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

Lecture 20:
The Frankencamera
A Programmable Camera Architecture

[Adams et al. 2010]

Kayvon Fatahalian
CMU 15-869: Graphics and Imaging Architectures (Fall 2011)

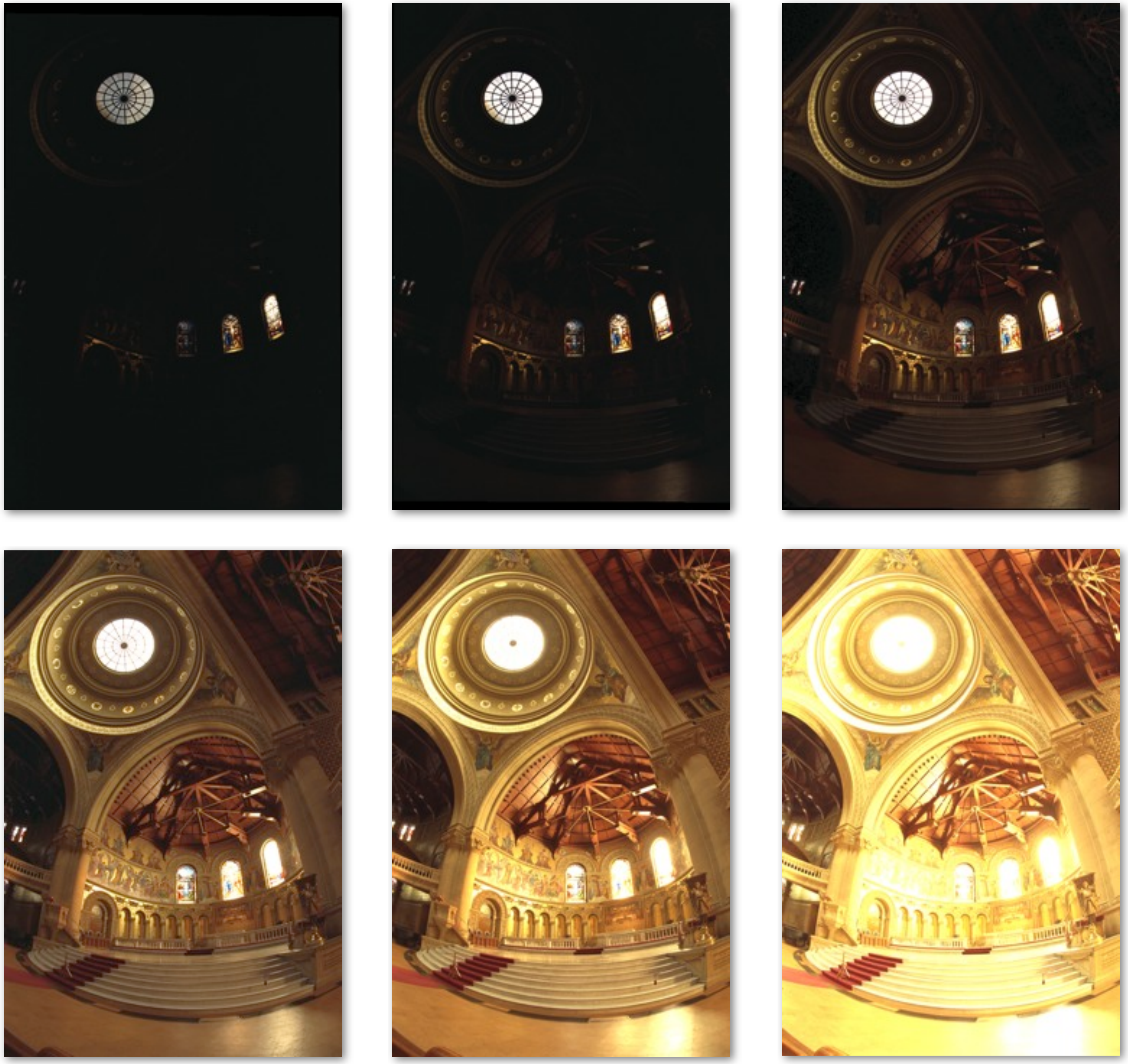
Context

