices can improve this further by the adoption of designs that drive closer integration of different processing units, such as those developed by the Heterogeneous System Architecture (HSA) Foundation.

4. Conclusions

Mobile and wearable devices are on the verge of having the performance capability to enable responsive and power-efficient CV and AR applications. As the demand for this capability grows, hardware designs and APIs will be driven towards addressing the integration challenges outlined. We close by outlining how future once SoCs are capable, more explicit APIs will allow for more efficient streaming, and in turn more compelling applications.

References

LANMAN D., FUCHS H., MINE M., MCDOWALL I., AND ABRASH M. 2014. Put on your 3D glasses now: the past, present, and future of virtual and augmented reality. ACM SIGGRAPH 2014 Courses, Article 12