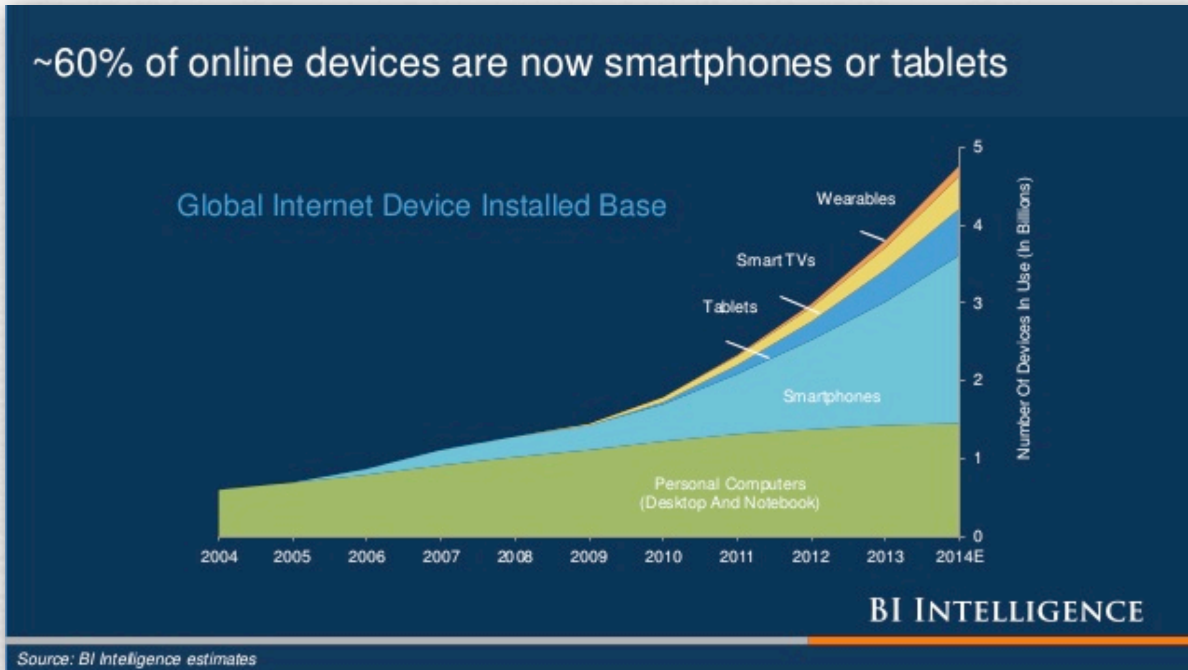
What "mobile-first" means for the future of computer science

UNC Computer Science
50th Anniversary Symposium
May 2, 2015

Marc Levoy
Engineering Manager
GoogleX

Professor, Emeritus
Computer Science Department
Stanford University

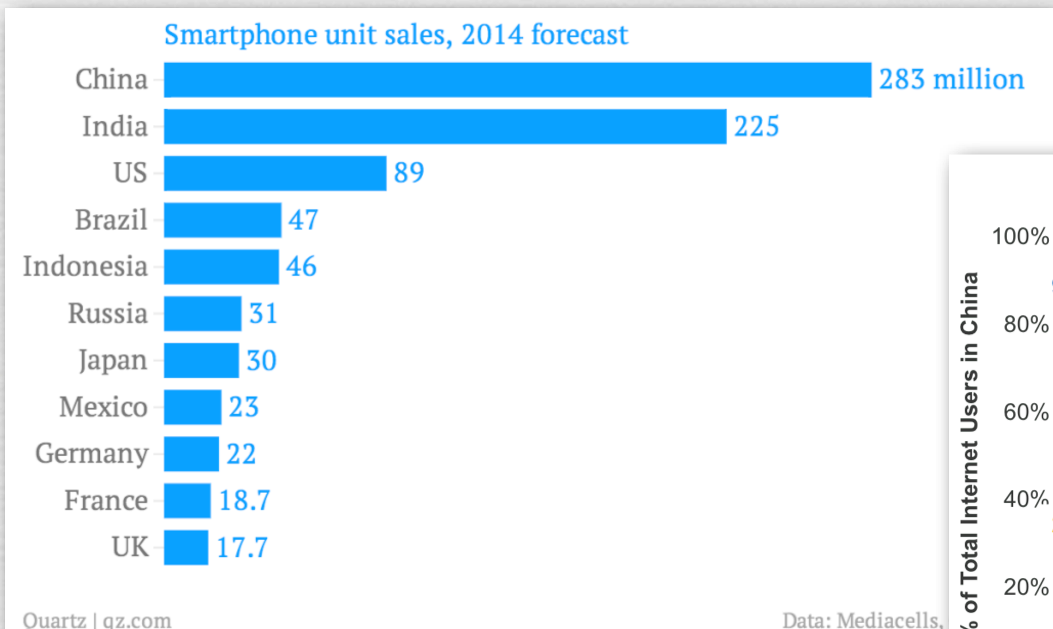
What does mobile-first mean?

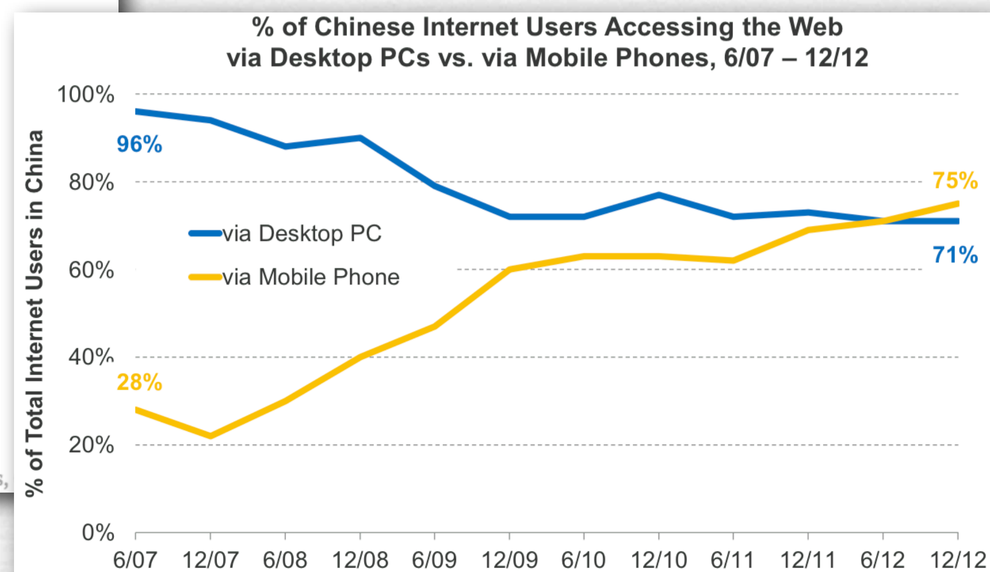


- all web sites should mobile friendly
- any desktop task should be do-able on your smartphone, although programming or writing will be inconvenient
- addressing the needs of the next billion users...

The next billion users

- probably don't speak English
- have paid dearly for their computing device
- will access the Internet mostly/only through a smartphone





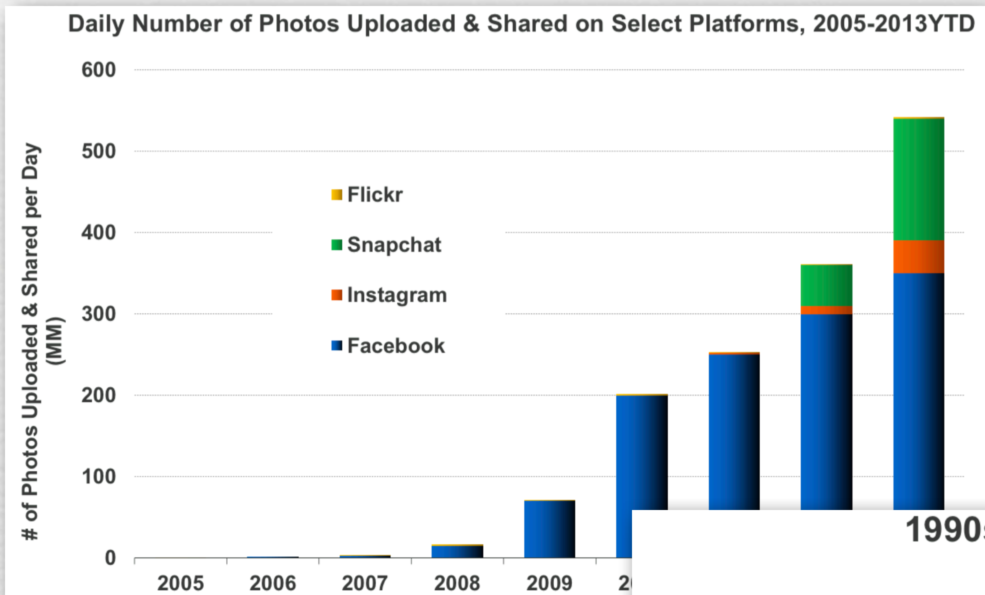
(Kleiner Perkins)

The next billion users

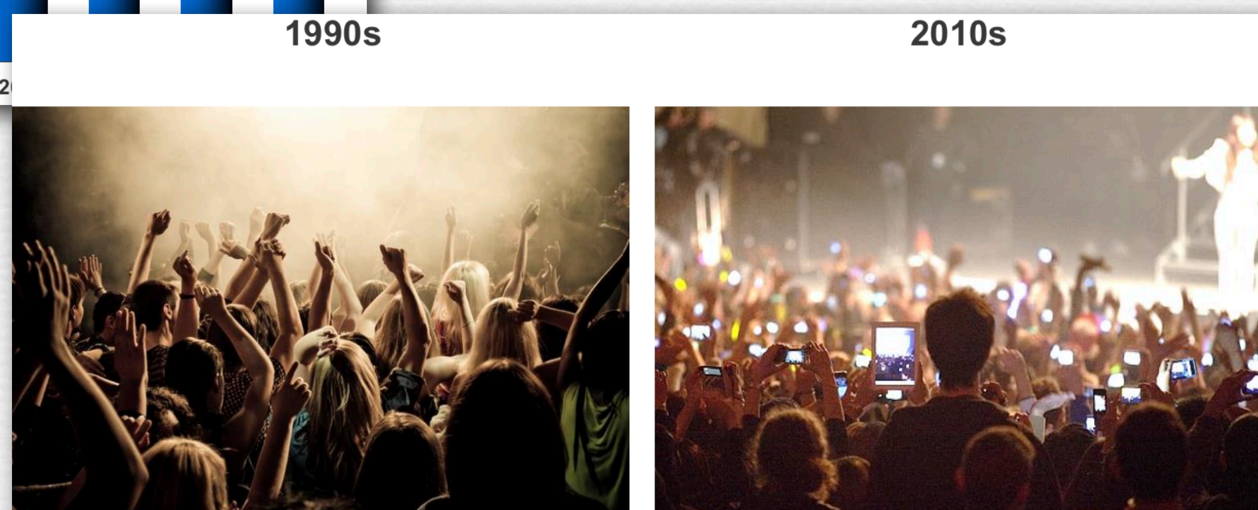
- probably don't speak English
- have paid dearly for their computing device
- will access the Internet mostly/only through a smartphone
- probably have mediocre connectivity (low bandwidth)
- cell phones give us convenience and entertainment;
  for them it means livelihood, freedom, and power
- the camera becomes an important tool...

Mobile cameras

- ~2B pictures are uploaded or shared per day



(Kleiner Perkins)



Mobile cameras

- ~2B pictures are uploaded or shared per day
- the best camera is the one you have with you
- mobile cameras are a powerful political tool ("liberation technology")

Demonstration in Kiev

Shooting of Walter Scott, North Charleston, SC

Mobile cameras

- ~2B pictures are uploaded or shared per day
- the best camera is the one you have with you
- mobile cameras are a powerful political tool
- wearable cameras are even more powerful...

What Google Glass means for the future of photography

University of North Carolina at Chapel Hill
October 28, 2013

Marc Levoy
Computer Science Department
Stanford University


hands free
(picture by Sebastian Thrun)

point of view


always available


instantly triggerable


Johnson
Cakes
WINNER!
2012
BARLO
OAKLAND
CHO
URBAN VILLAGE
your eyes are unobstructed

Why did Glass not launch in 2015?

- to be successful, Glass needed to be
  - *lightweight* enough,
  - *unobtrusive* enough,
  - *fashionable* enough, and
  - *useful* enough,

  to wear all day

- in the end,
  - it was *lightweight* and *fashionable,* but
  - the ratio of *useful* to *unobtrusive* was too low
  - and it was too *expensive* to build

smart watches, BEWARE!

- privacy was <u>not</u> a factor in canceling the launch

The challenges of mobile

- limited computing power
- always worried about battery life
- no precision pointing, just your finger(s)
- no keyboard, so can't program or write extensively
- small screen, difficult ambient lighting
- variable (or no) connectivity
- complicated computing platform

The challenges of mobile

- limited computing power
- always worried about battery life
- no precision pointing, just your finger(s)
- no keyboard, so can't program or write extensively
- small screen, difficult ambient lighting
- variable (or no) connectivity
- complicated computing platform
- might be tethered to a wearable...

