Computer Vision

Tushar B. Kute,
http://tusharkute.com

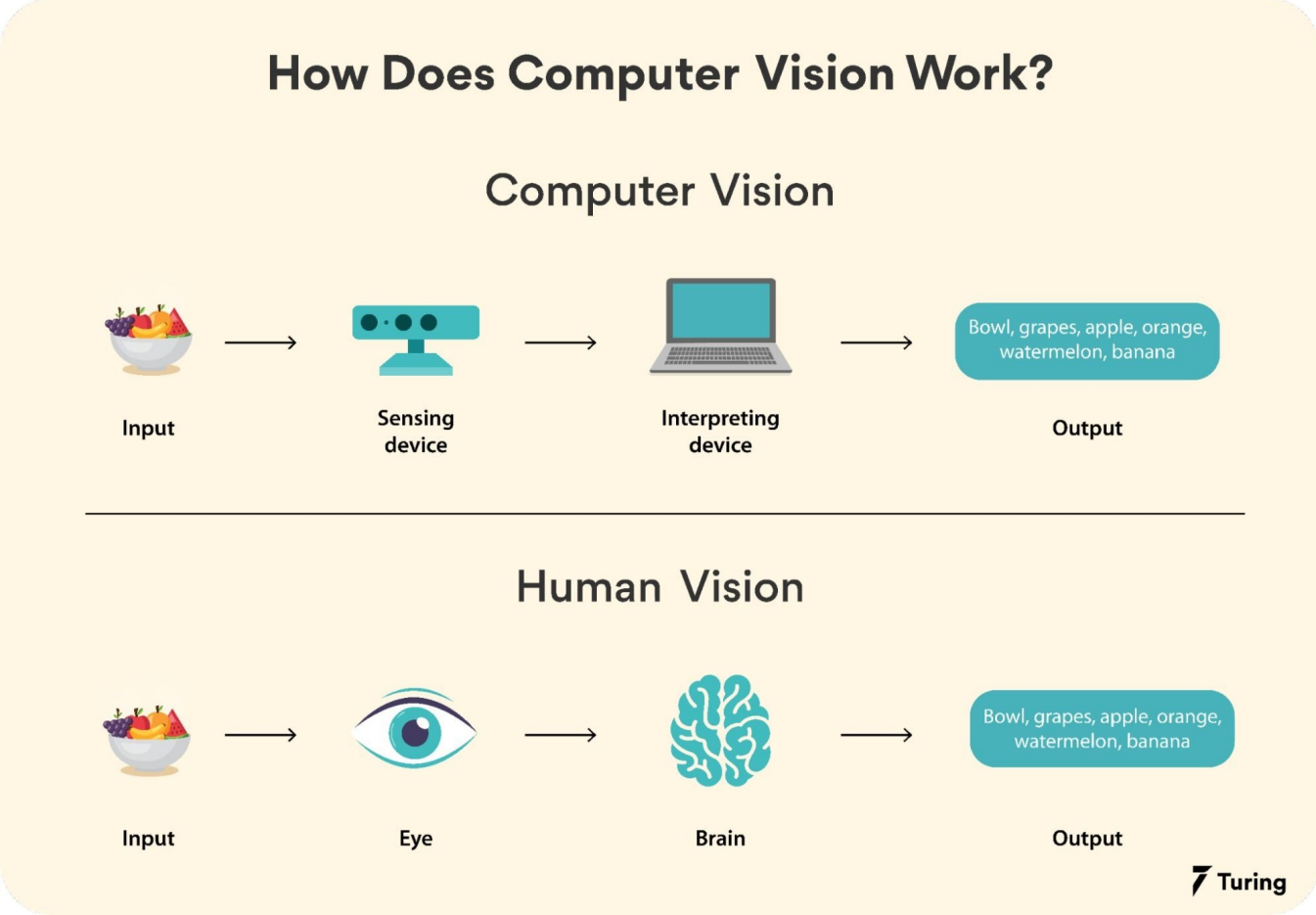
Computer Vision

- Computer vision is one of the fields of artificial intelligence that trains and enables computers to understand the visual world.
- Computers can use digital images and deep learning models to accurately identify and classify objects and react to them.
- Computer vision in AI is dedicated to the development of automated systems that can interpret visual data (such as photographs or motion pictures) in the same manner as people do.

Computer Vision

- The idea behind computer vision is to instruct computers to interpret and comprehend images on a pixel-by-pixel basis.
- This is the foundation of the computer vision field. Regarding the technical side of things, computers will seek to extract visual data, manage it, and analyze the outcomes using sophisticated software programs.
- The amount of data that we generate today is tremendous i.e. 2.5 quintillion bytes of data every single day.
- This growth in data has proven to be one of the driving factors behind the growth of computer vision.

Computer Vision: How?

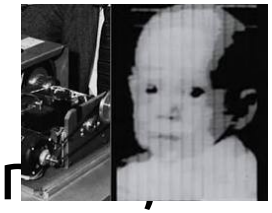


Computer Vision : Timeline

- Early Seeds (Pre-1960s):
  - Ancient Inspiration:
    - The earliest ideas for artificial vision can be traced back to ancient philosophers like Aristotle, who observed the principles of pinhole cameras.
  - Camera Obscura:
    - This optical device, dating back to the 5th century BC, projected an inverted image onto a screen, laying the foundation for capturing visual information.

Computer Vision : Timeline

- The Dawn of Computer Vision (1960s):



  - 1959: The first digital image scanner is inver , paving the way for computers to process visual data.
  - 1963: Larry Roberts, often called the "father of computer vision," publishes his thesis on extracting 3D information from 2D images, laying the groundwork for object recognition.
  - 1966: Marvin Minsky proposes the idea of connecting a camera to a computer, sparking the pursuit of machines that can "see" like humans.