- 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

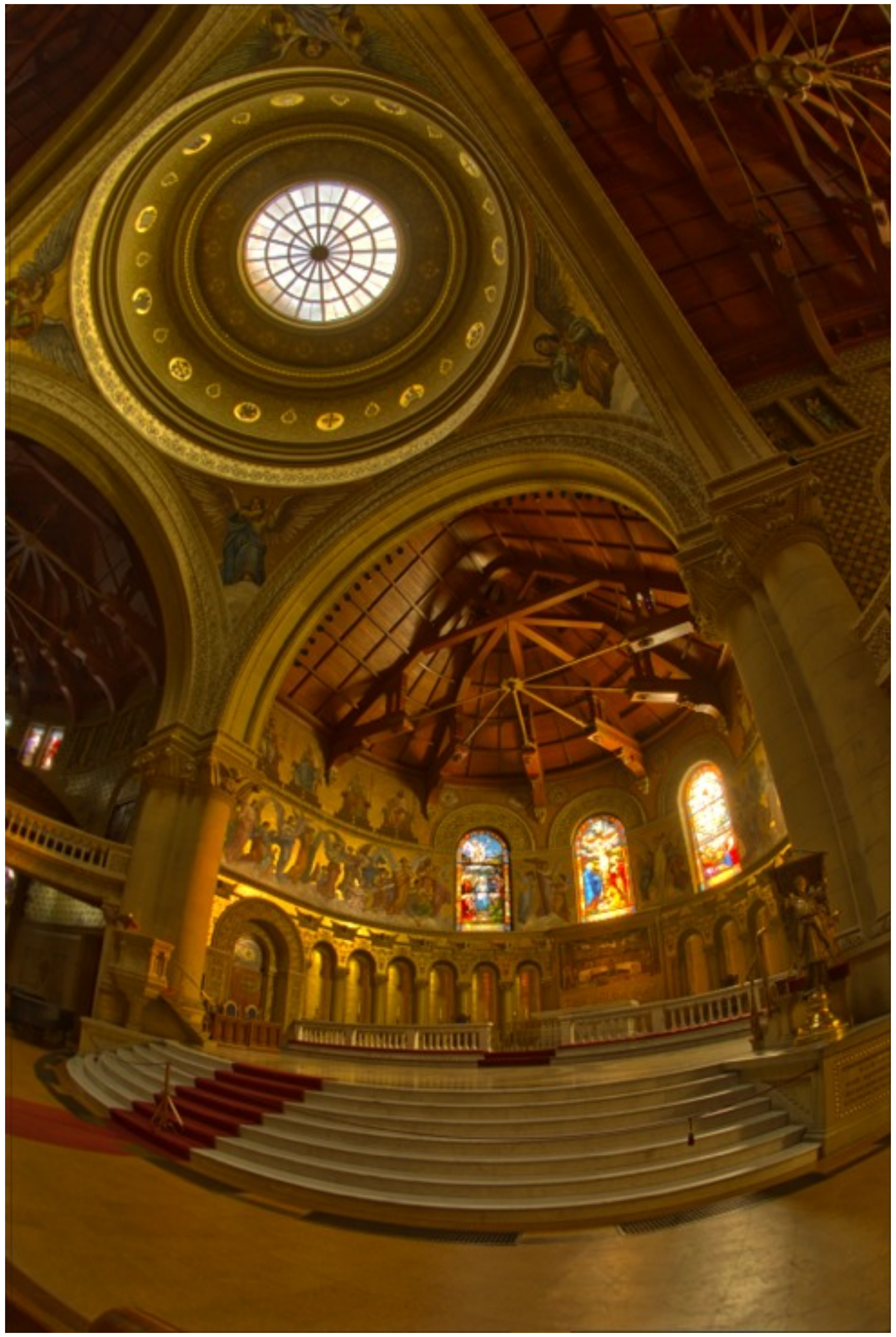
- But... ability to implement techniques on cameras was limited
  - Cameras not programmable by general public
  - Where some programmability did exist, interface too basic