The challenges of wearables

- even more limited computing and battery life
- even smaller display and cruder user interface
- even worse connectivity, and an extra hop

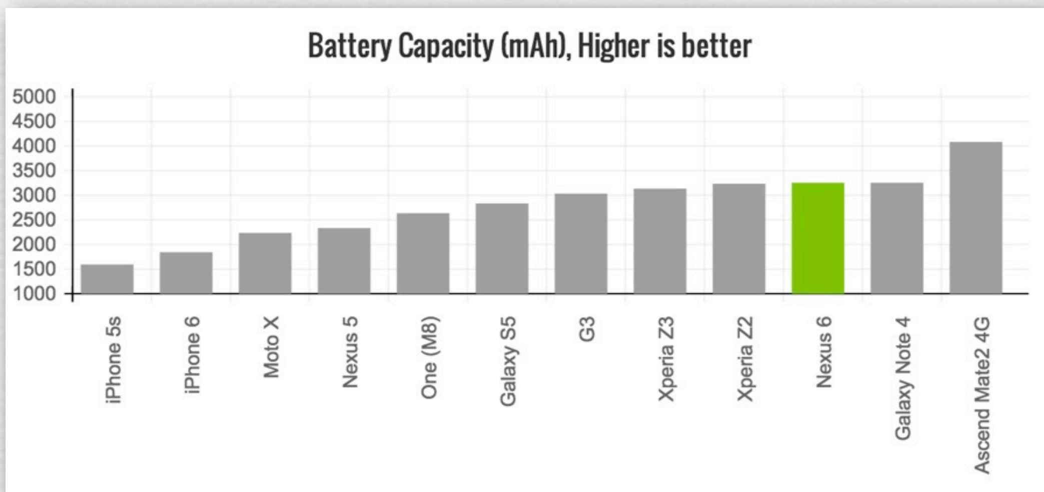
The challenges of wearables

- even more limited computing and battery life
- even smaller display and cruder user interface
- even worse connectivity, and an extra hop

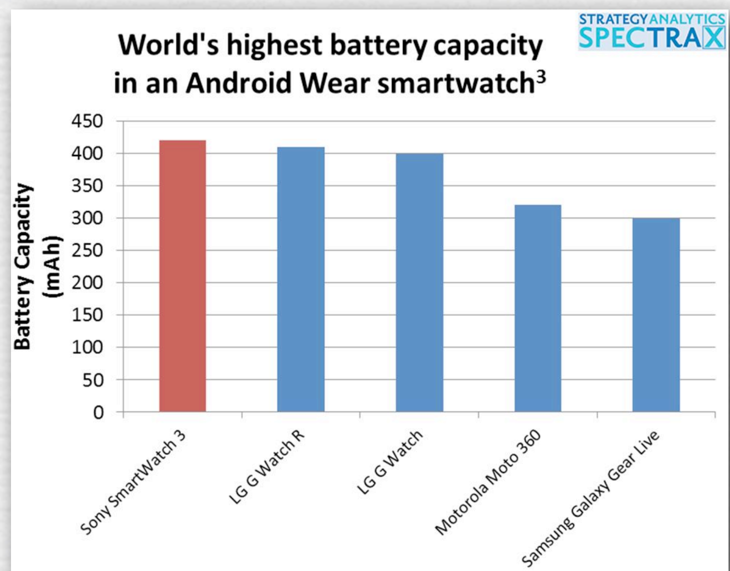
Performance is measured by speed <u>and</u> power

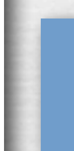
- cumulative usage (energy)
  - measured in milliwatt-hours
  - mobile devices must last all day

big challenge for watches!



(ubergizmo)





Apple watch

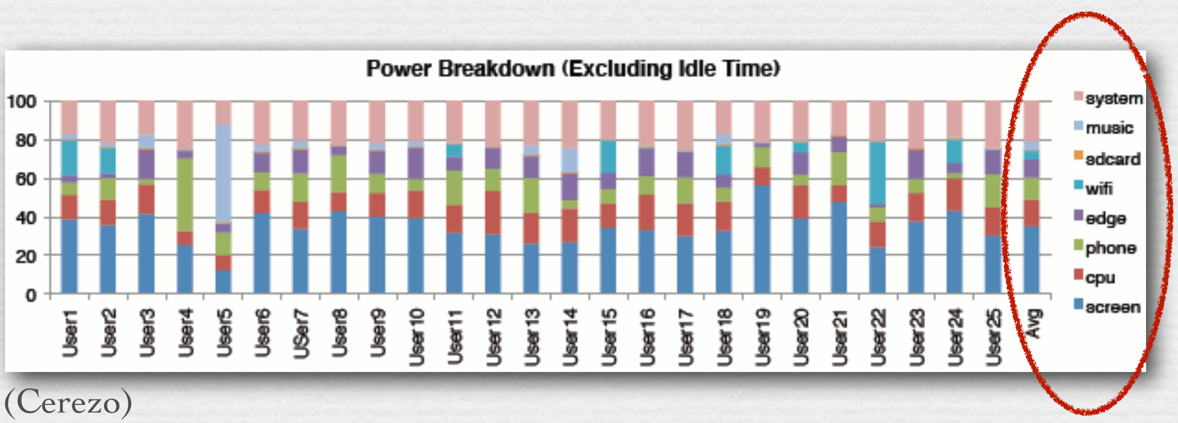
Performance is measured by speed and power

- ✦ cumulative usage (energy)
  - measured in milliwatt-hours
  - mobile devices must last all day

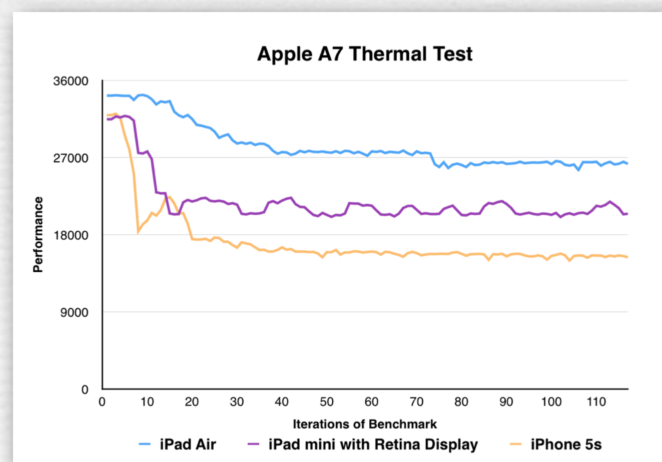
big challenge for watches!

- ✦ peak usage (power)
  - measured in milliwatts
  - limited by current draw on battery and heat dissipation
  - heat controlled by thermal throttling, e.g. cutting clock rate

big challenge for phones!



(Cerezo)



(anandtech)

Performance is measured by speed <u>and</u> power

- cumulative usage (energy)
  - measured in milliwatt-hours
  - mobile devices must last all day

big challenge for watches!

- peak usage (power)
  - measured in milliwatts
  - limited by current draw on battery and heat dissipation
  - heat controlled by thermal throttling, e.g. cutting clock rate

big challenge for phones!

Heavy computing is ok if it's over quickly.

Mobile devices need a breakthrough in <u>cooling</u>, not performance.

Upload data to cloud for computation?

- sending a burst of 10 × 5Mpix JPEG images (2MB@) over 3G to the cloud takes 50 secs at 400mA power
- for the same energy you could compute on an Android phone for 100 seconds
- 100 seconds × 2.7GHz × 4 cores = 22K operations on each pixel of our 50Mpix burst

It's almost never worth sending data to the cloud for processing.

Action items for computer scientists

1. embarrassingly parallel algorithms are not a panacea on mobile; you need algorithms that actually do less work

Functionality depends on connectivity

- a cell phone might contain 7 radios
  - CDMA, GSM, Wifi, Bluetooth, NFC, GPS, FM
- graceful degradation in functionality if connectivity is poor or intermittent or missing
  - seamless hand-off between wifi and cellular data
  - progressive streaming & rendering of images and video
  - ability to use device without cloud-based voice recognition

big challenge
for wearables!

Action items for computer scientists

1. embarrassingly parallel algorithms are not a panacea on mobile; we need algorithms that do less work
2. need better voice recognition / transcription on device, and the solution can't require a giant database

Functionality depends on connectivity

- a cell phone might contain 7 radios
  - CDMA, GSM, Wifi, Bluetooth, NFC, GPS, FM
- graceful degradation in functionality if connectivity is poor or intermittent or missing
  - seamless hand-off between wifi and cellular data
  - progressive streaming & rendering of images and video
  - ability to use device without cloud-based voice recognition
- ways of synchronizing content with the cloud
  1. must be online (web, email, chat), or
  2. cache most recent (Google Docs), or
  3. pin selected content (iTunes, iPhoto, Play Music), or
  4. cache everything on device (Dropbox, Evernote)

Action items for computer scientists

1. embarrassingly parallel algorithms are not a panacea on mobile; we need algorithms that do less work
2. need better voice recognition / transcription on device, and the solution can't require a giant database
3. robust synchronization of large, diverse databases across multiple, intermittently connected devices is still elusive

Mobile devices are insanely complicated

- heterogeneous mixture of computing resources
  - CPU
  - GPU
  - DSP
  - VLIW co-processor
  - "programmable" ISP

increasingly hard to program

Mobile devices are insanely complicated

- heterogeneous mixture of computing resources
- multiple vendors who barely talk to each other
  - IP provider (face detection circuitry)
  - SoC chipmaker (Qualcomm)
  - phone maker (Motorola, if Nexus 6)
  - OS writer (Google, if Android)
  - app writer (including independent developers)

unless all of them are Apple

Mobile devices are insanely complicated

- heterogeneous mixture of computing resources
- multiple vendors who barely talk to each other
- the software stack is deeper than you think
  - multiple languages
    (in Android: Java, C++, assembler, microcode)
  - 13 nested function calls to lock the focusing lens!

Mobile devices are insanely complicated

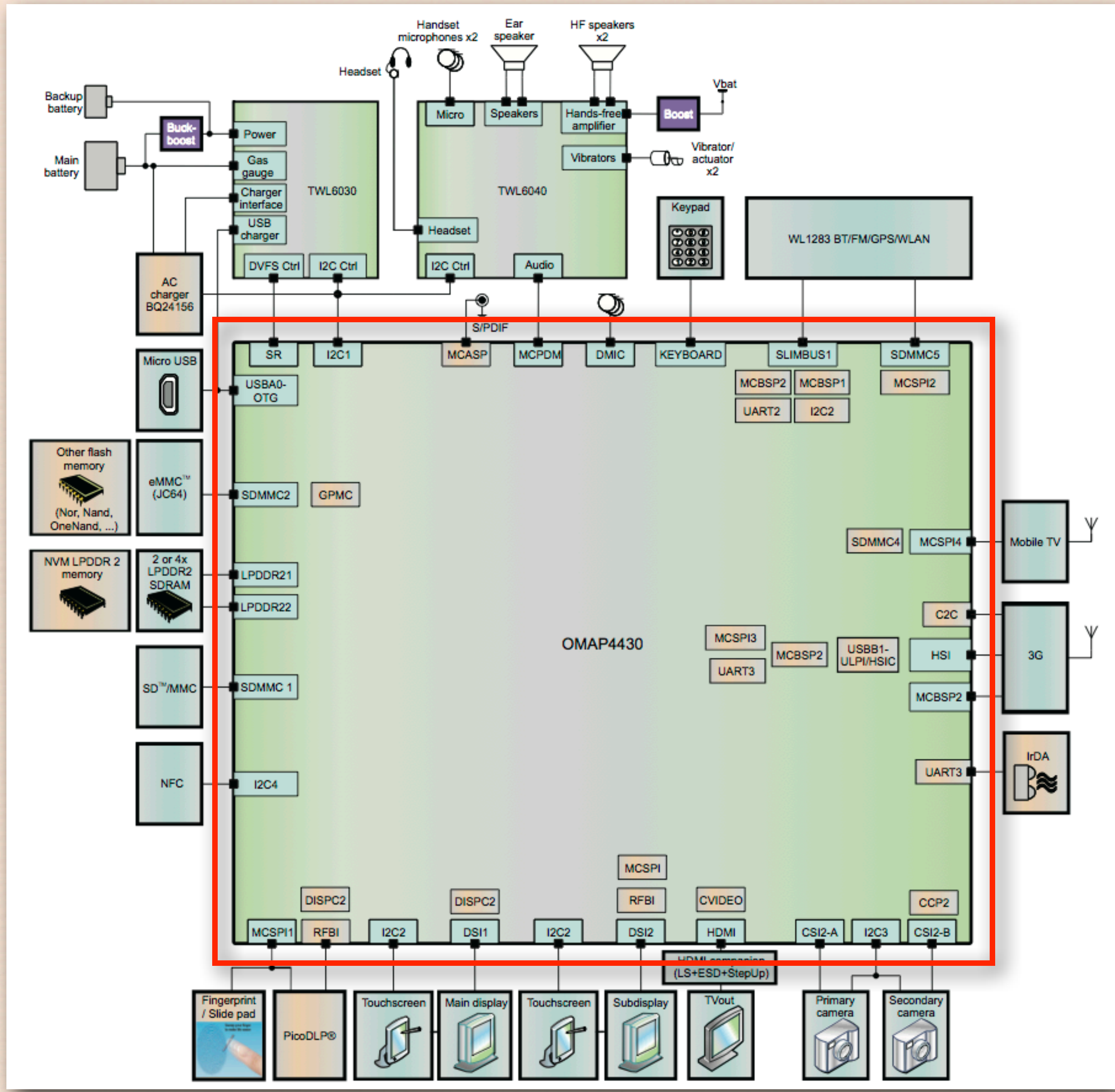
- heterogeneous mixture of computing resources
- multiple vendors who barely talk to each other
- the software stack is deeper than you think
- many functions are implemented in hardware...

