

Android 5.x's Stream-Based Camera Architecture

Balwinder Kaur

Senior Member, Technical Staff
ON Semiconductor

Android Builder's Summit | San Jose | 3.23.15

Tools of the Trade



Agenda

- The Basic Idea
- Enabling New Experiences
- Camera2 Architecture
- Camera2 APIs
 - Ready. Set. Go!
- The Other Use Cases
- The Devil is in the Details
- Play Store Filters
- Demo
- Summary



The Basic Idea

Limitations of original Camera Architecture

Limitations of android.hardware.Camera

3 Primary Operating Modes

- Preview, Capture & Video Recording

New Features – hard to implement

- Burst mode photography
- Zero Shutter Lag, Multi-Shot HDR, Panoramic Stitch

No Per-Frame Control

- Not Possible

No Support for RAW

- Most products returned null for the raw callback

Minimal Metadata

- Support for face detection data

Primitive Custom Settings

- Through Camera.Parameters class but no guarantees when the settings would go into effect

Enabling New Experiences

Camera2 Architecture Enables -

Point and Click Camera

Professional Camera

Professional Camera + Post
Processing on-device

Innovative Mobile Camera ++

Features and Requirements

Point and Click

- Preview
- Still
- Video Recording

Professional Camera

- Fine Grained Control of Lens, Sensor, Flash
- RAW Sensor Output with capture metadata

Computational Photography

- Individual Frame Control
- Algorithms & Compute Power
- HDR, Focus Stacking, Exposure Bracketing

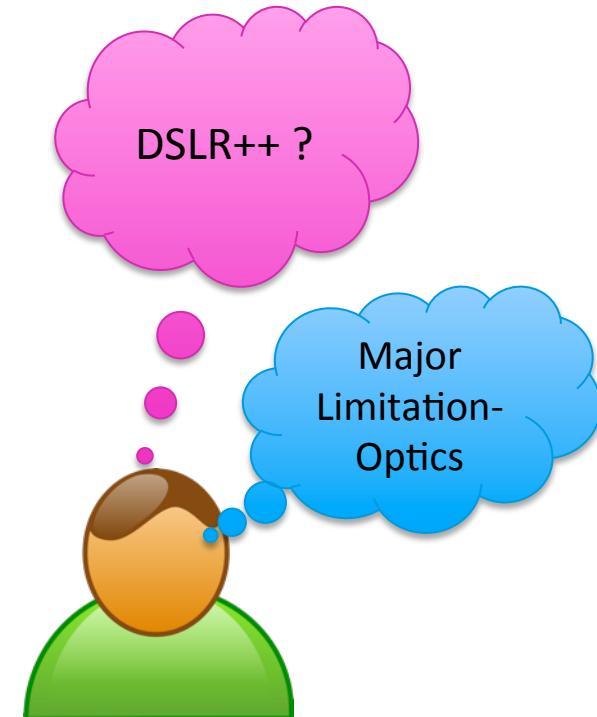
Innovative Mobile Cameras

- Sensors
- Location
- Connectivity/Cloud Power
- Compute Power

android.hardware.camera2

- Enables Professional Quality Photography
 - Think DSLR instead of Point-and-Shoot
- Enables Access to RAW Images
- Enables On-device processing of Camera Data
 - High Dynamic Range, Focus Stacking
- Enables New Use Cases combining Imaging with
 - Rich Sensor input
 - Inertial sensors, altitude et. al.
 - Compute Power
 - Multi-core CPU & GPU
 - Connectivity
 - Power of the Cloud & Access to Proximity (BLE, NFC)
 - Context
 - Location & User History

Computational Photography



Professional Quality Photography

- Fine grained control of the
 - Sensor
 - Flash
 - Lens
 - Image Signal Processing Pipeline
- Control on a per-frame basis, and deterministic behavior
- Processing Images at full resolution and full frame rate(30 fps)
- Meta data
 - Every frame is returned with the actual settings that it was taken with, and requested for.

Professional Camera

Computational Photography

Definition

- Computational photography refers to computational image capture, processing, and manipulation techniques that enhance or extend the capabilities of digital photography.

Professional Camera +
Post Processing on-device

Source: [Wikipedia](#)

Multi Shot HDR

Computational
Photography

- HDR = High Dynamic Range
- Multiple shots at different exposures and then blended together



Panoramic Stitching

Computational
Photography



Multiple Images with orientation information and fixed Exposure

Source: commons.wikimedia.org

Flash No-Flash Photography

Computational
Photography

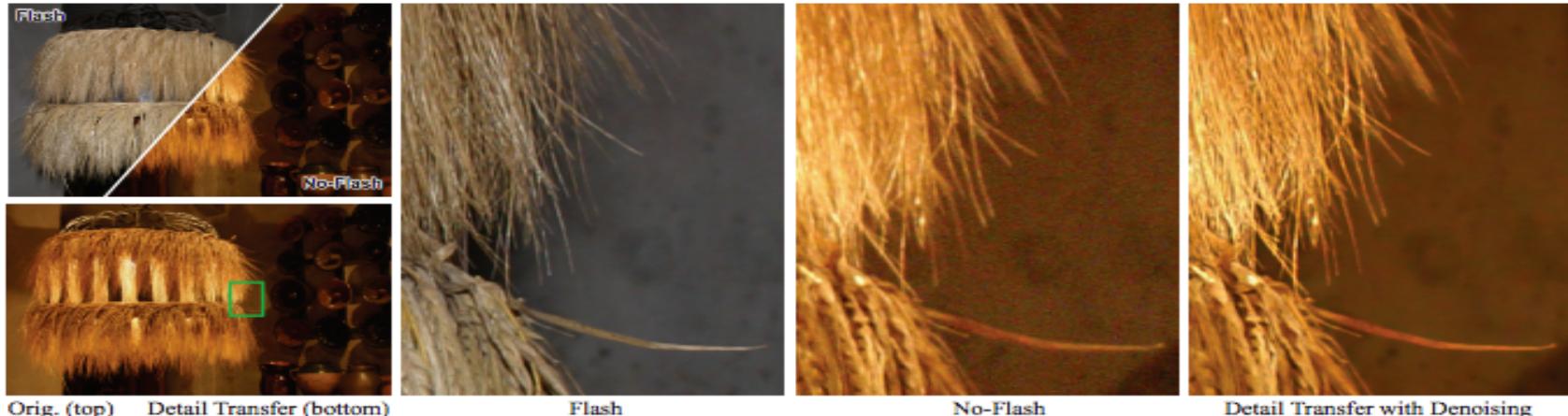
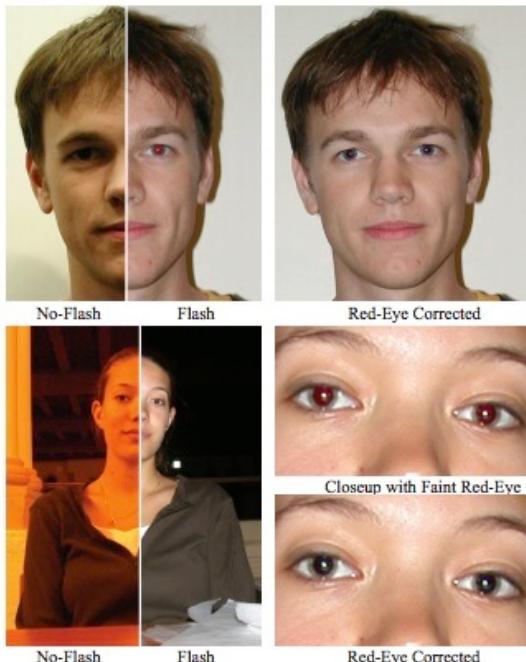