    (end result was that latency between two photos was high, mitigating utility of multi-shot techniques)

Example: high dynamic range images



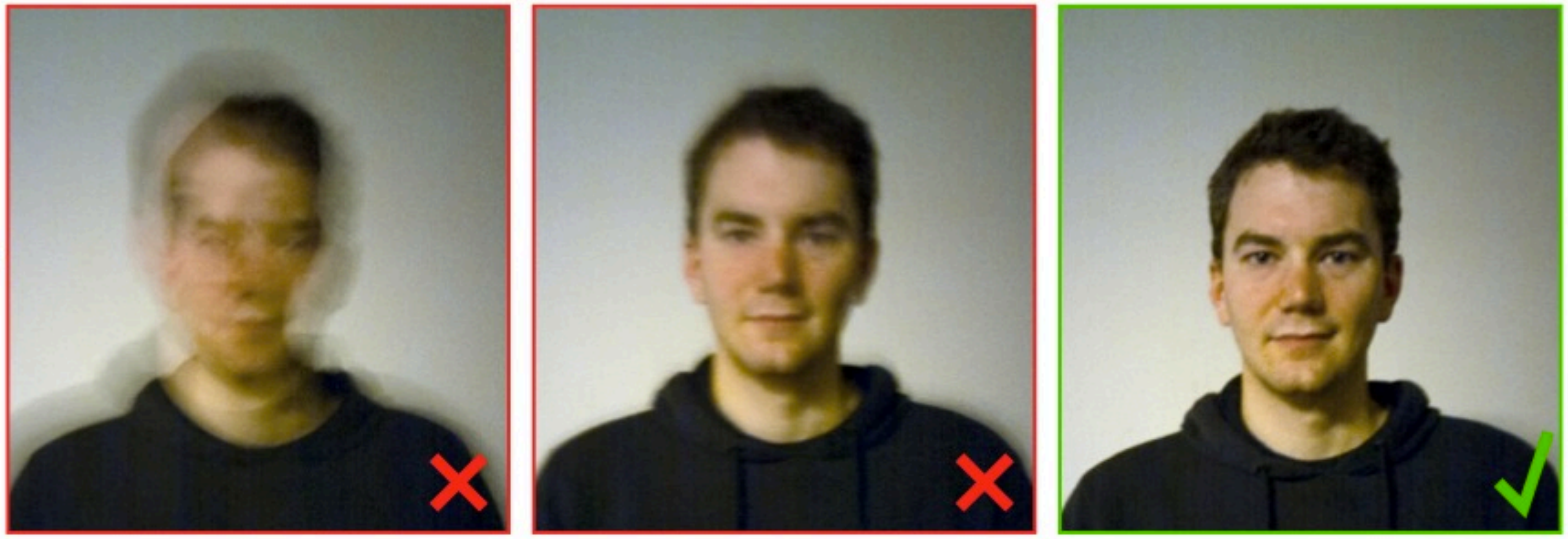
Source photographs: varying exposure



Tone mapped HDR image

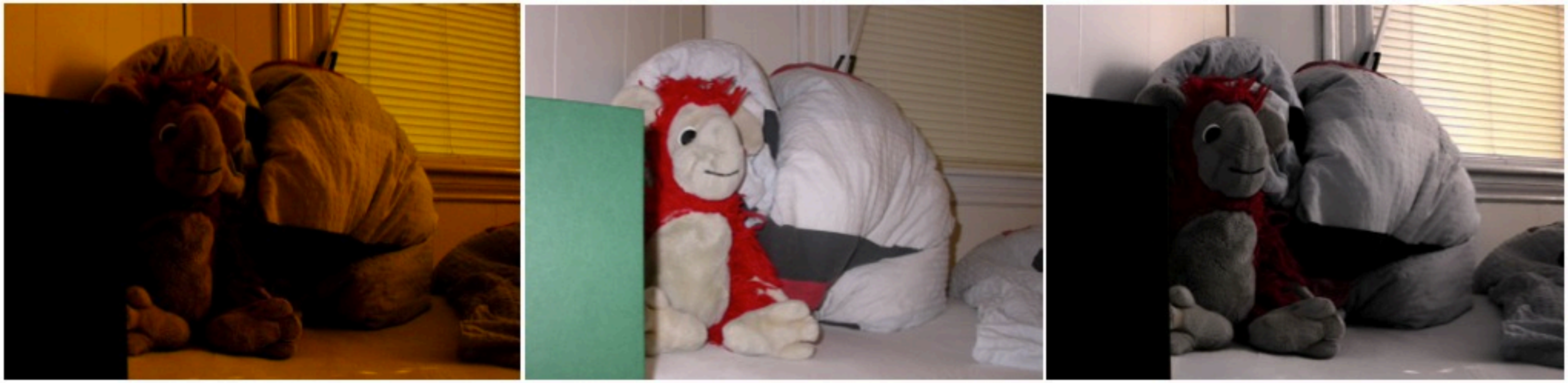
Credit: Debevec and Malik