Enabling hardware technologies for burst-mode computational photography

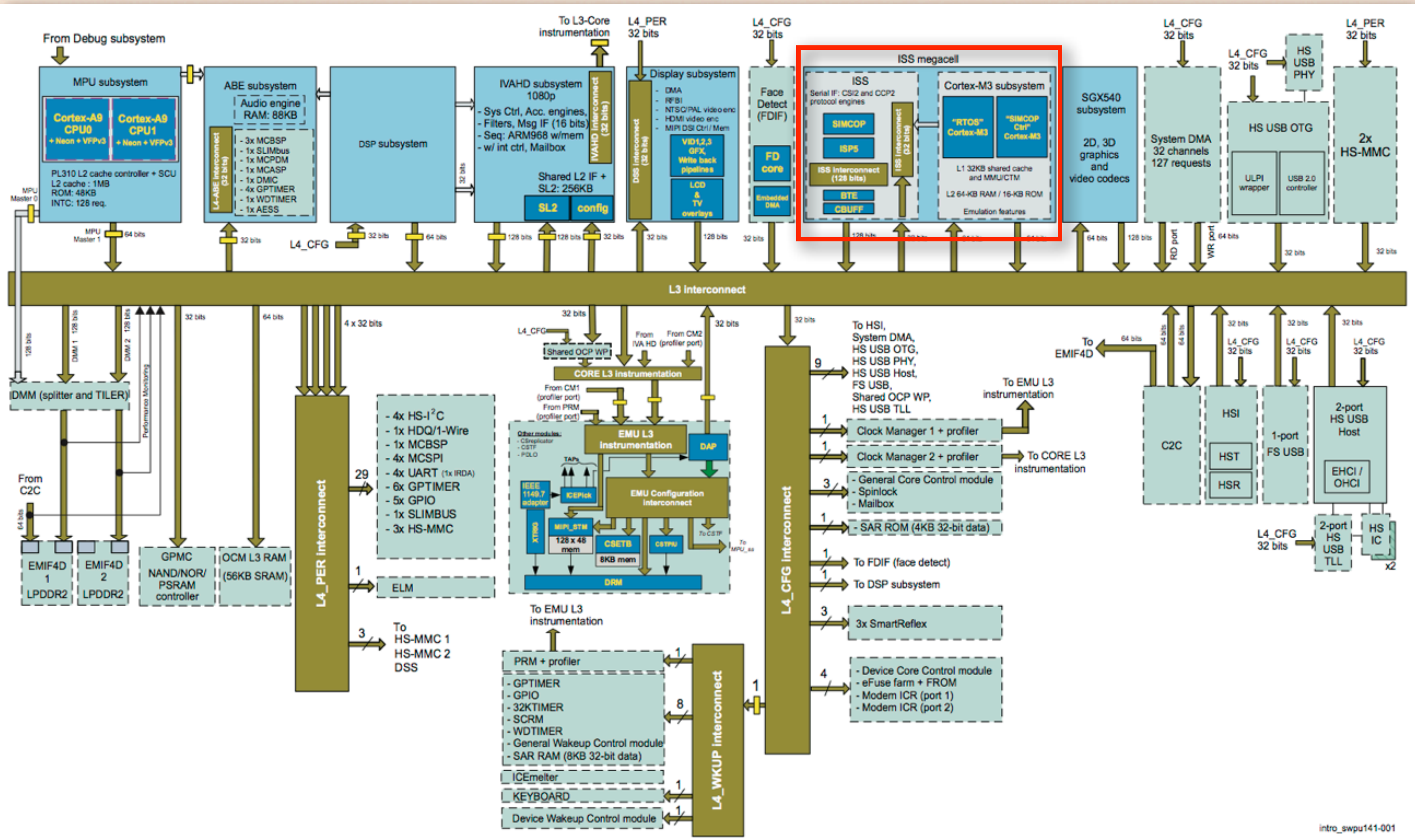
- fast sensor readout
  - 5Mpix @ 30fps on Google Glass
- fast processing
  - 5Mpix @ 30fps to YUV
- live viewfinder consists of processing at full-res to YUV, then downsizing to screen resolution
- this processing is implemented in ASIC hardware on most cameras

Texas Instruments OMAP4 SoC

(used in Google Glass)



Major subsystems



Imaging subsystem (ISS)

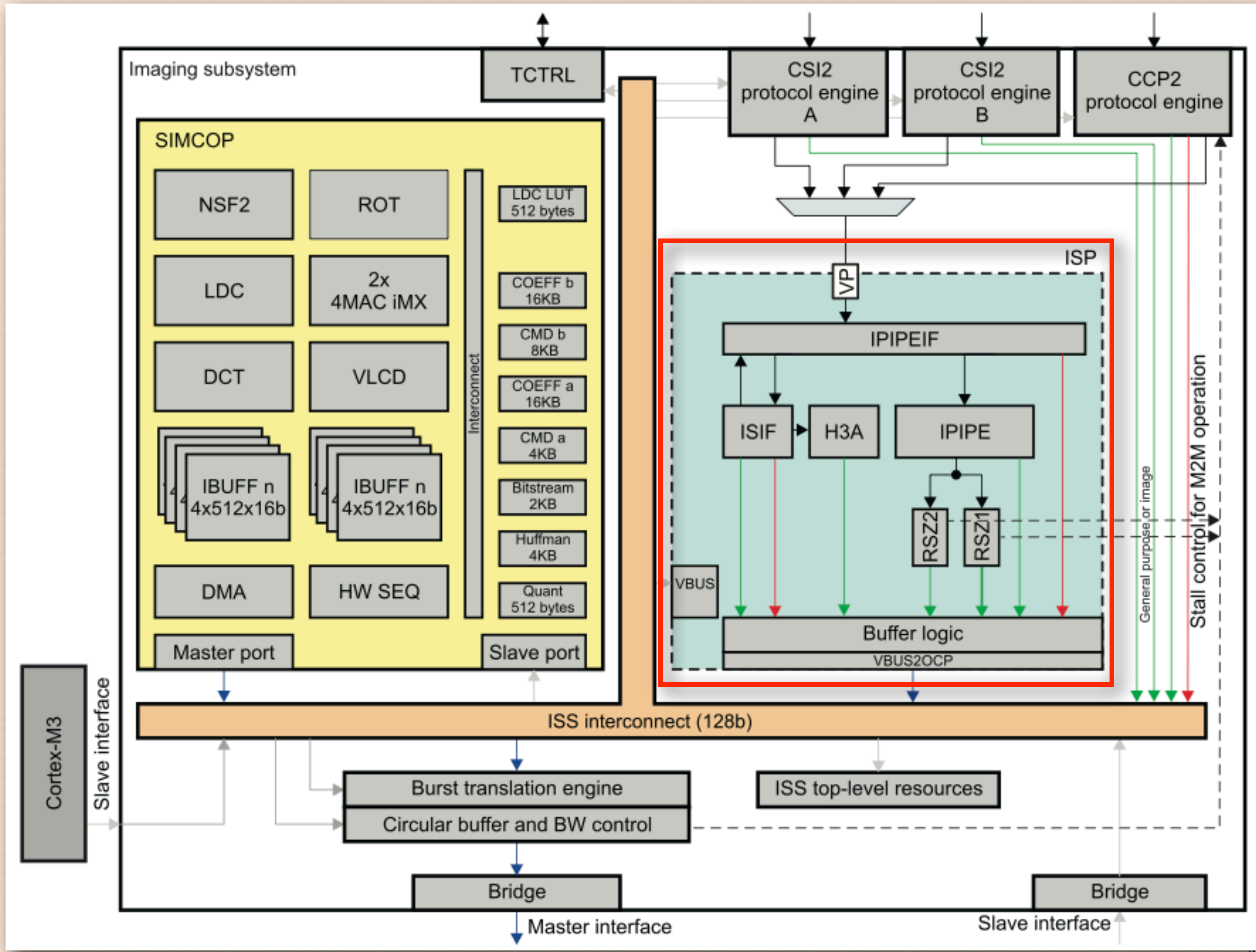


Image and signal processor (ISP)