Figure 6: An old European lamp made of hay. The flash image captures detail, but is gray and flat. The no-flash image captures the warm illumination of the lamp, but is noisy and lacks the fine detail of the hay. With detail transfer and denoising we maintain the warm appearance, as well as the sharp detail.

Source: [Microsoft Research](#)

Flash No-Flash Photography

Computational
Photography



- Take 2 shots of the scene— one with flash on, the other with flash off
- Computationally blend the 2 frames together

Source: [Microsoft Research](#)

Focus Stacking

Computational
Photography



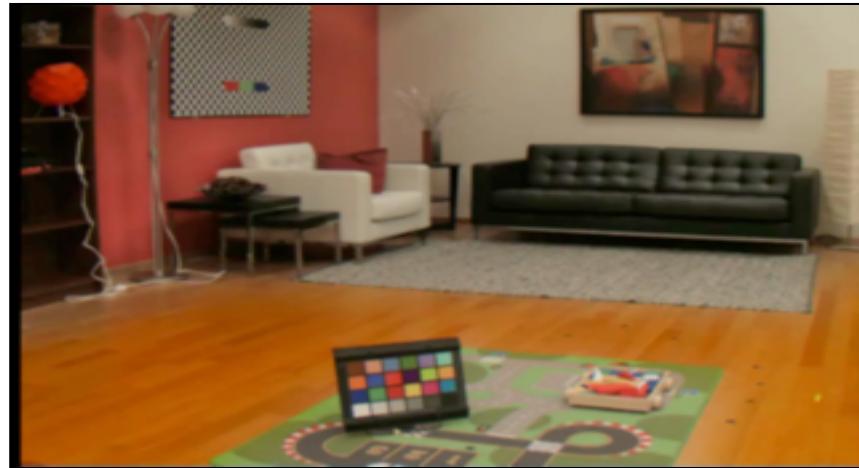
Focus Stacking for macro photography.
Also known as – all-in-focus photography

Source: [Wikipedia](#)

View Finder

Computational
Photography

- Today's smartphones enable features like
 - Touch to Expose & Touch to Focus
- Computational Photography can take it to the next level
 - Edit the frame in the view finder by selecting areas to sharpen, blur, brighten and maintain these during capture mode



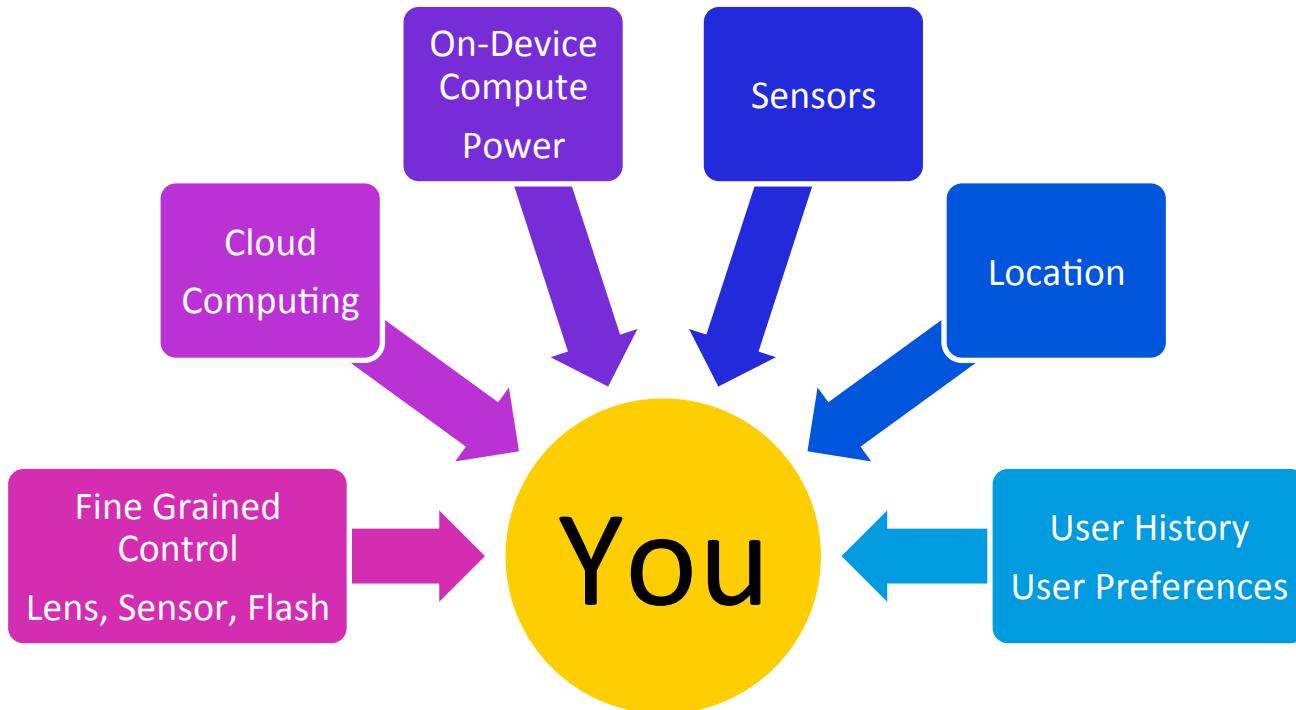
Source: [SIGGRAPH 2012](#)

Mobile Embedded Vision Applications

- Gesture Recognition
- Face Recognition, Object Tracking
- Visual Search
- 3D Mapping of the environment
- Augmented Reality

Natural User Interface
Companies are using the Front Facing Camera to provide gesture recognition solutions.