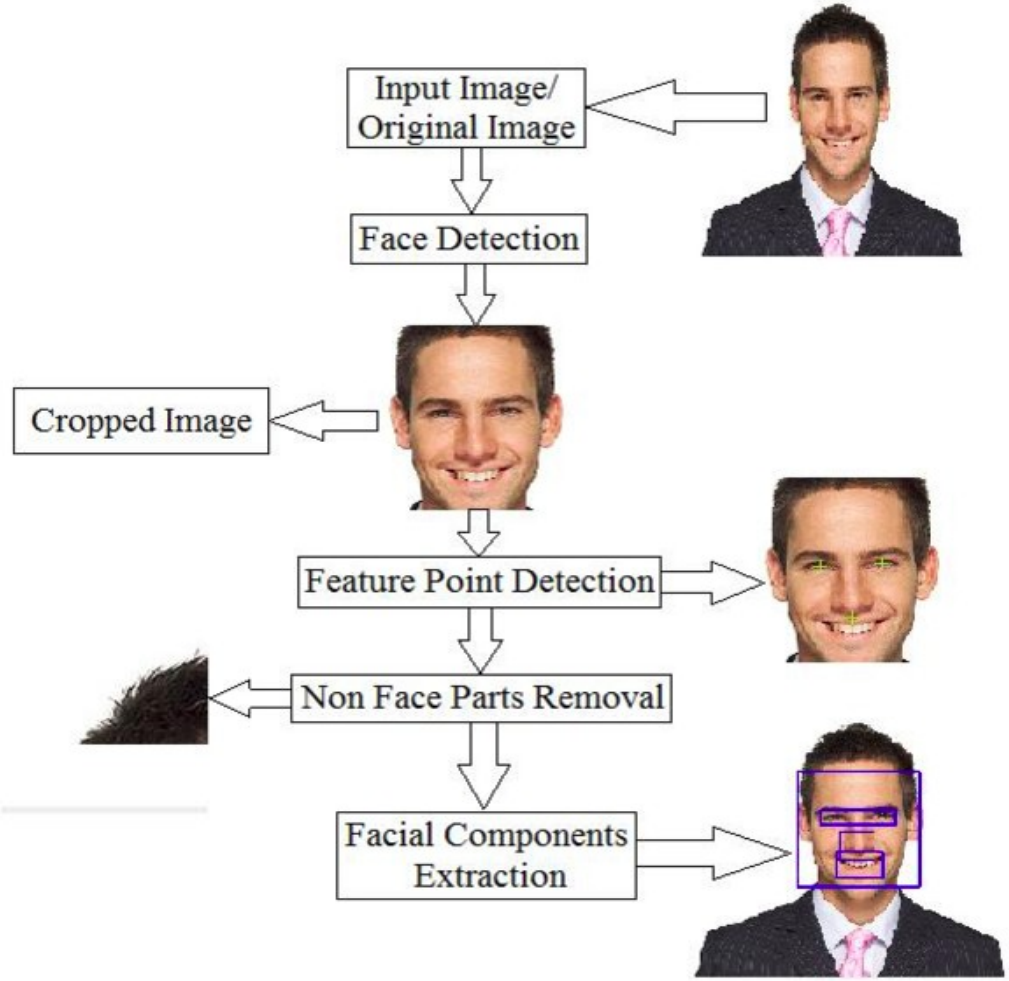
Computer Vision : Timeline

- Building the Foundations (1970s-1980s):
  - 1970s: Researchers focus on tasks like edge detection, line labeling, and 3D modeling, laying the groundwork for image understanding.
  - 1980s: Kunihiko Fukushima develops the "neocognitron," an early precursor to modern convolutional neural networks.
  - Advancements in image processing: Techniques like scale-space analysis and shape inference from shading and texture emerge, enabling more sophisticated image analysis.

Computer Vision : Timeline

- The Rise of Machine Learning (1990s-2000s):
  - 1990s: Camera calibration, multi-view stereo reconstruction, and image segmentation with graph cuts pave the way for advanced 3D scene understanding.
  - 2001: The Viola-Jones face detection algorithm revolutionizes real-time object recognition, leading to applications like facial recognition and surveillance.
  - 2009: Deep learning takes center stage with the rise of convolutional neural networks, bringing significant breakthroughs in image recognition and classification.

