CSCI 4270 and 6270 Computational Vision

Lecture 02 – Images and OpenCV

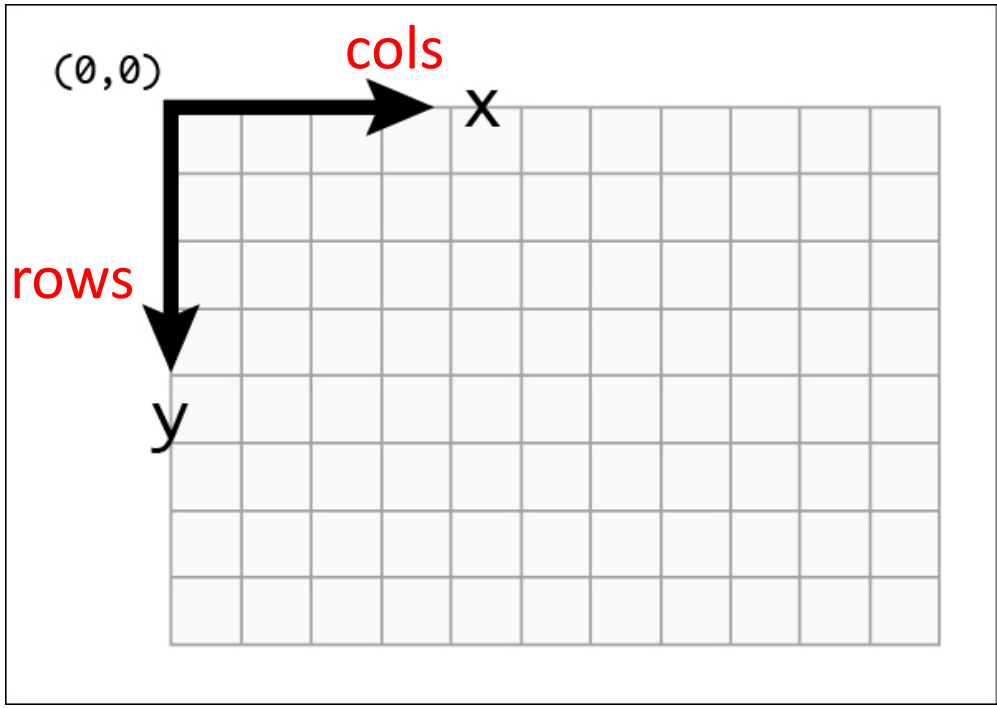
January 28, 2021

Overview

- Definitions
- Images and coordinate systems
- Color spaces
- Getting started with OpenCV, NumPy and MatPlotLib
- Some more advanced NumPy
- Examples:
  - Accessing and displaying
  - Combining images through averaging
  - Side-by-side display
  - Computing and displaying histograms
  - Stretching intensity ranges.
- Pointers to software tutorials are on the Submitty site

Images

- 2d array storing color or gray scale light intensity values
- Coordinate systems
  - Origin in upper left corner
  - Indexed x and y in OpenCV
  - But, indexed using rows and columns in NumPy
  - *Be aware of this difference in ordering!*



Pixel Values

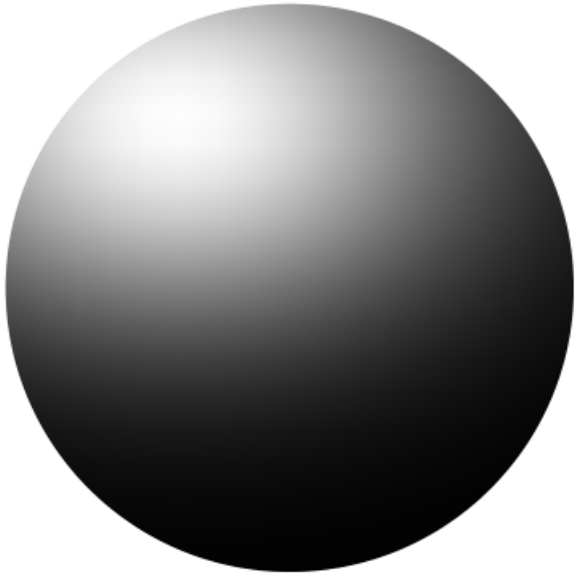
- Color:
  - RGB (red, green, blue) in most image formats
  - BGR in OpenCV
- Gray scale
  - Sometimes convert to gray scale before working with images
- Individual color and gray scale values:
  - Range [0,…, 255]
  - Unsigned byte
  - Computations generally use float, so need to convert back and forth
  - Sometimes we normalize floating point intensities into the range [0,1]
- Be aware of all of these issues as you start to program.