More multi-shot photography examples



"Lucky" Imaging

Take a bunch of photos in rapid succession: likely to find one without camera shake



no-flash

flash

result

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

Frankencamera goals

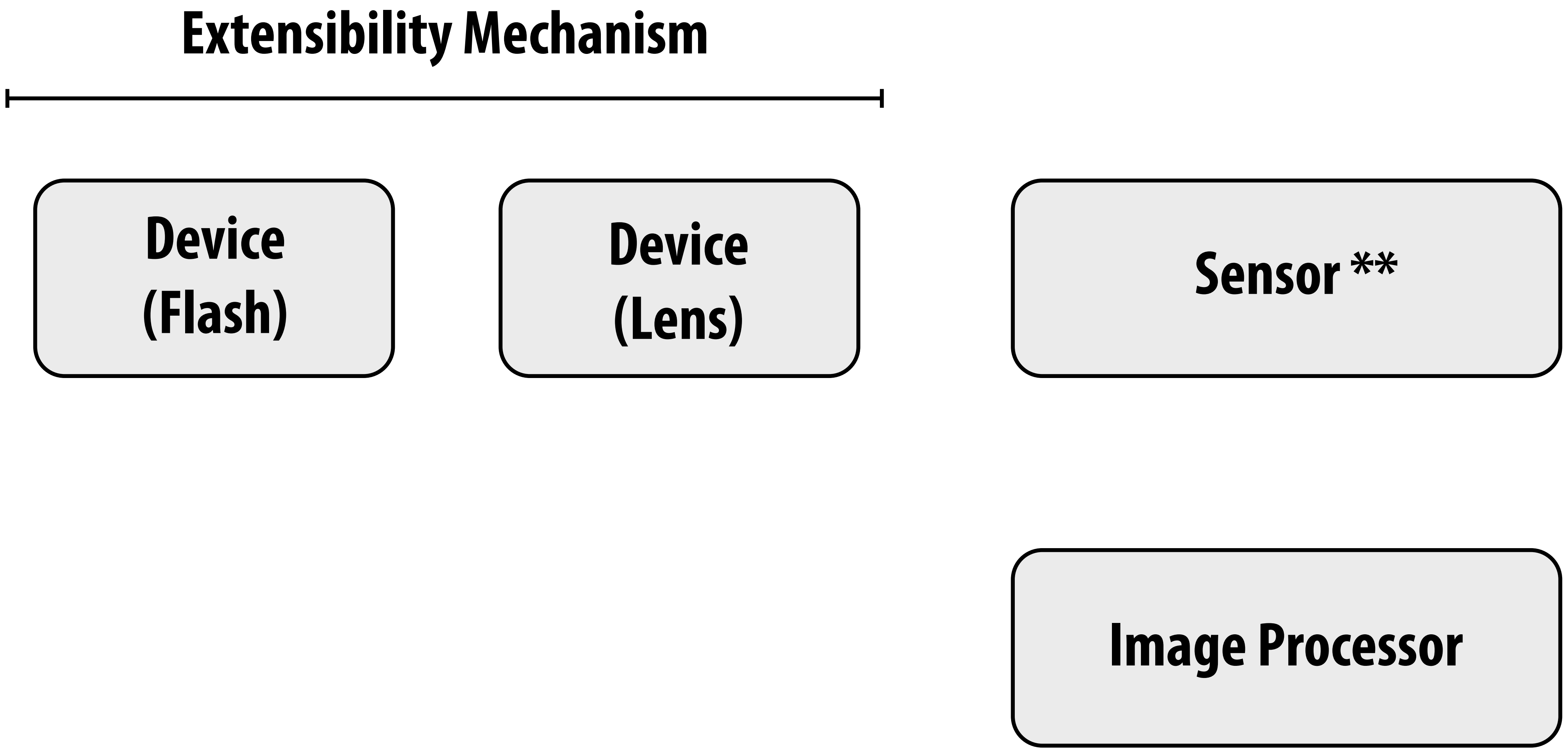
[Adams et al. 2010]