Innovative Mobile Camera ++



Camera2 Architecture

Limitations of android.hardware.Camera

3 Primary Operating Modes

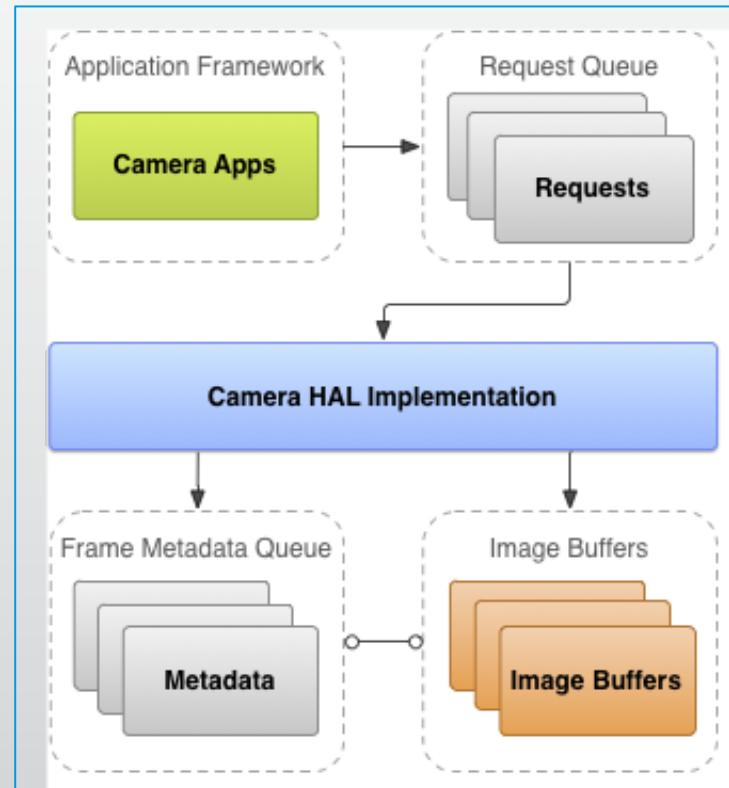
New Features – hard to implement

No Per-Frame Control

No Support for RAW

Minimal Metadata

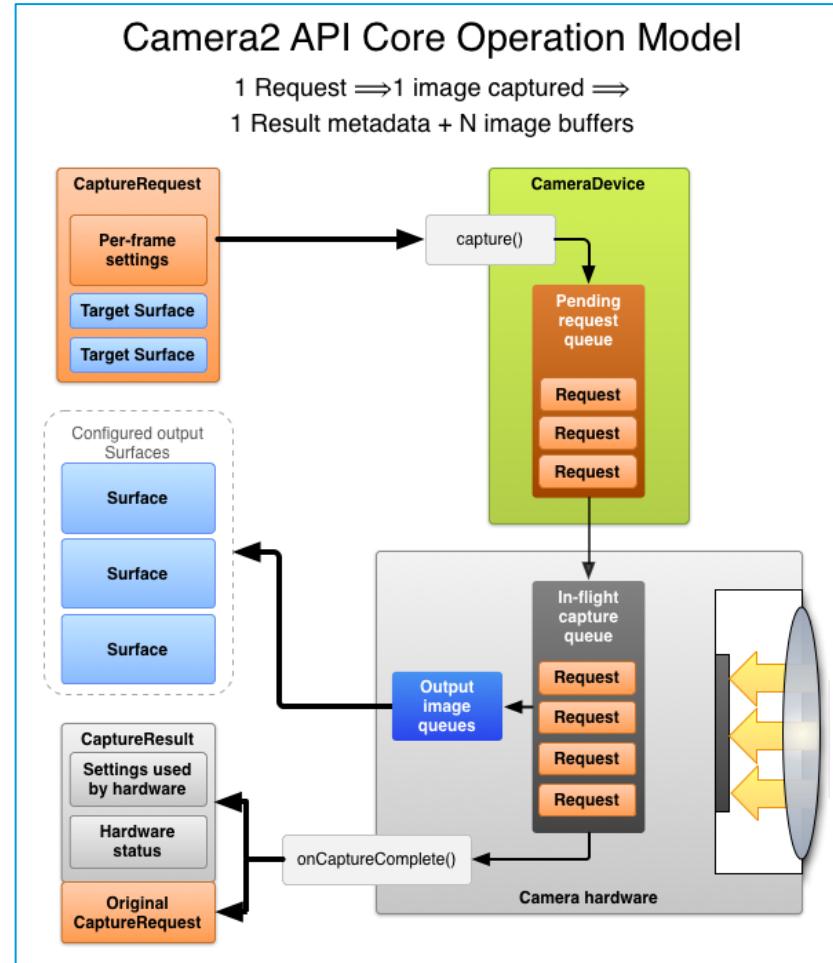
Custom Settings



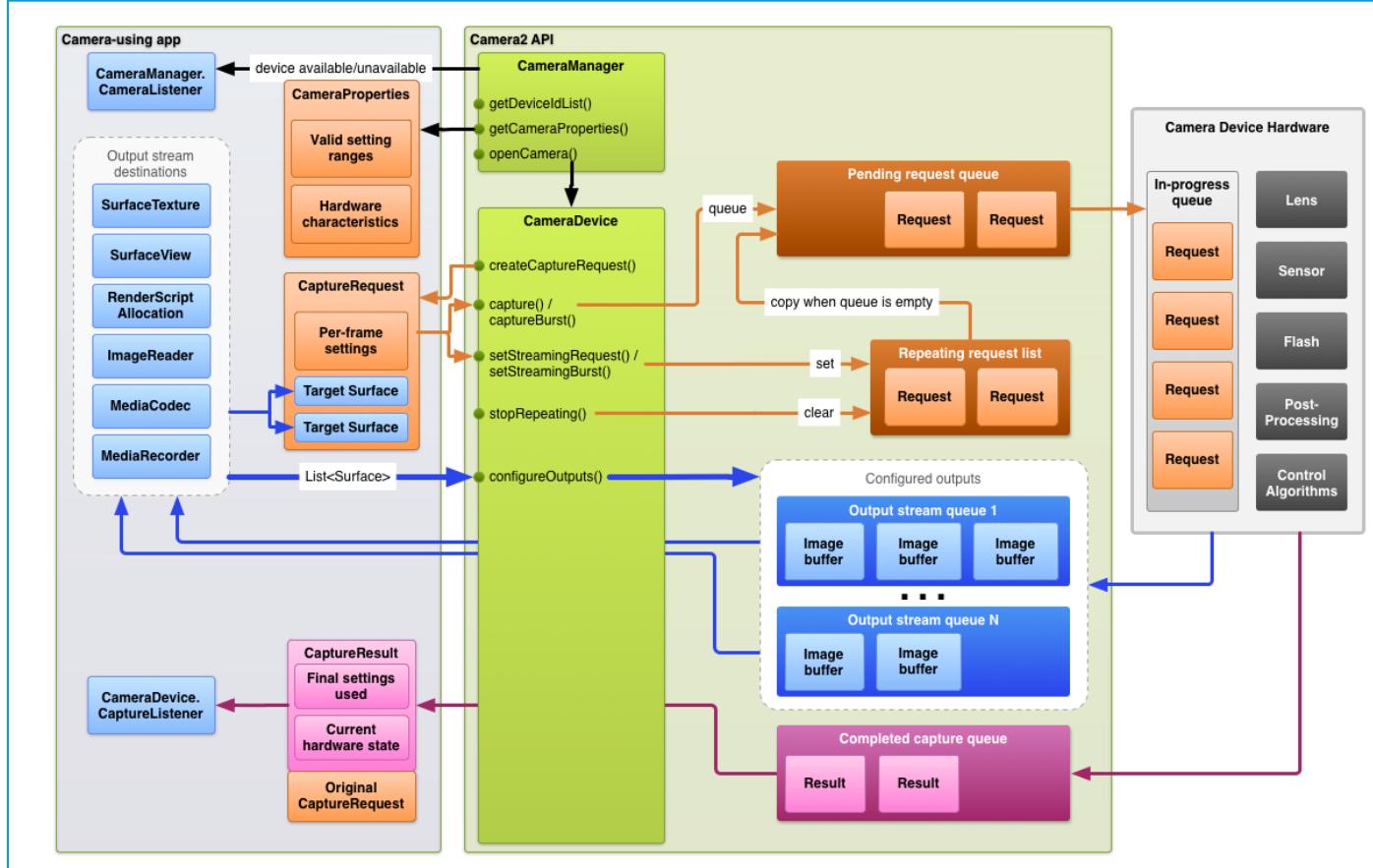
Camera2 Core Operation Model

- Per Frame Settings
- The Settings travel with individual requests and are no longer globally applied
- Image buffer delivered to all configured and requested Surfaces
- Metadata returned back to the application as requested separately
- Multiple requests and results in queue simultaneously

<http://source.android.com/devices/camera/camera.html>



Camera3 HAL Model



The Camera APIs

Android Lollipop

Camera related classes and interfaces from android.hardware deprecated
2 new packages

android.hardware.camera2

android.hardware.camera2.params

25+ classes, 1 Exception



Camera Model

From the Javadoc

“This package models a camera device as a pipeline, which takes in input requests for capturing a single frame, captures the single image per the request, and then outputs one capture result metadata packet, plus a set of output image buffers for the request. The requests are processed in-order, and multiple requests can be in flight at once. Since the camera device is a pipeline with multiple stages, having multiple requests in flight is required to maintain full frame-rate on most Android devices.”

Architecture walk through for primary use cases

Live
Preview

Still Capture
(JPG)

Still Capture
(RAW/DNG)

Still Capture
(Burst)

Video
Recording

Ready. Set. Go!

Figure out what's available

Concepts and Terms from Camera 2