Computer Vision : Timeline



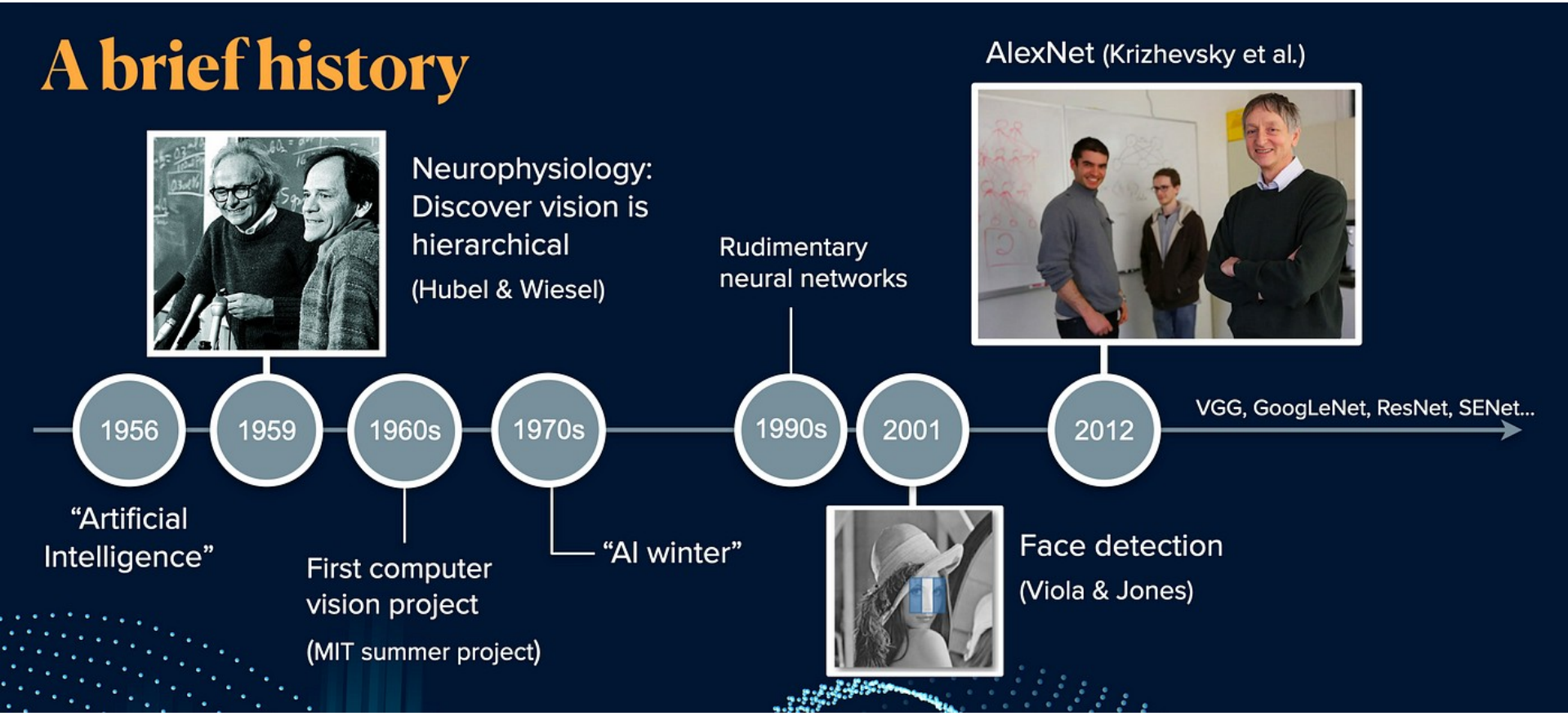
Computer Vision : Timeline

- The Deep Learning Revolution (2010s-Present):
  - ImageNet Challenge: This annual competition becomes a driving force for deep learning advancements, pushing the boundaries of image recognition accuracy.
  - Self-driving cars: Companies like Google and Tesla pioneer autonomous vehicles that rely heavily on computer vision for navigation and obstacle detection.

Computer Vision : Timeline

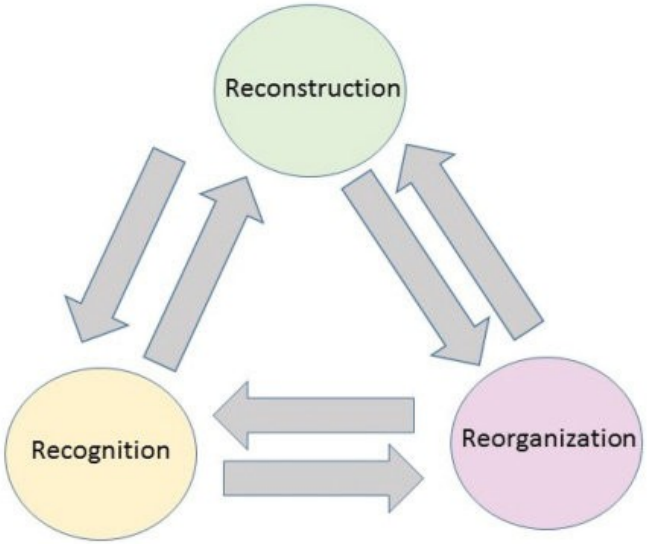
- The Deep Learning Revolution (2010s-Present):
  - Facial recognition: Advancements in deep learning make facial recognition more accurate and efficient, leading to its adoption in various fields like security and marketing.
  - Computer vision beyond images: Research expands into video analysis, 3D object understanding, and scene reconstruction, opening up new possibilities for robotics and augmented reality.

Computer Vision: History



Three R: Computer Vision

- The combination of natural language processing and computer vision involves three key interrelated processes: recognition, reconstruction, and reorganization.



Computer Vision

- Recognition:
  - This process involves assigning digital labels to objects within the image.
  - Examples of recognition are handwriting or facial recognition for 2D objects, and 3D assignments handle challenges such as moving object recognition which helps in automatic robotic manipulation.

Computer Vision

- Reconstruction:
  - This process refers to 3D scene rendering given inputs from particular visual images by incorporating multiple viewpoints, digital shading, and sensory depth data.
  - The outcome results in a 3D digital model that is then used for further processing.

Computer Vision

- Reorganization:
  - This process refers to raw pixel segmentation into data groups that represent the design of a pre-determined configuration.
  - Low-level vision tasks include corner detection, edges, and contours; while high-level tasks involve semantic segmentation, which can partly overlap with recognition processes.

Computer Vision Levels

- Computer vision is divided into three basic categories that are as following:
- Low-level vision:
  - includes process image for feature extraction.
- Intermediate-level vision:
  - includes object recognition and 3D scene Interpretation
- High-level vision:
  - includes conceptual description of a scene like activity, intention and behavior.

