ESAP – 2018

Image Processing Basics

Semester 1, 2018

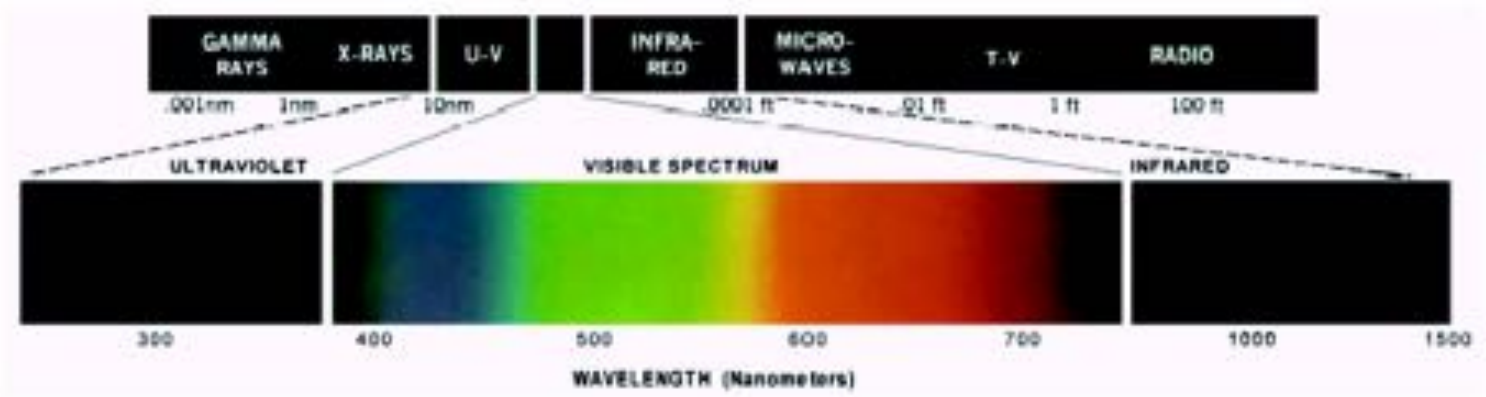
IMAGE REPRESENTATION

A Quick Introduction

LIGHT AND THE ELECTROMAGNETIC SPECTRUM

Light is just a particular part of the electromagnetic spectrum that can be sensed by the human eye

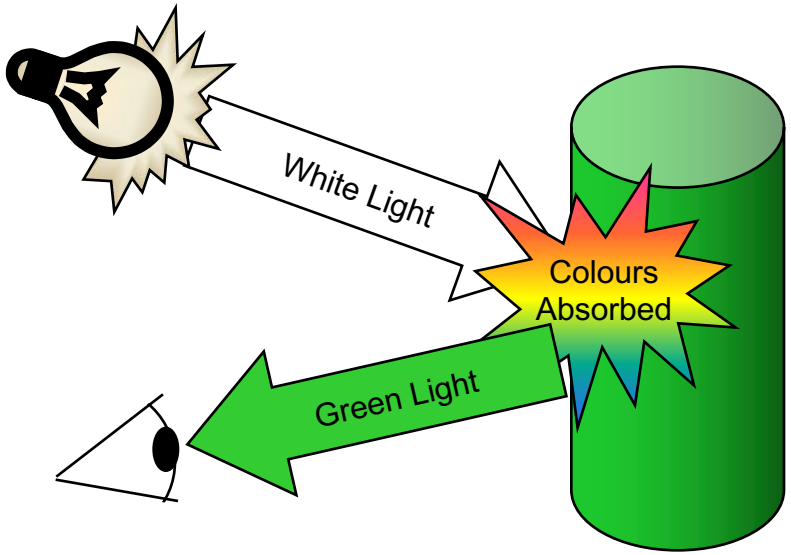
The electromagnetic spectrum is split up according to the wavelengths of different forms of energy



REFLECTED LIGHT

The colours that we perceive are determined by the nature of the light reflected from an object

For example, if white light is shone onto a green object most wavelengths are absorbed, while green light is reflected from the object



SAMPLING, QUANTISATION AND RESOLUTION

In the following slides we will consider what is involved in capturing a digital image of a real-world scene

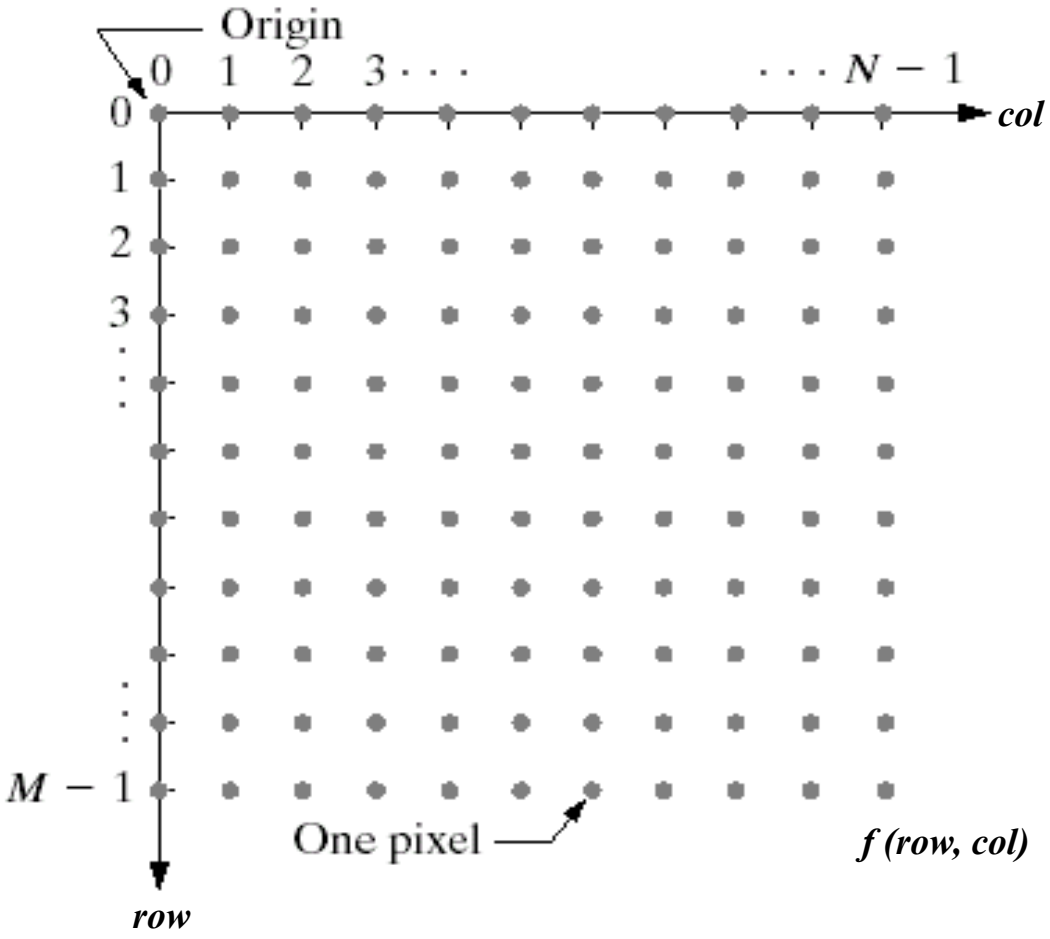
- Image sensing and representation
- Sampling and quantisation
- Resolution

IMAGE REPRESENTATION

Before we discuss image acquisition recall that a digital image is composed of M rows and N columns of pixels each storing a value

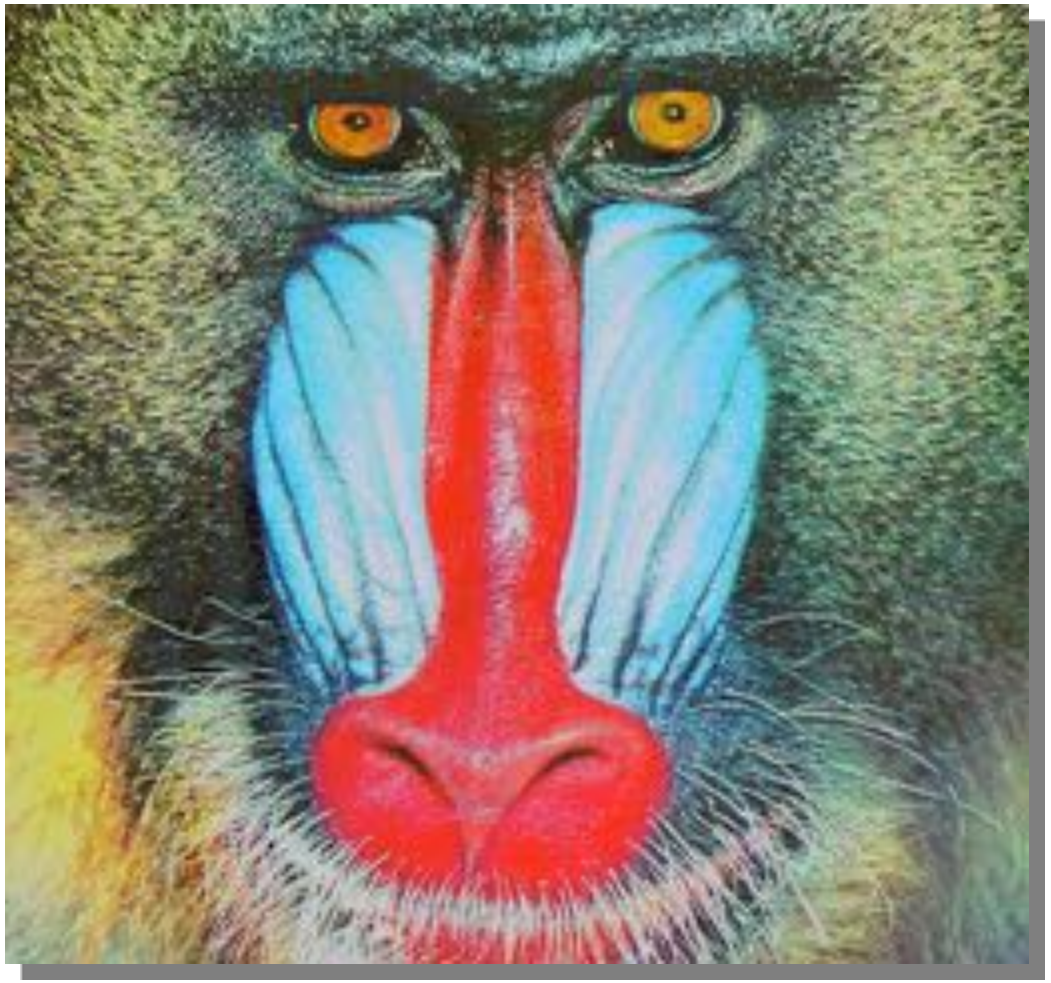
Pixel values are most often grey levels in the range 0-255(black-white)

We will see later on that images can easily be represented as matrices



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

COLOUR IMAGES



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

COLOUR IMAGES

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

COLOUR IMAGES – RGB

RGB

[RED+GREEN+BLUE]

- Additive colors
- Vibrant tones
- Used by monitors for displays
- 3 Channels : Red, Green, Blue
- Smaller files size

CMYK

[CYAN, MAGENTA, YELLO, BLACK]

- Subtractive colors
- Cool tones
- Only way for print media
- 4 channels : Cyan, Magenta, Yellow, Black
- Easier to color treat

red	*green*	*blue*	
255	0	0	*red*
0	255	0	*green*
0	0	255	*blue*
0	0	0	*black*
100	100	100	*dark gray*
255	255	255	*white*
255	255	0	*yellow*
255	0	255	*magenta*
9	90	166	*this color*

Some color values

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

IMAGE FILE FORMATS

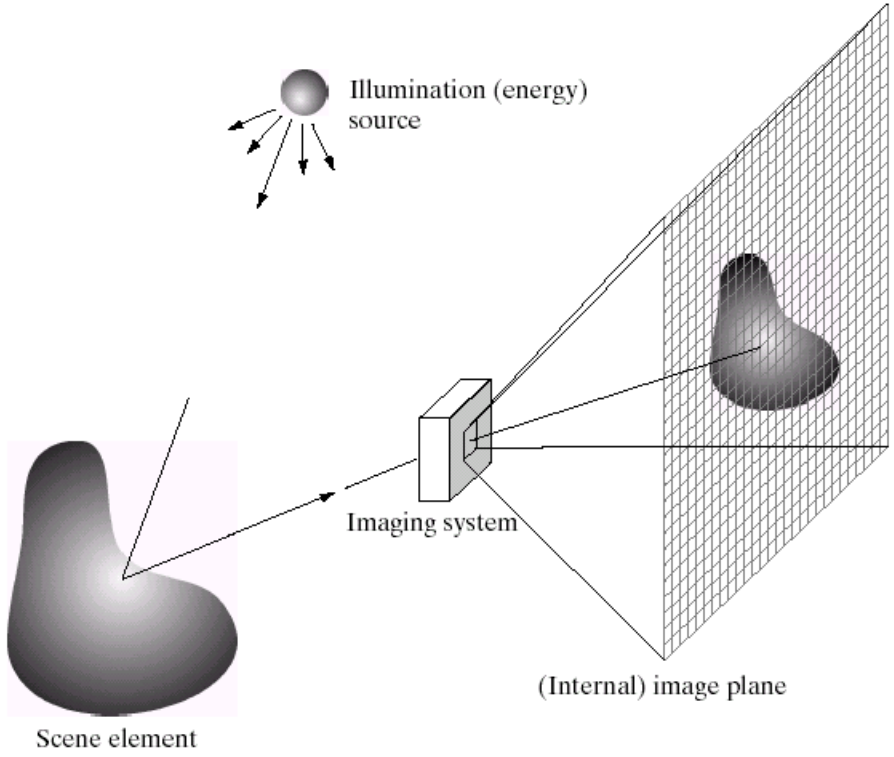
- **Image file formats** are standardized means of organizing and storing digital images.
  - Image files are composed of digital data in one of these formats that can be rasterized for use on a computer display or printer.
  - An image file format may store data in uncompressed, compressed, or vector formats.
  - Once rasterized, an image becomes a grid of pixels, each of which has a number of bits to designate its color equal to the color depth of the device displaying it.
- The **size of an image file** is dependent on the number of pixels in the image and the color depth (bits per pixel).
  - Primary image data has 3-color values for each picture element; the best known format uses 8-bit R (red), G (green) and B (blue) values, but there are many alternative color spaces & image formats.
  - Images can be compressed in various ways, however. A compression algorithm stores either an exact representation (lossless) or an approximation of the original image (lossy) in a smaller number of bytes that can be expanded back to the uncompressed form with a corresponding decompression algorithm.
  - Images with the same number of pixels & color depth can have very different compressed file size.
- JPEG (Joint Photographic Experts Group) is the best known lossy compression method; JPEG-compressed images are usually stored in the **JFIF** (JPEG File Interchange Format) file format and most image data stored on the Internet is in JPEG form.