- Camera Device
- Streams
- Capture Session
- Target Surfaces
- Capture Request
- Capture Result
 - Image Data
 - Metadata
 - Total Capture Result



1. How many Cameras do I have?

Query the System for capabilities

CameraManager provides information on the number of available cameras, or CameraDevices.

```
CameraManager manager =  
    (CameraManager) getSystemService(Context.CAMERA_SERVICE);  
  
String[] camids = manager.getCameraIdList();
```

2. What kind of a CameraDevice is it?

CameraCharacteristics provide static metadata for a given CameraDevice. This information is immutable for a given camera.

```
CameraCharacteristics characteristics =  
    manager.getCameraCharacteristics(camid);
```

CameraCharacteristics Keys

INFO_SUPPORTED_HARDWARE_LEVEL

3 Classes of Cameras are supported

Additionally, RAW may be supported

- ① FULL + RAW
- ② LIMITED + RAW



LEGACY

LIMITED

FULL

3. How many maximum streams does the system support?

3 Classes of Streams

- ① Processed & Non-Stalling
- ② Processed & Stalling
- ③ RAW (Bayer Domain)

The Maximum Number of Streams =
REQUEST_MAX_NUM_OUTPUT_PROC¹
+ REQUEST_MAX_NUM_OUTPUT_PROC_STALLING¹
+ REQUEST_MAX_NUM_OUTPUT_RAW¹

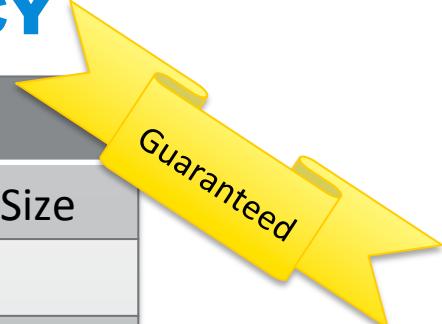
	PROCESSED & NON_STALLING (like YUV)	PROCESSED & STALLING (like JPEG)	RAW
FULL	>=3	>=1	>=0
LIMITED	>=2	>=1	>=0
LEGACY	>=2	>=1	0

¹ CameraCharacteristics.Key

Guaranteed Stream Configurations - LEGACY

Target 1		Target 2		Target 3	
Type	Max Size	Type	Max Size	Type	Max Size
PRIV ¹	Maximum ²				
JPEG	Maximum				
YUV	Maximum				
PRIV	Preview	JPEG	Maximum		
YUV	Preview	JPEG	Maximum		
PRIV	Preview	JPEG	Maximum		
PRIV	Preview	YUV	Preview		
PRIV	Preview	YUV	Preview	JPEG	Maximum

Still capture plus in-app processing.