Applications

- Autonomous Vehicles:
  - Self-driving automobiles use CV systems to gather information regarding their surroundings and interpret that data to determine their next actions and behavior.
- Robotic Applications:
  - Manufacturing robotic machines using CV, 'view' and 'comprehend' their surroundings to perform their scheduled tasks.
  - In manufacturing, such systems inspect assembly items to determine faults and tolerance limits - simply by 'looking' at them as they traverse the production line.

Applications

- Image Search and Object Recognition:
  - Applications use CV data vision theory to identify specific objects within digital images, search through catalogs of product images, and extract information from photos.
- Facial Recognition:
  - Businesses and Government departments use facial recognition technology (that have adopted CV) to 'see' precisely what an individual is trying to gain access to.

Future

- Designing:
  - Within the area of home design, designer clothes, jewelry making, etc., customer systems can understand verbal or written requirements and thereby automatically convert these instructions to digital images for enhanced visualization.
- Describing Medical Images:
  - computer vision systems can be trained to identify more modest human ailments and use digital imagery in finer detail than human medical specialists.

Future

- Converting Sign Language: to speech or written text to assist the deaf and hard of hearing individuals in interacting with their surroundings. This enhanced capability can ensure their better integration within society.
- Surrounding Cognition: Constructing an intelligent system that 'sees' its surroundings and delivers a (recorded) spoken narrative. This outcome will be of use for visually impaired individuals.
- Converting Words to Images: Producing intelligent systems that convert spoken content to a digital image may assist people who do not talk and hear.

Loading an image

- Writing code in a specific programming language? (e.g., Python, Java, JavaScript)
- Working with a particular software or platform? (e.g., Photoshop, Google Docs, HTML)
- Trying to display an image on a website or device?

Loading an image

- OpenCV (cv2):
  - Popular for real-time computer vision and image processing.
  - Efficient for handling large images and videos.

Loading an image

- Pillow (PIL):
  - Offers simple and intuitive image manipulation tools.
  - Works well for basic image processing tasks.

Loading an image

- Matplotlib:
  - Primarily for data visualization, but can also handle images.
  - Integrates well with other scientific Python libraries.

Loading an image

- scikit-image (skimage):
  - Comprehensive image processing library.
  - Offers advanced image analysis and manipulation tools.

Loading an image

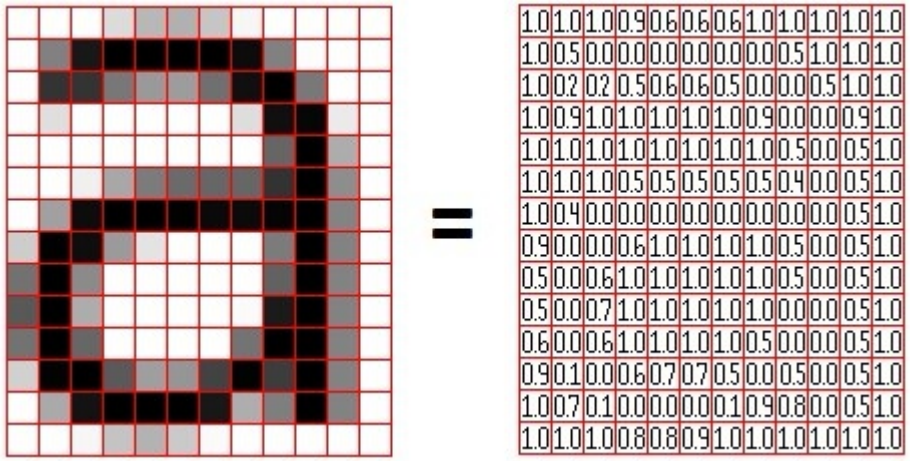
- TensorFlow:
  - Machine learning framework with image loading capabilities.
  - Useful when working with images for deep learning tasks.

Loading an image

- Key considerations when choosing a library:
  - Purpose: Consider the specific image processing tasks you need to perform.
  - Performance: Evaluate speed and memory usage for large images or videos.
  - Integration: Ensure compatibility with other libraries in your project.
  - Ease of use: Choose a library with a clear and concise API for your needs.

How images get converted to numbers?

- The transformation of an image into a series of numbers, like magic, is the foundation of how computers understand and process visual information.