IMAGE ACQUISITION

A Quick Introduction

IMAGE ACQUISITION

Images are typically generated by *illuminating* a *scene* and absorbing the energy reflected by the objects in that scene



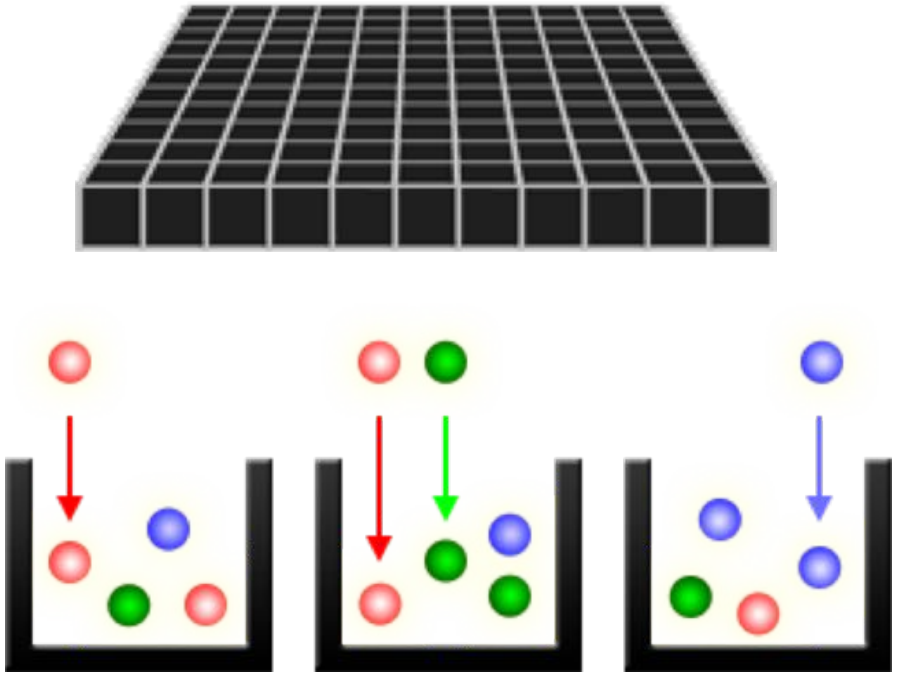
– Typical notions of illumination and scene can be way off:
  - X-rays of a skeleton
  - Ultrasound of an unborn baby
  - Electro-microscopic images of molecules

Images taken from Gonzalez & Woods, DigitalImage Processing (2002)

DIGITAL CAMERA SENSORS #1

MONO PIXEL ARRAY

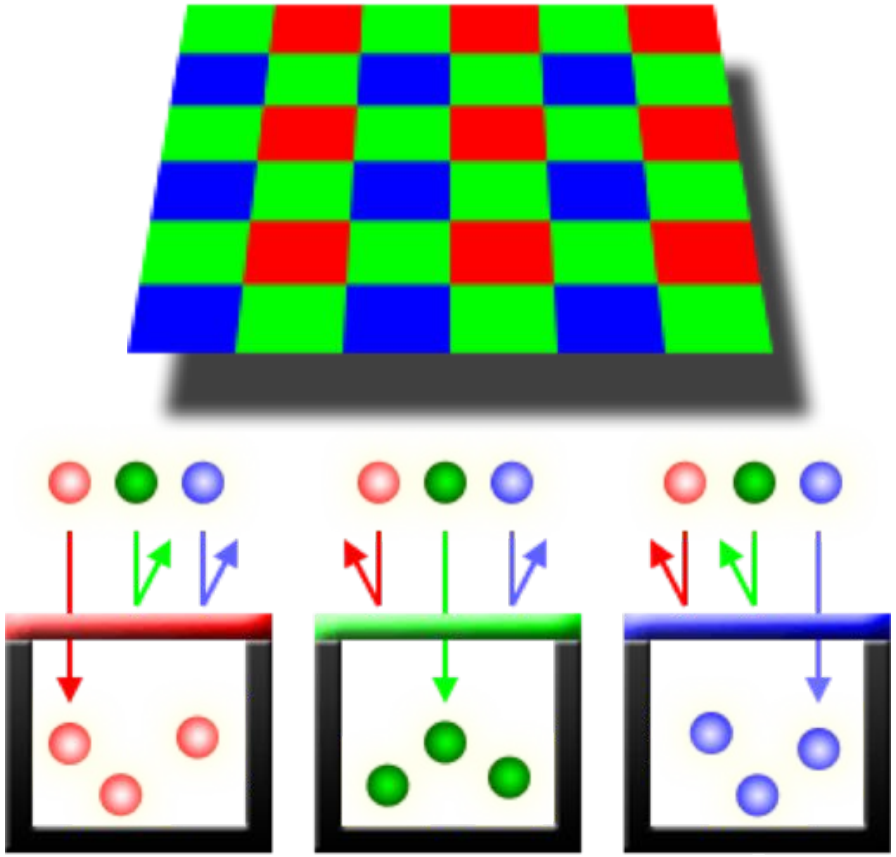
- A digital camera uses an array of millions of tiny light cavities or "photosites" to record an image.
  - When you press your camera's shutter button and the exposure begins, each of these is uncovered to collect and store photons.
  - Once the exposure finishes, the camera closes each of these photosites, and then tries to assess how many photons fell into each.
  - The relative quantity of photons in each cavity are then sorted into various intensity levels, whose precision is determined by bit depth (0 - 255 for an 8-bit image).



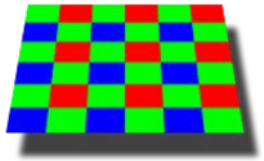
DIGITAL CAMERA SENSORS #2

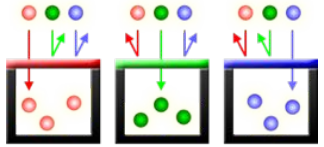
COLOR PIXEL ARRAY

- However, the previous set-up would only create grayscale images, since these cavities are unable to distinguish how much they have of each color.
- To capture color images, a filter has to be placed over each cavity that permits only particular colors of light.
- Virtually all current digital cameras can only capture one of three primary colors in each cavity, and so they discard roughly 2/3 of the incoming light.
- As a result, the camera has to approximate the other two primary colors in order to have full color at every pixel.
- The most common type of color filter array is called a "Bayer array," shown opposite.



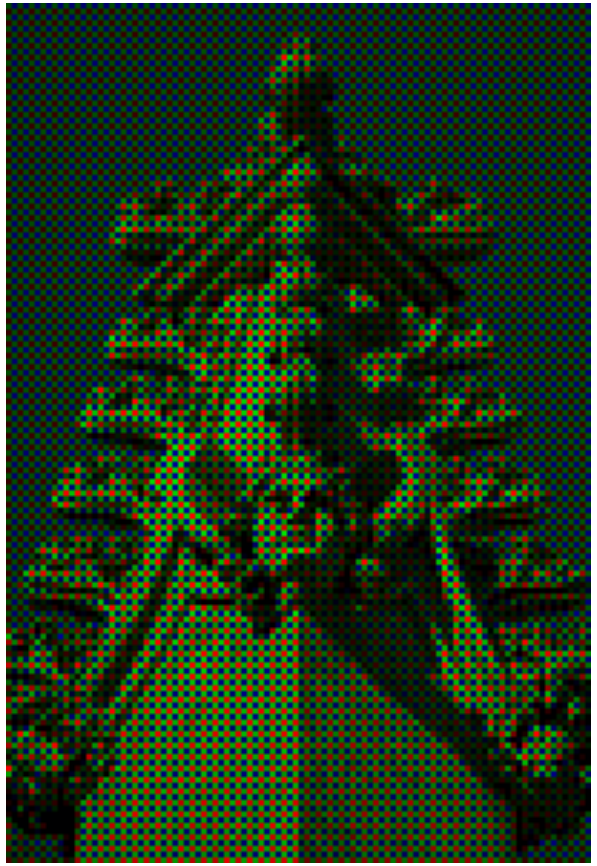
DIGITAL CAMERA SENSORS #3



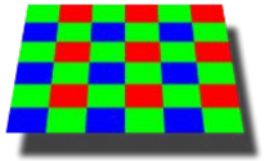


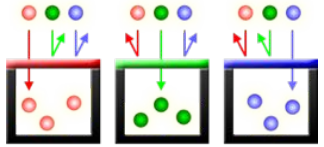
BAYER IMAGE

- A Bayer array consists of alternating rows of red-green and green-blue filters.
- Notice how the Bayer array contains twice as many green as red or blue sensors.
- Each primary color does not receive an equal fraction of the total area because the human eye is more sensitive to green light than both red and blue light.
- Redundancy with green pixels produces an image which appears less noisy and has finer detail than could be accomplished if each color were treated equally.
  - Noise in the green channel is less than for the other two primary colors simply because there are twice as many pixels.
- Bayer's technique is > 30 years old – clearly a robust engineering approximation!



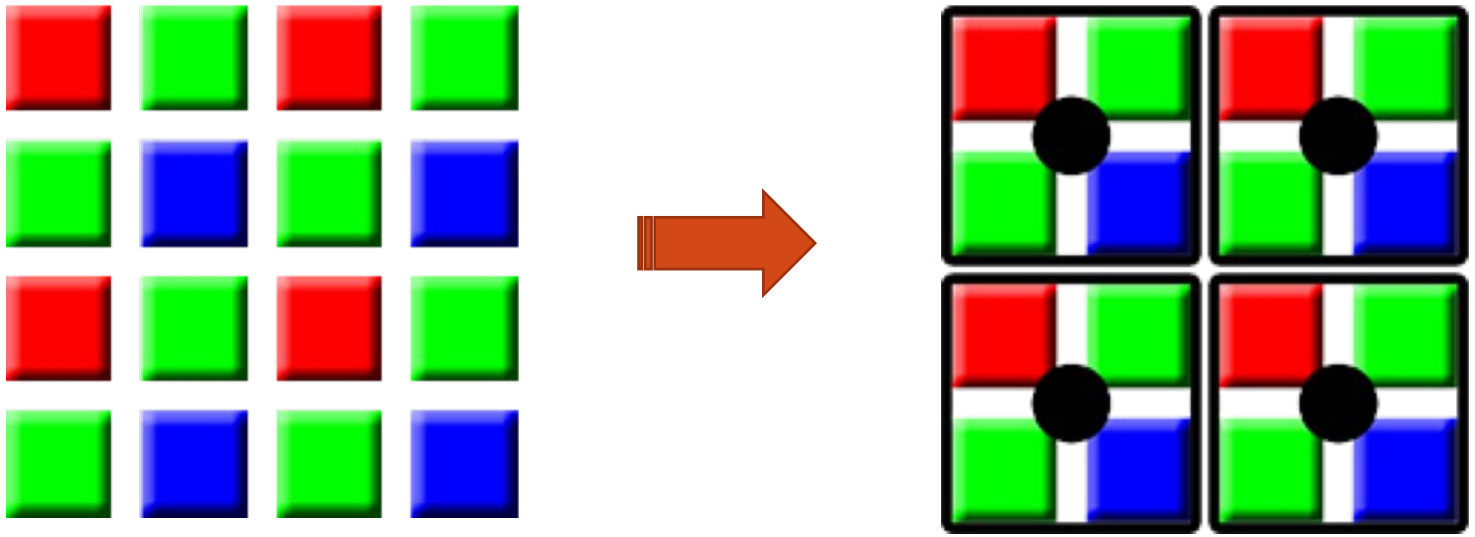
DIGITAL CAMERA SENSORS #4



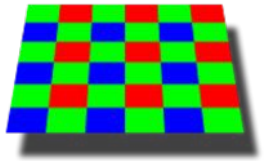


DEMOSAICING THE BAYER IMAGE

- Bayer "demosaicing" is the process of translating this Bayer array of primary colors into a final image which contains full color information at each pixel.
- How is this possible if the camera is unable to directly measure full color?
- One way of understanding this is to instead think of each 2x2 array of red, green and blue as a single full color cavity.

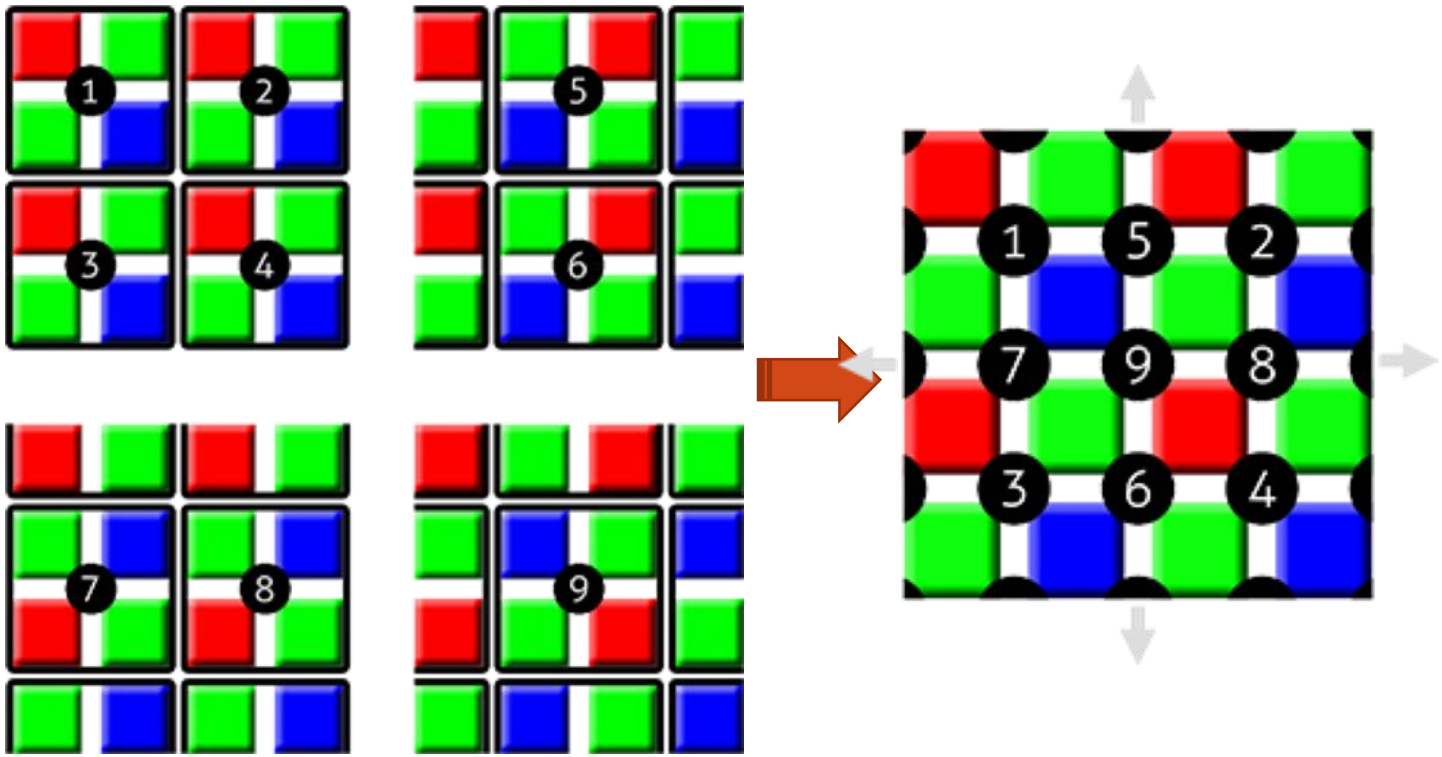


DIGITAL CAMERA SENSORS #5



DEMOSAICING THE BAYER IMAGE #2

- This would work fine, however most cameras take additional steps to extract even more image information from this color array.
  - If the camera treated all of the colors in each 2x2 array as having landed in the same place, then it would only be able to achieve half the resolution in both the horizontal and vertical directions.
  - On the other hand, if a camera computed the color using several overlapping 2x2 arrays, then it could achieve a higher resolution than would be possible with a single set of 2x2 arrays.
  - The following combination of overlapping 2x2 arrays could be used to extract more image information.
- In fact there is a large body of research literature on different ways to implement the demosaicing process.



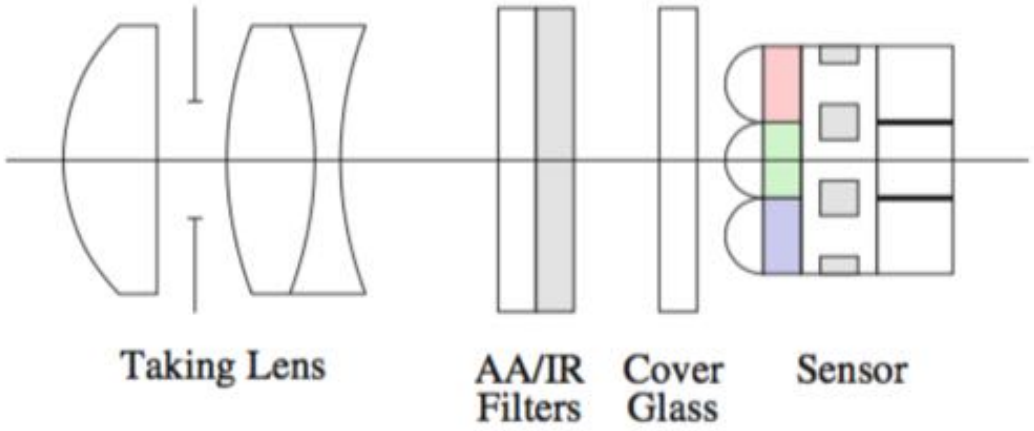
DIGITAL CAMERA TECHNOLOGIES

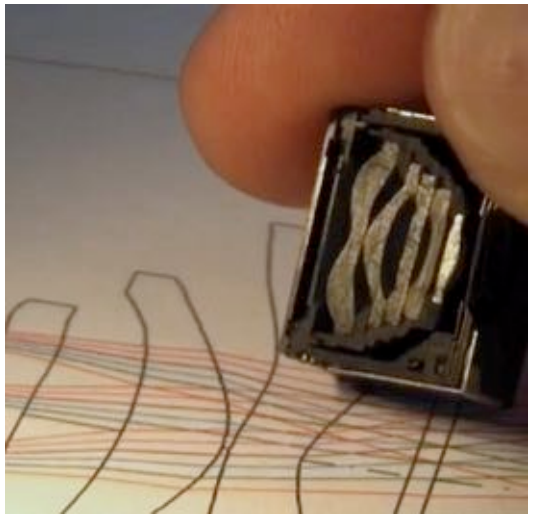
Where Most Images Come From …

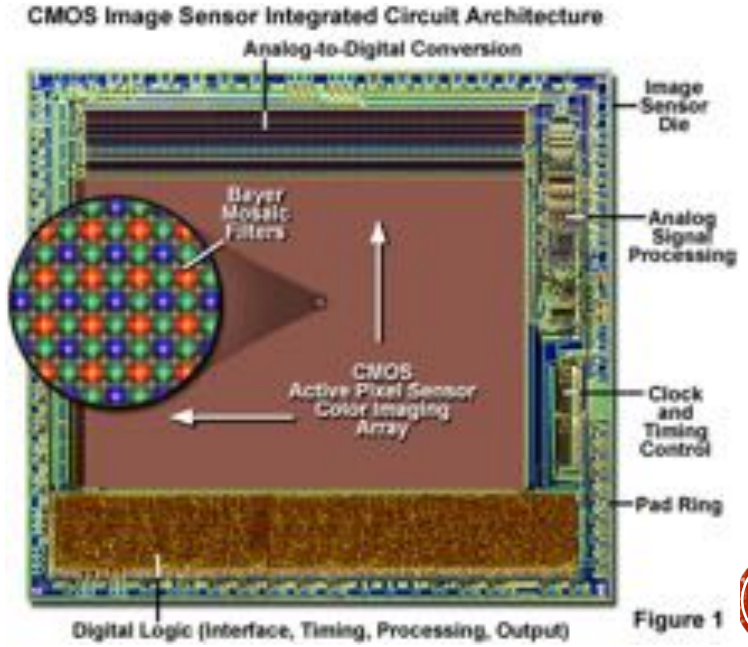
DIGITAL CAMERA TECHNOLOGIES #1

BASICS – THE OPTICAL IMAGE PATH #1

- Multi-Element Lens
  - Typically at least 5-element
  - Telecentric (see reading #1)
  - Small Point-Spread Function (PSF)
- Anti-Aliasing Filter
  - Removes High-Frequency (Spatial) Artifacts
- Infrared Cutoff Filter
  - Silicon is sensitive to NIR
  - NIR focus is different to Visible
- Sensor
  - Bayer Color Filter Array (CFA)
  - Back-Illuminated