Guaranteed

¹ Device Opaque Format, No Visibility into Format for Application

² Device Maximum Sensor Output

Guaranteed Stream Configurations - LIMITED

Target 1		Target 2		Target 3	
Type	Max Size	Type	Max Size	Type	Max Size
PRIV	Preview	PRIV	Record ¹		
PRIV	Preview	YUV	Record		
YUV	Preview	YUV	Record		
PRIV	Preview	PRIV	Record	JPEG	Record
PRIV	Preview	YUV	Record	JPEG	Record
YUV	Preview	YUV	Preview	JPEG	Maximum

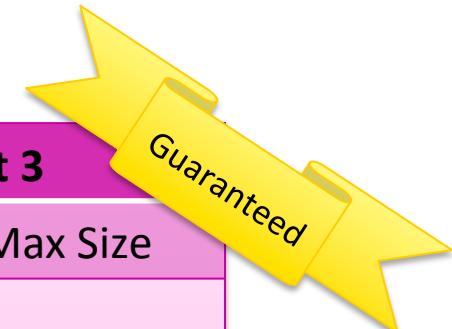


Guaranteed

¹ Maximum resolution supported by the device for media recording

Guaranteed Stream Configurations - FULL

Target 1		Target 2		Target 3	
Type	Max Size	Type	Max Size	Type	Max Size
PRIV	Preview	PRIV	Maximum		
PRIV	Preview	YUV	Maximum		
YUV	Preview	YUV	Maximum		
PRIV	Preview	PRIV	Preview	JPEG	Maximum
YUV	640x480	PRIV	Preview	YUV	Maximum
YUV	640x480	YUV	Preview	YUV	Maximum



Guaranteed Stream Configurations - RAW

Target 1		Target 2		Target 3	
Type	Max Size	Type	Max Size	Type	Max Size
RAW	Maximum				
PRIV	Preview	RAW	Maximum		
YUV	Preview	RAW	Maximum		
PRIV	Preview	PRIV	Preview	RAW	Maximum
PRIV	Preview	YUV	Preview	RAW	Maximum
YUV	In-app processing with simultaneous JPEG and DNG.				
PRIV	Preview	JPEG	Maximum	RAW	Maximum
YUV	Preview	JPEG	Maximum	RAW	Maximum

Guaranteed

4. How can these streams be configured?

```
CameraCharacteristics characteristics =  
cameraManager.getCameraCharacteristics(cameraId);  
StreamConfigurationMap configs = characteristics.get(  
    CameraCharacteristics.SCALER_STREAM_CONFIGURATION_MAP);
```

StreamConfigurationMap provides information on

- Supported Stalling and Non-Stalling Image Formats
- Supported Image Sizes for Supported Image Formats
- Supported Video Sizes and fps(frames per Second) Speeds
- Minimum Frame Duration and Stall Duration for supported formats and sizes
- Determine whether a given Surface is supported by this CameraDevice or not

FULL Camera Devices

30 FPS at
sensor max resolution
more than 20fps is required

Per Frame Control

android.sync.maxLatency
PER_FRAME_CONTROL

Manual Sensor Control
Manual Post Processing
Control

android.request.availableCapabilities

Arbitrary Cropping
Region

android.scaler.croppingType ==
FREEFORM

Minimum 3 Processed
Non-Stalling Output
Streams
(think YUV)

Stream Configurations
Available
as a Configuration Map

LIMITED and LEGACY Camera Device

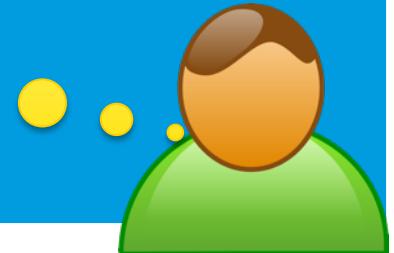
LEGACY

- NO Per- Frame Control
- NO Manual Sensor Control
- NO Manual Post Processing
- NO Arbitrary Cropping Regions
- No Stringent Performance Constraints
- Not a candidate for a Superior Camera Experience

LIMITED

- Query for individual capabilities!

That's
a lot of
checks!



LEGACY

LIMITED

FULL

- ① FULL + RAW
- ② LIMITED + RAW

Ready. Set. Go!

Open the Camera

1. Open a Full Camera Device

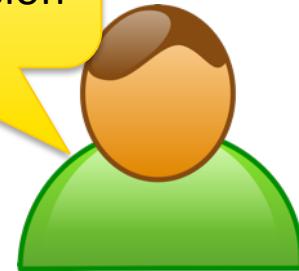
- ① Use the instance of the CameraManager to open a Camera passing in its camera id.
- ② Get a handle to a CameraDevice via the CameraDevice.StateCallback method when it is successfully opened.

```
CameraDevice.StateCallback mStateCallback = new  
    CameraDevice.StateCallback () {  
        ... } ;  
manager.openCamera(camid, mStateCallback, ...) ;
```

2. Create a Capture Session

- Expensive Operation
 - Involves allocation of buffers beneath the HAL
 - May take up to several hundred milliseconds
 - All modes cannot be supported concurrently
 - so Sessions may have to be torn down and setup again.

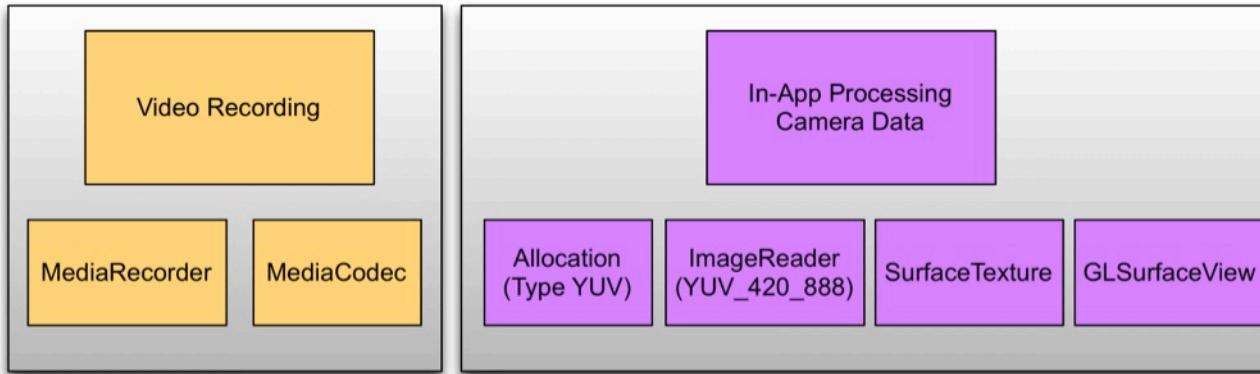
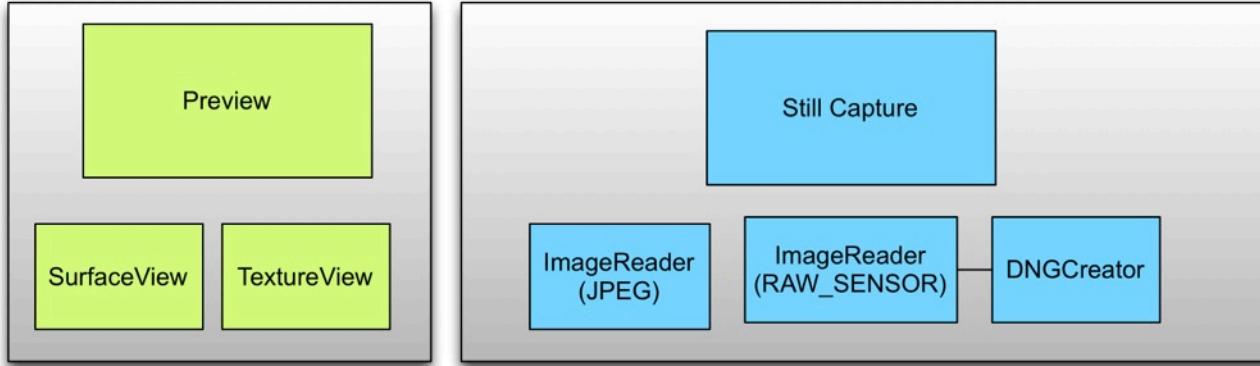
Minimize calls to
createCaptureSession



```
CameraCaptureSession.StateCallback mCaptureCallback=
new CameraCaptureSession.StateCallback () {...};
List<Surface> outputs = new ArrayList<Surface>(...);
createCaptureSession (outputs, mCaptureCallback, ...);
// a handle to a capture session in mCaptureCallback
method
```