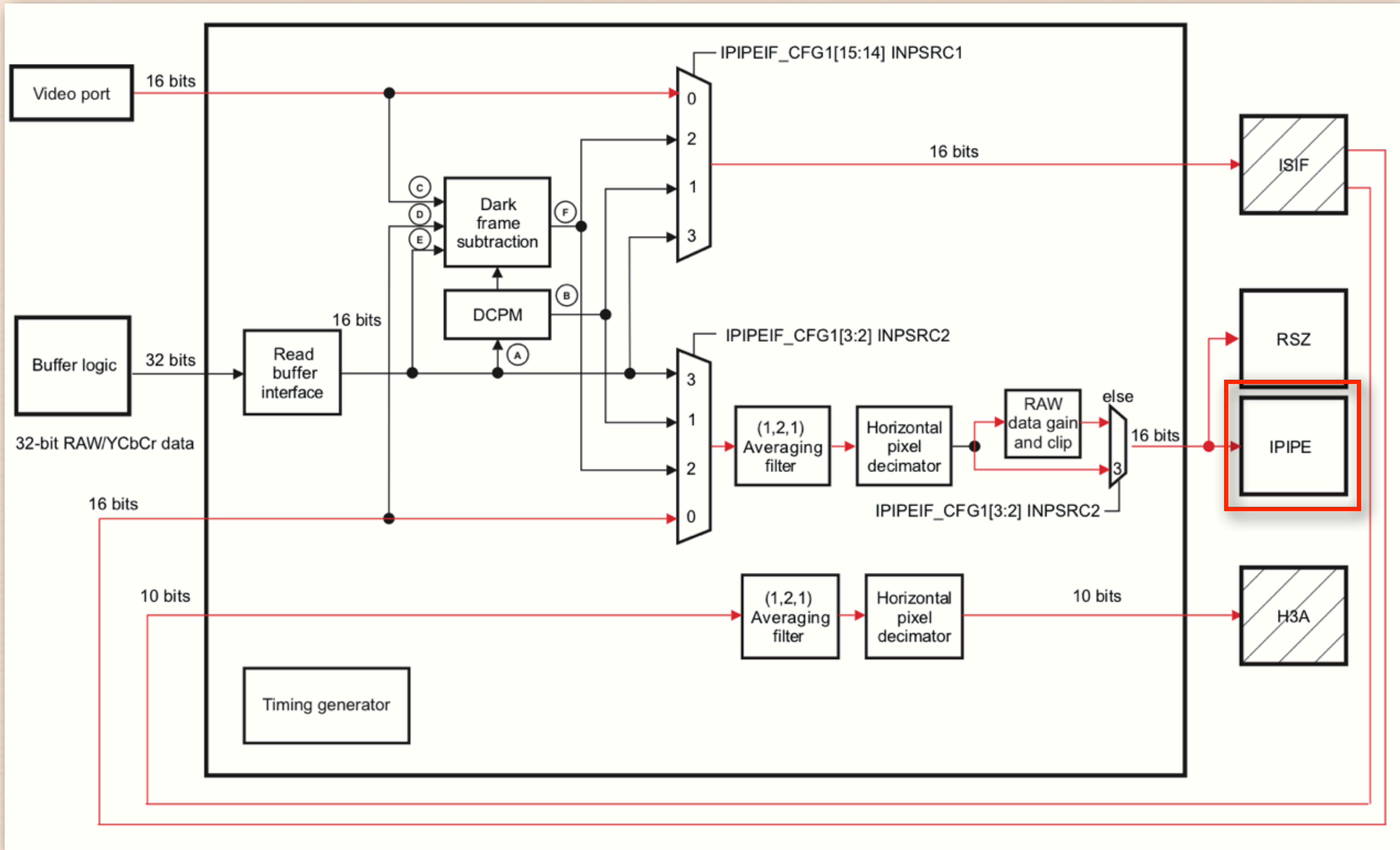
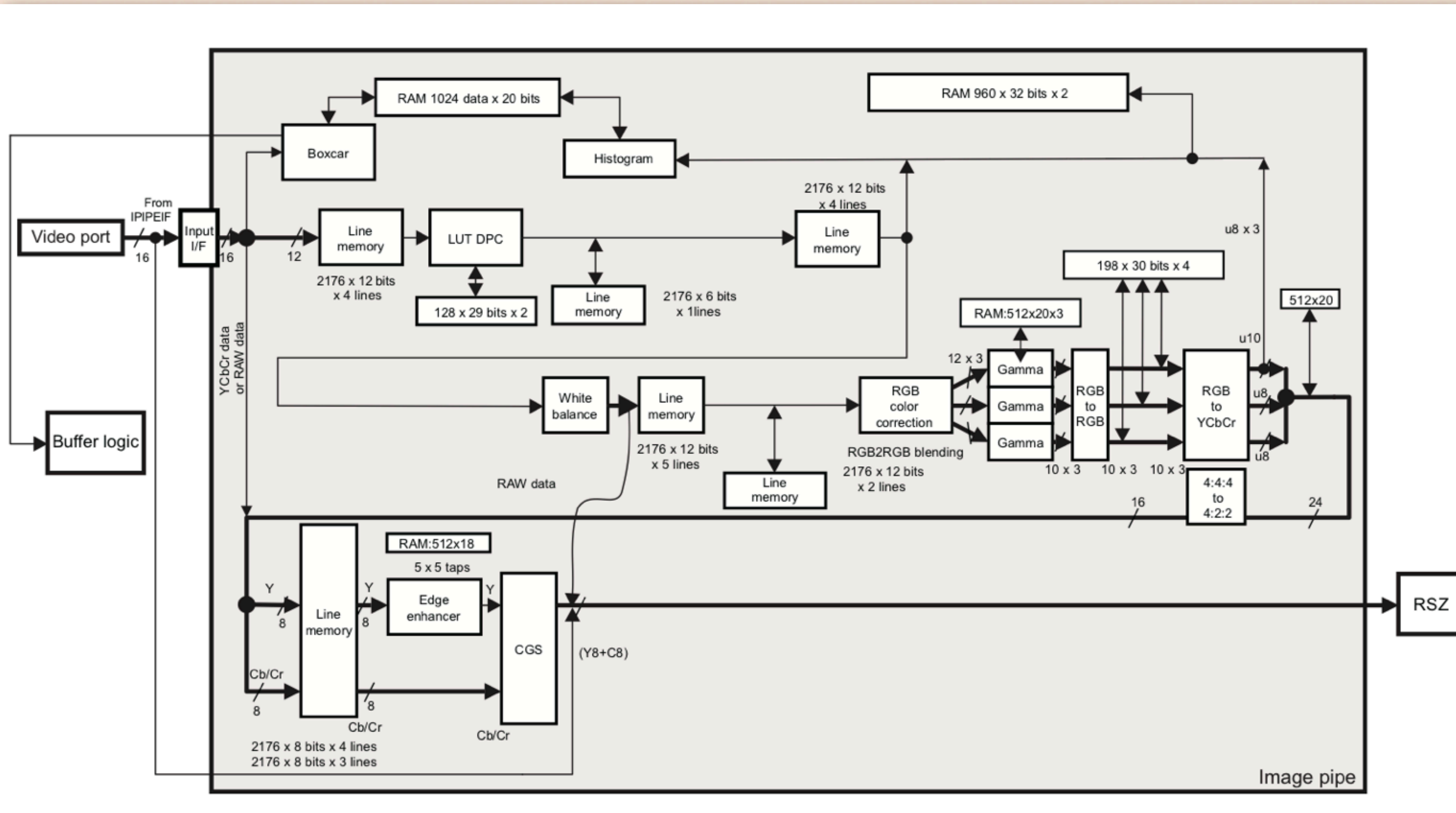
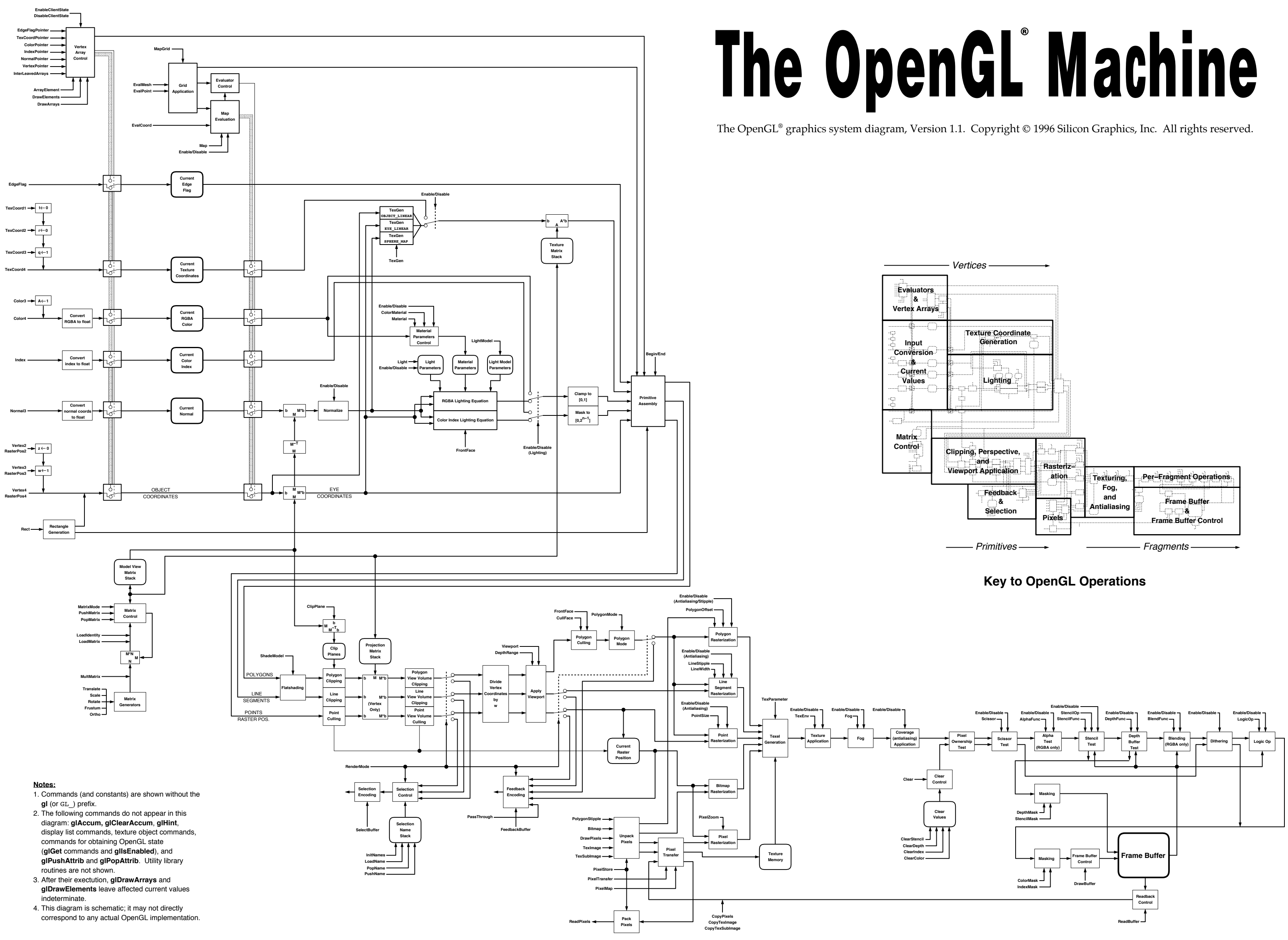


Image processing pipeline (IPIPE)
(public version of documentation)



The OpenGL® Machine



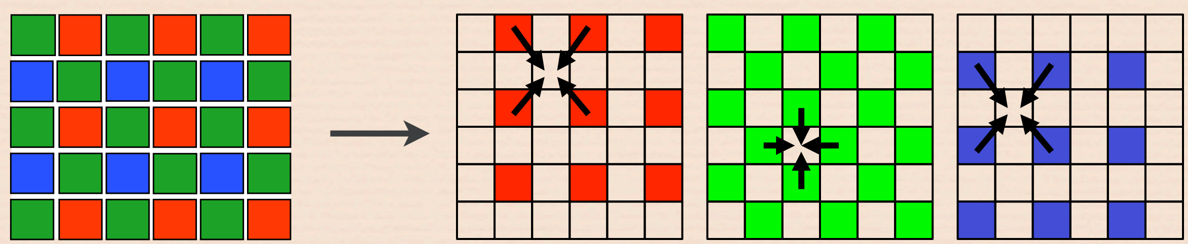
Notes:

1. Commands (and constants) are shown without the **gl** (or GL_) prefix.
2. The following commands do not appear in this diagram: **glAccum, glClearAccum**, **glHint**, display list commands, texture object commands, commands for obtaining OpenGL state (**glGet** commands and **glIsEnabled**), and **glPushAttrib** and **glPopAttrib**. Utility library routines are not shown.
3. After their exectution, **glDrawArrays** and **glDrawElements** leave affected current values indeterminate.
4. This diagram is schematic; it may not directly correspond to any actual OpenGL implementation.

Key to OpenGL Operations

Typical pipeline

- dark frame subtraction
- lens shading correction
- sensor linearization
- gain and offset controls
- statistics gathering
- pixel defect correction
- initial denoising
- demosaicking
- color correction
- tone mapping
- edge sharpening/denoising
- warping / resizing

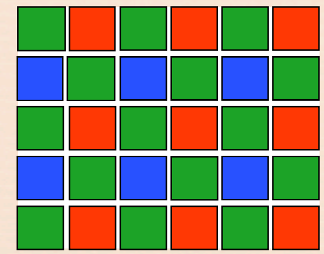


Bayer mosaic

YUV

What if we could reconfigure it?

- dark frame subtraction
- lens shading correction
- sensor linearization
- gain and offset controls
- statistics gathering
- pixel defect correction
- initial denoising
- demosaicking
- color correction
- tone mapping
- edge sharpening/denoising
- warping / resizing



tap-out of Bayer mosaic

re-injection of Bayer mosaic

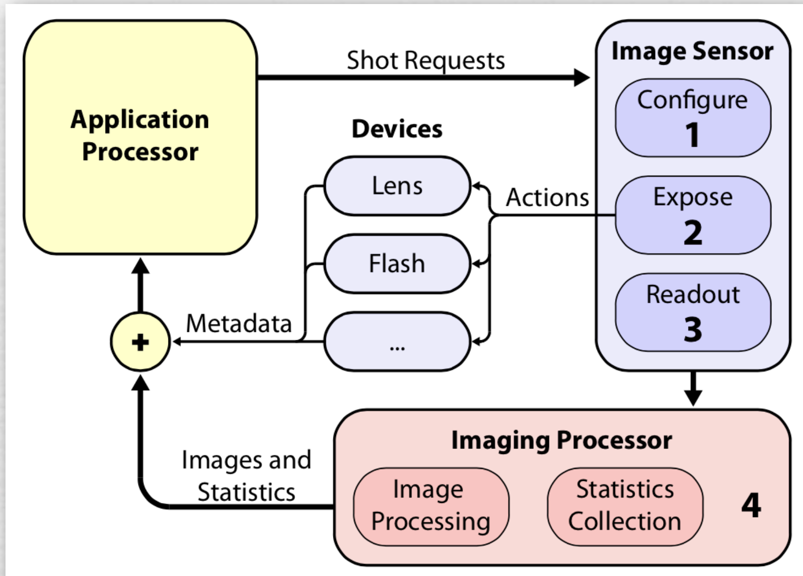
Using handshake to avoid demosaicking

1. read frames, process to RAW
2. align features with pixel precision
3. hope for an R,G,B in every pixel
4. re-inject but suppress demosaicing

Mobile devices are insanely complicated

- heterogeneous mixture of computing resources
- multiple vendors who barely talk to each other
- the software stack is deeper than you think
- many functions are implemented in hardware
- key is finding the right points of abstraction
  - for computer graphics:
    Jim Clark's Geometry Engine →
    OpenGL → GPU shading languages
  - for computational photography:
    Frankencamera architecture →
    Camera2 API → camera shading languages?
  - for computer vision: ??