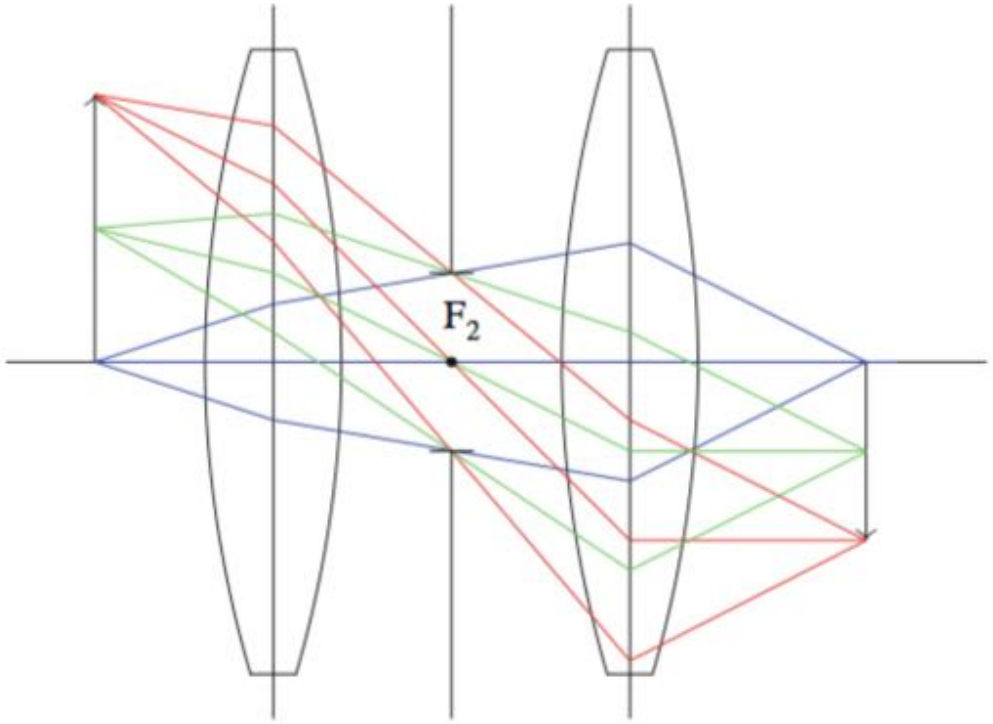


DIGITAL CAMERA TECHNOLOGIES #2

BASICS – THE OPTICAL IMAGE PATH #2

- Telecentric Lens
  - Ensures that projection from 3D space to 2D sensor is Orthographic
  - Objects at different distances appear with a constant magnification
  - Most lenses are entocentric – meaning that distant objects have lower magnification
  - Practical telecentric multi-lens systems are challenging to design but reduce distortions (lens shading) at the sensor
  - Modern smartphone lens systems are very close to ideal – quite an achievement!

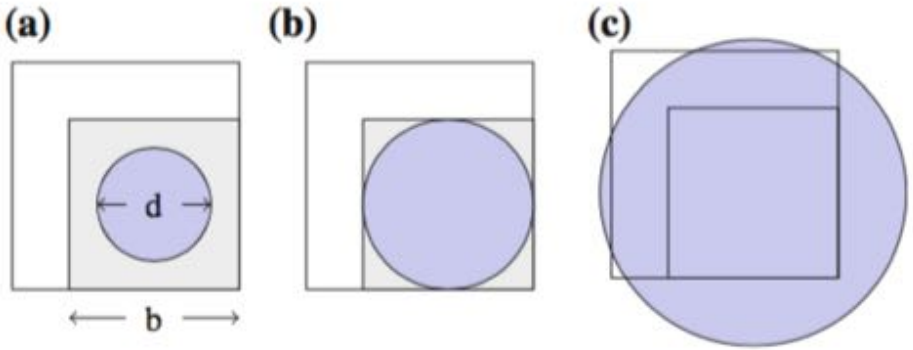
A 2-Element Telecentric Lens

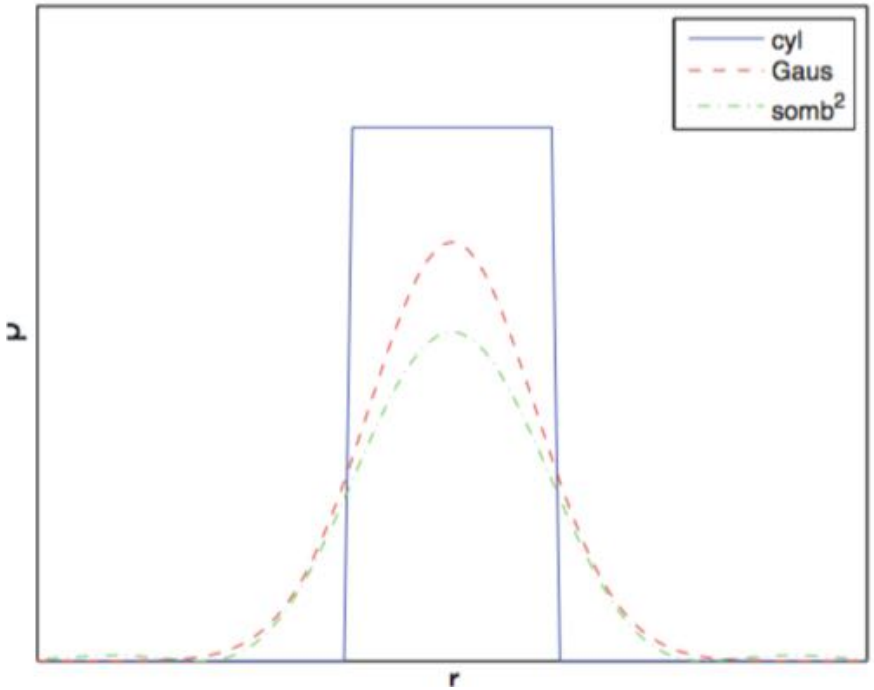


DIGITAL CAMERA TECHNOLOGIES #3

BASICS – THE OPTICAL IMAGE PATH #3

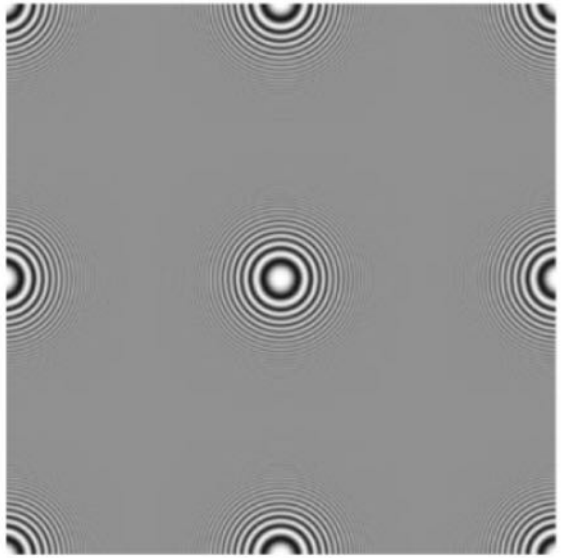
- Point-Spread Function (PSF)
  - This is the image projected onto the sensor from a 'point' light source
  - Ideally it would produce a similar sized 'point' on the sensor;
  - In reality the light (luminance) is spread over a larger areas due to the imperfect lens
- This is modelled with a Gaussian (or equivalent) luminance distribution – bottom figure
- The PSF of the lens should be matched to the pixel size of the sensor; top figure shows (a) PSF too narrow; (b) just right and (c) too wide
- Lens material and shape affects PSF; multi-lens system is even more complex; typically PSF is measured empirically.



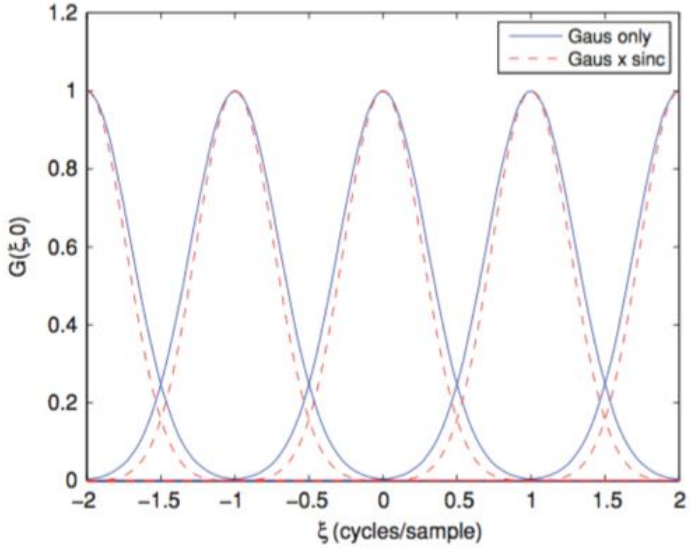


DIGITAL CAMERA TECHNOLOGIES #4

BASICS – THE OPTICAL IMAGE PATH #4



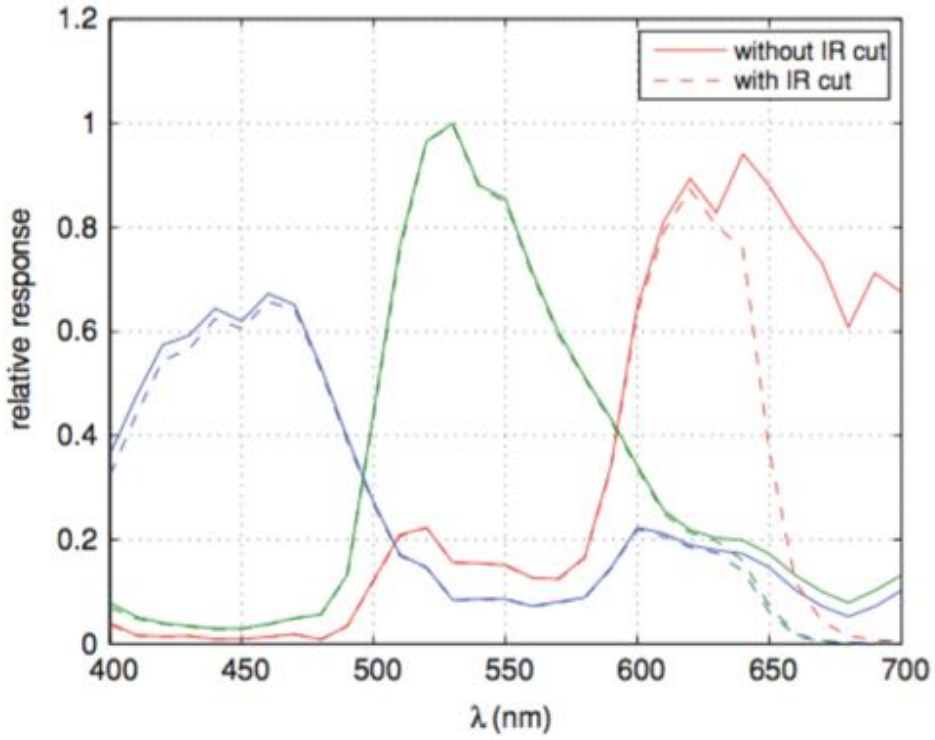
- Anti Aliasing Filter
  - The digital camera sensor consists of a rectilinear grid of pixels.
  - As such it senses a *sampled* version of the image formed by the lens
  - This can lead to high-freqency 'aliases' being generated; the spatial frequency response is show in the lower figure, the resulting *aliasing* of the central 'peak' is shown at the edges of the top figure, opposite
- Solutions
  - Increase the sampling rate – more pixels on sensor
    - *Disadvantages:* requires narrowing PSF & more wiring thus increased cost;
  - Bandlimiting – an optical low-pass filter eliminates the high-frequency components that produce aliasing
    - *Disadvantages:* can reduce image sharpness



DIGITAL CAMERA TECHNOLOGIES #6

BASICS – THE OPTICAL IMAGE PATH #6

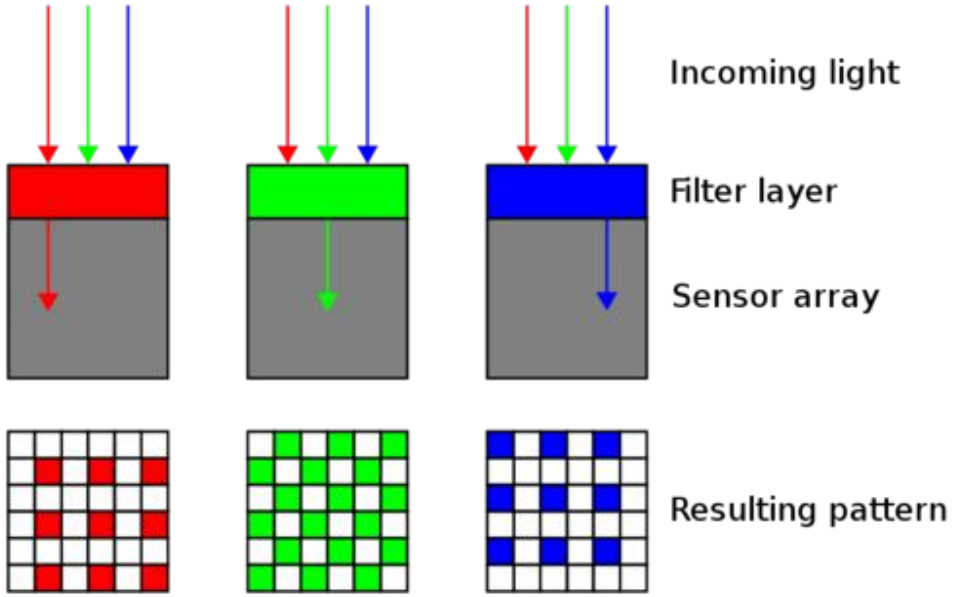
- Figure opposite shows the wavelength/frequency response of a typical CMOS sensor with and without the IR-cut filter
  - Note the blue response is silghtly elevated
  - The green response is identical
  - The red response is significantly elevated leading to a strong reddish hue in the image
- There are some specialized sensors that have a fourth pixel that is NIR sensitive
  - NIR pixel allows to measure NIR and remove its effects from R, G and B pixels in post-processing
  - However these specialized sensors are not (yet) in mainstream devices.



DIGITAL CAMERA TECHNOLOGIES #7

BASICS – THE IMAGE SENSOR #1

- Bayer Color Filter Array
  - A Bayer filter mosaic is a color filter array (CFA) for arranging RGB color filters on a square grid of photosensors.
  - Its particular arrangement of color filters is used in almost all single-chip digital image sensors used in digital cameras, camcorders, and scanners to create a color image.
  - The filter pattern is 50% green, 25% red and 25% blue, hence is also called RGBG, GRGB, or RGGB.