Failure to create a capture session may result in exceptions being thrown

Different Target Output Streams



3. Create a Capture Request

- ① Pick the Capture Request Template
 - TEMPLATE_PREVIEW
 - TEMPLATE_STILL_CAPTURE
 - TEMPLATE_VIDEO_SNAPSHOT
 - TEMPLATE_RECORD
 - TEMPLATE_MANUAL

```
CaptureRequest.Builder mPreviewBuilder = mCameraDevice  
.createCaptureRequest(CameraDevice.TEMPLATE_PREVIEW);
```

3. Create a Capture Request – contd.

- ② Pick an appropriate target output
- e.g. TextureView for the correct width/height for Preview

```
mPreviewBuilder.addTarget(mPreviewSurface); //where  
mPreviewSurface is a Surface initialized to the width and  
height of the Preview Desired
```

- ③ Pick the Frequency of the Capture Request
- capture
 - captureBurst
 - setRepeatingRequest
 - setRepeatingBurst
 - stopRepeating
 - abortCaptures()

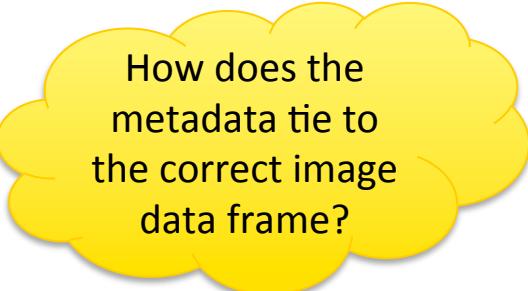
Ready. Set. Go!

Get your pictures

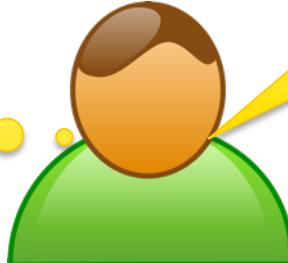
Receiving Image and Metadata

```
mSession.setRepeatingRequest(mPreviewBuilder.build(),  
    listener, backgroundHandler);
```

- The Image data is typically received in a listener associated with the Output Surface.
- The Metadata is received in the onCaptureCompleted callback method of CameraCaptureSession.CaptureCallback via the TotalCaptureResult object.



How does the metadata tie to the correct image data frame?



Through the Time Stamp!
`CaptureResult.SENSOR_TIMESTAMP`
& `image.getTimestamp()`

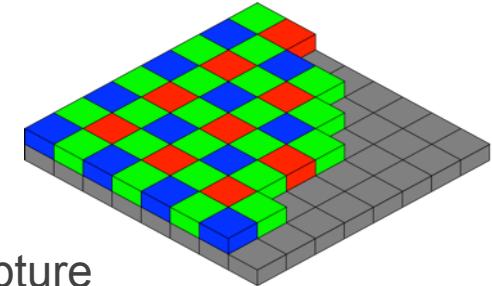
The Other Use Cases

Still Capture - JPEG

- ① Use `TEMPLATE_STILL_CAPTURE` for the `CaptureRequest`
- ② Create an `ImageReader` object with `format == ImageFormat.JPEG`
- ③ Pick `CameraCaptureSession.capture` for the frequency of capture
- ④ Image Data is received in the `ImageReader.onImageAvailableListener`
- ⑤ Metadata is received in the `CameraCaptureSession.CaptureCallback.onCaptureCompleted` callback

Still Capture - RAW

- ① Use TEMPLATE_STILL_CAPTURE for the CaptureRequest
- ② Create an ImageReader object with format == ImageFormat.
[ImageFormat.RAW_SENSOR](#)
- ③ Pick CameraCaptureSession.capture for the frequency of capture
- ④ Image Data is received in the ImageReader.onImageAvailableListener
- ⑤ Metadata is received in the
CameraCaptureSession.CaptureCallback.onCaptureCompleted callback
- ⑥ [Use an instance of DngCreator to convert RAW to DNG \(Digital Negative\)](#)



```
final File dngFile = new File(path, filename);
outs = new FileOutputStream(dngFile);
if (mRawTotalResult != null) {
    DngCreator d = new DngCreator(characteristics, mRawTotalResult);
    d.writeImage(outs, image);
}
```

Still Capture - BURST

- ① Use TEMPLATE_STILL_CAPTURE for the CaptureRequest
- ② Create an ImageReader object with format == ImageFormat.ImageFormat.JPEG
- ③ Pick [CameraCaptureSession.captureBurst](#) for the frequency of capture
- ④ Image Data is received in the ImageReader.onImageAvailableListener
- ⑤ Metadata is received in the CameraCaptureSession.CaptureCallback.onCaptureCompleted callback



Video Recording

- ① Use `TEMPLATE_RECORD` for the `CaptureRequest`
- ② Create a `MediaRecorder` Object and use its surface as the output stream
- ③ Pick `CameraCaptureSession.setRepeatingRequest` for the frequency of capture
- ④ Image Data is recorded in the Image file set via `MediaRecorder.setOutputFile` call.
- ⑤ Metadata is received in the `CameraCaptureSession.CaptureCallback`.
`onCaptureCompleted` callback



