PART 11 Runtime Support for Interactive Perception Applications on Mobile Devices

• Ramesh Govindan, Faculty, PI*
• Lily Mummert, Staff
• Padmanabhan Pillai, Staff
• Moo-Ryong Ra, Graduate Student*
• Anmol Sheth, Staff
• David Wetherall, Faculty

* Primary Contact

Overview

Resource constrained mobile devices need to leverage computation on nearby servers to run responsive applications that recognize objects, people, or gestures from real-time video. However, it is unclear what computation to offload, and how to structure parallelism across the mobile device and server. To answer these questions, we develop and evaluate three interactive perceptual applications. We find that offloading and parallelism choices should be dynamic, even for a given application, as performance depends on scene complexity as well as environmental factors such as the network and device capabilities. To this end we develop Odessa, a novel, lightweight runtime that automatically and adaptively makes offloading and parallelism decisions for mobile interactive perception applications. Our evaluation shows that the incremental greedy strategy of Odessa converges to an operating point that is close to an ideal offline partitioning. It provides more than a 3x improvement in application performance over partitioning suggested by domain experts. Odessa works well across a variety of execution environments, and is agile to changes in the network, device and application inputs.

Approach

As the processing, communication and sensing capabilities of mobile devices increase, a new class of mobile interactive perception applications is emerging. These applications use cameras and other high-data rate sensors to perform perception tasks, like face or object recognition, and enable natural human machine interfaces and interactive augmented reality experiences on mobile devices. For example, face recognition could be used by a social networking application that recognizes people as the user sweeps the camera across a crowded room; a gesture recognition based natural user interface could be used to control a media application running on the mobile device; and object and pose recognition can be used by an augmented reality shopping application that overlays information about an object in the user’s hand.

Interactive perception applications have a unique set of requirements that stress the capabilities of mobile devices. First, interactive applications require crisp response. For example, to feel responsive, an augmented reality application would need to display results well under a second. Second, these applications require continuous processing of high data rate sensors such as cameras to maintain accuracy. For example, a low frame rate may miss intermediate object poses or human gestures. Third, the computer vision and machine learning algorithms used to process this data are compute intensive. For example, in one of the applications we study, extracting features from an image can take 7 seconds on a netbook. Finally, the performance of these algorithms is highly variable and depends on the content of the data, which can vary greatly.

These requirements cannot be satisfied on today’s mobile devices alone. Moreover, even though the computing and communication capabilities of these platforms are improving, interactive perception applications will continue to push platform limits as new, more accurate but more compute intensive algorithms are developed. However, two techniques can help make mobile interactive perception a reality: offloading one or more of the compute-intensive application components to an Internet-connected server, and using parallelism on multi-core systems to improve responsiveness and accuracy of the applications. Fortunately, interactive perception applications can be structured
for offloading, and provide considerable opportunities to exploit parallel processing. In this work, we develop a runtime called Odessa that automatically and adaptively determines how best to use these techniques.

System Description
Motivated by the lessons from the earlier measurement study, Odessa adaptively exploits pipelining, data-parallelism and offloading to improve performance and accuracy of these applications. The Odessa runtime runs on the mobile device; this enables Odessa to transparently improve application performance across different mobile platforms when the mobile is disconnected from the server. The design the Odessa runtime has three goals, in order of decreasing importance:

- It must simultaneously achieve low makespan and high throughput in order to meet the needs of mobile interactive-perception applications.
- It must react quickly to changes in input complexity, device capability, or network conditions. This goal ensures that transient changes in makespan or throughput are minimized or avoided.
- It must have low computation and communication overhead.

Prior approaches for offloading frame the problem using a discrete or graph optimization formulation. For this approach to be effective, accurate estimates of stage execution time are required on both the mobile device and the server, which are often obtained by offline profiling. However, the results in our measurement study show that the execution time can vary significantly and cannot easily be modeled offline. Odessa uses a greedy algorithm that periodically acquires information from a lightweight application profiler to estimate the bottleneck in the current configuration. Then, its decision engine uses simple predictors based on nominal processor frequencies, and a recent history of network measurements, to estimate whether offloading or increasing the level of parallelism of the bottleneck stage would improve performance. Surprisingly, this greedy and incremental approach works very well to improve makespan and throughput, and incurs negligible overhead. Rarely, Odessa’s decision may need to be reversed because its estimators may be off, but it has a built-in self-correcting mechanism to maintain stability.

Accomplishments
We have explored the design of a runtime, called Odessa, that enables interactive perception applications on mobile devices. The unique characteristics of the applications drive many of the design decisions in Odessa, whose lightweight online profiler and simple execution time predictors help make robust and efficient offloading and parallelization decisions. Our evaluation of Odessa shows that it can provide more than 3x improvement in performance compared to application configurations by domain experts as in Figure 2. Additionally, Odessa can adapt quickly to changes in scene complexity, compute resource availability, and network bandwidth.

Future Directions
Much work remains, including exploring the performance of Odessa under a broader range of applications, extending it to take advantage of the public cloud, and exploring easy deployability.