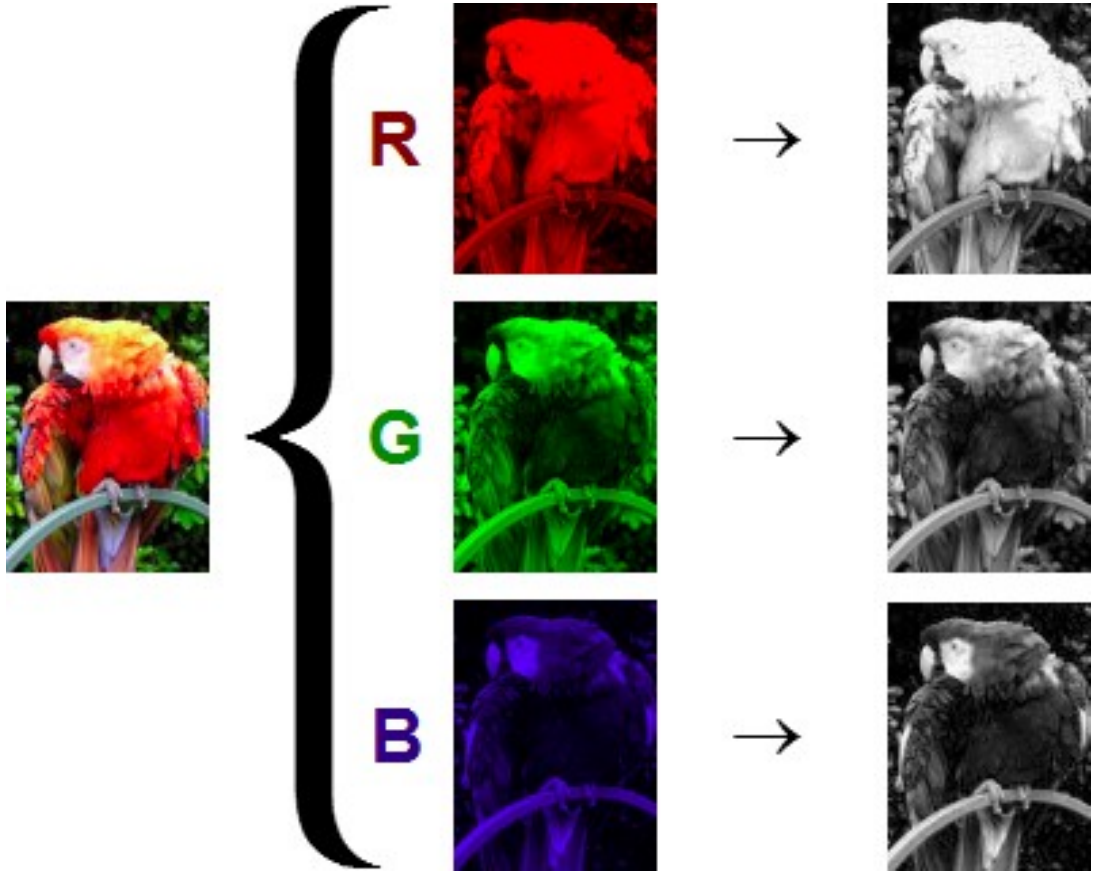
Pixel Power

- Imagine your favorite image like a mosaic.
- Each tiny square tile in that mosaic is called a pixel.
- It's the smallest unit of information in a digital image, and like a tiny paintbrush, it holds the color information for that specific point.

Color Decoding

- Computers don't directly understand colors like we do. They speak the language of numbers. So, each pixel's color needs to be translated into a numerical value. This depends on the image format:
  - Black and white: Each pixel gets a value between 0 (black) and 255 (white).
  - Grayscale: Similar to black and white, but with more tonal variations, ranging from 0 to 255, where higher values represent lighter shades.
  - Color (RGB): For vibrant hues, we have red, green, and blue (RGB) channels. Each channel gets a value between 0 and 255, and these values are combined to create the final color we see. Think of it like mixing primary colors to paint!

Color Decoding



Grid to numbers

- Now, picture all the pixels lined up neatly, forming a grid.
- Each pixel has its own numerical value based on its color.
- This grid of numbers becomes the digital representation of the image in the computer's memory.

Beyond the surface

- Simple images might only have one layer of numbers (like grayscale). But for complex images with depth and detail, there might be additional layers:
  - Alpha channel: This controls transparency, allowing pixels to be partially visible or invisible.
  - Depth channel: This encodes depth information for 3D scenes, used in applications like augmented reality.

Binary Symphony

- Ultimately, all these numbers are stored in the computer's memory as binary digits, or bits.
- These 0s and 1s are the fundamental language of computers, and the intricate arrangement of these bits represents the image in its entirety.
- So, the next time you see a digital image, remember the hidden symphony of numbers dancing behind the scenes, allowing computers to perceive and interpret the visual world just like us!

Grayscale vs. Color Images

- Colour pixels are different from grayscale pixels.
- Colour pixels are RGB, meaning they have three pieces of information associated with them, namely the Red, Green and Blue components.
- Grayscale pixels have one component, a gray tone derived from a graduate scale from black to white.
- A colour pixel is generally 24-bit (3 × 8-bit), and a gray pixel is just 8-bit.

Grayscale vs. Color Images

- This basically means that a colour pixel has a triplet value comprised of 0..255 for each of red, green and blue components, whereas a grayscale pixel has a single values 0..255.
- The figure below compares a colour and grayscale pixel.
- The colour pixel has the R-G-B value 61-80-136.The grayscale pixel has the value 92.

Grayscale vs. Color Images

- It is easy to convert a pixel from colour to grayscale (like applying a monochrome filter in a digital camera).
- The easiest method is simply averaging the three values of R, G, and B. In the sample above, the grayscale pixel is actually the converted RGB: (61+80+136)/3 = 92.

Grayscale vs. Color Images

- Now colour images also contain regions that are gray in colour – these are 24-bit “gray” pixels, as opposed to 8-bit grayscale pixels.
- The example below shows a pixel in a grayscale image, and the corresponding “gray” pixel in the colour image.
- Grayscale pixels are pure shades of gray. Pure shades of gray in colour images are often represented with RGB all having the same value, e.g. R=137, G=137, B=137.

Grayscale vs. Color Images

Grayscale

- Advantages:
  - Smaller file size, making them more efficient for storage and transmission.
  - Simpler and faster to process, making them suitable for real-time applications.
  - Can highlight certain features or textures that might be masked by color in the original image.
  - Can evoke a sense of nostalgia or timelessness.

Grayscale

- Disadvantages:
  - Lack the color and vibrancy of the original image, making it visually less appealing.
  - Certain tasks like object recognition might be more challenging due to the absence of color information.
  - Not suitable for applications requiring accurate color representation.

Color Images

- Advantages:
  - More realistic and visually appealing, showcasing the full richness of the visual world.
  - Useful for tasks like object recognition and classification, as color can be a key discriminant feature.
  - Generally preferred for artistic expression and emotional impact.

Color Images

- Disadvantages:
  - Larger file size due to the presence of three channels.
  - Increased processing requirements for manipulation and analysis.
  - Certain features or details might be obscured due to color dominance.

Which one to choose?