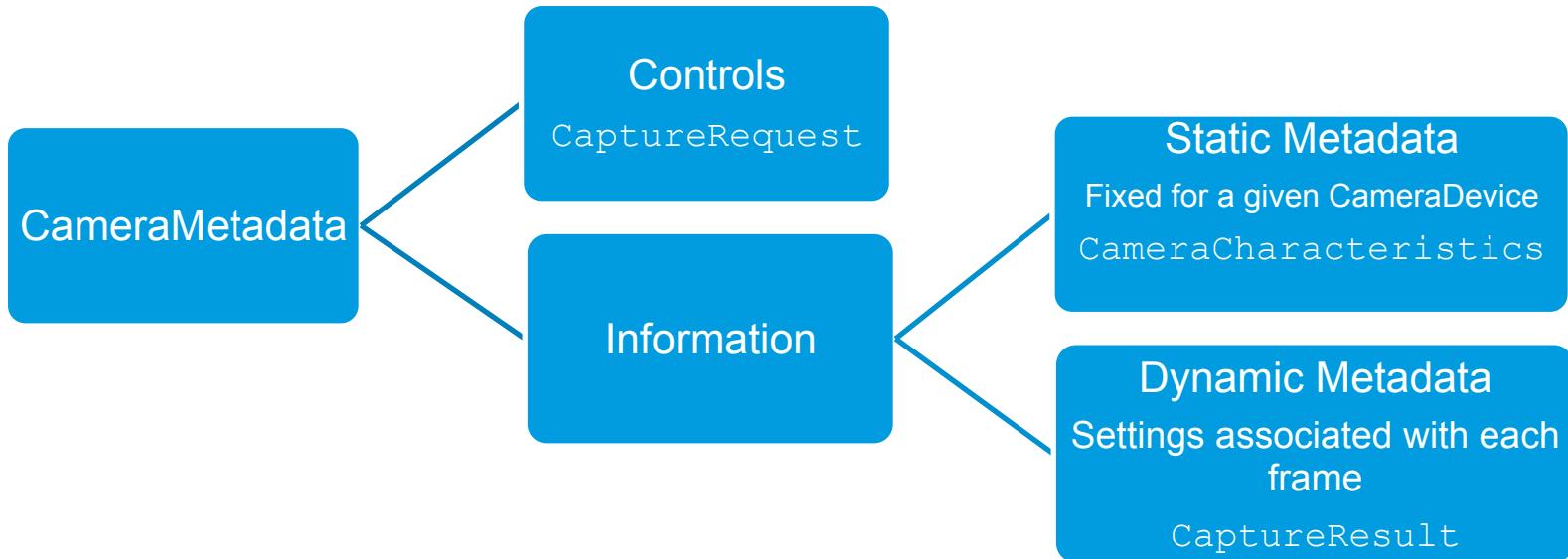
In-App Processing of Camera Data

- ① Use an appropriate Template for the CaptureRequest; including TEMPLATE_MANUAL if the use case so demands
- ② Create an object like Allocation, Renderscript, ImageReader (with Format YUV_420_888), SurfaceTexture, or GLSurfaceView that provides a Surface
- ③ Pick an appropriate frequency for capture
- ④ Image Data can be found in an appropriate callback for the surface chosen
- ⑤ Metadata is received in the CameraCaptureSession.CaptureCallback.onCaptureCompleted callback

Some use cases for In-App Processing

- Use Cases
 - Filters
 - Computational Photography Use cases
 - Custom Post Processing of the Imaging Pipeline
 - Computer Vision Applications
- Uses
 - Compute Framework
 - Renderscript, OpenCL
 - Low level Graphics APIs
 - OpenGL ES

And then there was Metadata....



Key Value Pairs for Hardware Control (Lens, Sensor, Flash), Image Processing Pipeline, Control Algorithms and Output Buffers

Fine Grained Control

a.k.a “The Devil is in the Details”



Manual Sensor Control

Sensor

Sensitivity
(Gain/ISO)

Sensor
Exposure Time

Sensor Frame
Duration

Flash (if present)

Flash
Mode

Lens (if adjustable)

Lens Aperture

Lens Filter Density

Lens Focal Length

Lens Focus Distance

Black Level Lock

Black
Level
Lock
On/Off

Manual 3A Control

- A Camera Request can override the 3A Algorithm Controls
 - Auto Exposure (AE)
 - Auto Focus(AF)
 - Auto White Balance (AWB)
- There is an overall control for all Auto Algorithms
 - android.control.mode

- When set to OFF, application has full control of the pipeline
- When set to AUTO, the individual controls are in effect