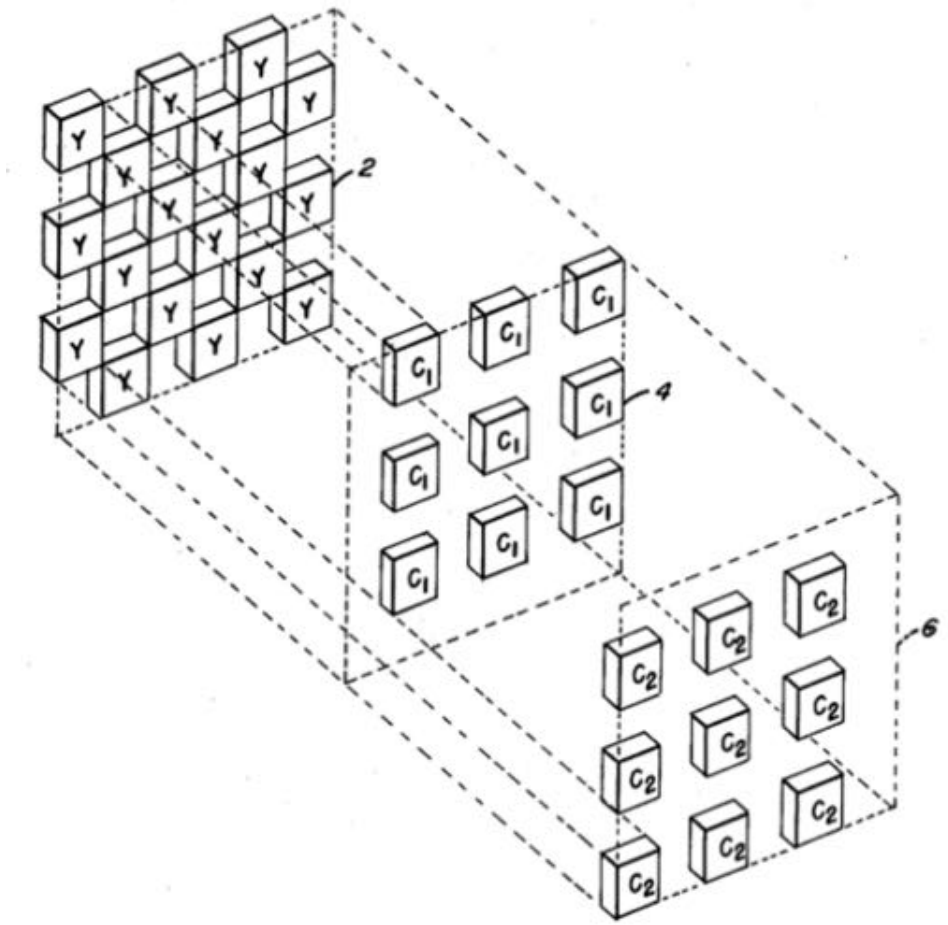


Bottom line – it is easier to eliminate NIR wavelengths with a dedicated optical filter! Almost every **consumer device** has such a NIR filter

DIGITAL CAMERA TECHNOLOGIES #8

BASICS – THE IMAGE SENSOR #2

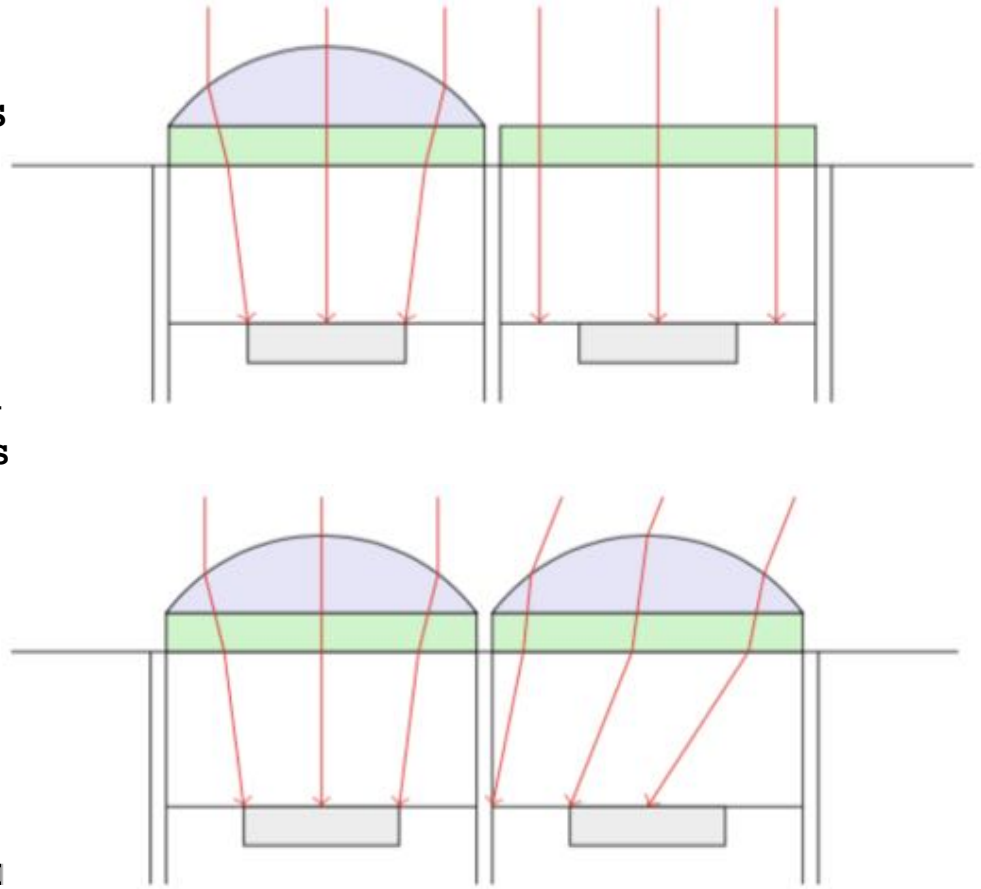
- Bayer Color Filter Array
  - Bryce Bayer's patent (U.S. Patent No. 3,971,065) in 1976 – provided with your readings
  - It refers to the green photo-sensors as *luminance-sensitive* elements and the red and blue ones *chrominance-sensitive* elements.
  - He used twice as many green elements as red or blue to mimic the physiology of the human eye.
    - The luminance perception of the human retina uses M and L cone cells combined, during daylight vision, which are most sensitive to green light.
  - These elements are referred to as *sensor elements*, *sensels*, *pixel sensors*, or simply **pixels**;
  - sample values sensed by these **pixels**, after interpolation, become image pixels.



DIGITAL CAMERA TECHNOLOGIES #9

BASICS – THE IMAGE SENSOR #3

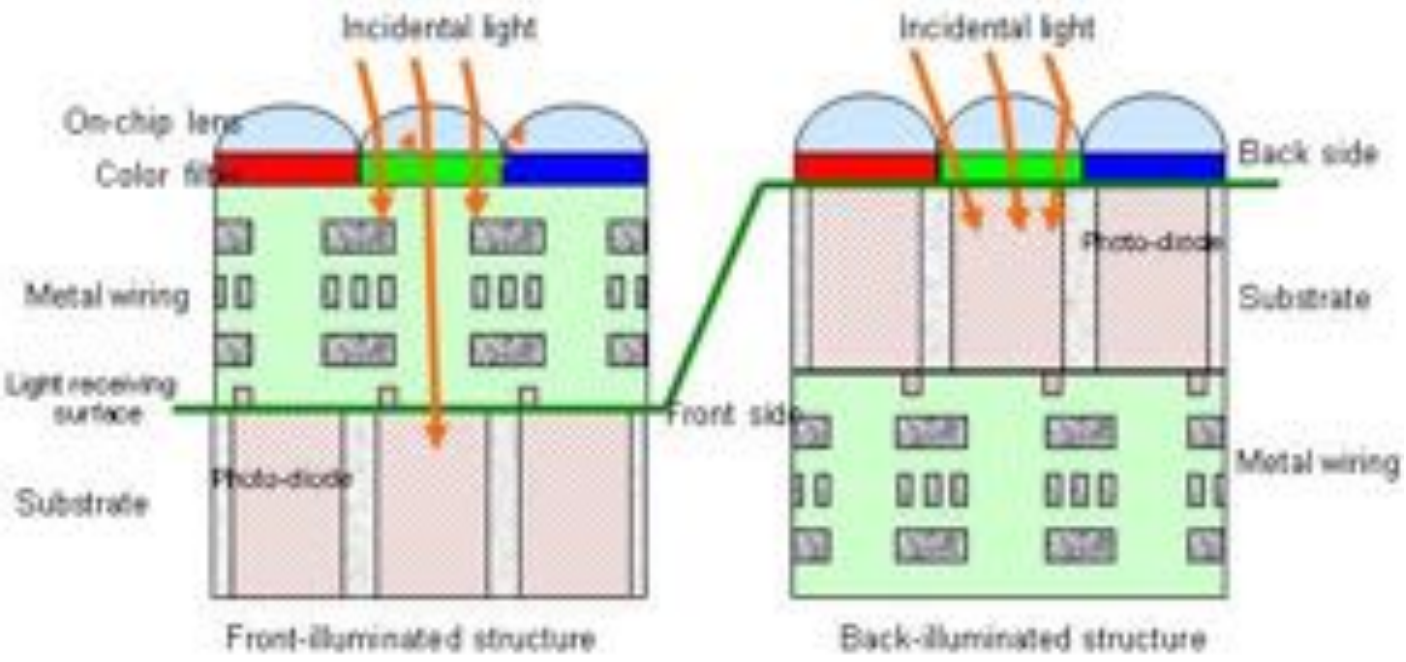
- Lenslet Array
- Array of lenslets is aligned over the sensor; each pixel has its own lenslet;
- Many ways to fabricate lenslet arrays
  - most common commercial methods involve a two-step process of first "carving" basic lenslets forms (usually cylinders) and then melting ("reflowing") them into hemispherical lens shapes
  - It is also possible to control the curvature of the lenslet along only the horizontal and vertical axes and to allow the other orientations (i.e., diagonals) to vary
    - As a result, the profile of the lenslets assumes a more "pillow" or flattened shape
  - research into new and improved methods is active
- Lenslets improve the light gettting the the sensor, especially for sensors with a low 'fill-factor';
- Lenslets are direction-sensitive and tend to limit the useful field of view of the sensor; this can be a challenge for modern consumer devices where wider fields of view are preferred



DIGITAL CAMERA TECHNOLOGIES #10

BASICS – THE IMAGE SENSOR #4

- Back Illuminated Sensors
  - Also known as backside illumination (BSI or BI) sensor
  - It is a type of digital image sensor that uses a novel arrangement of the imaging elements to increase the amount of light captured and thereby improve low-light performance.
  - The technique was used for some time in specialized roles like low-light security cameras and astronomy sensors
  - Complex to build and required refinement before becoming widely used.
  - Sony was the first to reduce these problems and their costs sufficiently to introduce a 5-megapixel 1.75 μm BSI CMOS sensor at general consumer prices in 2009
  - BI sensors from OmniVision Technologies have since been used in consumer electronics from other manufacturers

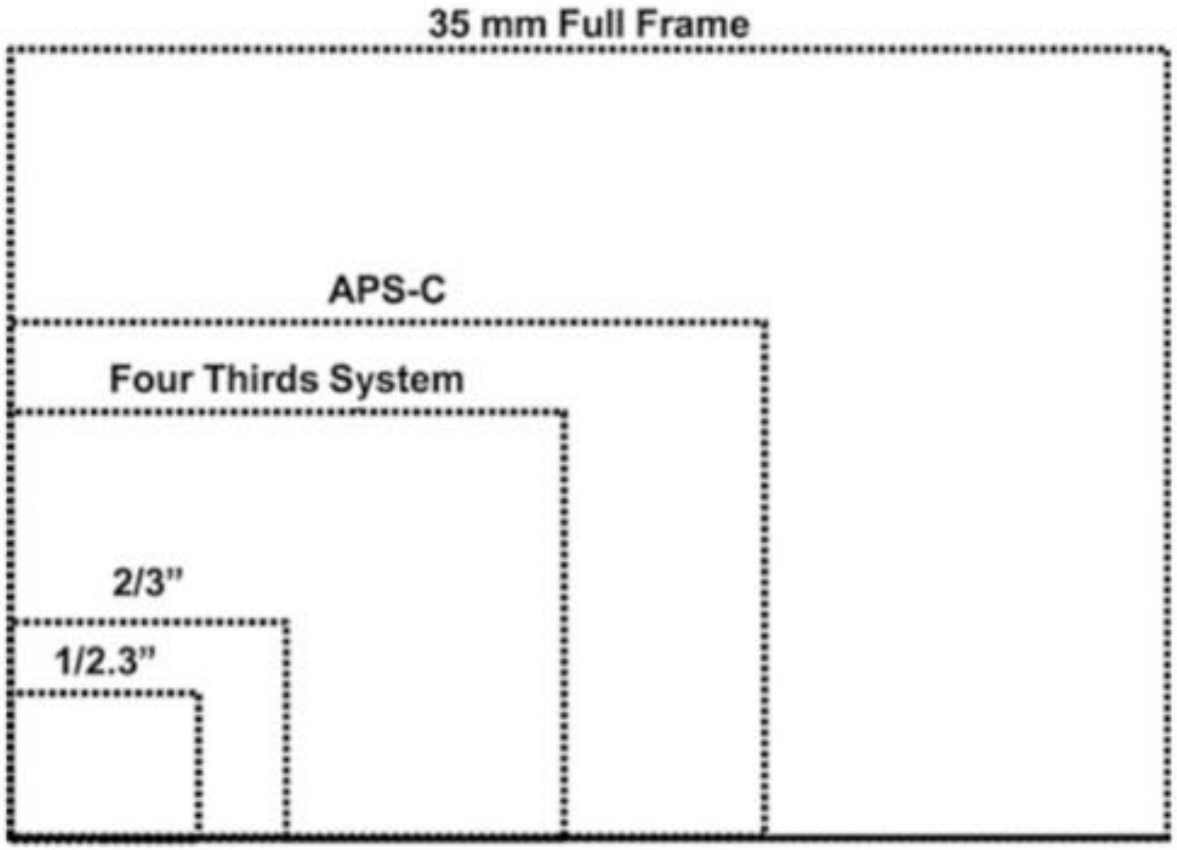


Today almost all consumer devices employ BSI sensors because of their high 'fill-factor' and higher sensitivity to light compared with traditional CMOS processes

DIGITAL CAMERA TECHNOLOGIES #11

BASICS – THE IMAGE SENSOR #5

- The iPhone 5 uses a 1/3.2" or 4.54 x 3.42 mm-sized sensor – even smaller than the smallest size shown in the figure opposite
- This is why the low-light performance of most smartphones is so poor – the individual pixels are thus much smaller on smartphone cameras.
- The undisputed king of smartphone sensors, used by Nokia PureViewTM devices, has a 1/1.2 sensor (10.7 x 8 mm).
- In recent devices the trend to more pixels has been reversed – the latest Samsung Galaxy reverted to 12 megapixels from the 16 megapixel sensor in the Galaxy S6
- Other optical considerations come into play for smartphone-sized devices;
  - The thinness of modern devices limits the focal length of the optics which must be less than 5mm from the sensor surface – a short focal length introduces additional distortions and severely restricts the optical design.
  - There is a fundamental maximum to the resolution of any optical system due to diffraction; an optical system with the ability to produce images with angular resolution as good as the instrument's theoretical limit is said to be **diffraction limited** - see next slide.



A comparison of sensor size; DSLR cameras have Full-Frame (35mm) size sensors; modern consumer cameras are APS-C or 4/3 size; smartphones are limited to much smaller sizes due to their thinness

DIGITAL CAMERA TECHNOLOGIES #15

BASICS – SENSOR OPERATION #3

- **Sensor Artifacts**
  - **Sensor Defects** - are many causes of defects: high dark current arising from an impurity in the silicon crystal, a flaw in a filter or lenslet, a surface flaw on the sensor (dirt or scratch), an electrical fault, or even a flaw on the sensor cover glass
  - **Dark Current** - electrons are generated from thermal effects as well as from photon collection. The current is relatively constant, so the charge collected during integration depends linearly upon integration time. Dark current also varies exponentially with temperature.
  - Dark correction must be precise over a wide range of temperatures and integration times, light-shielded (also known as dark, or optical black) pixels are placed around the periphery of the image sensor to provide accurate data on the dark current accumulated during each exposure
  - **Fixed Pattern Noise** - Fixed pattern noise has two main components: variation in the dark floor, and variation in gain or sensitivity. Variation in dark signal is primarily driven by slight changes in dark current from pixel to pixel. With short exposure times and low dark current levels, the variation in dark signal is usually insignificant.
    - Long exposure times can bring the dark signal up high enough to be visible, and summing or averaging multiple exposures can bring down temporal noise, making the fixed pattern noise more visible. Fixed pattern noise is usually more significant and complex in CMOS sensors than CCDs because of the more complex circuitry in the array
  - **Crosstalk** - most sensors suffer from crosstalk between pixels, where photons that should be collected in one pixel are collected in a neighboring one.
    - Some crosstalk is optical, where light that passes through one lenslet and CFA filter ends up being collected in an adjacent pixel..
    - Crosstalk is also caused by charge diffusion, especially for longer wavelength light. Longer wavelength light penetrates farther into silicon before being absorbed and converted into charge and some photons get absorbed in the substrate rather than in the photosensitive pixel. The charge generated in the substrate can then diffuse to a nearby pixel.
  - With the presence of a CFA, crosstalk presents several artifacts. For example, in the Bayer CFA, the green channel is composed of two classes of pixels. Half of the green pixels have horizontal neighbors that are blue and vertical neighbors that are red. The other half have the converse—horizontal neighbors are red and vertical neighbors are blue.
    - Because of this layout, crosstalk makes the effective spectral sensitivity of each red (or blue) pixel combine with a portion of the sensitivity of neighboring green pixels.
    - Another crosstalk artifact is non-uniformity in response of pixels in a color channel, based upon their neighbors. Thus, half of the green pixels have spectral sensitivity that has more of a red contribution, while the other half has more of a blue contribution. This change in sensitivity is usually a small percentage of the total sensitivity, but the highly patterned nature of the artifact can cause trouble for demosaicing.
  - **Saturation** - Image sensors can exhibit artifacts in and around regions of the image where pixels are saturated, that is, where a pixel accumulates more charge than it can contain. When a pixel is saturated, charge can overflow from one pixel to adjacent pixels, creating what are known as blooming artifacts