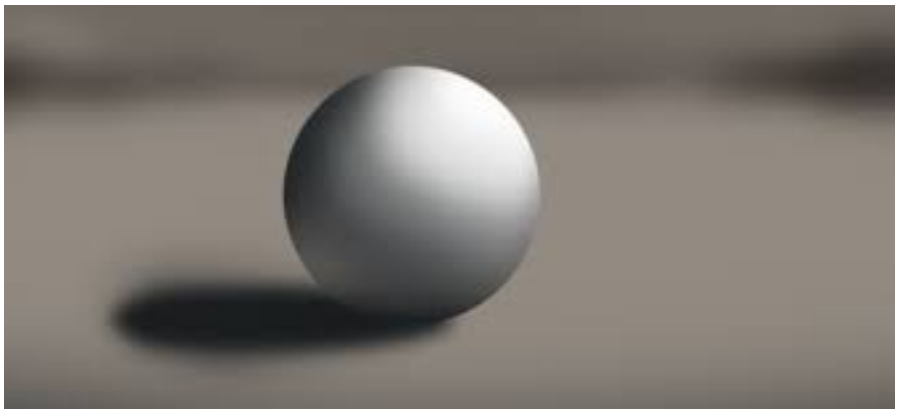
Image and Video Dimensions

- Large dimensions
  - 4K x 6K or larger is common
  - We tend to downsample images to 1K max linear dimension to work with them.
    - See OpenCV function *resize*
- Video resolution is typically anywhere from 320x240 to 1920x1080, with frame-rates of 30 images per second or more.

Light and Color: The Source of Intensities

- Intensities are formed from a combination of light source, surface normals, reflectance properties, air, lens, sensor and digital characteristics

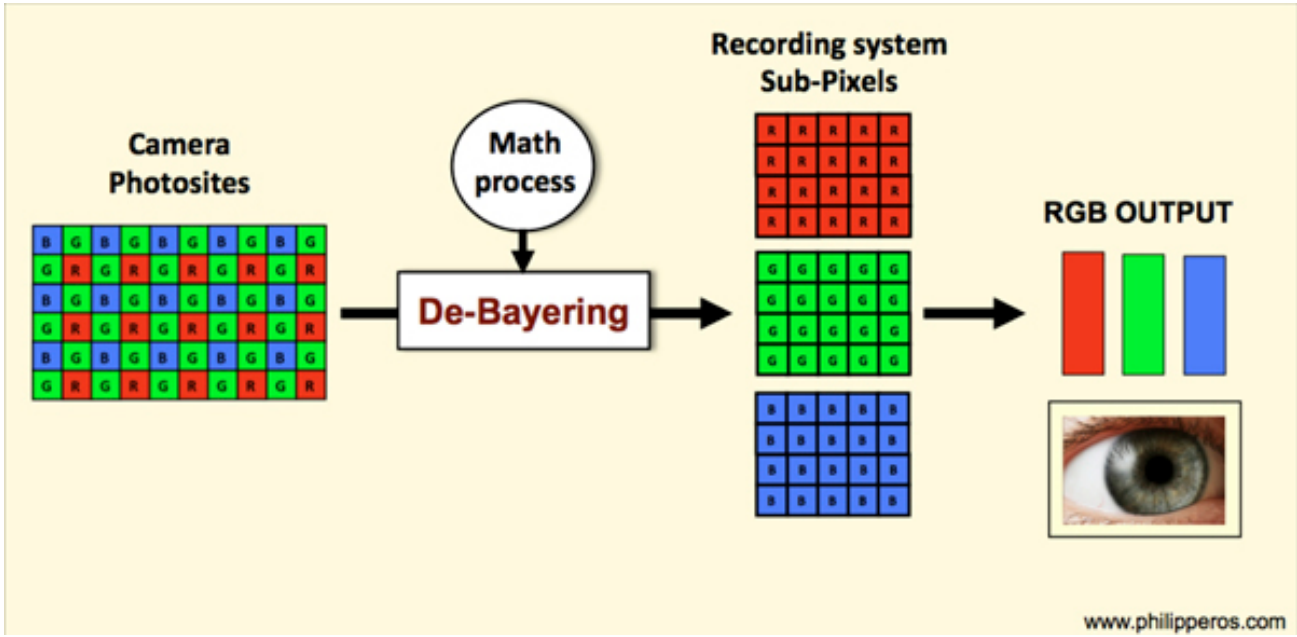




What shape are these? Where is the light source in each case?