Application Controlled (Auto) Algorithms

android.control.mode= AUTO

android.control.a*mode = OFF

Auto Focus

- Application uses android.lens.focusDistance to control the Lens

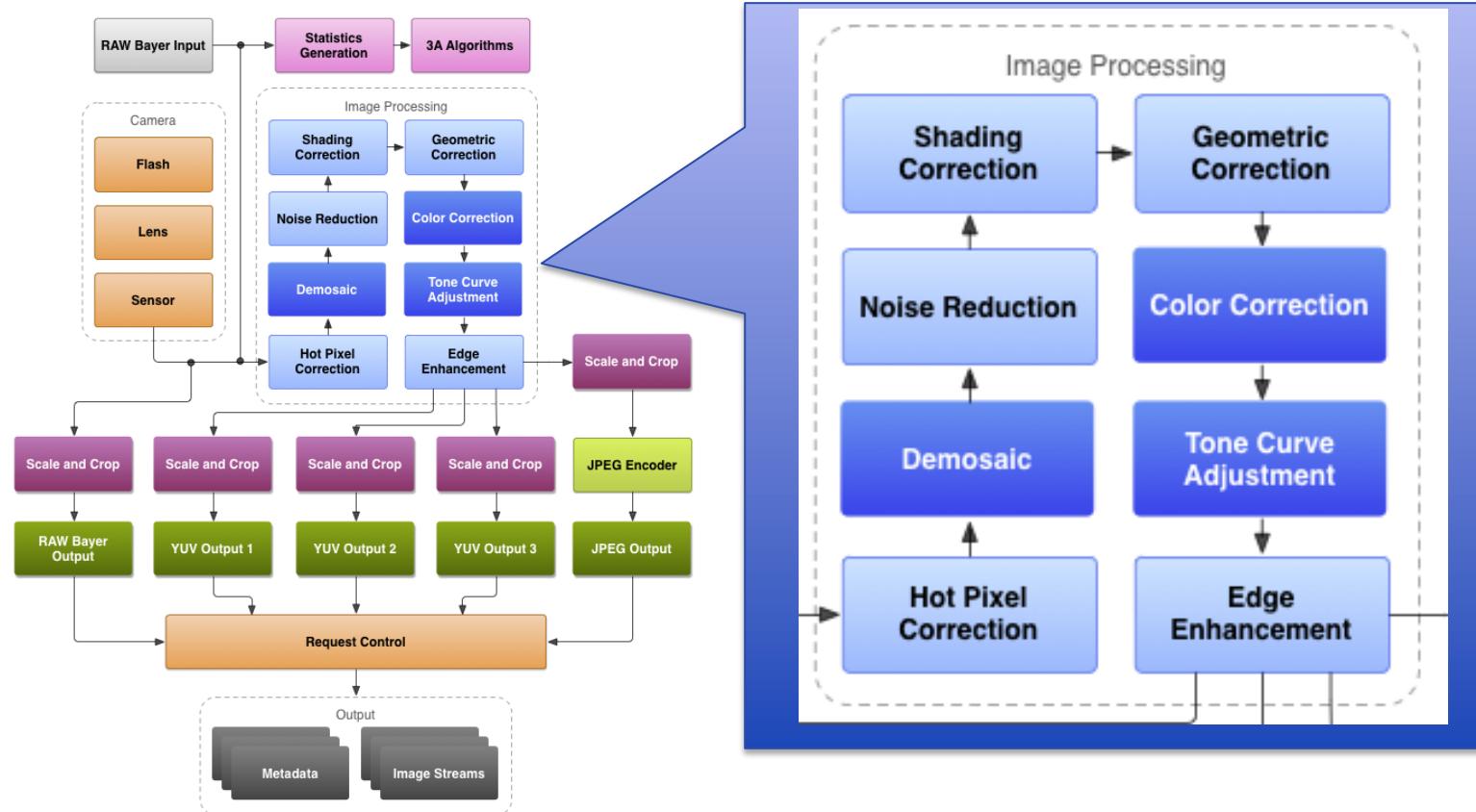
Auto Exposure

- Application uses android.sensor.exposureTime, android.sensor.sensitivity and android.sensor.frameDuration + android.flash.* to control exposure

Auto White Balance

- Application uses android.colorCorrection.transform & android.colorCorrection.gains to control the white balance

Manual Post Processing



<http://source.android.com/devices/camera/camera.html>

Image Processing Pipeline

Hot Pixel
Mode

Black Level
Correction

Demosaic

Noise
Reduction
Mode

Shading Mode
(lens shading)

Color
Correction
Mode

Tone Curve
Adjustment

Edge
Enhancement

Manual Post Processing

Manual Tonemap control

- android.tonemap.*

Manual White Balance control

- android.colorCorrection.*

Manual Lens Shading map control

- android.shading.*

Manual aberration correction control (if supported)

- android.colorCorrection.aberration.*

The other camera2 package **android.hardware.camera2.params**

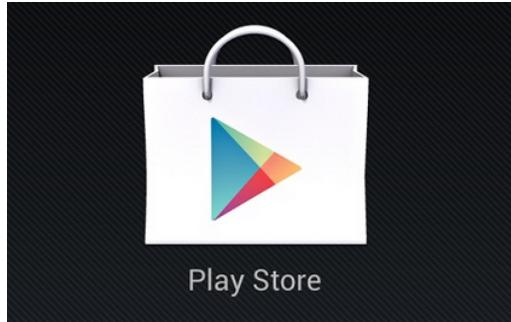
- BlackLevelPattern
- ColorSpaceTransform
- Face
- LensShadingMap
- MeteringRectangle
- RggbChannelVector
- StreamConfigurationMap
- TonemapCurve

Play Store Feature Filters

Android Manifest <uses-feature>

Play Store Feature Filters

- android.hardware.camera.hardware_level.full
- android.hardware.camera.capability.raw
- android.hardware.camera.capability.manual_sensor
- android.hardware.camera.capability.manual_post_processing

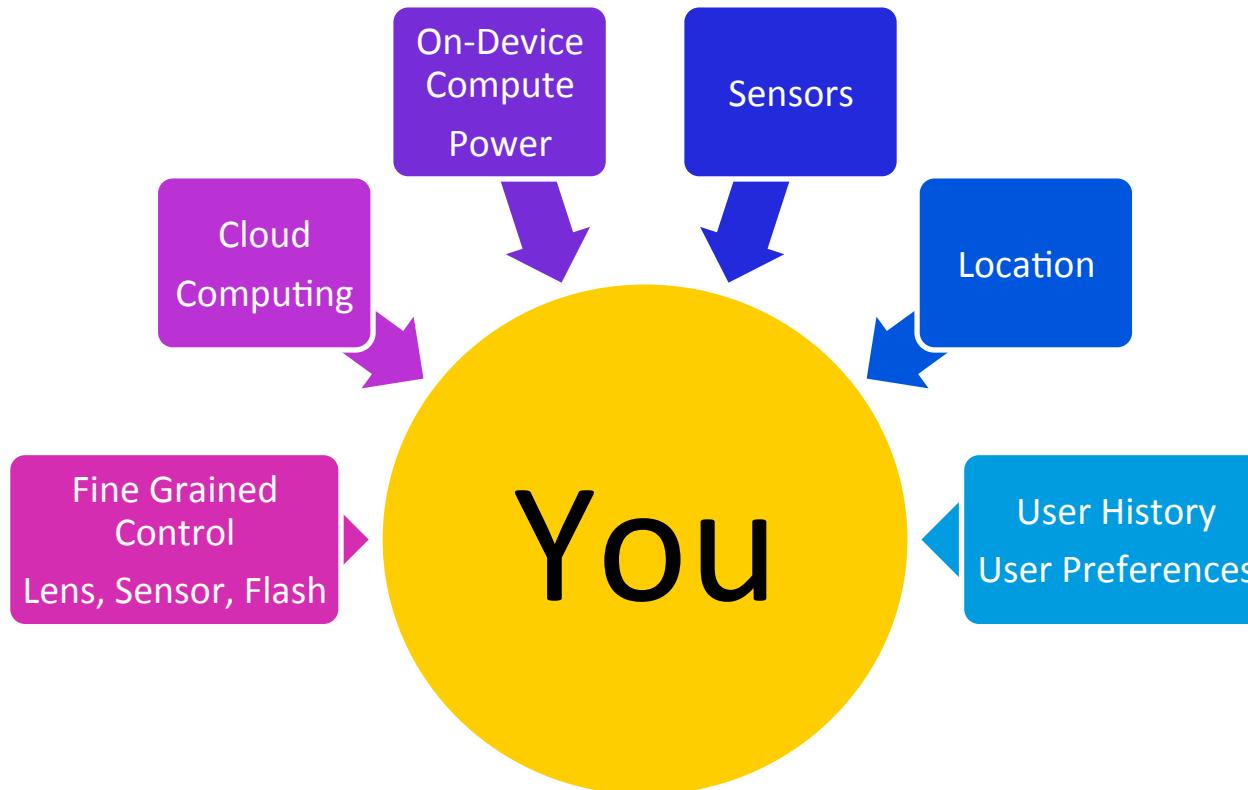


[See Also: Versioning](#)

Demo Time

Summary

It's all about you and your power tools!



Q & A

and **THANK YOU** for your time.

Balwinder Kaur

balwinder.kaur@onsemi.com
balwinder.x.kaur@gmail.com

<http://www.slideshare.net/lbk003> <https://github.com/lbk003>