DIGITAL CAMERA TECHNOLOGIES #16

BASICS – THE IMAGE PIPELINE #1

- To fully understand the complexity of what happens in a modern digital camera, we need to illustrate the concept of the image processing pipeline (IPP) – the sequence of underlying manipulations of the original image data to get to the image that you see on the main camera screen.
  - (more details in reading #2)

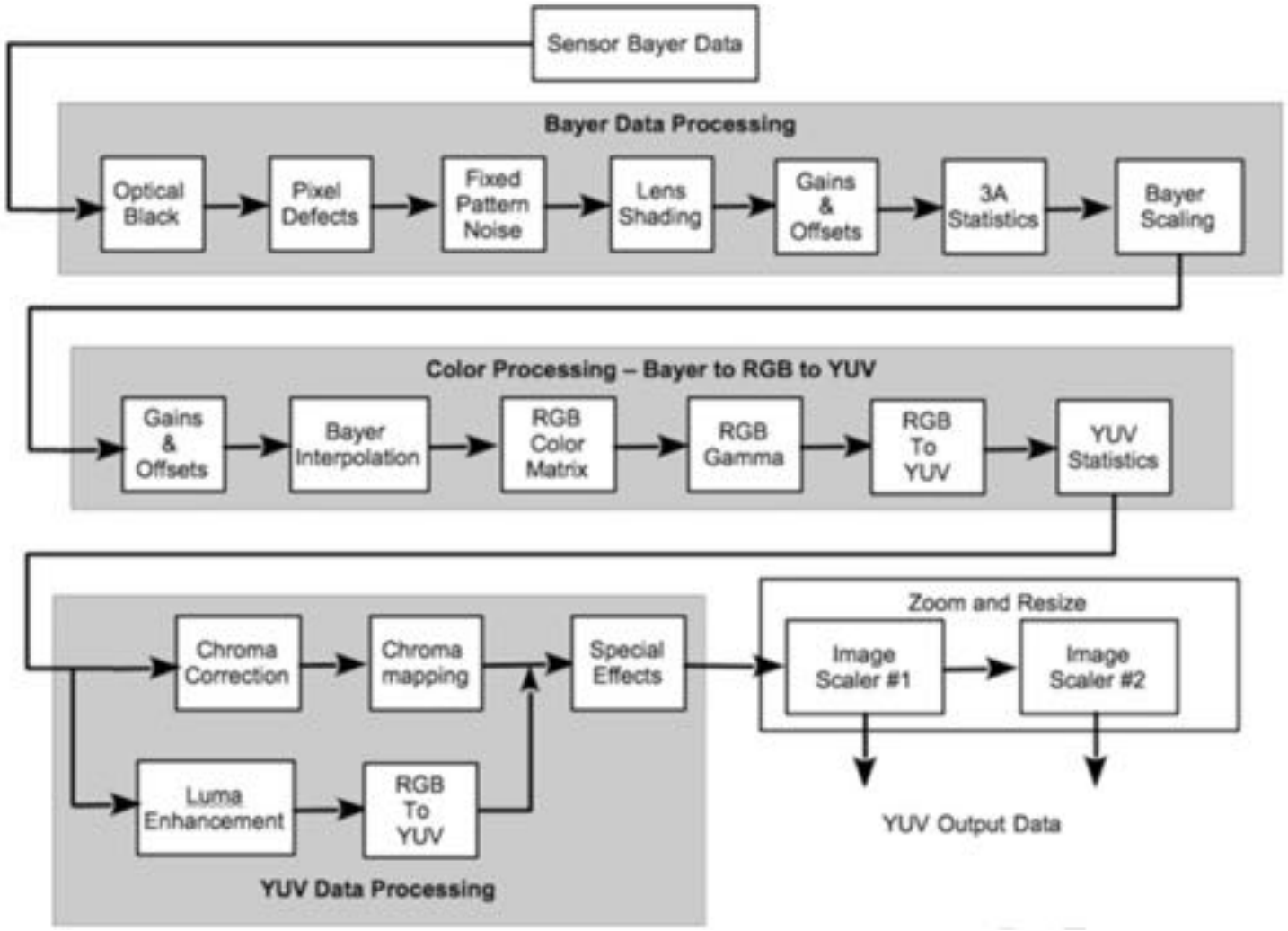


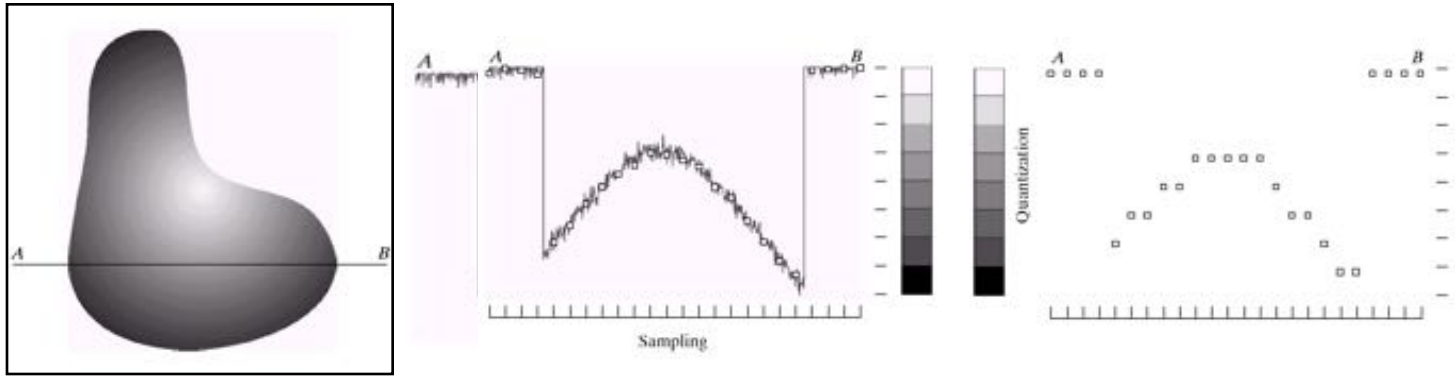
IMAGE SAMPLING & QUANTIZATION

The Practicalities of Image Representation

IMAGE SAMPLING AND QUANTISATION

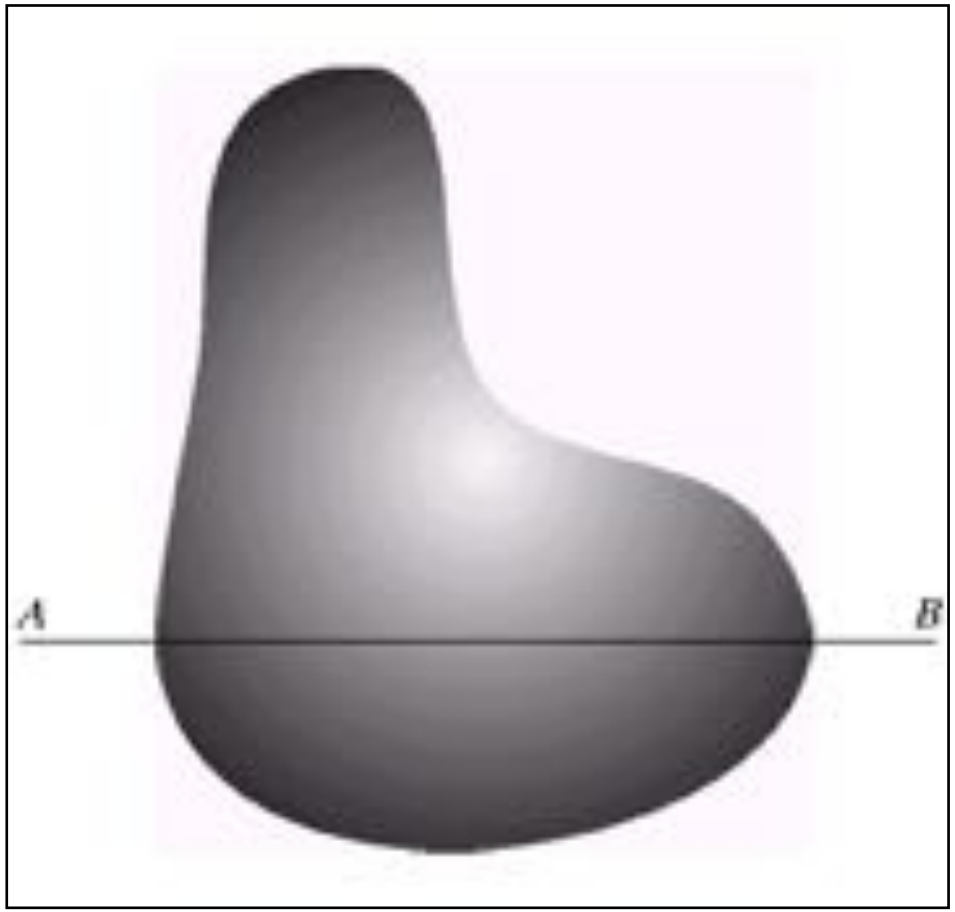
A digital sensor can only measure a limited number of **samples** at a **discrete** set of energy levels

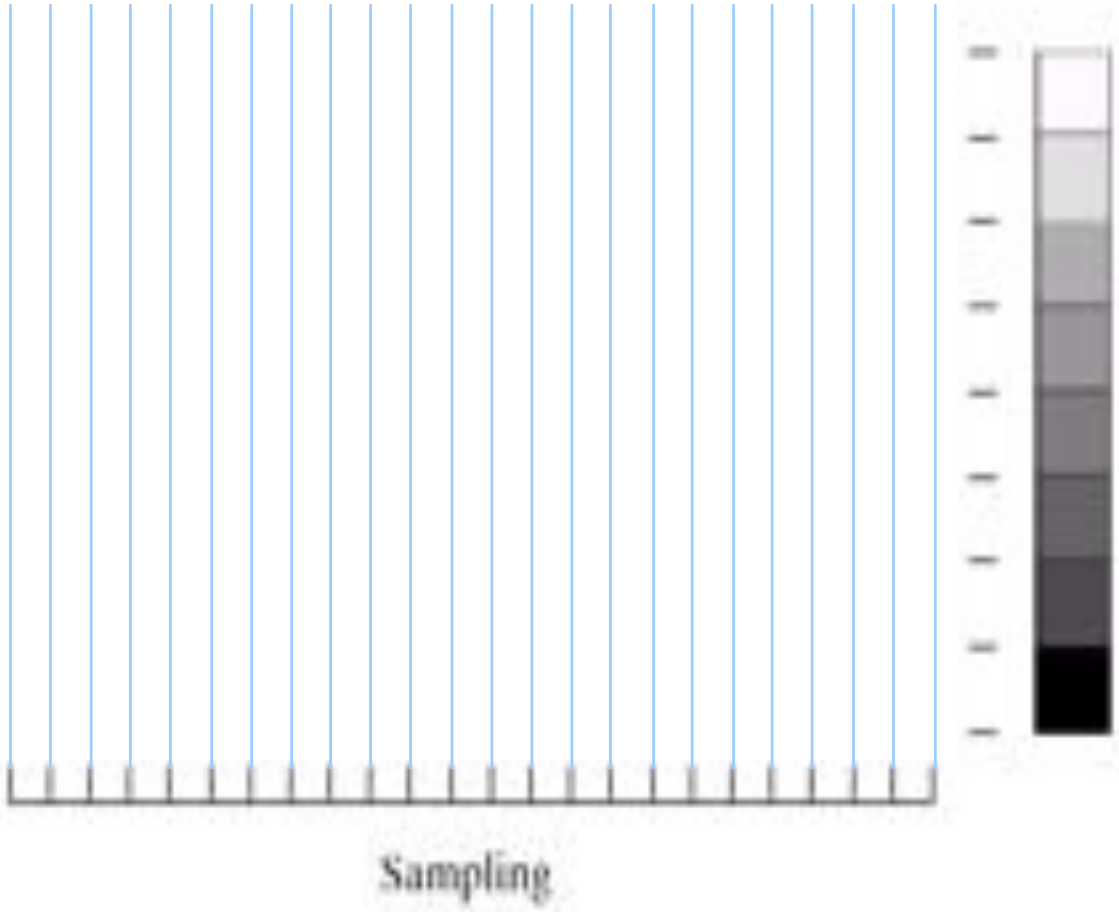
Quantisation is the process of converting a continuous **analogue** signal into a digital representation of this signal



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

IMAGE SAMPLING AND QUANTISATION

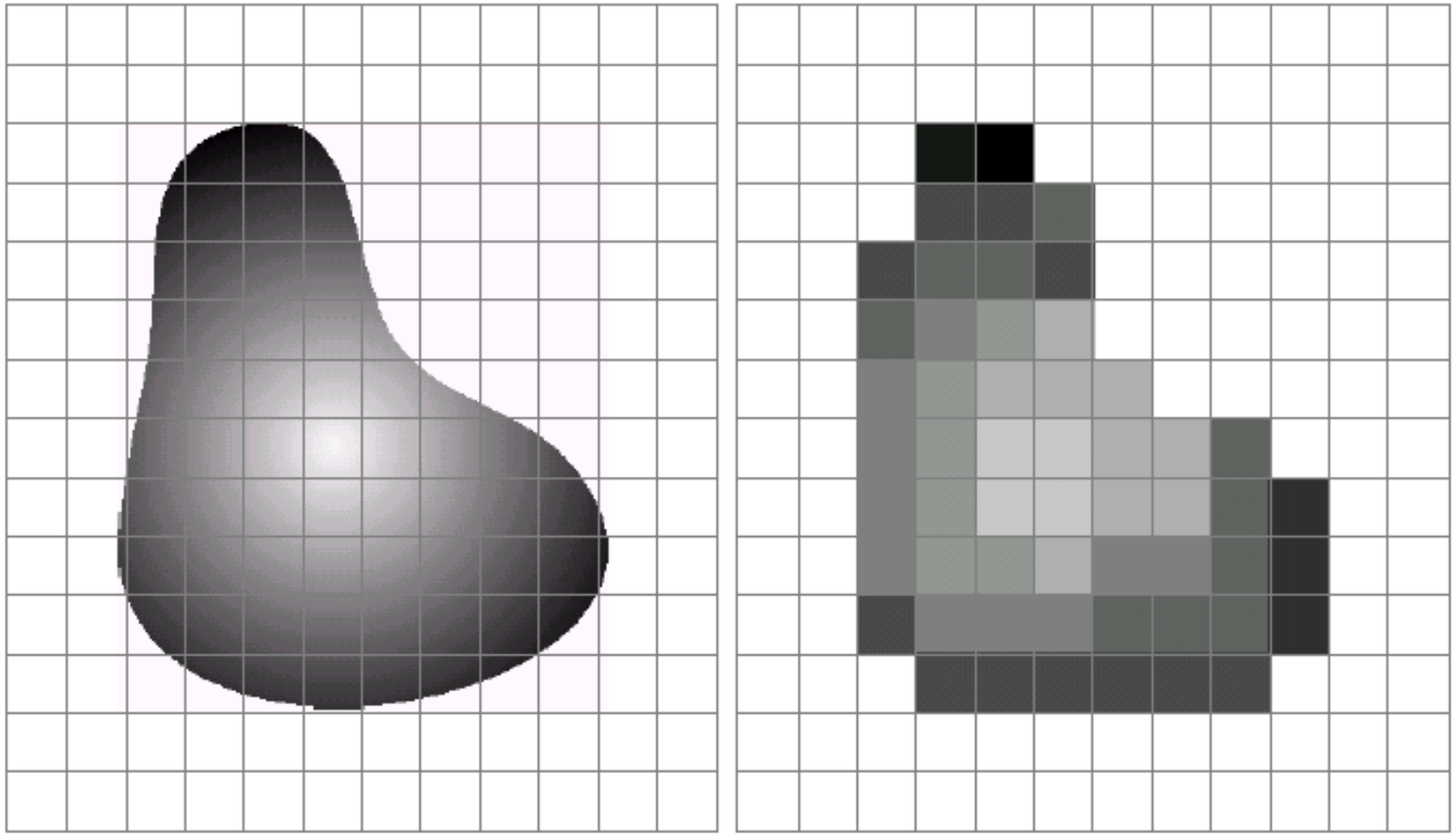




Images taken from Gonzalez & Woods, Digital Image Processing (2002)

IMAGE SAMPLING AND QUANTISATION (CONT…)

Remember that a digital image is always only an **approximation** of a real world scene