- The importance of color:
  - If color is crucial for understanding the content or conveying emotions, then color images are the better choice.
- File size and processing requirements:
  - If storage space or processing speed is a concern, grayscale might be a better option.
- Target audience and application:
  - Who will be viewing the images and for what purpose? Consider their preferences and the specific needs of the application.

Grayscale vs. Color: Summary

- Grayscale images have a single channel representing intensity, while color images have three channels (RGB).
- Grayscale conversion is often used for tasks like edge detection, object recognition, and reducing processing overhead.

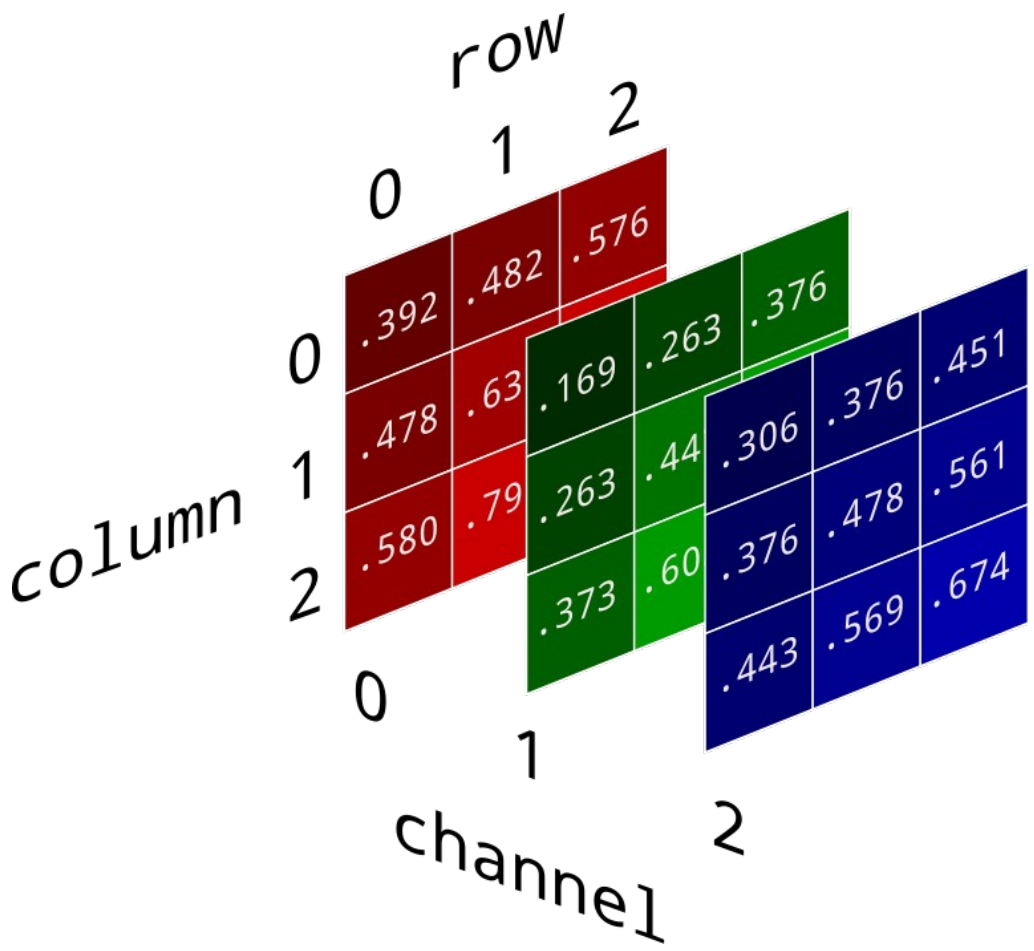
Saving Images

- cv2.imwrite('saved_image.jpg', img)
- img.save('saved_image.png') # PIL
- plt.imsave('saved_image.bmp', img)
- io.imsave('saved_image.tif', img) # scikit
- tf.io.write_file('saved_image.png', tf.image.encode_png(img))

RGB Channels

- RGB channels," refers to the three primary color channels that combine to form the vast majority of digital images and colors we see on screens.
- Imagine each pixel in an image as a tiny chamber, and within that chamber exist three separate compartments, each one dedicated to a specific color:
  - Red, Green, and Blue.

RGB Channels



RGB Channels

- Red:
  - This channel controls the intensity of the red component in the pixel's overall color.
  - Higher values of red will make the pixel appear more reddish, while lower values will bring out other colors or tend towards black.

RGB Channels

- Green:
  - Similar to red, the green channel governs the green component in the pixel's color. Adjusting its value influences how much green is present, affecting the overall hue and vibrancy.
- Blue:
  - Lastly, the blue channel manages the blue component, adding depth and coolness to the pixel's color. Its intensity plays a crucial role in creating various shades and tones.

RGB Channels: Combination

- Mixing high red and green results in yellow, while combining red and blue creates purple.
- Balancing all three channels at their maximum level (255 each) produces pure white, and setting them to zero yields black.
- The interplay of these values in every pixel across an image forms the intricate tapestry of colors and details we perceive.

BGR Channels

- While RGB (Red, Green, Blue) is the most commonly used and intuitive way to represent color channels in digital images, some systems, particularly in computer vision and image processing, utilize BGR (Blue, Green, Red) instead.

BGR Channels: Why?

- Hardware Optimization: Some early image processing hardware was optimized for BGR order, making it the default for certain libraries and frameworks.
- OpenCV Legacy: OpenCV, a popular library for computer vision, adopted BGR as its default due to compatibility with older hardware and software.
- Channel Interpretation: While the interpretation of red, green, and blue remains the same, the order simply swaps the blue and red channels. Both RGB and BGR effectively capture the full spectrum of colors.