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 <u>entire state</u> of the system
    - Defines precise timing of the commands (no OpenGL analogy)

Frame

- **A frame describes the <u>result</u> of a shot**

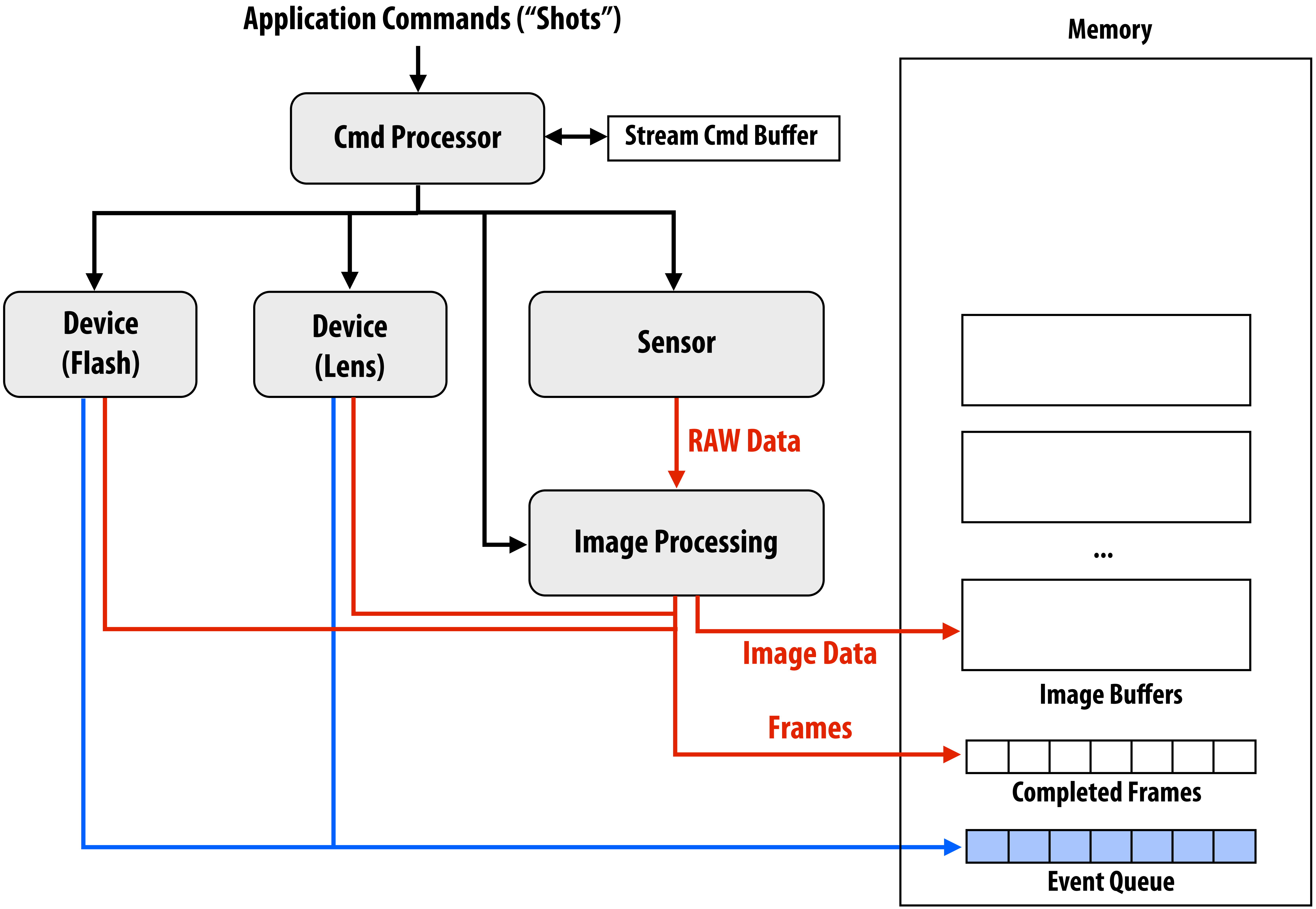
- **A frame contains:**
  - Reference to corresponding image buffer
  - Statistics for image (computed by image processor)
  - Shot configuration data (what was specified by app)
  - Actual configuration data (configuration actually used when acquiring image)

