Lecture 17:

Image Processing Architectures (and their future requirements)

Visual Computing Systems
CMU 15-869, Fall 2013
Smart phone processing resources
Qualcomm Snapdragon

Krait 400 CPU
- features 28nm HPM process technology
- superior 2GHz+ performance

Adreno 330 for advanced graphics

Hexagon QDSP6 for ultra low power applications and custom programmability

Integrated 802.11ac, USB 3.0 and BT 4.0 offers broad array of high speed connectivity

Multimedia
- Audio, Video and Gesture

Camera
- 21MP with dual ISP
- Support for up to 2560x2048 display
- Miracast 1080p
- HD support

Navigation
- IZat GNSS with support for three GPS constellations

Connectivity
- 4G LTE, WiFi, USB, BT and FM

Image credit: Qualcomm
Apple A7 (iPhone 5s)

- Chipworks estimates:
  - CPU + cache: 17% area
  - GPU: 22% area
  - (Imagination PowerVR Series 6 (G6430))
  - Big SRAM block above GPU: 3-4 MB?

Discussion

- Traditional rule of thumb in system design is to design simple, general-purpose components. This is not the case with mobile processing systems (perf/watt)

- Needs of high bandwidth sensing and media processing are a big part of these designs [image/video/audio processing, 2D/3D graphics]
  - User interfaces are visually rich
  - Games
  - Speech recognition
  - Photography/video
    - Acquire signal, compute images (not directly measuring an image)
    - More processing --> smarter sensing
    - More processing --> more flexibility?

- Questions for architects:
  - Re-homogenize, or become increasingly heterogeneous?
  - How does an application developer think about these systems?
Frankencamera
Frankencamera context

- Cameras are cheap and ubiquitous
- Significant processing capability on cameras
- Many techniques for combining multiple photos to overcome deficiencies in traditional camera systems

- But... ability to implement techniques on cameras was limited
  - Cameras not programmable by general public
  - Where some programmability did exist, interface too basic
    (end result was that latency between two photos was high, mitigating utility of multi-shot techniques)
Example: high dynamic range images

Source photographs: varying exposure

Tone mapped HDR image

Credit: Debevec and Malik
More multi-shot photography examples

“Lucky” imaging

Take several photos in rapid succession: likely to find one without camera shake

Flash-no-flash photography [Eisemann and Durand]
(use flash image for sharp, colored image, infer actual room lighting from no-flash image)
Frankencamera goals

1. Create open, handheld camera platform for researchers

2. Define system architecture for computational photography applications
   - Motivated by impact of OpenGL on graphics application and graphics hardware development (portable apps despite highly optimized GPU implementations)
   - Motivated by proliferation of smart-phone apps

Note: Apple was not involved in Frankencamera's industrial design. ;-)

F2 Reference Implementation

Nokia N900 Smartphone Implementation
F-cam components

**Sensor** is really just a special case of a device
Shot

- **A shot is a command**
  - Actually it’s a set of commands
  - Encapsulates both “set state” and “perform action(s)” commands

- **Defines state (configuration) for:**
  - Sensor
  - Image processor
  - Relevant devices

- **Defines a timeline of actions**
  - Exactly one sensor action: expose
  - Optional actions for devices
  - Note: timeline extends beyond length of exposure ("frame time")
Shot

- **Interesting analogy:**
  - An F-cam shot is very similar to an OpenGL display list
  - A shot is really a series of commands (both action commands and state manipulation commands)
    - State manipulation commands specify the entire state of the system
    - Defines precise timing of the commands (no OpenGL analogy for this)
Frame

- A frame describes the **result** of a shot

- A frame contains:
  - Reference to corresponding image buffer
  - Statistics for image (computed by image processor)
  - Shot configuration data (what was specified by app)
  - Actual configuration data (configuration actually used when acquiring image)
Question

- What problem in conventional camera interface designs does F-cam address: throughput or latency?
Aside: latency in camera systems

- Often in this class our focus is on achieving high throughput
  - Triangles per clock
  - Pixel per clock
- But low latency is critical in many visual computing domains
  - Camera metering, focus
  - Optical flow, tracking

Example: CMU smart headlight project
[Charette et al. 2012]
**Problem:** current sensor, projector interfaces operate at frame (not pixel or pixel row) granularity.

<table>
<thead>
<tr>
<th>Frame #</th>
<th>Pipeline Stages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Stage 1</td>
</tr>
<tr>
<td>Frame 1</td>
<td>Capture</td>
</tr>
<tr>
<td>Frame 2</td>
<td>Capture</td>
</tr>
<tr>
<td>Frame 3</td>
<td>Capture</td>
</tr>
<tr>
<td>Frame 4</td>
<td>Capture</td>
</tr>
</tbody>
</table>

| Time (ms) | 5 | 9 | 14 | 18 | 27 | 36 |
F-cam “streaming” mode

- System repeats shot (or series of shots) in infinite loop
- Stops only when application says so
- Intended for “live view” (digital viewfinder) or metering mode
F-cam as an architecture

Application Commands ("Shots")

- **Cmd Processor**
  - Stream Cmd Buffer
  - Device (Flash)
  - Device (Lens)
  - Sensor
    - RAW Data
  - Image Processing
    - Image Data
      - Frames
    - Completed Frames
- Memory
  - Image Buffers
  - Completed Frames
  - Event Queue

RAW Data

Image Buffers
F-cam scope

- F-cam provides a set of abstractions that allow for manipulating configurable camera components
  - Timeline based specification of actions
  - Feed-forward: no feedback loops

- F-cam architecture performs image processing, but...
  - This functionality is not programmable
  - F-cam does not provide an image processing language
  - Other than work performed by the image processing stage, F-cam applications do all their own image processing (e.g., on smartphone/camera's CPU or GPU resources)
NVIDIA Chimera

- Software framework for writing computational photography pipelines
- Idea: application provides kernel functions for CPU, GPU (or specifies how to configure/use the ISP) and describes how to connect up the kernels
F-cam extension: programmable image processing

Application Commands ("Shots")

- Device (Flash)
- Device (Lens)
- Sensor

Image Processing

- RAW Data
- Image Data
- Frames

Cmd Processor

Stream Cmd Buffer

Memory

- Image Buffers
- Completed Frames
- Event Queue

Completed Frames

Event Queue
Class design challenge 1

- If there was a programmable image processor, application would probably seek to use it for more than just on data coming off sensor

- E.g., HDR imaging app
Class design challenge 2

- Question: How does auto-focus work in F-cam?

- How might we extend the F-cam architecture to model a separate autofocus/metering sensor?
Class design challenge 3

- Should we add a face detection unit to the architecture?
- How might we abstract a face detection unit?
- Or a feature extractor?
Architecture is hard.
Discussion

- Is there a need for a camera “App Store”?