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

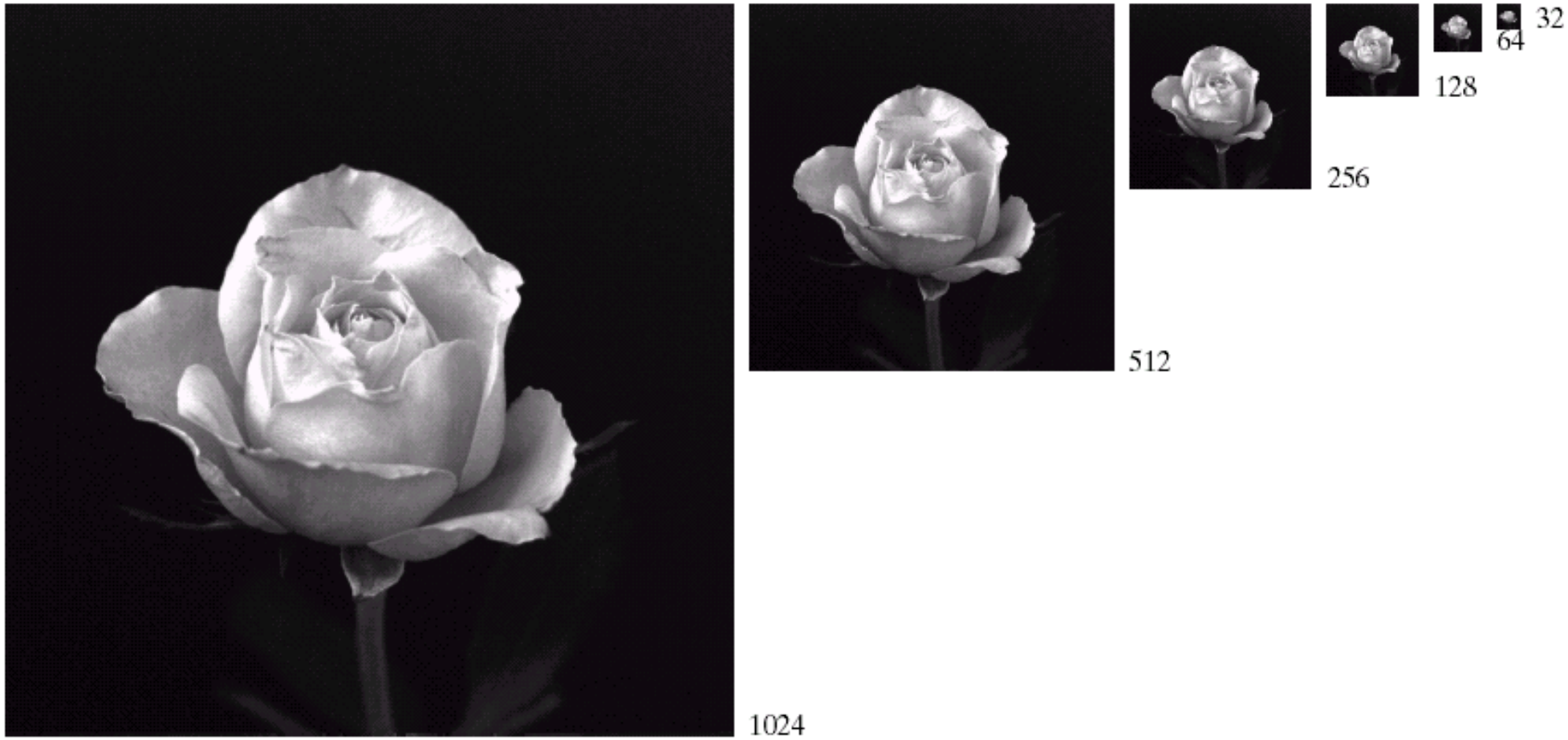
SPATIAL RESOLUTION

The spatial resolution of an image is determined by how sampling was carried out

Spatial resolution simply refers to the smallest discernable detail in an image

- Vision specialists will often talk about pixel size
- Graphic designers will talk about *dots per inch* (DPI)

SPATIAL RESOLUTION (CONT…)



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

SPATIAL RESOLUTION (CONT…)

1024 * 1024 | 512 * 512 | 256 * 256

128 * 128 | 64 * 64 | 32 * 32

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

INTENSITY LEVEL RESOLUTION

Intensity level resolution refers to the number of intensity levels used to represent the image

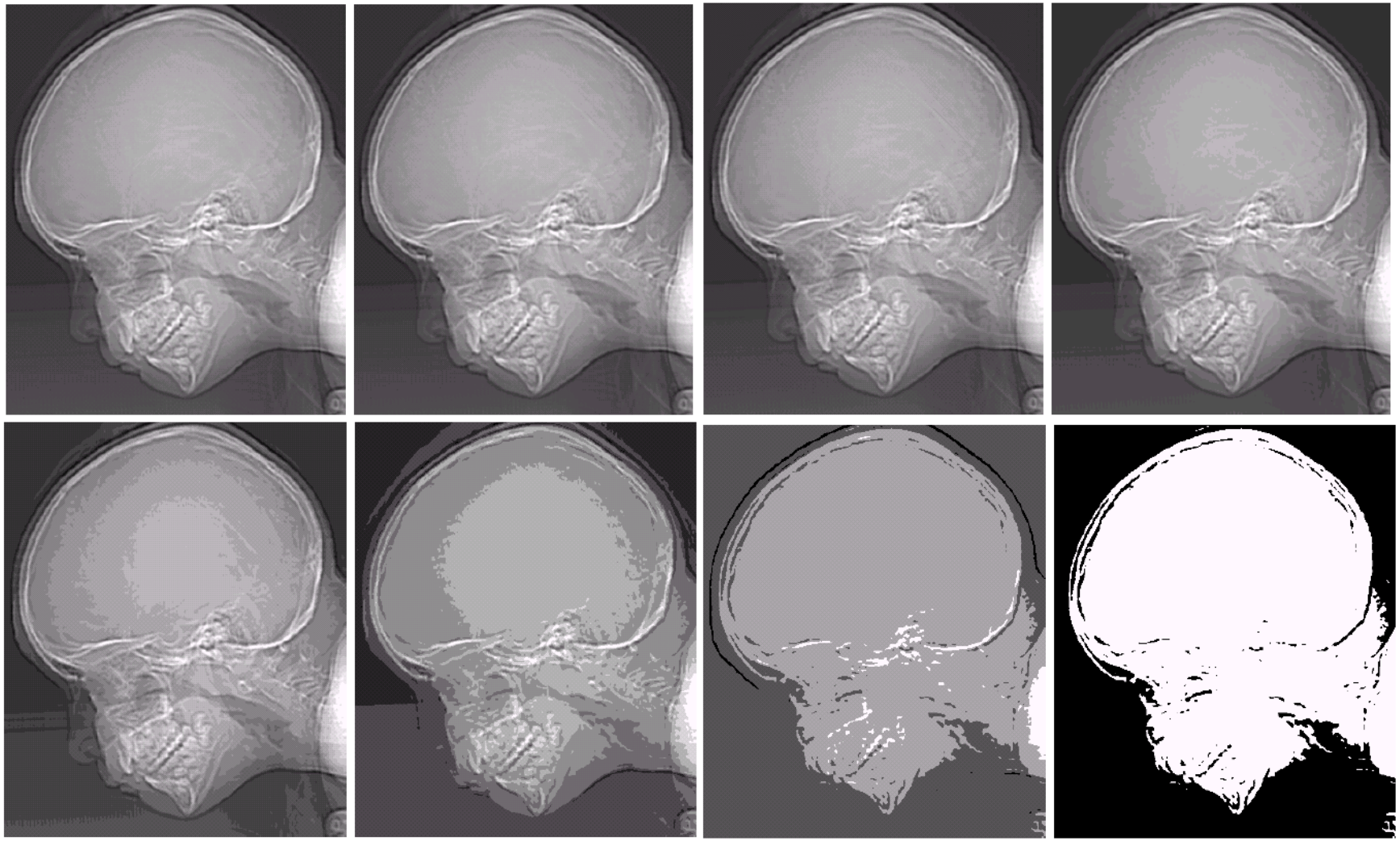
- The more intensity levels used, the finer the level of detail discernable in an image
- Intensity level resolution is usually given in terms of the number of bits used to store each intensity level

Number of Bits	Number of Intensity Levels	Examples
1	2	0, 1
2	4	00, 01, 10, 11
4	16	0000, 0101, 1111
8	256	00110011, 01010101
16	65,536	1010101010101010

INTENSITY LEVEL RESOLUTION (CONT…)

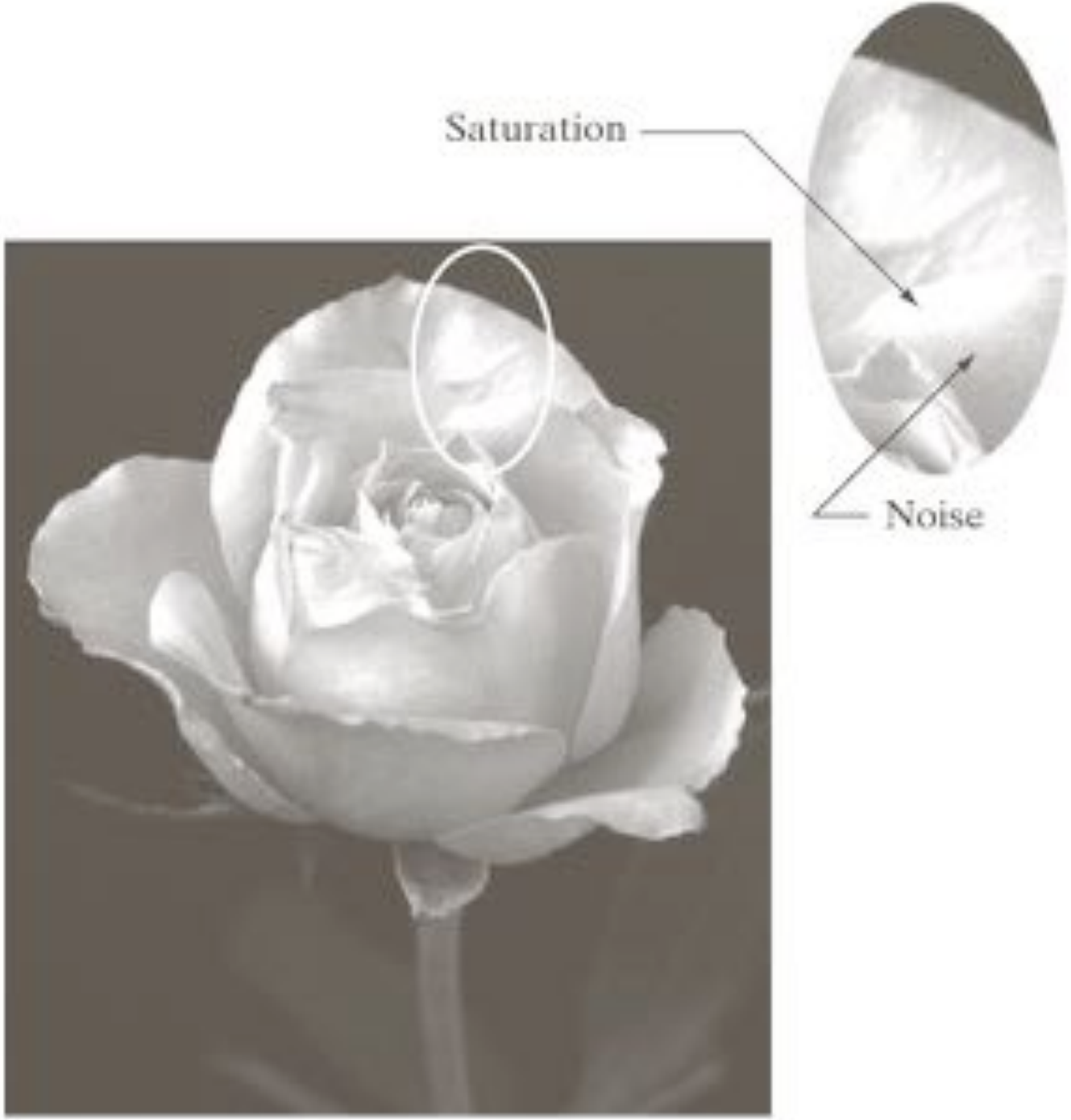
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

256 grey levels (8 bits per pixel) | 128 grey levels (7 bpp) | 64 grey levels (6 bpp) | 32 grey levels (5 bpp)



16 grey levels (4 bpp) | 8 grey levels (3 bpp) | 4 grey levels (2 bpp) | 2 grey levels (1 bpp)

SATURATION & NOISE



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

RESOLUTION: HOW MUCH IS ENOUGH?

The big question with resolution is always *how much is enough*?

- This all depends on what is in the image and what you would like to do with it
- Key questions include
  - Does the image look aesthetically pleasing?
  - Can you see what you need to see within the image?

RESOLUTION: HOW MUCH IS ENOUGH? (CONT…)

The picture on the right is fine for counting the number of cars, but not for reading the number plate

INTENSITY LEVEL RESOLUTION (CONT…)

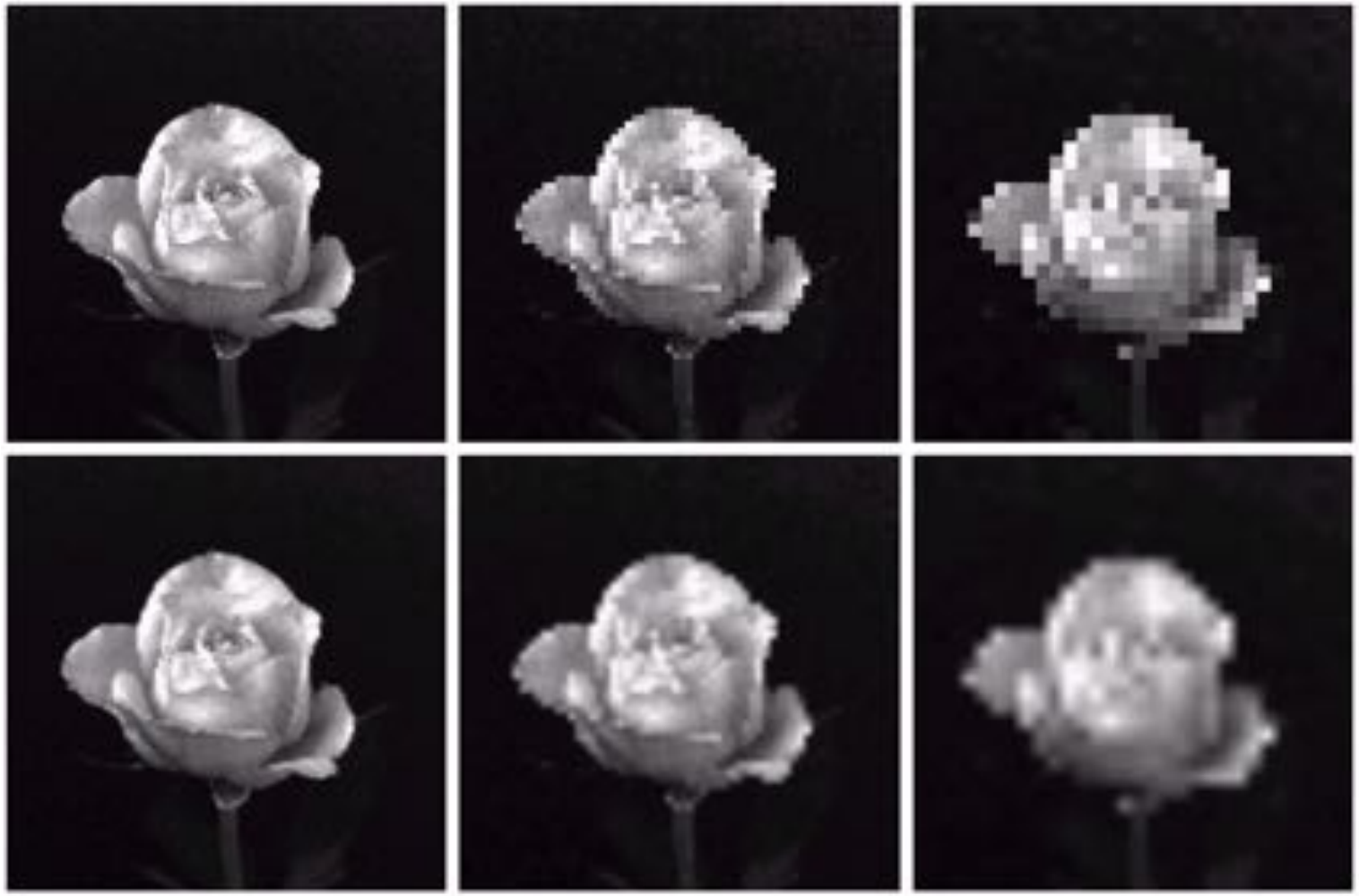
Low Detail

Medium Detail

High Detail

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

INTERPOLATION



a b c
d e f

FIGURE 2.25 Top row: images zoomed from 128 × 128, 64 × 64, and 32 × 32 pixels to 1024 × 1024 pixels, using nearest neighbor gray-level interpolation. Bottom row: same sequence, but using bilinear interpolation.

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

OPEN_CV (COMPUTER VISON S/W LIBRARY)

A Quick Introduction

WHAT IS OPENCV??