BGR Channels: Understanding

- While BGR might seem unfamiliar at first, knowing its existence and potential reasons for usage is crucial when working with computer vision and image processing libraries.
  - Always check the documentation of the specific library or framework you're using to determine the default color channel order.
  - Be aware of potential conversion needs when comparing or transferring data between systems using different conventions.

How to convert?

- OpenCV:
  - cv2.cvtColor(img, cv2.COLOR_BGR2RGB) or cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
- Pillow:
  - img.convert('RGB') or img.convert('BGR')
- Matplotlib:
  - Similar to Pillow, using the appropriate color space when saving or displaying the image.

RGB vs. BGR: Function

- Both RGB and BGR represent the same color information within a pixel.
- They control the intensity of red, green, and blue components to create the entire spectrum of colors we see on screens.

RGB vs. BGR: Order

- The key difference lies in the order of the channels.
  - RGB: This is the more common and intuitively understood order in many fields like graphic design, media, and consumer electronics.
  - BGR: This order is primarily used in computer vision and image processing libraries like OpenCV, often due to historical conventions and hardware optimization for early image processing setups.

RGB vs. BGR: Interplay

- Both RGB and BGR channels work together to produce vibrant colors.
- Adjusting their values influences the overall hue and saturation of a pixel.
- Higher values of a specific channel increase its presence, while lower values bring out other colors or tend towards black.

RGB vs. BGR: Conversion

- Switching between RGB and BGR is usually straightforward:
  - Most libraries and frameworks offer built-in functions for conversion, like cv2.cvtColor() in OpenCV or img.convert() in Pillow.
  - Be aware of the default color space used by the specific library or framework you're working with.

RGB vs. BGR: Impact

- The choice between RGB and BGR primarily impacts technical aspects:
  - Data compatibility:
    - When transferring data between systems using different conventions, conversion might be necessary.
  - Library-specific code:
    - If using BGR-based libraries like OpenCV, be mindful of color channel order in calculations and interpretations.

cmap

- In the realm of OpenCV, cmap acts as a magical paintbrush, enabling you to create captivating visualizations of images that transcend beyond the ordinary.
- It's often used with functions like applyColorMap() to apply different colormaps, each revealing unique information and enhancing visual perception.

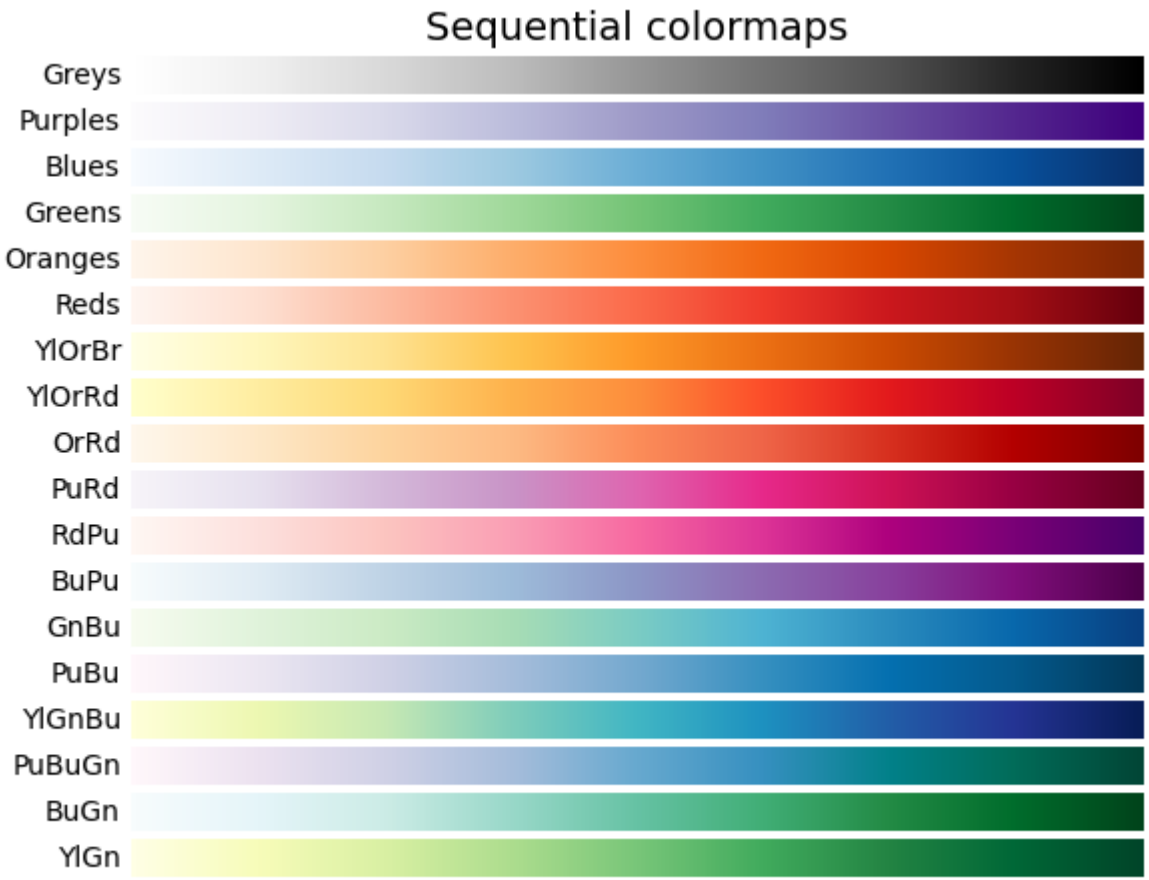
ColorMap

- It's a predefined set of colors that are mapped to numerical values within an image.
- It acts as a translator, converting numerical data into visually interpretable colors.
- OpenCV offers a diverse palette of built-in colormaps, each catering to specific visualization needs.