Stanford Frankencamera architecture and FCam API [Adams SIGGRAPH 2010]

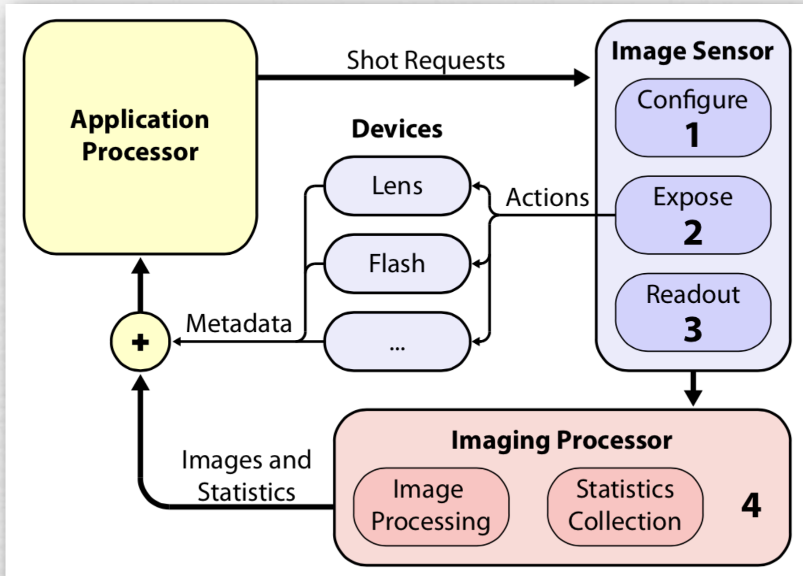



```
Sensor sensor;
Flash flash;
vector<Shot> burst(2);

burst[0].exposure = 1/200.;
burst[1].exposure = 1/30.;

Flash::FireAction fire(&flash);
fire.time = burst[0].exposure/2;
burst[0].actions.insert(fire);

sensor.stream(burst);

while (1) {
  Frame flashFrame =
    sensor.getFrame();
  Frame noflashFrame =
    sensor.getFrame();
}
```

Demonstration applications

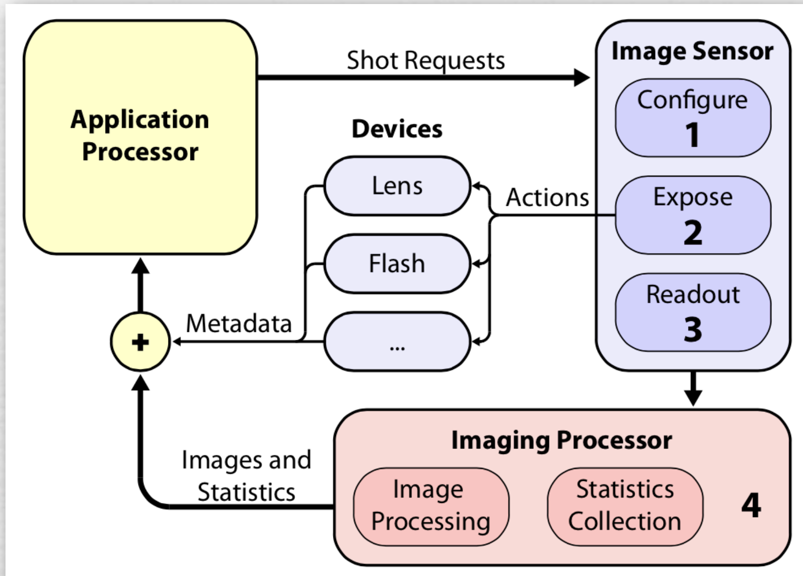
- Canon 430EX (smaller flash) strobed continuously
- Canon 580EX (larger flash) fired once at end of exposure

Stanford Frankencamera architecture and FCam API [Adams SIGGRAPH 2010]



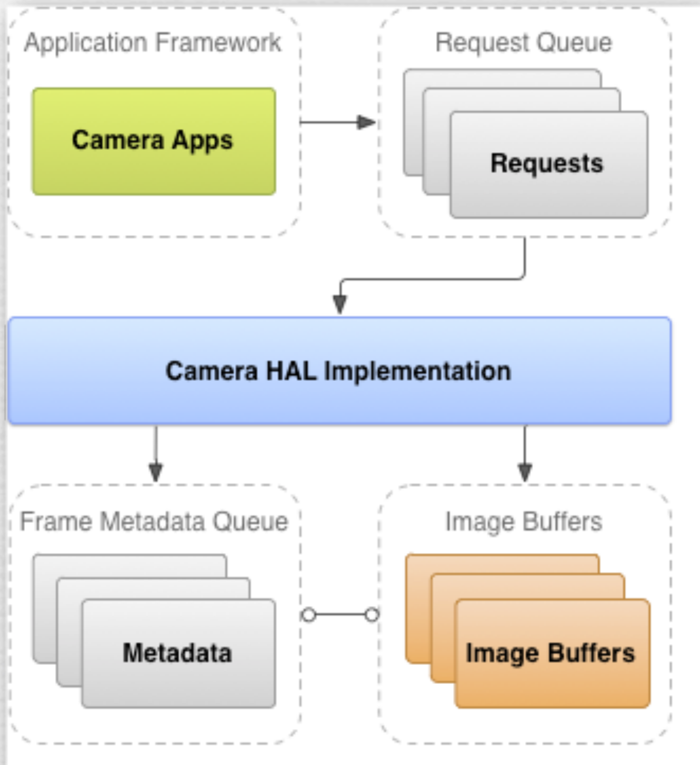

```
Sensor sensor;
Flash flash;
vector<Shot> burst(2);

burst[0].exposure = 1/200.;
burst[1].exposure = 1/30.;

Flash::FireAction fire(&flash);
fire.time = burst[0].exposure/2;
burst[0].actions.insert(fire);

sensor.stream(burst);

while (1) {
  Frame flashFrame =
    sensor.getFrame();
  Frame noflashFrame =
    sensor.getFrame();
}
```

Android Camera HAL 3 architecture and Camera2 API (Eddy Talvala and others)




```
Sensor sensor;
Flash flash;
vector<Shot> burst(2);

burst[0].exposure = 1/200.;
burst[1].exposure = 1/30.;

Flash::FireAction fire(&flash);
fire.time = burst[0].exposure/2;
burst[0].actions.insert(fire);

sensor.stream(burst);

while (1) {
  Frame flashFrame =
    sensor.getFrame();
  Frame noflashFrame =
    sensor.getFrame();
}
```

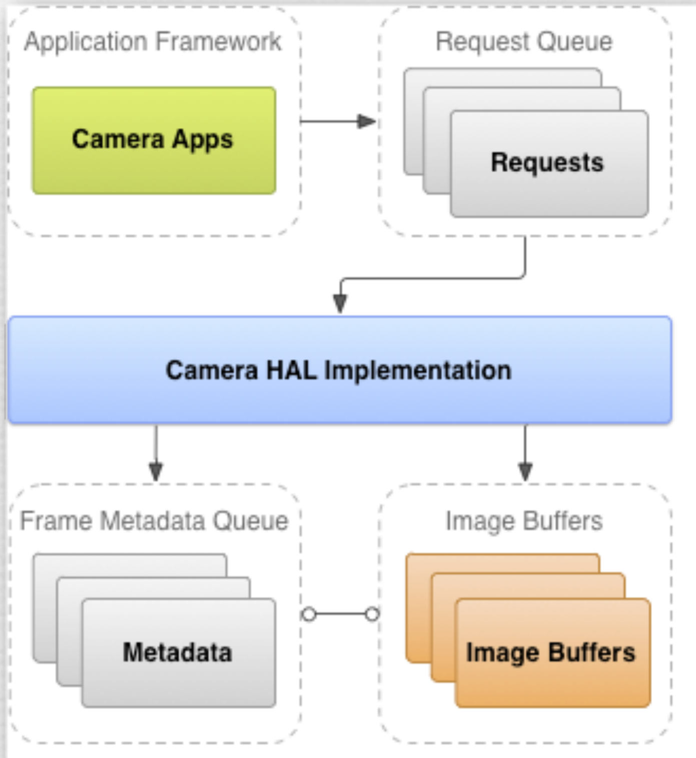
Android Camera HAL 3 architecture and Camera2 API (Eddy Talvala and others)




```
Sensor sensor;
Flash flash;
vector<Shot> burst(2);

burst[0].exposure = 1/200.;
burst[1].exposure = 1/30.;

Flash::FireAction fire(&flash);
fire.time = burst[0].exposure/2;
burst[0].actions.insert(fire);

sensor.stream(burst);

while (1) {
  Frame flashFrame =
    sensor.getFrame();
  Frame noflashFrame =
    sensor.getFrame();
}
```

Android Camera HAL 3 architecture and Camera2 API (Eddy Talvala and others)

```
// This is how to tell the camera to trigger.
mPreviewRequestBuilder.set(CaptureRequest.CONTROL_AE_PRECAPTURE_TRIGGER,
        CaptureRequest.CONTROL_AE_PRECAPTURE_TRIGGER_START);
// Tell #mCaptureCallback to wait for the precapture sequence to be set.
mState = STATE_WAITING_PRECAPTURE;
mCaptureSession.capture(mPreviewRequestBuilder.build(), mCaptureCallback,
        mBackgroundHandler);
    } catch (CameraAccessException e) {
        e.printStackTrace();
    }
}

/**
 * Capture a still picture. This method should be called when we get a response in
 * {@link #mCaptureCallback} from both {@link #lockFocus()}.
 */
private void captureStillPicture() {
    try {
        final Activity activity = getActivity();
        if (null == activity || null == mCameraDevice) {
            return;
        }
        // This is the CaptureRequest.Builder that we use to take a picture.
        final CaptureRequest.Builder captureBuilder =
                mCameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_STILL_CAPTURE);
        captureBuilder.addTarget(mImageReader.getSurface());

        // Use the same AE and AF modes as the preview.
        captureBuilder.set(CaptureRequest.CONTROL_AF_MODE,
                CaptureRequest.CONTROL_AF_MODE_CONTINUOUS_PICTURE);
        captureBuilder.set(CaptureRequest.CONTROL_AE_MODE,
                CaptureRequest.CONTROL_AE_MODE_ON_AUTO_FLASH);

        // Orientation
        int rotation = activity.getWindowManager().getDefaultDisplay().getRotation();
        captureBuilder.set(CaptureRequest.JPEG_ORIENTATION, ORIENTATIONS.get(rotation));

        CameraCaptureSession.CaptureCallback CaptureCallback
                = new CameraCaptureSession.CaptureCallback() {

            @Override
            public void onCaptureCompleted(CameraCaptureSession session, CaptureRequest request,
                                           TotalCaptureResult result) {
                Toast.makeText(getActivity(), "Saved: " + mFile, Toast.LENGTH_SHORT).show();
                unlockFocus();
            }
        };

        mCaptureSession.stopRepeating();
        mCaptureSession.capture(captureBuilder.build(), CaptureCallback, null);
    } catch (CameraAccessException e) {
        e.printStackTrace();
    }
}
```



- allows control over the camera
- doesn't accelerate image processing

open problem!

Mobile devices are insanely complicated

- heterogeneous mixture of computing resources
- multiple vendors who barely talk to each other
- the software stack is deeper than you think
- many functions are implemented in hardware
- key is finding the right points of abstraction
- we also need the right programming model
  - library (API)
  - general language
  - domain-specific language
  - low-level language (machine instructions)

Halide?

Separating algorithms from schedules

[Ragan-Kelley 2012]

(a) Clean C++ : 9.94 ms per megapixel

```
void blur(const Image &in, Image &blurred) {
 Image tmp(in.width(), in.height());

 for (int y = 0; y < in.height(); y++)
  for (int x = 0; x < in.width(); x++)
   tmp(x, y) = (in(x-1, y) + in(x, y) + in(x+1, y))/3;

 for (int y = 0; y < in.height(); y++)
  for (int x = 0; x < in.width(); x++)
   blurred(x, y) = (tmp(x, y-1) + tmp(x, y) + tmp(x, y+1))/3;
}
```

Separating algorithms from schedules

[Ragan-Kelley 2012]

(b) Fast C++ (for x86) : 0.90 ms per megapixel

```
void fast_blur(const Image &in, Image &blurred) {
 __m128i one_third = _mm_set1_epi16(21846);
 #pragma omp parallel for
 for (int yTile = 0; yTile < in.height(); yTile += 32) {
  __m128i a, b, c, sum, avg;
  __m128i tmp[(256/8)*(32+2)];
  for (int xTile = 0; xTile < in.width(); xTile += 256) {
   __m128i *tmpPtr = tmp;
   for (int y = -1; y < 32+1; y++) {
    const uint16_t *inPtr = &(in(xTile, yTile+y));
    for (int x = 0; x < 256; x += 8) {
     a = _mm_loadu_si128((__m128i*)(inPtr-1));
     b = _mm_loadu_si128((__m128i*)(inPtr+1));
     c = _mm_load_si128((__m128i*)(inPtr));
     sum = _mm_add_epi16(_mm_add_epi16(a, b), c);
     avg = _mm_mulhi_epi16(sum, one_third);
     _mm_store_si128(tmpPtr++, avg);
     inPtr += 8;
   }}
   tmpPtr = tmp;
   for (int y = 0; y < 32; y++) {
    __m128i *outPtr = (__m128i *)(&(blurred(xTile, yTile+y)));
    for (int x = 0; x < 256; x += 8) {
     a = _mm_load_si128(tmpPtr+(2*256)/8);
     b = _mm_load_si128(tmpPtr+256/8);
     c = _mm_load_si128(tmpPtr++);
     sum = _mm_add_epi16(_mm_add_epi16(a, b), c);
     avg = _mm_mulhi_epi16(sum, one_third);
     _mm_store_si128(outPtr++, avg);
}}}}}
```

Separating algorithms from schedules

[Ragan-Kelley 2012]

(c) Halide : 0.90 ms per megapixel

```
Func halide_blur(Func in) {
 Func tmp, blurred;
 Var x, y, xi, yi;

 // The algorithm
 tmp(x, y) = (in(x-1, y) + in(x, y) + in(x+1, y))/3;
 blurred(x, y) = (tmp(x, y-1) + tmp(x, y) + tmp(x, y+1))/3;

 // The schedule
 blurred.tile(x, y, xi, yi, 256, 32).vectorize(xi, 8).parallel(y);
 tmp.chunk(x).vectorize(x, 8);

 return blurred;
}
```

Why is Halide spreading so fast?

