Lecture 20: The Frankencamera
A Programmable Camera Architecture

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

- Cheap and ubiquitous cameras
- Significant processing capability on cameras
- Lot’s of techniques on how to combine 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 a bunch of 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-hone apps

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)”

- 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
  - It 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)
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)
“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

Device (Flash)

Device (Lens)

Sensor

Image Processing

RAW Data

Image Data

Frames

Completed Frames

Event Queue

Image Buffers

...
Code examples
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 (like graphics pipeline)

- 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 image processing stage, F-cam applications do all their own image processing (e.g., on camera’s CPU)
F-cam extension: programmable image processing

Application Commands (“Shots”) → Cmd Processor

Device (Flash) → Sensor

Device (Lens) → Sensor

Sensor → Image Processing

Image Processing → Image Buffers

Image Buffers → Completed Frames

Completed Frames → Event Queue

RAW Data → Stream Cmd Buffer

Frames → Stream Cmd Buffer

Image Data → Stream Cmd Buffer

...
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 abstract a separate autofocus/metering sensor?
Class design challenge 3

- Should we add a face detection unit?
- How might we abstract a face detection unit?
- Or a feature extractor?
Architecture is hard.
Class discussion

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