“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



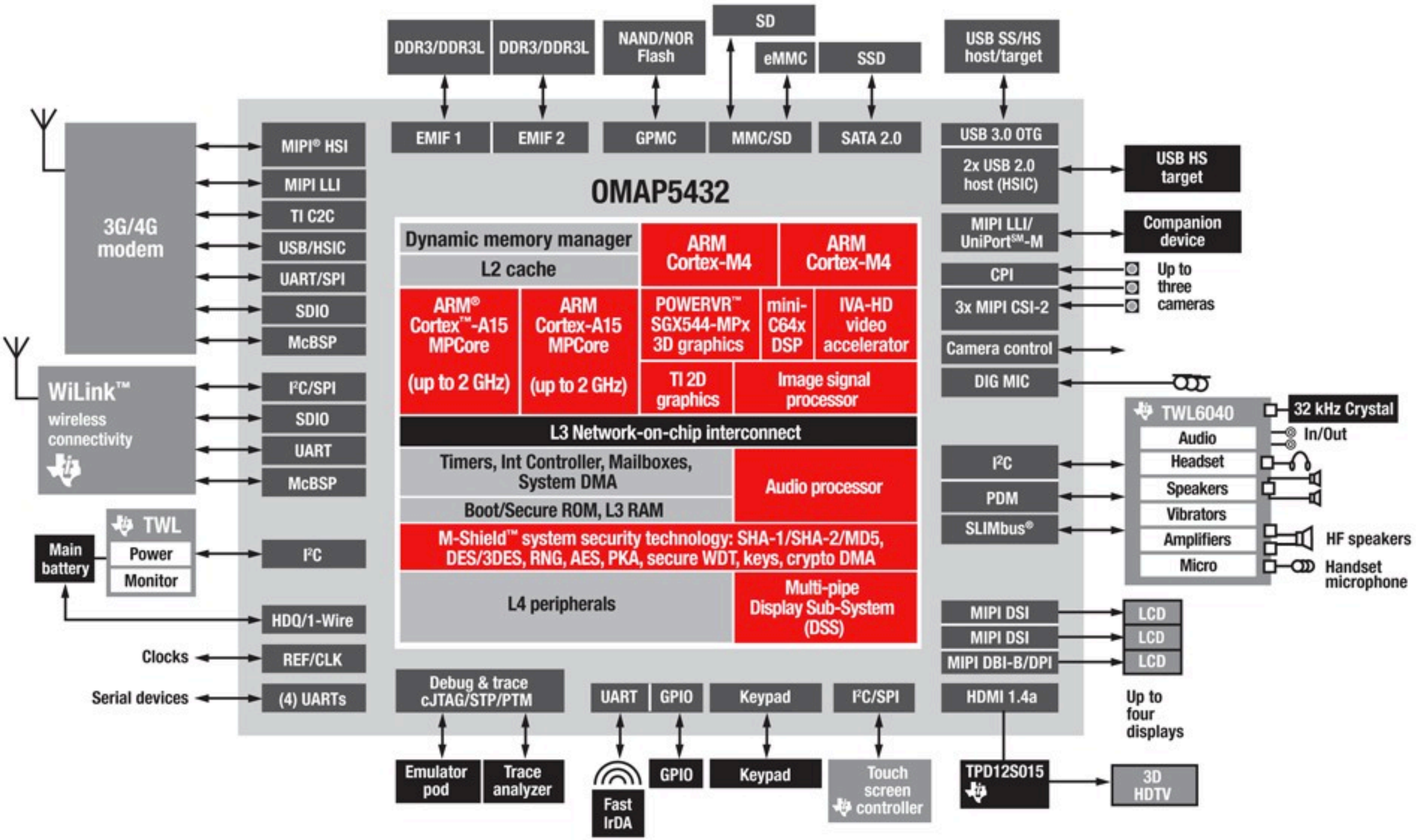
Code examples

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)