ColorMap: Sequential

- Ideal for representing continuous data with a natural progression of colors.
- Examples:
  - Grayscale: Showcases intensity variations in shades of gray.
  - Jet: Progresses from blue to red, often used for heatmaps and gradients.
  - Hot: Emphasizes higher values with a transition from black to red.
  - Bone: Simulates X-ray images with a bone-like appearance.

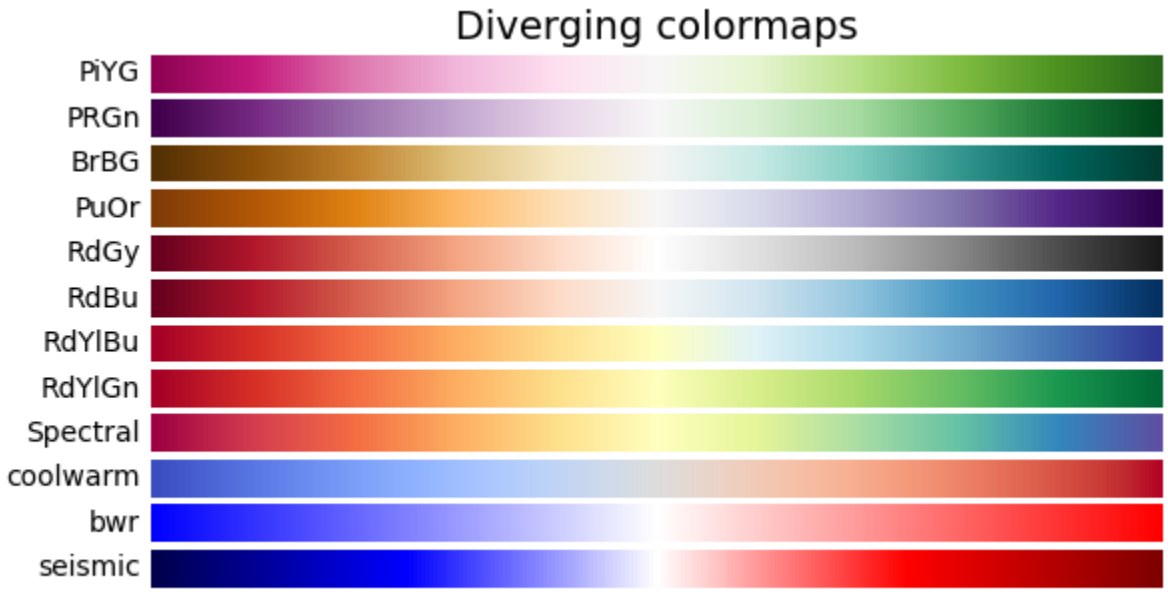
ColorMap: Sequential



ColorMap: Diverging

- Designed to highlight both high and low values, diverging from a central color.
- Examples:
  - Coolwarm: Transitions from blue to red, useful for representing bipolar data.
  - Spectral: Spans a wide range of hues, often used for scientific visualizations.
  - Seismic: Employs a seismic-like color scheme with distinct divisions.

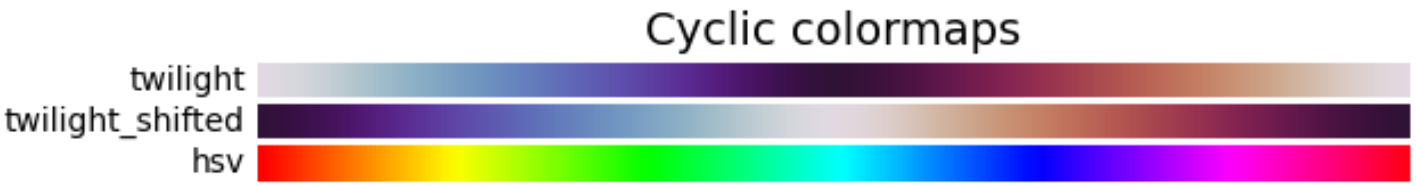
ColorMap: Diverging



ColorMap: Cyclic

- Wraps colors around a circular spectrum, useful for representing periodic data.
- Examples:
  - HSV: Represents Hue, Saturation, and Value, creating a circular color wheel.
  - Rainbow: Mimics a rainbow's spectrum, often used for categorical data.
  - Turbo: Offers a visually balanced colormap with smooth transitions.

ColorMap: Cyclic



Colormap in matplotlib

- Sequential: Ideal for representing numerical data with a continuous range, where high and low values are clearly distinguished. Examples include:
  - viridis: Perceptually uniform and widely used for its visual clarity.
  - plasma: Offers a visually pleasing purple-to-yellow gradient.
  - inferno: Emphasizes mid-range values with a fiery color scheme.

Colormap in matplotlib

- Diverging: Optimized for data with a central neutral point and values that diverge towards positive and negative extremes. Examples include:
  - RdBu: Transitions from red to blue, effectively highlighting differences.
  - coolwarm: Shifts from cool colors (blue) to warm colors (red), useful for representing opposing trends.

Colormap in matplotlib

- Cyclical: Suitable for data with a cyclical nature, where the highest and lowest values smoothly blend. Examples include:
  - hsv: Cycles through hues, useful for representing angular quantities.
  - twilight: Mimics the colors of twilight, ideal for periodic data.

Colormap in matplotlib

- Qualitative: Best for categorical data where different colors represent distinct categories rather than numerical values. Examples include:
  - tab10: Ten distinct colors for representing up to ten categories.
  - Set2: Eight aesthetically pleasing colors for categorical distinctions.

Thank you

@mitu_skillologies

@mITuSkillologies

@mitu_group

@mitu-skillologies

@MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com