- because with a bit of portable code you can write
  - faster matrix multiply than Eigen
  - faster Gaussian blur than Intel Performance Primitives
  - faster Fourier transform than fftw
- or maybe because it...
  - runs on device and in the cloud
  - is supported on Linux, Windows, OSX, iOS, Android
  - compiles to x86, ARM, MIPS, native client, OpenCL, OpenGL, CUDA, JavaScript, RenderScript (ISPs soon)
- companies writing Halide code
  - Apple, Intel, Adobe, Microsoft, Nvidia, Google, Facebook, Qualcomm, Sony, Datexim, Algolux, ContextVision, Leap Motion, Nodasys, Nikon, Vicomtech, Ubisoft, Idruna, Imgtec, Lytro

Action items for computer scientists

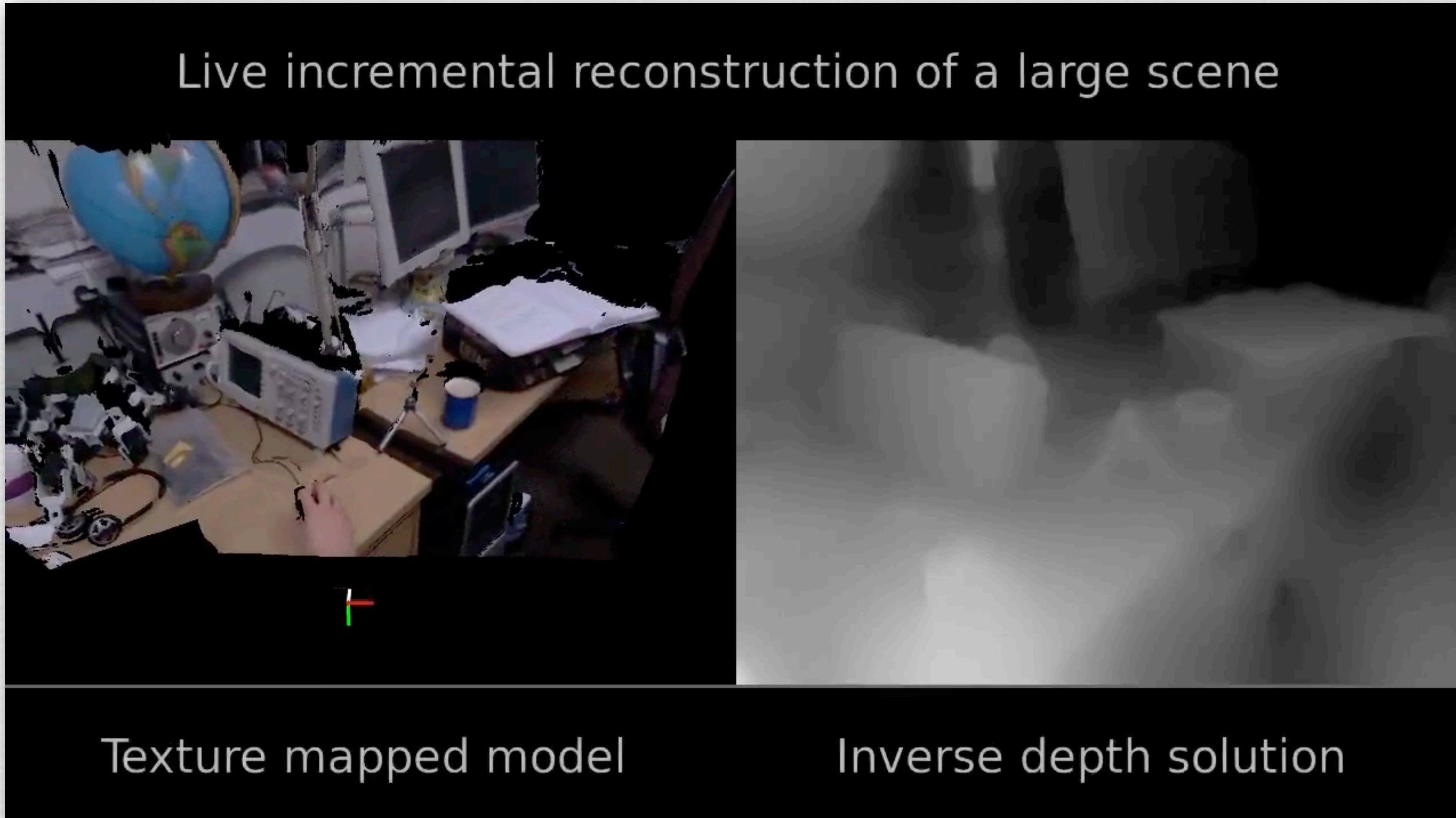
1. embarrassingly parallel algorithms are not a panacea on mobile; we need algorithms that do less work
2. need better voice recognition / transcription on device, and the solution can't require a giant database
3. robust synchronization of large, diverse databases across multiple, intermittently connected devices is still elusive
4. need architectures for accelerating image processing and computer vision, and good ways to program them

CS's biggest successes in 25 years

- deep learning + big data is replacing hand-built algorithms for many tasks, including photography
- computer vision is beginning to work
  - Google image search no longer relies solely on text
  - can estimate camera pose from sensed imagery ("visual odometry") in real-time
  - can compute stereo (at low-res) in real time

DTAM: dense tracking and mapping in real-time

[Newcombe, ICCV 2011]



- becoming possible on a mobile device (Google Tango)
- in the future, JPEG files will include depth (RGBZ)

CS's biggest successes in 25 years

- deep learning + big data is replacing hand-built algorithms for many tasks, including photography
- computer vision is beginning to work
  - Google image search no longer relies solely on text
  - can estimate camera pose from sensed imagery ("visual odometry") in real-time
  - can compute stereo (at low-res) in real time
  - can build 3D models in real time
  - lots of applications, including VR, AR

Word Lens

(app for iOS and Android)

- mediocre translation, but clever user interface
- recently bought by Google, runs on Glass

CS's biggest successes in 25 years

- deep learning + big data is replacing hand-built algorithms for many tasks, including photography
- computer vision is beginning to work
  - Google image search no longer relies solely on text
  - can estimate camera pose from sensed imagery ("visual odometry") in real-time
  - can compute stereo (at low-res) in real time
  - can build 3D models in real time
  - lots of applications, including VR, AR
  - pressure on hardware, abstractions, languages
  - brain drain from academia

Action items for computer scientists

1. embarrassingly parallel algorithms are not a panacea on mobile; we need algorithms that do less work
2. need better voice recognition / transcription on device, and the solution can't require a giant database
3. robust synchronization of large, diverse databases across multiple, intermittently connected devices is still elusive
4. need architectures for accelerating image processing and computer vision, and good ways to program them
5. allow faculty to rotate through industry, or spend 50% of their time in industry, without losing tenure

Mobile systems are hard to teach

- competition and patent lawsuits leads companies to keep their technologies secret
- mobile device manufacturers are EEs, not CSers, so their devices have poor, opaque, and inflexible software
- as a result, there are few textbooks about mobile systems technologies (or cameras), and few courses
  - How does auto white balancing work on real cameras?
  - Or auto exposure metering?
  - Or auto focusing?
  - Or denoising?

Mobile systems are hard to teach

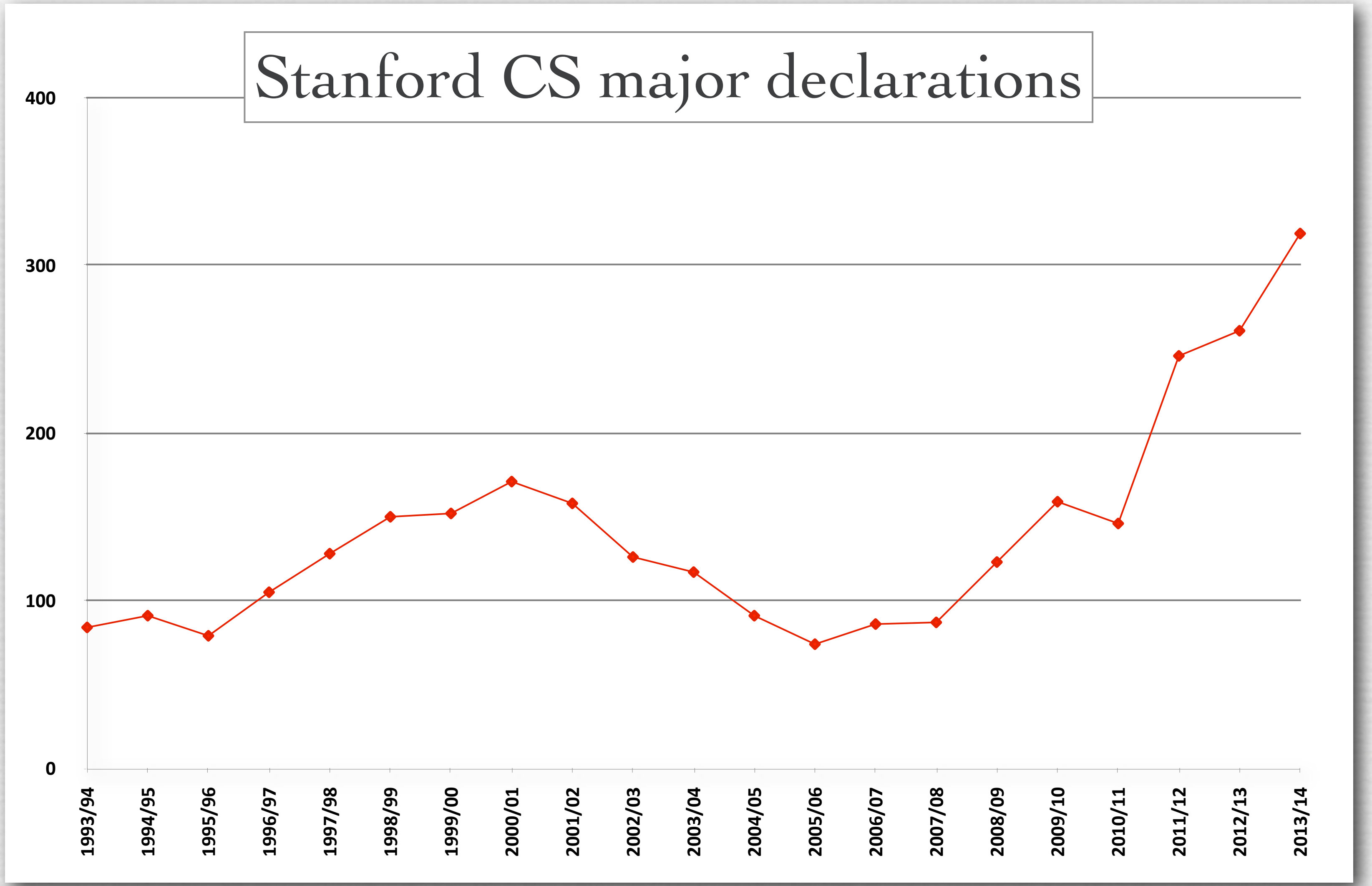
- competition and patent lawsuits leads companies to keep their technologies secret
- mobile device manufacturers are EEs, not CSers, so their devices have poor, opaque, and inflexible software
- as a result, there are few textbooks about mobile systems technologies (or cameras), and few courses
- students come out of school without the skills they need to succeed in industry
  - machine learning should be mandatory
  - so should web development, security, NLP
  - and mobile systems