- An open source BSD licensed computer vision library.
  - Patent-encumbered code isolated into “non-free” module.
    - SIFT, SURF, some of the Face Detectors, etc.
- Available on all major platforms
  - Android, iOS, Linux, Mac OS X, Windows
- Written primarily in C++
  - Bindings available for Python, Java, even MATLAB (in 3.0).
- Well documented at http://docs.opencv.org
- Source available at https://github.com/Itseez/opencv

OpenCV

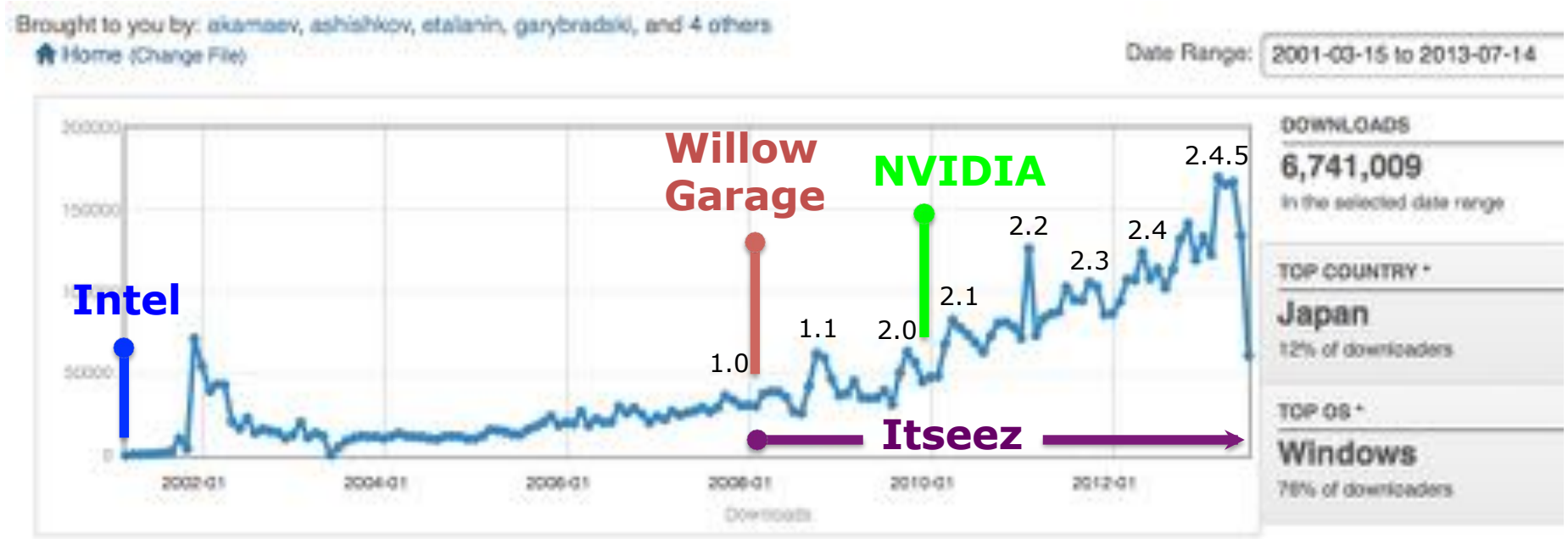
HISTORY OF OPENCV

- OpenCV started by Intel Research in 1998.
- Goals originally were:-
  - Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.
  - Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.
  - Advance vision-based commercial applications by making portable, performance-optimized code available for free—with a license that did not require to be open or free themselves.
- Originally released at CVPR 2000.

OPENCV THEN AND NOW…..

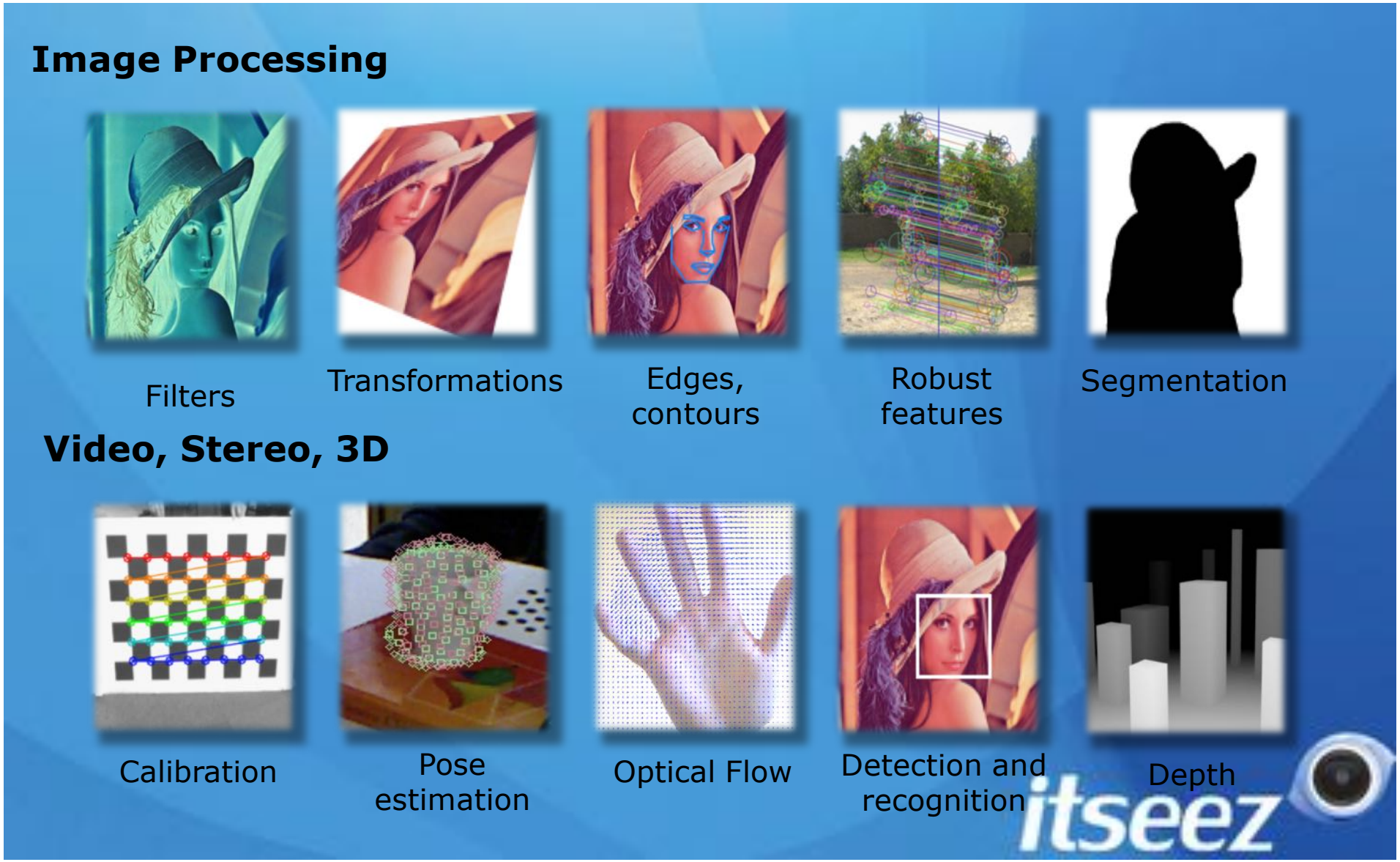
- Version 1.0 was released in 2006.
- In 2008 obtained corporate support from Willow Garage (Robotics Company).
- OpenCV 2 was released in 2009.
  - Included major changes for C++ (mostly C beforehand).
- In 2012 support for OpenCV was taken over by a non-profit foundation OpenCV.org.
- OpenCV 3 was released in 2014.
  - Seems to be under corporate support from Itseez.
  - More on these changes soon.

OPENCV THEN AND NOW….



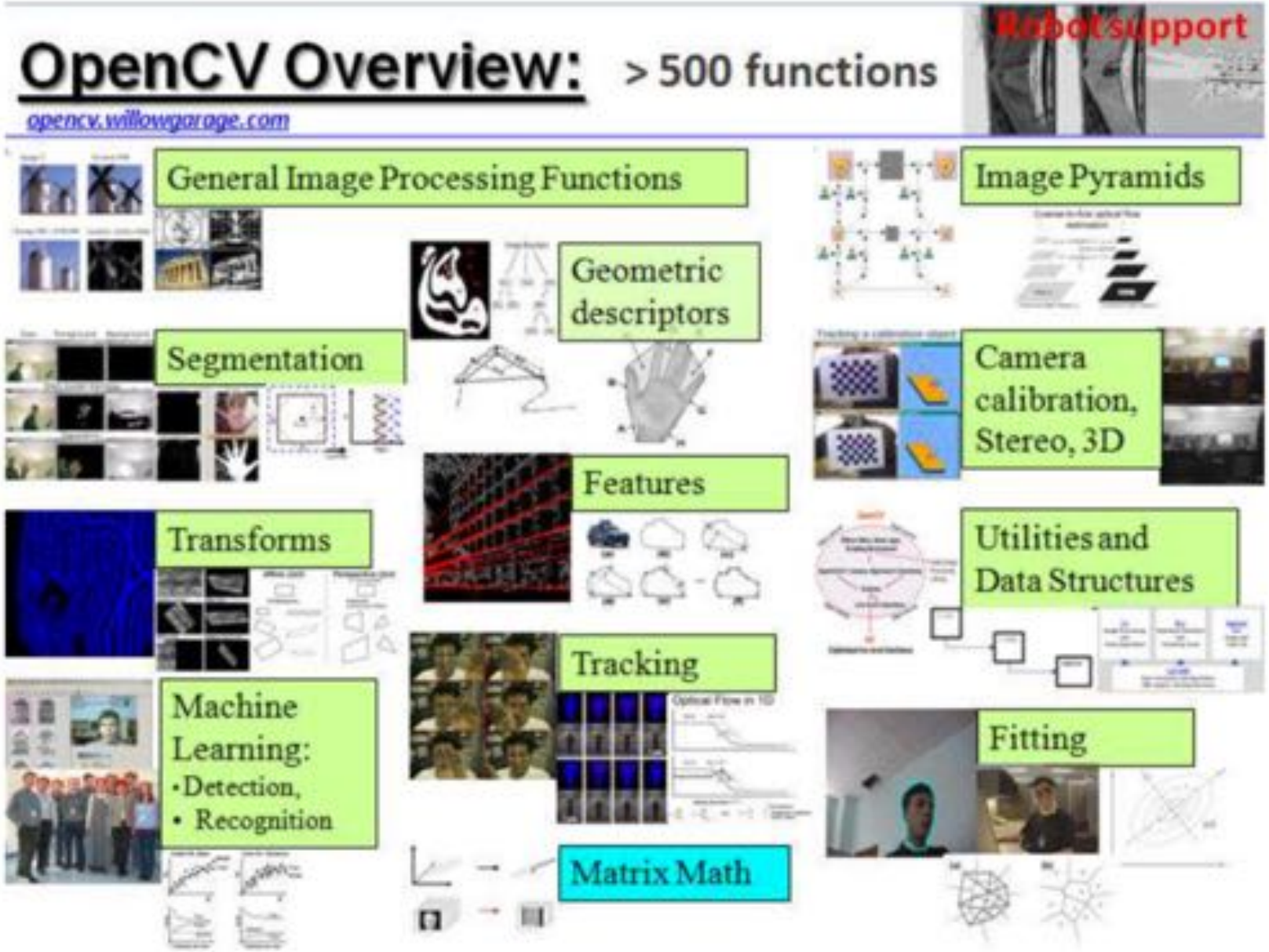
Taken from OpenCV 3.0 latest news and the roadmap.

WHAT CAN OPENCV DO?



Taken from OpenCV 3.0 latest news and the roadmap.

WHAT CAN OPENCV DO? #2



OPENCV 3.0

Migration is relatively smooth from 2.4

- Mostly cleanings
  - Refined C++ API
  - Use **cv::Algorithm** everywhere
- API changes
  - C API will be marked as deprecated
  - Old Python API will be deprecated
  - Monstrous modules will be split into micromodules – Extra modules

OPENCV 3.0

- Sufficiently improved CUDA and OpenCL modules
  - Mobile CUDA support
  - Universal OpenCL binaries (CPU, GPU)
- Hardware Abstraction Layer (HAL)
  - IPP, FastCV-like low-level API to accelerate OpenCV on different HW.
- Open-source NEON optimizations
  - iOS, Android, Embedded.
  - Latest NEWS - 40 NEON optimized functions in 3.0.
- Check out the transition guide.

WHICH VERSION WILL BE USING?

- OpenCV 3.0 is brand new, and is well worth a look and play.
- Most vision tutorials are still in OpenCV 2.4.X.
- OpenCV 2.4.X is still the de facto library for computer vision and image processing.
- Will remain like this until 3.0 matures.

CAUTION!

- Danger with OpenCV is that it allows you to do a lot with very little understanding for what is going on.
- It is also assumed that you know C++ going forward.

KEY OPENCV CLASSES

Point_	Template 2D point class
Point3_	Template 3D point class
Size_	Template size (width, height) class
Vec	Template short vector class
Matx	Template small matrix class
Scalar	4-element vector
Rect	Rectangle
Range	Integer value range
Mat	2D or multi-dimensional dense array (can be used to store matrices, images, histograms, feature descriptors, voxel volumes etc.)
SparseMat	Multi-dimensional sparse array
Ptr	Template smart pointer class

(Taken from "OpenCV 2.4 Cheat Sheet")

MATRIX BASICS

Create a matrix

```
Mat image(240, 320, CV_8UC3);
```

[Re]allocate a pre-declared matrix

```
image.create(480, 640, CV_8UC3);
```

Create a matrix initialized with a constant

```
Mat A33(3, 3, CV_32F, Scalar(5));
Mat B33(3, 3, CV_32F); B33 = Scalar(5);
Mat C33 = Mat::ones(3, 3, CV_32F)*5.;
Mat D33 = Mat::zeros(3,3, CV_32F)  + 5.;
```

Create a matrix initialized with specifted values

```
double a = CV_PI/3;
Mat A22 = (Mat_<float>(2, 2) «
   cos(a), -sin(a), sin(a), cos(a));
float B22data[] ={cos(a),  -sin(a), sin(a), cos(a)};
Mat B22 = Mat(2,  2,CV_32F, B22data).clone();
```

(Taken from "OpenCV 2.4 Cheat Sheet")

OpenCV 2.4 Cheat Sheet (C++)

*The OpenCV C++ reference manual is here: http://docs.opencv.org. Use **Quick Search** to find descriptions of the particular functions and classes*

Key OpenCV Classes

Point_	Template 2D point class
Point3_	Template 3D point class
Size_	Template size (width, height)class
Vec	Template short vector class
Matx	Template small matrix class 4-
Scalar	element vector
Rect	Rectangle
Range	Integer value range
Mat	2D or multi-dimensional dense array (can be used to store matrices, images, histograms, feature descriptors, voxel volumes etc.)
SparseMat	Multi-dimensional sparse array
Ptr	Template smart pointer class

Matrix Basics

Create a matrix

```
Mat image(240, 320, CV_8UC3);
```

[Re]allocate a pre-declared matrix

```
image.create(480, 640, CV_8UC3);
```

Create a matrix initialized with a constant

```
Mat A33(3, 3, CV_32F, Scalar(5));
Mat B33(3, 3, CV_32F); B33 = Scalar(5);
Mat C33 = Mat::ones(3, 3, CV_32F)*5.; Mat
D33 = Mat::zeros(3, 3, CV_32F) + 5.;
```

Create a matrix initialized with specifted values

```
double a = CV_PI/3;
Mat A22 = (Mat_<float>(2, 2) «
   cos(a), -sin(a), sin(a), cos(a));
float B22data[] = {cos(a), -sin(a), sin(a), cos(a)};
Mat B22 = Mat(2, 2, CV_32F, B22data).clone();
```

Initialize a random matrix

```
randu(image, Scalar(0), Scalar(256)); // uniform dist
randn(image, Scalar(128), Scalar(10)); // Gaussian dist
```

Convert matrix to/from other structures (without copying the data)

```
Mat image_alias = image;
float* Idata=new float[480*640*3];
Mat I(480, 640, CV_32FC3,Idata);
vector<Point> iptvec(10);
Mat iP(iptvec); // iP – 10x1 CV_32SC2 matrix IplImage*
oldC0 = cvCreateImage(cvSize(320,240),16,1); Mat newC=
cvarrToMat(oldC0);
IplImage oldC1 = newC; CvMat oldC2 = newC;
```

... (with copying the data)

```
Mat newC2 = cvarrToMat(oldC0).clone();
vector<Point2f> ptvec = Mat_<Point2f>(iP);
```

Access matrix elements

```
A33.at<float>(i,j) = A33.at<float>(j,i)+1;
Mat dyImage(image.size(), image.type());
for(int y = 1; y <image.rows-1; y++) {
   Vec3b* prevRow = image.ptr<Vec3b>(y-1);
   Vec3b* nextRow = image.ptr<Vec3b>(y+1);
   for(int x = 0; x <image.cols; x++)
      for(int c = 0; c <3; c++)
         dyImage.at<Vec3b>(y,x)[c] =
         saturate_cast<uchar>(
         nextRow[x][c] - prevRow[x][c]);
}
Mat_<Vec3b>::iterator it = image.begin<Vec3b>(),
   itEnd = image.end<Vec3b>();
for(; it != itEnd; ++it)
   (*it)[1] ^= 255;
```

Matrix Manipulations: Copying, Shuffling, Part Access

src.copyTo(dst)	Copy matrix to another one
src.convertTo(dst,type,scale,shift)	Scale and convert to another datatype
m.clone()	Make deep copy of a matrix
m.reshape(nch,nrows)	Change matrix dimensions and/or number of channels without copying data
m.row(i), m.col(i)	Take a matrix row/column
m.rowRange(Range(i1,i2)) m.colRange(Range(j1,j2))	Take a matrix row/column span
m.diag(i)	Take a matrix diagonal
m(Range(i1,i2),Range(j1,j2)), m(roi)	Take a submatrix
m.repeat(ny,nx)	Make a bigger matrix from a smaller one
flip(src,dst,dir)	Reverse the order of matrix rows and/or columns
split(...)	Split multi-channel matrix into separate channels
merge(...)	Make a multi-channel matrix out of the separate channels
mixChannels(...)	Generalized form of split() and merge()
randShuffle(...)	Randomly shuffle matrix elements

Example 1. Smooth image ROI in-place

```
Mat imgroi = image(Rect(10, 20, 100, 100));
GaussianBlur(imgroi, imgroi, Size(5, 5), 1.2, 1.2);
```

Example 2. Somewhere in a linear algebra algorithm

```
m.row(i) += m.row(j)*alpha;
```

Example 3. Copy image ROI to another image with conversion

```
Rect r(1, 1, 10, 20);
Mat dstroi = dst(Rect(0,10,r.width,r.height));
src(r).convertTo(dstroi, dstroi.type(), 1, 0);
```

Simple Matrix Operations

OpenCV implements most common arithmetical, logical and other matrix operations, such as

- add(), subtract(), multiply(), divide(), absdiff(), bitwise_and(), bitwise_or(), bitwise_xor(), max(), min(), compare()
  – correspondingly, addition, subtraction, element-wise multiplication ... comparison of two matrices or a matrix and a scalar.