Colors

- Camera sensor chips record with 2x the green sensors than red or blue, creating a “Bayer” pattern
  - Mimics the sensitivity of the human retina.
- The result is converted to RGB at each pixel through a process called “De-Bayering”



Color Spaces

- RGB is just one of many possible color representations
- RGB is not “perceptually uniform”
- Other, more uniform spaces, include
  - CIE and its variants, especially L*a*b
  - XYZ
  - HSV (hue, saturation, value)
- See http://www.easyrgb.com for conversions

Software Tools

- Three primary: OpenCV, NumPy, MatPlotLib
- OpenCV:
  - Use as a Python module
  - Manipulate and process images through function calls rather than objects
  - The C++ version is more object-oriented
- NumPy:
  - OpenCV functions work with and return images that are NumPy array objects
  - 2d (grayscale) or 3d (color) arrays
- MatPlotLib:
  - Tools for displaying images and graphs

Five Examples to Get Started

1. Access and display images
2. Combine two images through weighted averaging
3. Show two images side-by-side
4. Compute and display image histograms
5. Stretch the intensity range of an image to increase its contrast

Source code will be distributed separately and posted on the Submitty site.

Ex 1: Access and Display

Open and access an image. Display it using one of four different methods

- cv2 functions:
  - imread
  - imshow
  - resize
  - cvtColor
- NumPy:
  - Color images as 3d arrays, gray scale as 2d
  - Accessing pixels through indexing and the item function
  - Ordering differences in row/col vs. x/y and RGB vs. BGR
- MatPlotLib – pyplot
  - imshow
- Other Python:
  - random
  - Use of os functions and command-line arguments
  - List comprehensions

Ex 2: Combine Images

Create a new image from a linear combination of two equal-sized images.

- cv2 functions:
  - addWeighted
- NumPy:
  - Arithmetic operations
  - astype method
- MatPlotLib – pyplot
  - subplot to generate a grid of image outputs
- Other Python:
  - Random selection of images until a pair is found, each with the same dimensions

Ex 3: Side by Side

Create a new image showing side-by-side scaled images.

- cv2 functions:
  - imwrite to output an image to a file
  - resize
- NumPy:
  - Use of slicing to insert copies of one image into another.
  - Concatenation creates the same effect
- MatPlotLib – pyplot:
  - None
- Other Python:
  - None

Ex 4: Histograms

Display an image side-by-side with its histogram. Depending on the user's request, the histogram could be of the intensity or of RGB each separately.

- cv2 functions:
  - calcHist
  - What happens when imread fails?
- NumPy:
  - Converting the histogram counts to pixel percentages.
  - np.max function
- MatPlotLib – pyplot
  - Plotting a graph rather than an image
  - One or three curves in the graph
  - Controlling the color of the plot
  - Controlling the color of the image

Ex 5: Linear Stretch

Scale the intensities in an image so that they cover the entire range from 0 to 255. (See notes in code.)

- cv2 functions:
  - imread with automatic conversion to gray scale
- NumPy (a lot here, so we will work on examples in class):
  - histogram
  - cumsum
  - Converting to a true distribution
  - np.where
  - Calculating the cutoff
  - Boolean arrays
  - Boolean indexing of arrays
- MatPlotLib – pyplot
  - Displaying intensity values by providing a function to access the image

Summary

- Note the lack of explicit loops
  - Explicit loops are slow (!) and obscure the core logic
  - Learn to use the power of NumPy
    - (Very low grades if you don't :D)
- Interplay between OpenCV and NumPy:
  - OpenCV has algorithms implemented internally and accessed through function calls
  - NumPy is used to manipulate images directly and make up our own new algorithms
- Image display:
  - Mostly use MatPlotLib's PyPlot for displaying and debugging
  - Homework solutions will require explicit output of images to files to streamline grading.