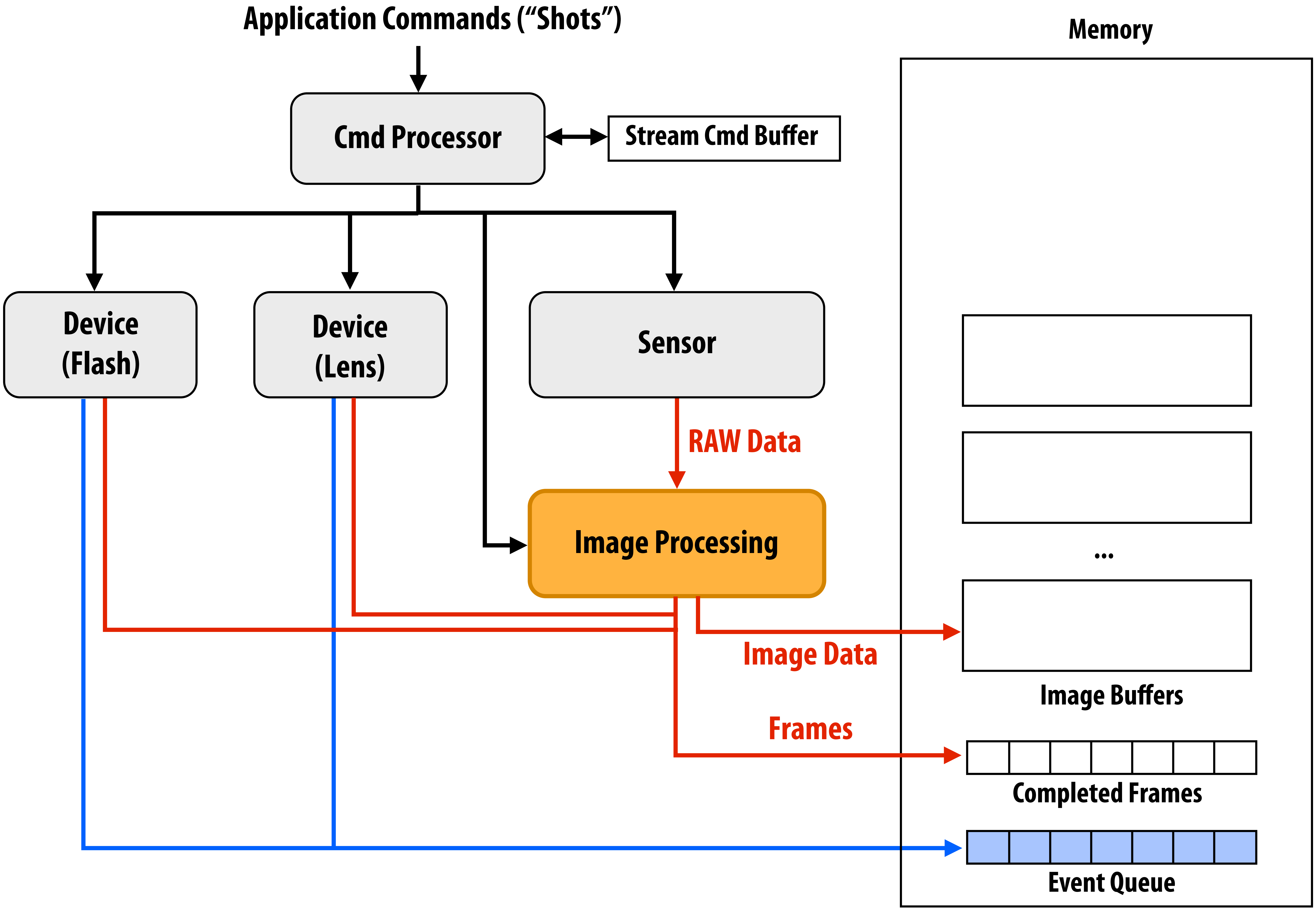
Texas Instruments OMAP 5



DDR3/DDR3L
DDR3/DDR3L
NAND/NOR Flash
SD
eMMC
SSD
USB SS/HS host/target
EMIF 1
EMIF 2
GPMC
MMC/SD
SATA 2.0
USB 3.0 OTG
2x USB 2.0 host (HSIC)
USB HS target
OMAP5432
MIPI LLI/ UniPort^{SM}-M
Companion device
CPI
3x MIPI CSI-2
Up to three cameras
Camera control
DIG MIC
3G/4G modem
MIPI® HSI
MIPI LLI
TI C2C
USB/HSIC
UART/SPI
SDIO
McBSP
WiLink™ wireless connectivity
I^2C/SPI
SDIO
UART
McBSP
TWL
Power
Monitor
Main battery
I^2C
HDQ/1-Wire
Clocks
REF/CLK
Serial devices
(4) UARTs
Dynamic memory manager
L2 cache
ARM Cortex-M4
ARM Cortex-M4
ARM® Cortex™-A15 MPCore (up to 2 GHz)
ARM Cortex-A15 MPCore (up to 2 GHz)
POWERVR™ SGX544-MPx 3D graphics
mini-C64x DSP
IVA-HD video accelerator
TI 2D graphics
Image signal processor
L3 Network-on-chip interconnect
Timers, Int Controller, Mailboxes, System DMA
Boot/Secure ROM, L3 RAM
Audio processor
M-Shield™ system security technology: SHA-1/SHA-2/MD5, DES/3DES, RNG, AES, PKA, secure WDT, keys, crypto DMA
L4 peripherals
Multi-pipe Display Sub-System (DSS)
TWL6040
32 kHz Crystal
Audio
In/Out
Headset
Speakers
Vibrators
Amplifiers
HF speakers
Micro
Handset microphone
I^2C
PDM
SLIMbus®
MIPI DSI
MIPI DSI
MIPI DBI-B/DPI
LCD
LCD
LCD
Up to four displays
HDMI 1.4a
TPD12S015
3D HDTV
Debug & trace cJTAG/STP/PTM
Emulator pod
Trace analyzer
UART
GPIO
Fast IrDA
GPIO
Keypad
Keypad
I^2C/SPI
Touch screen controller

Image credit: TI

F-cam extension: programmable image processing



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"?**