  Example. Alpha compositing function:

```
void alphaCompose(const Mat&rgba1,
   const Mat&rgba2, Mat&rgba_dest)
{
   Mat a1(rgba1.size(), rgba1.type()), ra1;
   Mat a2(rgba2.size(), rgba2.type());
   int mixch[]={3, 0, 3, 1, 3, 2, 3, 3};
   mixChannels(&rgba1, 1, &a1, 1, mixch, 4);
   mixChannels(&rgba2, 1, &a2, 1, mixch, 4);
   subtract(Scalar::all(255), a1, ra1);
   bitwise_or(a1, Scalar(0,0,0,255), a1);
   bitwise_or(a2, Scalar(0,0,0,255), a2);
   multiply(a2, ra1, a2, 1./255);
   multiply(a1, rgba1, a1, 1./255);
   multiply(a2, rgba2, a2, 1./255);
   add(a1, a2, rgba_dest);
}
```

- sum(), mean(), meanStdDev(), norm(), countNonZero(), minMaxLoc(),
  – various statistics of matrix elements.
- exp(), log(), pow(), sqrt(), cartToPolar(), polarToCart()
  – the classical math functions.
- scaleAdd(), transpose(), gemm(), invert(), solve(), determinant(), trace(), eigen(), SVD,
  – the algebraic functions + SVD class.
- dft(), idft(), dct(), idct(),
  – discrete Fourier and cosine transformations

For some operations a more convenient algebraic notation can be used, for example:

```
Mat delta = (J.t()*J + lambda*
   Mat::eye(J.cols, J.cols, J.type()))
   .inv(CV_SVD)*(J.t()*err);
```

implements the core of Levenberg-Marquardt optimization algorithm.

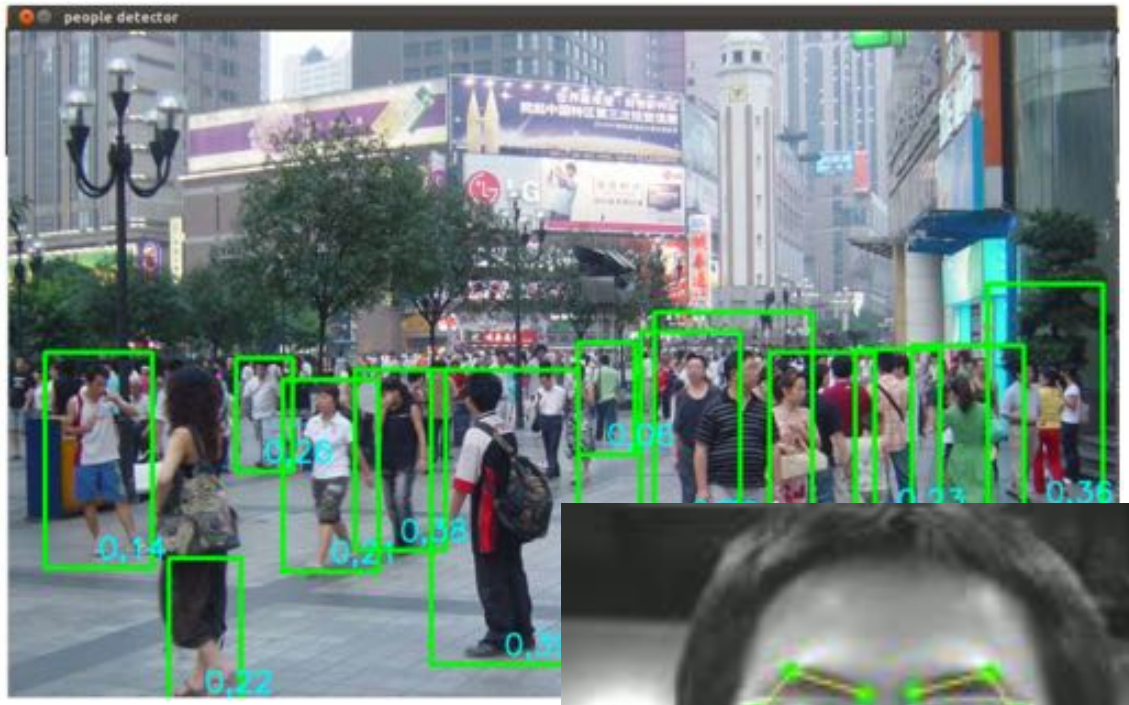
Image Processsing

Filtering

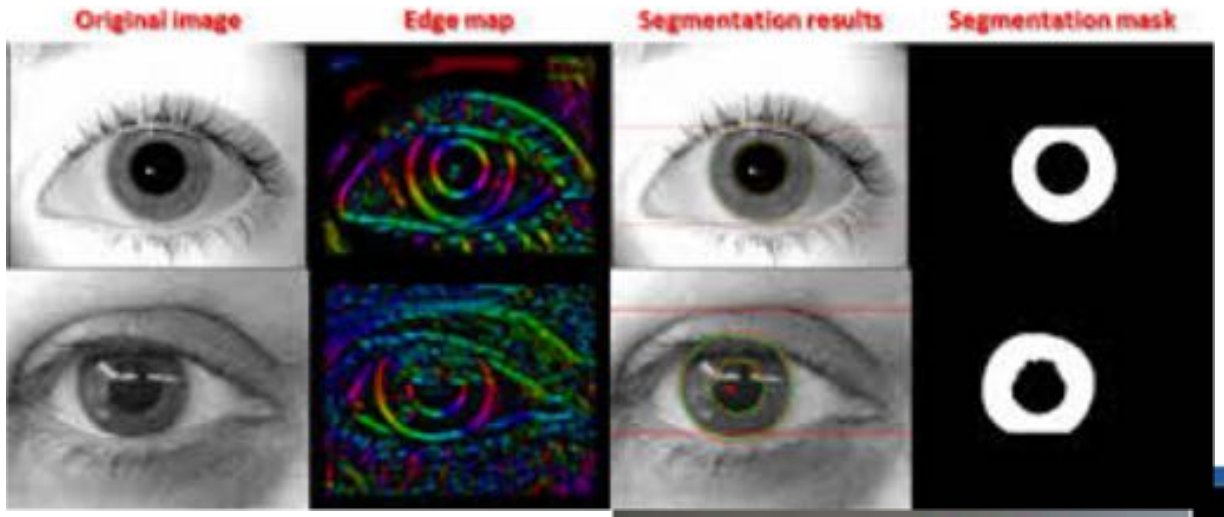
filter2D()	Non-separable linear filter
sepFilter2D()	Separable linear filter
boxFilter(), GaussianBlur(), medianBlur(), bilateralFilter()	Smooth the image with one of the linear or non-linear filters
Sobel(), Scharr()	Compute the spatial image derivatives
Laplacian()	compute Laplacian: $\nabla I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$
erode(), dilate()	Morphological operations

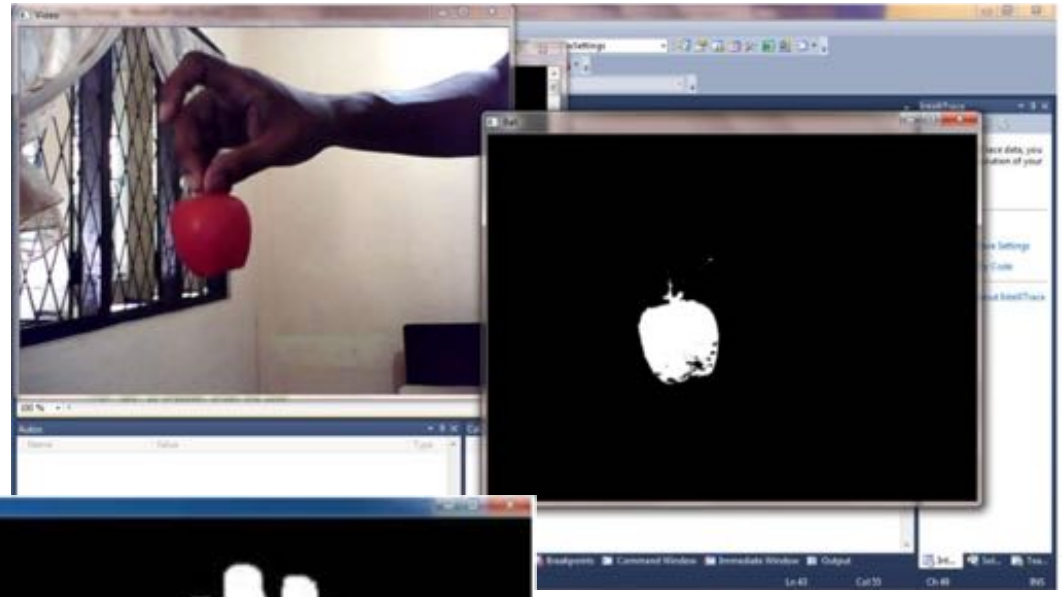
(Taken from "OpenCV 2.4 Cheat Sheet")

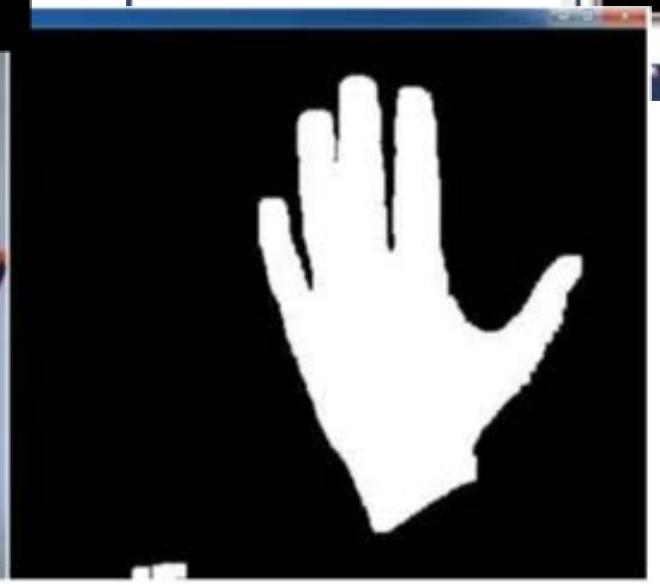
SOME EXAMPLE USES



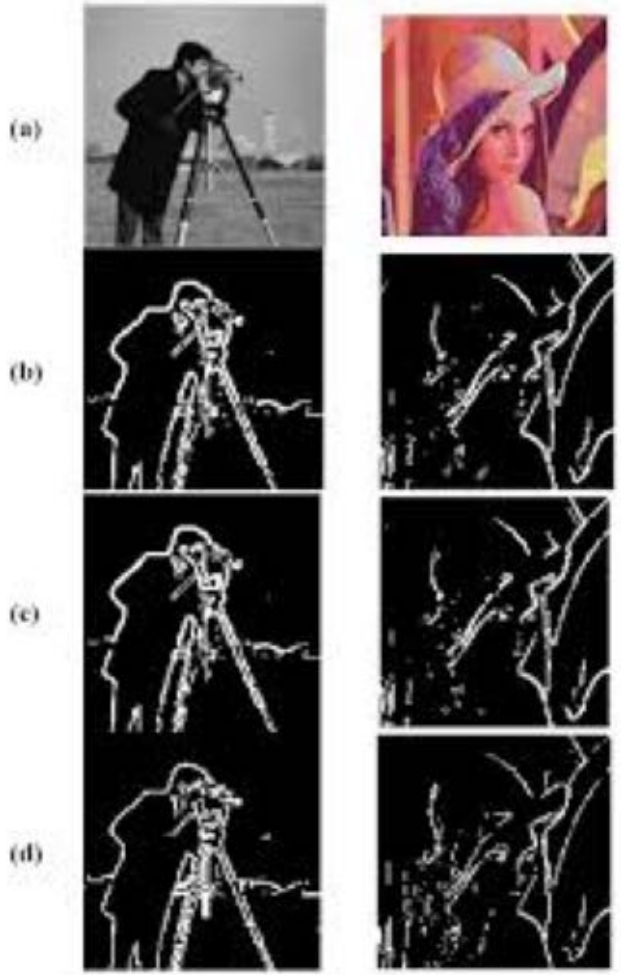
SOME EXAMPLE USES - SEGMENTATION

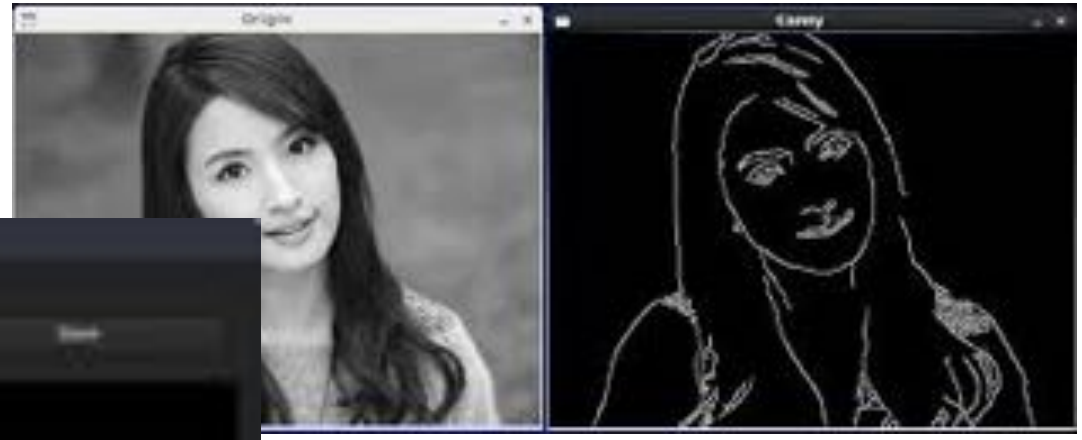


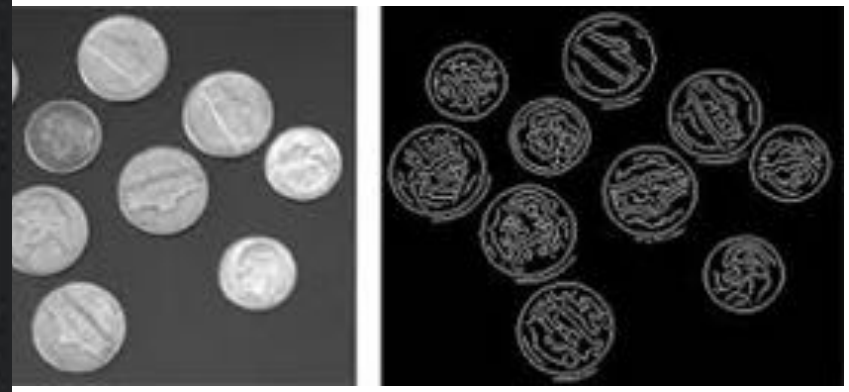




SOME EXAMPLE USES – EDGE DETECTION





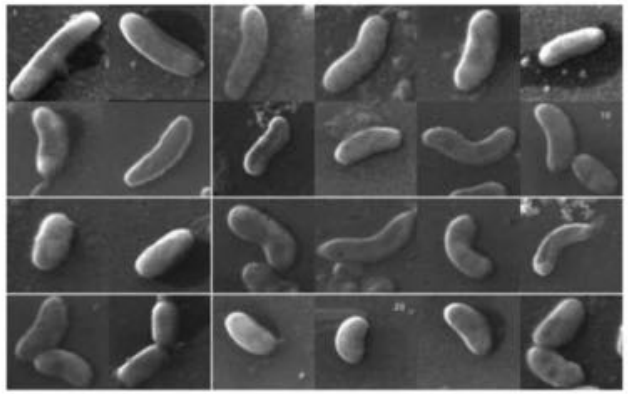


SOME EXAMPLE USES - APPLICATIONS

Features Extraction- Chalky grains

characterization of diseases

Biotechnology



??????????

? Questions ?

??????????