Udacity's business model

Action items for computer scientists

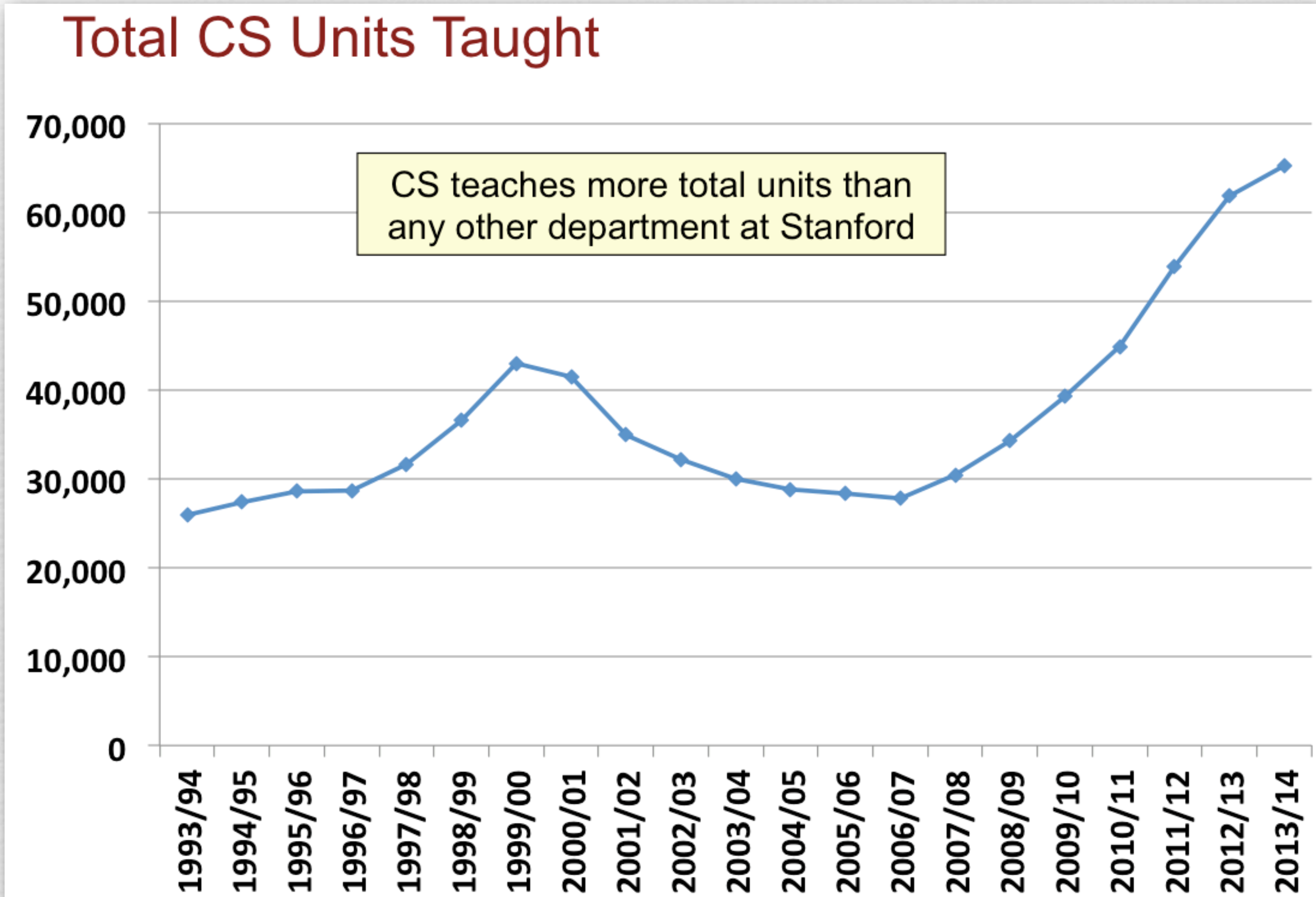
1. embarrassingly parallel algorithms are not a panacea on mobile; we need algorithms that do less work
2. need better voice recognition / transcription on device, and the solution can't require a giant database
3. robust synchronization of large, diverse databases across multiple, intermittently connected devices is still elusive
4. need architectures for accelerating image processing and computer vision, and good ways to program them
5. allow faculty to rotate through industry, or spend 50% of their time in industry, without losing tenure
6. develop platforms and write textbooks to enable teaching of mobile systems, especially via lab courses

Lab courses? With these enrollments?!



(Mehran Sahami)

Lab courses? With these enrollments?!



(Mehran Sahami)

Superhero vision

- seeing in the dark

Digital photography can easily exceed human vision

(Jesse Levinson Canon 10D, 28mm f/4, 3 min, ISO 100, 4 image pano)

- required a tripod
- can't currently do this using a cell phone, but it's not impossible
  - dark current (if one shot) or read noise (if a burst) must be very low

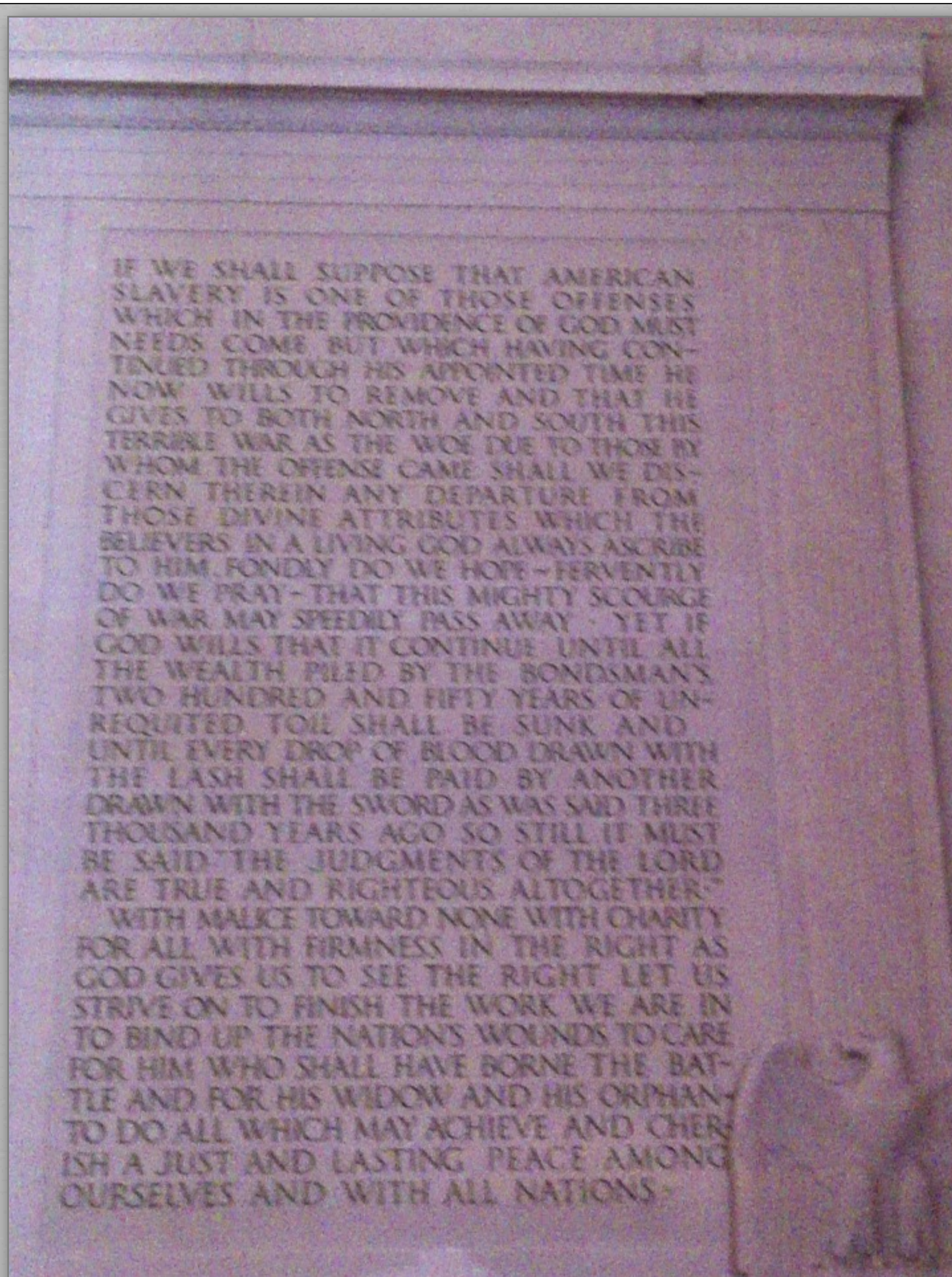
Low-light imaging using burst-mode computational photography

single frame
(iPhone 4)

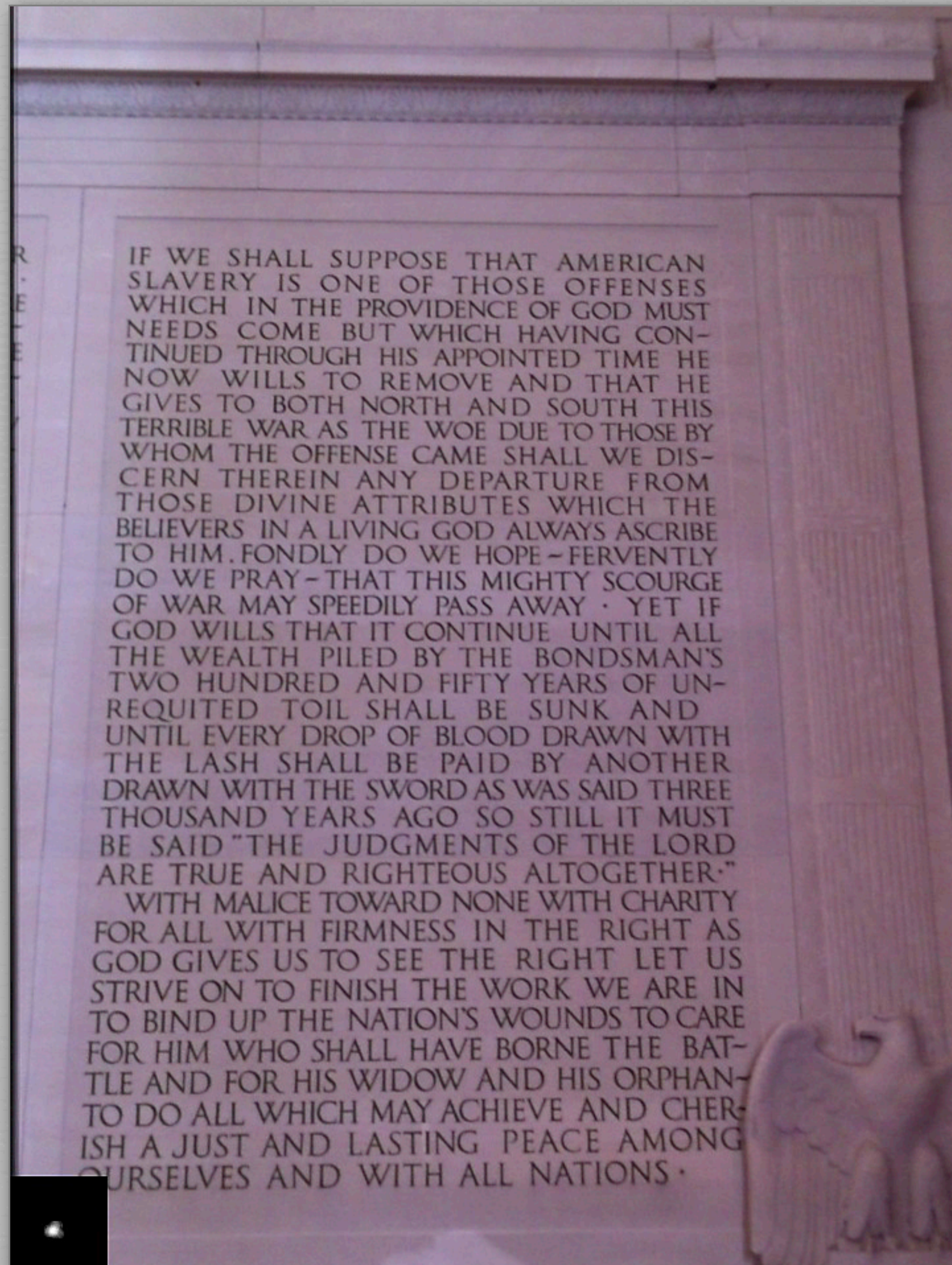
Low-light imaging using burst-mode computational photography

SNR increases as sqrt(# of frames)

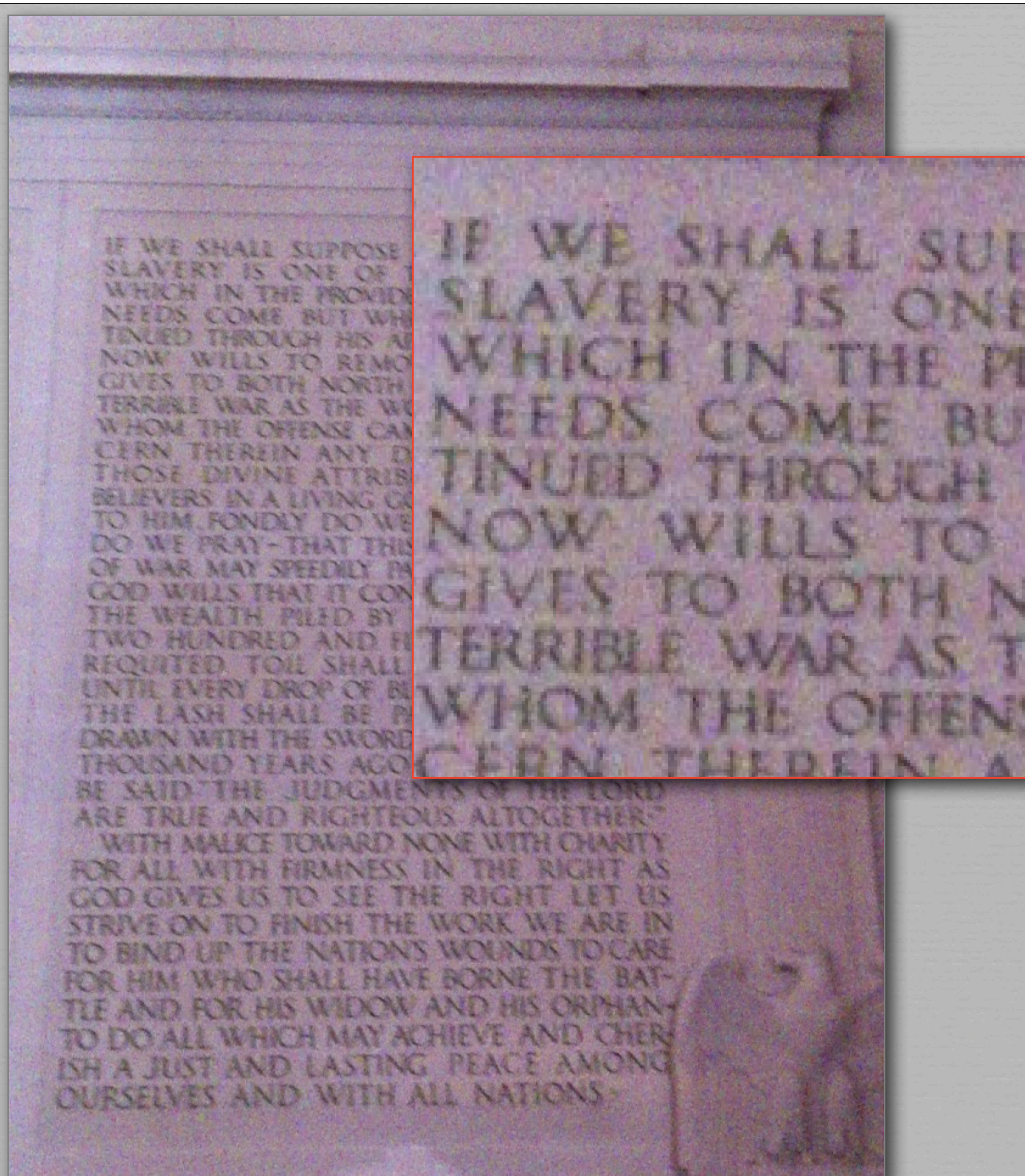
average of ~30 frames (SynthCam)



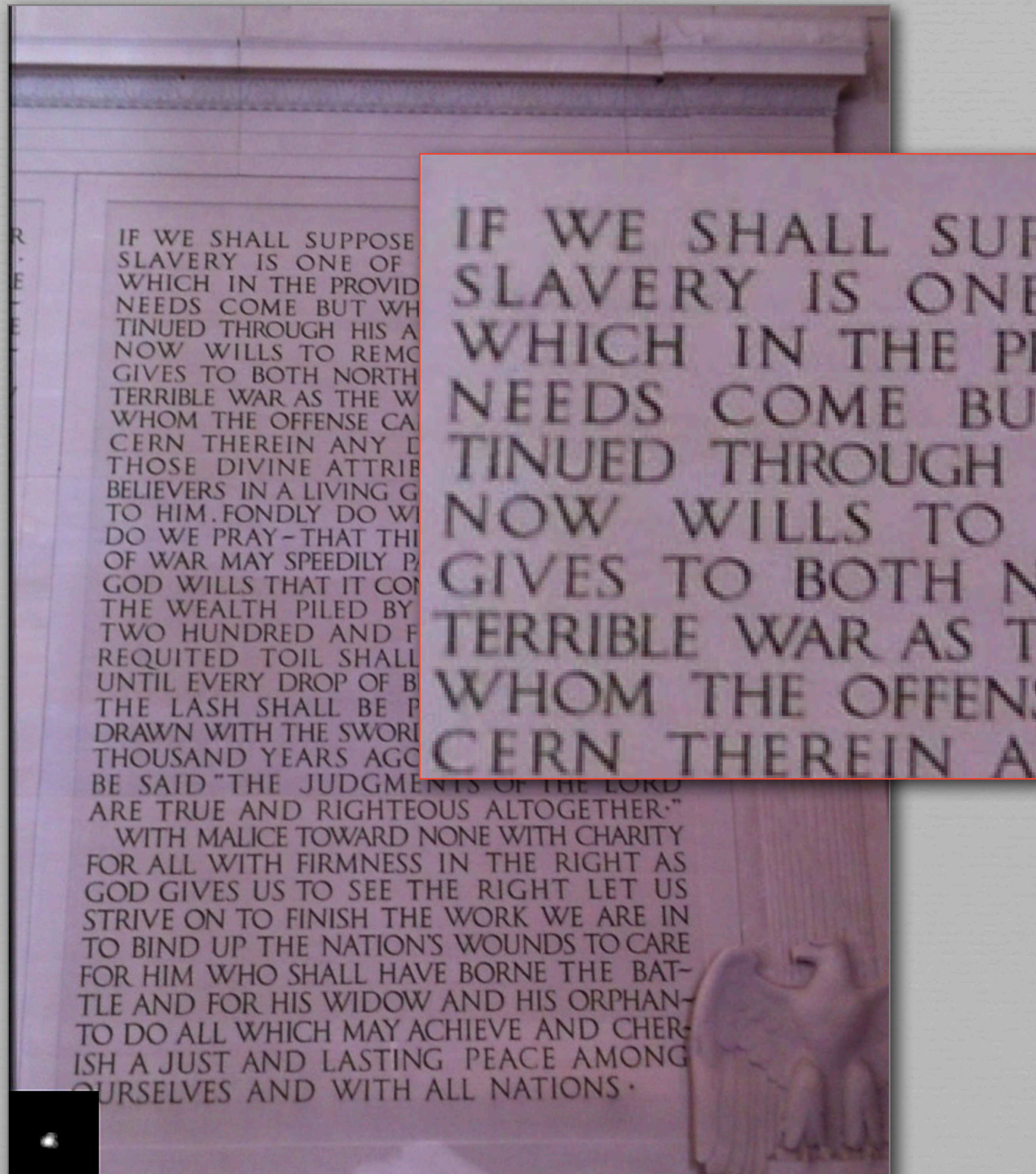
single frame



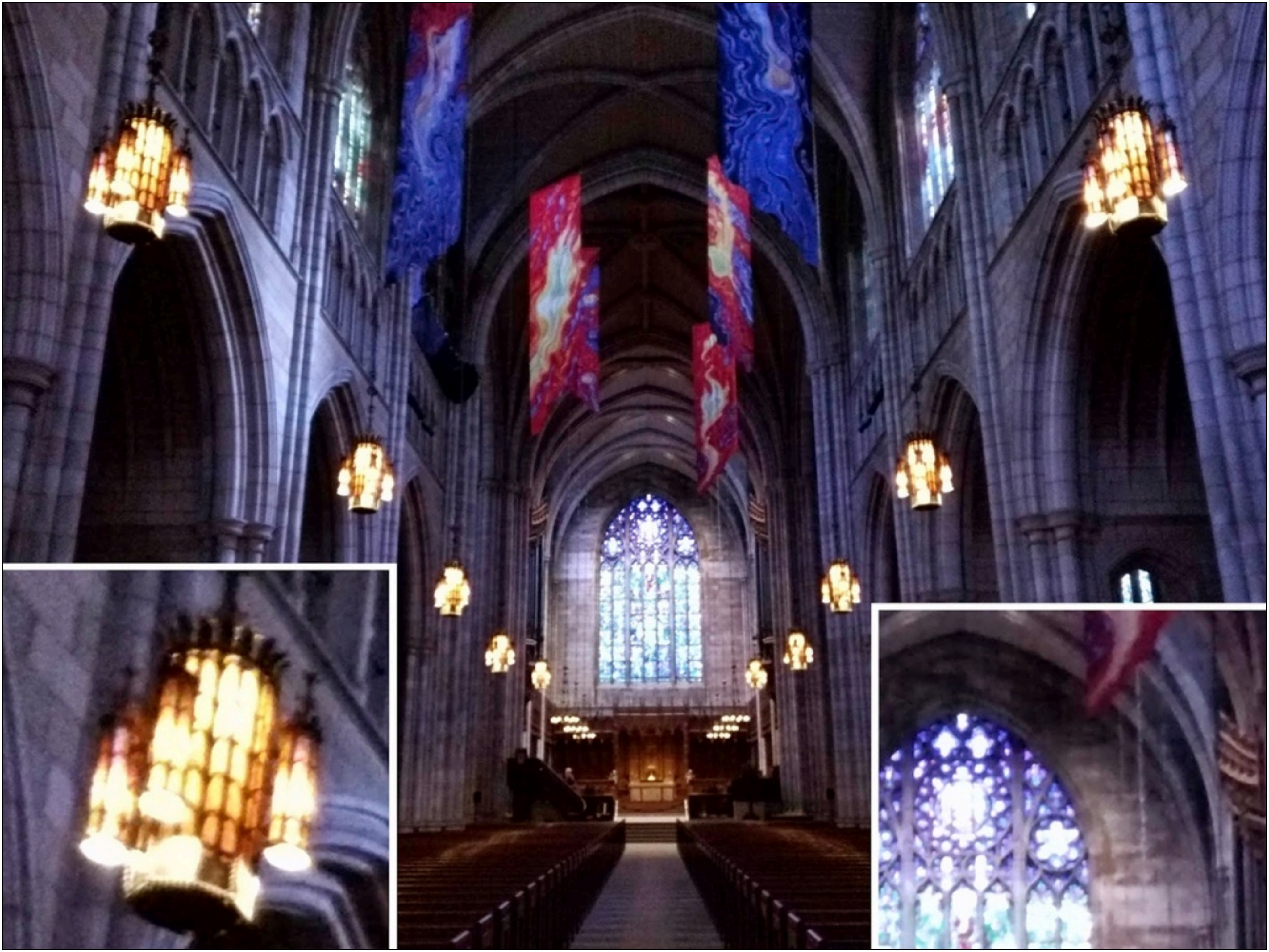
average of
~30 frames



single frame



average of
~30 frames



Superhero vision

- seeing in the dark
- seeing through objects

Removing foreground objects by translating the camera

- align the shots
- match histograms
- apply median filter

Superhero vision

- seeing in the dark
- seeing through objects
- magnifying glass, telescopic vision

Camera-based magnifiers

- optical zoom
  - requires a long optical path

- digital zoom (cropping)
  - requires a high pixel count, hence a thick camera

Nokia 808

- super-resolution
  - results typically look oversharpened

Beyond SLRs: Superhero vision

- seeing in the dark
- seeing through objects
- magnifying glass, telescopic vision
- slowing down motion

Superhero vision

- seeing in the dark
- seeing through objects
- magnifying glass, telescopic vision
- slowing down motion
- motion magnification, change magnification

Motion magnification

[Liu, SIGGRAPH 2005]

- can this be done using a (shaky) handheld camera?
- can it be computed on a (slow) mobile device?

Change magnification

[Wu, SIGGRAPH 2012]

- how much SNR is needed to detect this signal?
- is it socially acceptable to run this on Glass?

Superhero vision

- seeing in the dark
- seeing through objects
- magnifying glass, telescopic vision
- slowing down motion
- motion magnification, change magnification
- face recognition

If you met this man at a party...

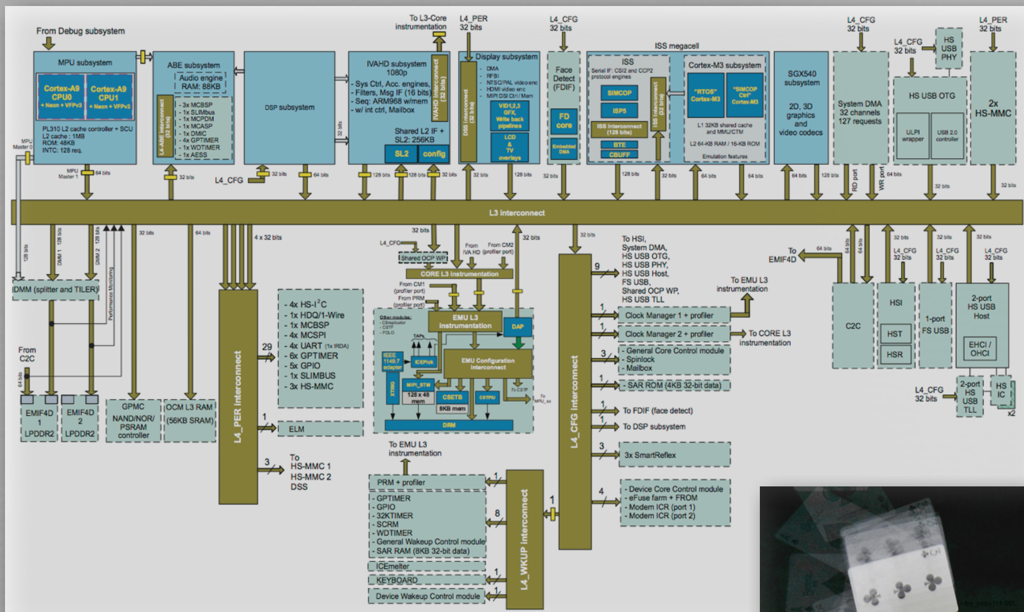
- *name:* Jack Sparrow
- *address:* Black Pearl
- *profession:* pirate
- *net worth:* zero
- *spouse:* many
- *criminal record:* long

Face recognition

- recognition from uncontrolled photos is still sci-fi
- Google pro-actively prohibited it on Glass

- it could eventually work
- if it does, someone will build a device to do it

- anonymity is so...*20th century;* get over it
- giving up anonymity ≠ giving up privacy



```
Sensor sensor;
Shot low, med, high;

low.exposure = 1/80.;
med.exposure = 1/20.;
high.exposure = 1/5.;

sensor.capture(low);
sensor.capture(med);
sensor.capture(high);

Frame frames[3];
frames[0] = sensor.getFrame();
frames[1] = sensor.getFrame();
frames[2] = sensor.getFrame();

fused = mergeHDR(frames);
```